

FUJITSU Software Interstage Studio



User's Guide

B1WD-3158-02ENZ0(00)
November 2013

Preface

Interstage Studio is a Java integrated development environment for developing Java EE 5 and Java EE 6 Web applications, and for developing Enterprise JavaBeans (EJB). Workbench is based on the currently widely used Eclipse open source development environment, and enables development to be performed with industry-wide standard operability. It is also possible to develop J2EE1.4 applications, but Java EE applications are recommended.

Purpose of This Manual

This manual explains development methods and operation procedures and provides development-related notes for workbench functions.

Intended Readers

This manual presents the information required by users who use Workbench to develop various types of applications. This manual assumes that the reader has a basic knowledge of Java EE Web applications and EJB.

Conventions

This manual uses the following conventions:

- For easy recognition, button names, menu options, dialog box names, and context menu options are displayed within [].
For example: Select [Add] > [Template] from the context menu.
- The names of files, items and relationships are displayed within " ".
- This product creates a group name of [Interstage Studio Vxx] on the Apps screen (in Windows 8 and Windows Server 2012) and the Start menu (in other Windows systems). For the actual group name, check the "Software Release Guide" and substitute this name when reading.
- For details on how to read install folders, refer to the "Software Release Guide", section "Folder Structure and Files".
Example:
<Workbench installation folder> is the install folder for all functions listed in the "Folder Structure and Files" correspondence table in the "Software Release Guide".

Workbench name

Explanations common to all workbenches are labeled with "Workbench" in this document. When it is necessary to explain different workbench features, "Standard workbench" labels the explanations for workbenches installed as standard when the product is installed, and "Java EE 6 workbench" labels the explanations for workbenches installed when the Java EE 6 development feature is selected.

Export Controls

This document or a portion thereof may not be exported (or re-exported) without authorization from the appropriate government authorities in accordance with the pertinent laws.

Trademarks

- Microsoft, Active Directory, ActiveX, Excel, Internet Explorer, MS-DOS, MSDN, Visual Basic, Visual C++, Visual Studio, Windows, Windows NT, Windows Server, Win32 are either registered trademarks or trademarks of Microsoft Corporation in the United States and/or other countries.
- Oracle and Java are registered trademarks of Oracle and/or its affiliates. Other names may be trademarks of their respective owners.
- Other company and product names in this documentation are trademarks or registered trademarks of their respective owners.

Revision History

Date of Publication and Version	Manual Code
November 2013: 2nd Version	B1WD-3156-02ENZ0(00)/B1WD-3156-02ENZ2(00)
November 2012: 1st Version	B1WD-3156-01ENZ0(00)/B1WD-3156-01ENZ2(00)

Copyright Notice

Copyright 2012-2013 FUJITSU LIMITED

How to Read this Manual

This section explains how to read this document.

Structure and overview of manuals

This document explains development with the standard workbench and the Java EE 6 workbench. Follow the explanations in this section to read the parts of the manual related to the workbench you are using for development.

Examples for the standard workbench include IJServer cluster examples. There are functions that can be used with both the IJServer cluster and the Interstage Java EE DAS service. Refer to the "Interstage Application Server Java EE Operator's Guide" for information on the differences.

When you are developing the Java EE 6 Application, please start reading from "[Chapter 9 Developing Java EE 6 Applications](#)".

Procedure explanations

The different types of application (Web, EJB, etc.) to be developed are explained in different chapters. Samples are used in the "Introduction" sections of the chapters to help explain the procedures for developing the applications. The procedures explained in the "Introduction" sections use the standard workbench procedures. The items to be set when developing with the Java EE 6 workbench may be different.

When the Java EE 6 workbench is used

- Target Runtime /Configuration

Target Runtime	Interstage Application Server V11.1 (Java EE 6)
Configuration	Default Configuration for Interstage Application Server V11.1 (Java EE 6)

- Helpful procedures and section names

Contents	References	
	Help	Contents
9.2.1.3 Developing Web Applications	Web Tools Platform User Guide	Creating Web applications
9.3.1.3 Developing EJB	Web Tools Platform User Guide	Developing EJB applications
9.4.1.3 Developing Applications using JPA	Dali Java Persistence Tools User Guide	-
9.5.1.3 Developing Web Service Applications	Web Tools Platform User Guide	Developing Web service applications

Workbenches corresponding to the chapters

The following table describes the information in each chapter and the workbench it relates to. Regarding the development of applications in the Java EE 6 workbench, the points that are different to the standard workbench are explained in "Description". Also read "Procedure explanations" above.

Contents	Description	Workbench	
		Standard workbench	Java EE 6 workbench
Chapter 1 Workbench Overview	Explains the workbench components such as editor and view, development resource formats such as workspace and project, and basic workspace operations.	Y	Y

Contents	Description	Workbench	
		Standard workbench	Java EE 6 workbench
Chapter 2 Developing Web Applications	Provides an overview of Web applications, and explains how to create Web applications.	Y	Y
Chapter 3 Developing Enterprise JavaBeans (EJB)	Provides an overview of Enterprise JavaBeans, and explains how to create Enterprise Bean and EJB client applications.	Y	Y
Chapter 4 Developing Applications that use Java Persistence API (JPA)	Provides an overview of the Java Persistence API, and explains how to create applications using JPA.	Y	Y
Chapter 5 Developing Web Service Applications	Provides an overview of Web services, and explains how to create Web service applications.	Y	Y
Chapter 6 Items Common to Java EE 5 Applications	Provides some common explanations for creating Java EE 5 applications.	Y	-
Chapter 7 Developing Java Applications	Provides an overview of Java applications, and explains how to create Java applications.	Y	Y
Chapter 8 Database Operation	Provides an overview of databases, and explains how to operate databases. The correct behavior of the functions described in this document depends on the JDBC drivers used by the various products. Use the tools and applications provided by the various products for database operations such as creating, updating, and deleting content.	Y	Y
Chapter 9 Developing Java EE 6 Applications	Provides an overview of Java EE 6 applications, and explains the process from application creation to deployment.	-	Y
Chapter 10 Tips	Explains features useful when using workbenches and points to consider regarding Java EE application programming.	Y	Y
Appendix A How to use Samples	Explains the samples that are provided.	Y	Y
Appendix B Migrating Resources from a Previous Version	Explains the migration of resources that were developed using older versions of the workbench.	Y	Y
Appendix C Procedure for Development Using JDK 6	Describes the procedure for using JDK 6 in development on a Java EE 6 workbench.	-	Y
Appendix D Migrating J2EE 1.4 Applications	Explains how to develop J2EE1.4 applications.	Y	-
Appendix E Troubleshooting	Explains how to resolve problems that occur in the workbench.	Y	Y
Appendix F Tutorial	Explains the procedure for application development with examples.	Y	-

Y: Supported.

-: Not supported.

Contents

Chapter 1 Workbench Overview.....	1
1.1 Basic Workbench Concepts.....	1
1.1.1 Workbench and its Elements.....	1
1.1.2 Development Resources.....	2
1.1.3 Basic Workbench Tasks.....	2
1.2 Basic Workbench Operations.....	3
1.2.1 Operations Related to Development Resources.....	3
1.2.2 Operations Related to the Workbench.....	4
1.2.3 Operations Related to the Main Tasks.....	4
Chapter 2 Developing Web Applications.....	6
2.1 Overview.....	6
2.1.1 What are Web Applications?.....	6
2.1.1.1 Modifications since J2EE1.4.....	7
2.1.2 Web Application Development.....	8
2.2 Introduction.....	8
2.2.1 Application to be Created.....	8
2.2.2 Development Flow.....	9
2.2.3 Development Procedures.....	10
2.3 Tasks.....	19
2.3.1 Preparing the Environment for Creating the Web Application.....	19
2.3.2 Creating a Servlet.....	20
2.3.3 Creating an HTML File.....	21
2.3.3.1 Creating a New HTML File.....	21
2.3.3.2 Editing the HTML File.....	22
2.3.3.3 Validating the HTML Tags.....	23
2.3.4 Creating a JSP File.....	23
2.3.4.1 Creating a New JSP File.....	23
2.3.4.2 Editing the JSP File.....	24
2.3.4.3 Validating the JSP.....	25
2.3.5 Graphically Editing the HTML/JSP Files.....	25
2.3.6 Creating the JavaScript.....	26
2.3.7 Creating the CSS.....	27
2.3.8 Editing the web.xml.....	28
2.3.9 Verifying the Web Application Behavior.....	28
2.3.10 Verifying JavaScript Operation.....	28
2.3.11 Distributing the Web Application to the Operating Environment.....	30
Chapter 3 Developing Enterprise JavaBeans (EJB).....	31
3.1 Overview.....	31
3.1.1 What is an EJB?.....	31
3.1.1.1 Modifications since J2EE1.4.....	32
3.1.2 Developing EJBs.....	34
3.2 Introduction.....	35
3.2.1 Application to be created.....	35
3.2.2 Development Flow.....	35
3.2.3 Development Procedures.....	37
3.3 Tasks.....	46
3.3.1 Preparing the Environment to create EJBs.....	47
3.3.2 Creating Session Beans.....	47
3.3.3 Creating Message-driven Beans.....	48
3.3.4 Using Databases.....	49
3.3.5 Creating EJB Clients.....	49
3.3.5.1 Creating Clients that invoke Session Beans.....	49
3.3.5.2 Creating Clients that send Messages to Message-driven Beans.....	50

3.3.6 Checking EJB Operation.....	51
3.3.7 Distributing EJBs to the Operating Environment.....	51
Chapter 4 Developing Applications that use Java Persistence API (JPA).....	52
4.1 Overview.....	52
4.1.1 What is JPA?.....	52
4.1.1.1 Modifications since EJB2.1.....	52
4.1.2 Developing Applications that use JPA.....	53
4.2 Introduction.....	54
4.2.1 Application to be created.....	54
4.2.2 Development Flow.....	54
4.2.3 Development Procedures.....	57
4.3 Tasks.....	69
4.3.1 Preparing the Environment to create Applications that use JPA.....	69
4.3.1.1 Setting the JPA Facet.....	70
4.3.2 Developing Persistence Units.....	71
4.3.3 Creating Entity Classes.....	72
4.3.3.1 Creating an Entity taking Data Structures into Account.....	73
4.3.3.2 Mapping the Entity and Database.....	75
4.3.3.3 Defining Relationships between Entities.....	76
4.3.3.4 Generating Entity Classes from Tables.....	78
4.3.4 Operating Database with JPA.....	78
4.3.5 Checking Operation of Applications that use JPA.....	80
4.3.6 Distributing Applications that use JPA to the Operating Environment.....	80
Chapter 5 Developing Web Service Applications.....	81
5.1 Overview.....	81
5.1.1 What is a Web Service?.....	81
5.1.1.1 Modifications since J2EE1.4.....	81
5.1.2 Web Services Development.....	83
5.2 Introduction.....	83
5.2.1 Application to be created.....	83
5.2.2 Development Flow.....	84
5.2.3 Development Procedures.....	85
5.3 Tasks.....	96
5.3.1 Preparing the Environment to create the Web Service.....	96
5.3.2 Creating the Web Service.....	97
5.3.2.1 Exposing a Java Class as a Web Service.....	97
5.3.2.2 Creating a Web Service from WSDL.....	97
5.3.3 Exposing a Stateless Session Bean as a Web Service.....	98
5.3.4 Creating WSDL.....	99
5.3.5 Creating a Service Endpoint Interface from WSDL.....	100
5.3.6 Creating a Web Service Client.....	101
5.3.7 Checking Web Service Operation.....	102
5.3.7.1 Using the Web Service Explorer as a Web Service Client.....	102
5.3.7.2 Using the TCP/IP Monitor to check Web Service Messages.....	103
5.3.8 Distributing Web Services to the Operating Environment.....	104
Chapter 6 Items Common to Java EE 5 Applications.....	105
6.1 Overview.....	105
6.1.1 What is Java EE 5?.....	105
6.1.2 Developing Java EE 5 Applications.....	105
6.2 Tasks.....	105
6.2.1 Preparing the Deployment Destination for Verifying Application Operation.....	106
6.2.1.1 Creating an IJServer Cluster (MyDebugJEE).....	106
6.2.2 Preparing to Create Applications.....	107
6.2.2.1 Creating New Projects.....	107
6.2.2.2 Creating Enterprise Applications.....	108

6.2.2.3 Setting Classpaths.....	109
6.2.2.4 Developing Using a Java EE Module that has no Development Resources.....	109
6.2.3 Creating the Java Class and the Interface.....	109
6.2.4 Creating XML Files.....	110
6.2.5 Detecting and Correcting Problems.....	111
6.2.5.1 Java Compiler.....	112
6.2.5.2 Validation.....	112
6.2.5.3 Interstage Java EE Validator.....	113
6.2.5.4 FindBugs.....	113
6.2.6 Checking Application Behavior.....	117
6.2.6.1 Debugging.....	120
6.2.6.2 Starting the Interstage Java EE Admin Console.....	121
6.2.7 Distributing to the Operating Environment.....	121
6.2.8 Developing Java EE Applications Using Entity Beans.....	122
Chapter 7 Developing Java Applications.....	123
7.1 Overview.....	123
7.1.1 What is a Java Application?.....	123
7.1.2 Developing Java Applications.....	123
7.2 Introduction.....	124
7.2.1 Applet that is Created.....	124
7.2.2 Development Flow.....	124
7.2.3 Development Procedures.....	125
7.3 Tasks.....	131
7.3.1 Preparing the Environment for Creating Java Applications.....	132
7.3.2 Creating a Class.....	132
7.3.3 Creating an Applet.....	132
7.3.3.1 Applet Information.....	133
7.3.4 Creating a Form.....	133
7.3.4.1 Creating a Frame, Dialog, Panel.....	134
7.3.4.2 Creating a Screen Control Panel, Screen Control Tabbed Panel, Window Control, and Object Control.....	134
7.3.4.3 Java Form Information.....	135
7.3.5 Creating JavaBeans.....	135
7.3.5.1 JavaBeans Information.....	136
7.3.6 Editing a Java Form.....	136
7.3.6.1 Setting a Bean.....	137
7.3.6.2 Laying out a Bean.....	137
7.3.6.3 Referencing or Setting Bean Properties.....	137
7.3.6.4 Creating Bean Relationships.....	139
7.3.6.5 Defining a Menu.....	140
7.3.6.6 Defining an Exclusive Selection Group.....	140
7.3.7 Editing the Screen Control Panel and Window Control Panel.....	141
7.3.7.1 Editing a Screen Control Panel.....	141
7.3.7.2 Editing a Window Control Panel.....	142
7.3.8 Defining BeanInfo.....	143
7.3.9 Describing User Interface Processing.....	144
7.3.9.1 Describing Processing of an Application with a User Interface.....	144
7.3.9.2 Operating a Form.....	144
7.3.9.3 Operating a Bean.....	145
7.3.9.4 Describing Event Processing.....	145
7.3.10 Performing Environment Setup for Java Applications.....	146
7.3.10.1 Registering a Bean.....	146
7.3.10.2 Custom Wizards.....	146
7.3.10.3 Setting Options.....	146
7.3.10.4 List of Forms.....	148
7.3.11 Verifying Java Application Operation.....	149
7.3.12 Distributing a Java Application to the Operational Environment.....	149

7.3.13 Developing Java Applications with JDK 6.....	150
7.3.13.1 Preparing the Environment for Creating Java Applications.....	150
7.3.14 Notes.....	150
Chapter 8 Database Operation.....	153
8.1 Overview.....	153
8.1.1 What is a Database?.....	153
8.1.2 Functions that Operate Databases.....	153
8.1.3 Limitations.....	154
8.2 Introduction.....	154
8.2.1 Database to be Created.....	154
8.2.2 Development Flow.....	154
8.2.3 Development Procedures.....	155
8.3 Tasks.....	159
8.3.1 Connecting to the Database.....	159
8.3.2 Referencing Database Contents.....	160
8.3.3 Executing SQL Statements.....	160
8.3.4 Creating Tables.....	162
8.3.5 Deleting Tables.....	162
8.3.6 Updating Table Data.....	163
8.3.6.1 Editing the Data.....	163
8.3.6.2 Extracting and Loading Data.....	163
8.3.7 Supported Database Information.....	164
8.3.7.1 JDBC Drivers.....	164
8.3.7.2 Connection Profile Properties.....	167
8.3.7.3 Editable Data Types.....	169
8.3.8 Using the DB Access Class Wizard to Automatically Generate JDBC Processes.....	172
Chapter 9 Developing Java EE 6 Applications.....	175
9.1 Overview.....	175
9.1.1 What is Java EE 6?.....	175
9.1.2 Developing Java EE 6 Applications.....	175
9.2 Developing Web Applications.....	176
9.2.1 Overview.....	176
9.2.1.1 What are Web Application?.....	176
9.2.1.2 Changes from Java EE 5.....	176
9.2.1.3 Developing Web Applications.....	177
9.3 Developing Enterprise JavaBeans (EJB).....	177
9.3.1 Overview.....	177
9.3.1.1 What is an EJB?.....	177
9.3.1.2 Changes from Java EE 5.....	177
9.3.1.3 Developing EJB.....	178
9.4 Developing Applications using Java Persistence API.....	178
9.4.1 Overview.....	178
9.4.1.1 What is JPA?.....	178
9.4.1.2 Changes from Java EE 5.....	178
9.4.1.3 Developing Applications using JPA.....	179
9.5 Developing Web Service Applications.....	179
9.5.1 Overview.....	179
9.5.1.1 What is a Web Service?.....	179
9.5.1.2 Changes from Java EE 5.....	179
9.5.1.3 Developing Web Service Applications.....	180
9.6 Common Subject Matter Regarding Java EE 6 Applications.....	181
9.6.1 Preparing to Create Applications.....	181
9.6.1.1 Preparing to Operate Servers.....	181
9.6.1.2 Creating a New Project.....	183
9.6.1.3 Creating Enterprise Applications.....	184
9.6.1.4 Setting a Classpath.....	184

9.6.1.5 Development using a Java EE 6 Module that is not a Development Resource.....	185
9.6.2 Creating the Java Classes and Interfaces.....	185
9.6.3 Creating XML Files.....	185
9.6.4 Detecting and Correcting Problems.....	185
9.6.5 Checking Application Behavior.....	185
9.6.5.1 Debugging.....	186
9.6.6 Distribute the Files to the Operating Environment.....	187
Chapter 10 Tips.....	189
10.1 Tool Use.....	189
10.1.1 Settings.....	189
10.1.2 Editors.....	190
10.1.3 Coding Support.....	192
10.1.4 Searching.....	194
10.1.5 Build and Debugging.....	194
10.1.6 Help.....	199
10.1.7 Other.....	200
10.2 Programming Techniques.....	201
10.2.1 Notes on Encoding in Web Applications.....	201
10.2.2 Using JNDI Lookup to Obtain Objects.....	202
10.3 Eclipse Plug-in Use.....	204
10.3.1 Installing and Uninstalling Plug-ins.....	204
10.3.2 Notes on Installing Eclipse Plug-ins.....	205
10.4 Thin Client Environment Edition.....	206
10.5 Developing in an IPv6 Environment.....	206
10.5.1 Specifying IPv6 Addresses.....	206
10.5.2 Notes on Application Development in an IPv6 Environment.....	206
Appendix A How to use Samples.....	208
Appendix B Migrating Resources from a Previous Version.....	210
B.1 Notes on Migrating Resources of Up to V10.....	210
B.1.1 Notes on Workspace Migration.....	211
B.1.2 Notes on Project Migration.....	212
B.2 Notes on Migrating Resources of Up to V9.....	213
B.3 Notes on Migrating Resources of Up to V8.....	214
B.3.1 Notes on Project Migration.....	214
B.4 Notes on Migrating Resources of Up to V7.....	214
B.4.1 Notes on Workspace Migration.....	214
B.4.2 Notes on Project Migration.....	215
B.5 Version-Independent Notes.....	216
B.5.1 Notes on Java Migration.....	216
B.5.2 Notes on J2EE Application Migration.....	217
B.5.3 Cautions when Migrating to JDK 6/7.....	223
B.5.4 Notes on Migrating ComponentDesigner Resources.....	223
B.5.5 Notes on Applet or Java Form Migration.....	225
B.6 Automatic Update of the Workspace and Projects.....	227
Appendix C Procedure for Development Using JDK 6.....	229
Appendix D Migrating J2EE 1.4 Applications.....	231
D.1 Developing Web Applications.....	231
D.1.1 Preparing the Environment for Creating Web Applications.....	231
D.1.2 Creating a Servlet.....	231
D.1.3 Creating an HTML File.....	231
D.1.4 Creating a JSP File.....	231
D.1.5 Graphically Editing the HTML/JSP Files.....	231
D.1.6 Creating the JavaScript.....	231

D.1.7 Creating the CSS.....	231
D.1.8 Editing the web.xml.....	231
D.1.9 Verifying the Web Application Behavior.....	232
D.1.10 Distributing the Web Application to the Operating Environment.....	232
D.2 Developing Enterprise JavaBeans.....	232
D.2.1 Preparing the Environment to Create EJBs.....	232
D.2.2 Creating an Enterprise Bean.....	232
D.2.3 Developing a CMP Entity Bean.....	236
D.2.4 Creating an EJB Test Client.....	239
D.2.5 Creating EJB Clients.....	241
D.2.6 Checking EJB Operation.....	242
D.2.7 Distributing EJBs to the Operating Environment.....	242
D.2.8 Exposing a Stateless Session Bean as a Web Service.....	242
D.3 Developing Web Service Applications.....	244
D.3.1 Preparing the Environment for Creating Web Service Applications.....	244
D.3.2 Creating the Web Service.....	245
D.3.3 Exposing a Stateless Session Bean as a Web Service.....	248
D.3.4 Creating a Service Endpoint Interface from WSDL.....	248
D.3.5 Creating a Web Service Client.....	248
D.3.6 Checking Web Service Operation.....	251
D.3.7 Distributing Web Services to the Operating Environment.....	251
D.4 J2EE Application Common Items.....	251
D.4.1 Preparing the Deployment Destination that is used for Application Operation Verification.....	251
D.4.2 Preparing the Environment to Create an Application.....	252
D.4.3 Creating the Java Class and the Interface.....	252
D.4.4 Creating XML Files.....	252
D.4.5 Detecting and Correcting Problems.....	252
D.4.6 Checking Application Behavior.....	252
D.5 J2EE1.4 Application Notes.....	255
D.5.1 EJB Notes.....	255
D.5.2 Web Service Application Notes.....	256
D.5.3 Notes Common to All J2EE Applications.....	257
D.6 Compatibility Information.....	257
D.6.1 Migration from J2EE Container to Java EE Container.....	257
Appendix E Troubleshooting.....	261
E.1 General Problems Related to Workbench.....	261
E.2 Problems Related to Java.....	262
E.3 Problems Related to Databases.....	262
E.4 Problems Related to JavaScript.....	263
E.5 Problems Related to Editor.....	264
E.6 Problems Related to IJServer Clusters and Interstage Java EE DAS Services.....	264
E.7 Problems Related to Server Operation Verification.....	266
Appendix F Tutorial.....	267
F.1 Java Application.....	267
F.2 Client Application.....	267
F.2.1 Lesson1 Calendar.....	267
F.2.2 Lesson2 Input and Output Fields.....	284
F.2.3 Lesson3 Buttons.....	295
F.2.4 Lesson4 Dialog.....	302
F.3 Applet.....	324
F.3.1 Developing Applet.....	324
F.3.2 Developing Applet That Use Multiple Screens.....	337
F.4 JavaBeans.....	364
F.4.1 Developing JavaBeans.....	364
Index.....	404

Chapter 1 Workbench Overview

This chapter explains the basic concepts that a user who has no previous experience of Eclipse will need to know before using the workbench, and describes basic workbench operations.

1.1 Basic Workbench Concepts

This section describes the elements that configure workbench, the structure of development resources, and the basic tasks performed from the workbench.

1.1.1 Workbench and its Elements

Workbench

The workbench is the place where users create, develop, build, execute, debug, and manage development resources.

Editor

Workbench has one editor area. When a file is opened from the Project Explorer view, the editor is displayed in the editor area. Multiple editors are displayed in the editor area in layers by default. However, the editor area can be split horizontally or vertically to display two or more editors simultaneously.

A vertical ruler is displayed as the left edge of the editor area. It displays markers associated with the range displayed in the editor area. For example, error and warning markers, markers that indicate breakpoints or places that match search items and other markers can be displayed. An outline ruler is displayed at the right edge of the editor area. It shows where certain markers are associated in the file as a whole, not just the part of the file visible in the editor area. The user can click on a marker in the outline ruler to display in the editor area the part of the file near that marker.



Point

From the workbench, users can also start any desired application to use as an external editor. However, external editors are not displayed in the editor area, and the support functions described above are not available for them.

Views

Views provide support for development tasks. Some examples of views are:

- Views for displaying and managing development resources (for example, the Project Explorer view and the Navigator view)
- Views for supporting editing tasks (for example, the Outline view and the Snippet view)
- Views for supporting debug tasks (for example, the Breakpoint view and the Variable view)

Each view has its own toolbar and menu. Operations performed from a view toolbar or menu only affect items in that particular view.



Note

Only one display of the same type of view can be displayed in one workbench.

Perspectives

A perspective defines the initial set of editors and views displayed by the workbench, their layout, and the items displayed in their toolbars and menus. Perspectives are prepared in advance for specific tasks. For example, the Java EE perspective contains the views to be used when developing Java EE applications, and the Debug perspective contains the views to be used when debugging an application. Users can select the perspective that best suits the task content.

1.1.2 Development Resources

Workspace

The workspace is the place where development resources and the status of development work are saved. The workspace manages development resources as projects. Multiple projects can be created in a workspace. In addition, the workspace is used to save the status of work performed by users, such as workbench settings information and breakpoints that have been set in source files.

The workbench creates a workspace in the following default location:

Workbench

```
<User's document folder>\Interstage Studio\<Product version>\workspace
```

Java EE 6 workbench

```
<User's document folder>\Interstage Studio\<Product version>\workspace_jee6
```

Multiple workspaces can be created, if required. A user can select which workspace to use when launching the workbench. The workbench can open only one workspace at a time.



If there is an IVS character in the workspace folder name, an error may occur during the editing or building of the source file.

Therefore, do not include IVS characters in the workspace folder names.



After the workbench is launched, users can switch to another workspace by selecting [File] > [Switch Workspace] from the menu.

Projects

A project is the management unit for the development resources used to develop applications. The building and debugging of applications is performed by project. The classpath and other settings required when building and the various settings required when developing applications are also performed by project.

Project wizards are used to create projects. Interstage Studio provides project wizards for Web applications, EJB, and other project types. When these wizards are used, the settings required for application development are set during project generation.



Resources under projects are stored as ordinary files and folders. If the files and folders are modified in places other than the workbench, or the file is modified using System Editor, it is necessary to reflect the changes in the workbench.

For example, if a file and folder are modified in places other than the workbench, an error will occur after clicking [File] > [Export] on the menu. In this case, click [File] > [Refresh] on the menu to reflect the changes, and then export the file and folder. However, if the project folder is copied to the workspace folder, changes to the file cannot be reflected in the workbench using this method. Click [File] > [Import] on the menu to reflect the changes to the project in the workbench.

1.1.3 Basic Workbench Tasks

Creating Development Resources

The creation of development resources from the workbench typically involves using a wizard to generate files, then using an editor to edit those files. Interstage Studio provides wizards for generating various types of files and the editors for editing these files.

Build

Build is a generic term referring to the creation of executable files from source files. Workbench invokes a builder to execute build tasks. What kinds of builders are invoked is determined by the type of project being built.

For example, the task of compiling Java source files to create a class file is one build task, and the function is provided by the Java builder.

There are two types of build, as follows:

- Incremental build: A method that builds only the source that has changed
- Clean build: A method that deletes all existing build results, then rebuilds all the development resources

Usually, an incremental build is performed. To perform a clean build, the user must explicitly clean the project beforehand.

Build is performed on the following two occasions:

- Automatic build: When a source file is changed and saved, a build is performed automatically
- Manual build: When a user executes a build manually by using the menu

Automatic builds are performed as the default. Therefore, users do not need to be aware of build tasks.

If a compile error, warning, or other information (collectively referred to as "problems") is output during a build, these are displayed in the Problems view. When a user double-clicks on a problem displayed in the Problems view, the source line for which the problem was issued is displayed in an editor. Compile errors, warnings, and any other problems are also indicated in an editor. As an example of this, a marker indicating an error may be displayed in the vertical ruler at the left edge of the editor area, or a squiggle may be displayed at the place of the problem.



.....

If an automatic build is performed every time source files are saved during a large-scale development, this may hinder user operations. If so, the timing of the build function can be controlled by setting manual build in order to minimize the hindrance to user operations.

.....

Checking Application Operation

Applications that have been created can be executed and debugged from the workbench.

Use a launch configuration to execute and debug an application. A launch configuration registers the various settings required for execution of the application. For example, the application server and the Java VM launch options can be set. A launch configuration is convenient because, once it is created, it can be used for repeated execution and debugging of the application.

1.2 Basic Workbench Operations

This section describes the basic workbench operations.

1.2.1 Operations Related to Development Resources

Creating a New Project

New projects are created using the New Project wizard. Use one of the following procedures to launch the New Project wizard:

- From the workbench menu, select [File] > [New] > [Project].
- Click the down arrow on the right of the [New] button in the workbench toolbar, then select [Project] from the sub-menu.
- Select [New] > [Project] from the context menu of the Project Explorer view or the Navigator view.

After the New Project wizard has been launched, select the project that you want to create and launch the project wizard.

Creating a New Source File

New source files are created using the New wizard. Use one of the following procedures to launch the New wizard:

- From the workbench menu, select [File] > [New] > [Other].

- Click the [New] button in the workbench toolbar.
- Select [New] > [Other] from the context menu of the Project Explorer view or the Navigator view.

After the New wizard has been launched, select the item that you want to create and launch the source wizard.

Point

Frequently-used source wizards are listed in the [New] sub-menu of the context menu. Source wizards can also be launched directly from this sub-menu.

Changing Project Settings (Properties)

Select the project from the Project Explorer view or the Navigator view, then select [Properties] from the context menu. The Properties dialog box is displayed. Select the relevant properties from the left tree, and change them to the required settings.

1.2.2 Operations Related to the Workbench

Editing Files using an Editor

Double-click a file in the Project Explorer view or the Navigator view, or select [Open] from the context menu of the file, to open the file in an editor so it can be edited. When the file is edited, an asterisk (*) is displayed in the tab part of the editor to indicate that the file has not yet been saved.

Selecting an Editor and Opening a File

Select the file from the Project Explorer view or the Navigator view, and then select [Open with] from the context menu. The editors that can open the file are listed in the sub-menu. Select the editor you want to use.

Displaying a View

Select [Window] > [Show View] from the workbench menu to display a view.

Frequently-used views are listed in the [Show View] sub-menu. If the view you want to display is not in the list, select [Other]. The [Show View] dialog box is displayed, and the view you want to display can be selected from a complete list of views.

Changing the Workbench Settings

Select [Window] > [Preferences] from the workbench menu. The Preferences dialog is displayed. Select the page from the left tree, and change the settings to the required ones.

Opening a Perspective

Select [Window] > [Open Perspective] from the workbench menu, or click the [Open Perspective] button in the workbench toolbar to open a perspective.

Frequently-used perspectives are listed in the [Open Perspective] sub-menu. If the perspective you want to open is not in the list, select [Other]. The [Open Perspective] dialog box is displayed, and the perspective you want to open can be selected from a complete list of perspectives.

1.2.3 Operations Related to the Main Tasks

Switching Between Automatic and Manual Build

Use one of the following procedures to switch between automatic and manual build:

- Select [Project] > [Build Automatically] from the workbench menu. This menu item is a toggle item and switches between automatic and manual build each time it is clicked.
- The [Build automatically] checkbox, reached by selecting [General] > [Workspace] from the Preferences dialog box, can be used to switch between automatic and manual build.

Manually Building a Project

If the build setting is manual, use one of the following procedures to perform the build manually:

- Select the project from the Project Explorer view or the Navigator view, then select [Project] > [Build Project] from the workbench menu.
If [Build All] is selected instead of [Build Project], all projects in the workspace are built.
- For standard workbenches, select projects in the Project Explorer or Navigator views, then select [Build Project] from the context menu.

Cleaning a Project

To delete build results, select [Project] > [Clean] from the workbench menu. A dialog box will be displayed. Select the project that is to be cleaned.

Chapter 2 Developing Web Applications

This chapter presents an overview of Web applications, and describes how to create Web applications that run in a Java EE application execution environment.

2.1 Overview

This section presents an overview of Web applications and the support functions for Web application development.

2.1.1 What are Web Applications?

Web applications are applications that are used from a Web browser, which is operating as the client. This is based on an arrangement whereby dynamic contents usage technology from a servlet container is added to the HTTP Web communication functions from a Web server.

Web applications can be used to implement client layer user interfaces, presentation layers to business logic that's implemented for example by an EJB, and similar.

Web applications operate as follows:

1. The Web application URL is accessed.
2. The Web page is displayed on the Web browser.
3. The user enters data in an HTML form, and then sends the data (as an HTTP request).
4. The server processes the data, then returns the response page (as an HTTP response) to the client
5. The process returns to Step 2, and the processing can be repeated.

Brief explanations of the technology used in the creation of Web applications are given below.

HTTP

HTTP (HyperText Transfer Protocol) is the protocol used to send and receive requests between a Web server and a client (a browser or similar). When the client sends a request to the Web server, the Web server returns a response in accordance with the request.

Frequently-used requests include the GET method, which is a request to get a particular page, and the POST method, which is used to send data entered in a form to the Web server.

HTML

HTML is a convention devised to describe the logical structure of a document by means of attaching tags to the contents. HTML is used to publish information on the internet, and has become the de facto coding language for Web browsers. HTML conventions have been extended in line with advancements in Web browsers.

Cascading Style Sheet (CSS)

A CSS is a file that holds style information for displaying an HTML page. This file can be used to specify the layout of a page on a browser and do design details. Extensions to HTML conventions have enabled appearance information to be coded within HTML, but the original purpose of HTML was the coding of logical structures, and many HTML modifications are required to unify the appearance within a site. In order to lessen the impact of this problem, CSS files that keep appearance information separate from HTML were devised as style sheets.

JavaScript

JavaScript is a script language created in order to attach operations that display calculated results and other operations to HTML documents. These scripts currently run on almost all browsers. Care is required with implementation since different browsers implement APIs and so on in different ways.

Servlet

Servlets are a type of technology for dynamically generating HTML documents on a Web server, and thus enable data entered from a Web browser to be processed on a server. Servlets provide technology that solves the redundancy, session management, and other problems of the conventional Web server technology, such as the widespread CGI (Common Gateway Interface).

A servlet interface must be implemented to create a servlet class. The getting of parameters from a request object and the creation of a response object are implemented as methods. The creation of a response object conventionally outputs an HTML document. However, now that JSP has been devised, JSP is usually used to create dynamic HTML documents, and servlets are used for tasks such as allocating processing to EJB services and JSP.

A servlet is initialized at the first request, and the same instance is used for subsequent requests. Therefore, servlets must be created to be thread-safe.

JSP (JavaServer Pages)

Servlet technology is used to make dynamic HTML contents, but servlet programs contain a mix of logic processed by servers and HTML content, which makes coding difficult. JSP makes it possible to enter Java in HTML files in order to output dynamic contents (for example, data embedding).

During operation, JSP is converted to the servlet class then executed by the server. Thus, the basic technology for executing JSP is the same as for a servlet.

JSTL

JSTL (JSP Standard Tag Library) is a JSP tag library standardized by the Java Community Process (JCP) as JSR-052. JSTL provides frequently-used common functions grouped together as a tag library.

JSF (JavaServer Faces)

JSF (JavaServer Faces) is a specification formulated by the Java Community Process (JCP) as JSR-127. It is an application framework used for creating user interfaces for Web applications.

2.1.1.1 Modifications since J2EE 1.4

The specification for Web applications included in Java EE is Servlet 2.5. The JSP specification is JSP 2.1.

The following are the main changes in Servlet 2.5:

- Annotation can now be used.
- The web.xml description format has been changed.

The following are the main functional additions in JSP 2.1:

- Unified EL can now be used as an expression language.

Annotation

Annotation can now be used in Servlet 2.5, thus enabling Dependency Injection. As a result, JNDI lookup processing need not be coded in source, and referenced resource declarations, and so on, need not be coded in web.xml.

The main annotations are as follows:

- `@Resource`
Specify to perform Dependency Injection on the servlet container. The action is the same as the web.xml `<resource-ref>` element, `<env-ref>` element, and the `<resource-env-ref>` element.
- `@EJB`
Declares an EJB reference. The action is the same as the web.xml `<ejb-ref>` element and the `<ejb-local-ref>` element.
- `@DeclareRoles`
Specifies a security role. The action is the same as the web.xml `<security-role>` element.

- @RunAs
Specifies the role at execution. The action is the same as the web.xml <run-as> element.

web.xml Modifications

The web.xml description method has been improved. The description method has been changed so that multiple patterns and elements can be described as a group. As a result, the volume of descriptions has decreased and readability has improved.

- Wild cards can be used for filter-mapping.
An asterisk (*) can now be used as a wild card in the <servlet-name> element of <filter-mapping>. Previously, it was necessary to list all the servlets.
- Multiple specifications of url-pattern are possible.
Multiple pattern mappings can now be specified in <servlet-mapping> and <filter-mapping>.

Unified EL is usable

The Expression Language introduced by JSP2.0 has been unified with the JSF Expression Language. Under Unified EL, coding in both the \${} format and the #{} format can be used.

2.1.2 Web Application Development

An overview of Web application development is given below.

Preparing for Web Application Development

Before developing a Web application, a Web application project must be created. Create a dynamic Web project and set the required target runtime, classpath, and other project settings.

For details, refer to "[2.3.1 Preparing the Environment for Creating the Web Application](#)".

Creating Servlets/HTML/JSP/JavaScript/CSS

Use the various source generation wizards and editors to create the resources needed to display the Web page.

For details, refer to "[2.3.2 Creating a Servlet](#)", "[2.3.4 Creating a JSP File](#)", "[2.3.3 Creating an HTML File](#)", "[2.3.6 Creating the JavaScript](#)", and "[2.3.7 Creating the CSS](#)".

Building and Validating Web Applications

Build is performed automatically in the initial state when resources are saved. Build checks for compile errors and uses various validators to verify that resources have no problems.

For details, refer to "[Build](#)" and "[6.2.5.2 Validation](#)".

Checking Web Application Operation

Use the Servers view to verify the behavior of the created Web application.

For details, refer to "[2.3.9 Verifying the Web Application Behavior](#)".

Distributing Web Applications

The Web application must be archived in order for it to be distributed. Use the Export wizard to create an archive file.

For details, refer to "[2.3.11 Distributing the Web Application to the Operating Environment](#)".

2.2 Introduction

This section introduces the procedures for creating a Web application.

2.2.1 Application to be Created

Create a Web application that returns the country name and total population from a ranking list of the populations of countries when the ranking is entered, and displays the result on a Web page. The application handles information for only the countries ranked 1 to 10, and

returns an error if the entered numeric is 0 or less or is 11 or greater. An error also occurs if a character string is entered for the ranking input item.

2.2.2 Development Flow

Development of the above Web application proceeds as follows:

1. Create a project for the Web application.

In order to create the Web application, first create a dynamic Web application project. When the Web application project is created as directed by the wizard, the classpath and other settings required for the build are set automatically.

2. Create Java classes.

Create the two classes required for the application.

- Create the data class.
- Create the logic class.

3. Create a servlet class

Use the following procedure to create a servlet class:

- Create the pattern for the servlet class.
Use the wizard to create the servlet class pattern.
- Implement the servlet class.
Use the Java editor to implement the servlet class.

4. Create an I/O page.

Use the wizard to create the pattern for the application I/O page, then use an editor to edit the page.

- Create the input page pattern.
- Edit the input page.
- Create the output page pattern.
- Edit the output page.
- Create the error page pattern.
- Edit the error page.

5. Verify the application behavior.

Use the following procedure to verify the behavior of the application:

- Associate the project and the server.
Set the server on which the application is deployed.
- Set the breakpoints.
Set the breakpoints for the debugger to verify the program behavior during execution.
- Launch the server.
Launch the server so that the application can receive requests from Web browsers. Deployment is performed automatically before the server starts.

- Execute the application.
Launch the Web browser and access the application URL to start verifying the behavior of the application.
- Debug the application.
Debug the program and verify that the application is behaving correctly.

6. Distribute the application to the operating environment.

Use the following procedures to distribute the application to the operating environment:

- Export the application.
Create a WAR file to distribute the application to the operating environment.
- Distribute the application to the operating environment.
Deploy the WAR file from the Interstage Java EE Admin Console.

2.2.3 Development Procedures

The actual procedures used to develop the application are described below.

- 1) [Creating the Project for the Web Application](#)
- 2) [Creating the Java Classes](#)
- 3) [Creating the Servlet Class](#)
- 4) [Creating the I/O Page](#)
- 5) [Verifying the Application Behavior](#)
- 6) [Distributing the Application to the Operating Environment](#)

1) Creating the Project for the Web Application

Select [File] > [New] > [Project] from the menu bar to display the [New Project] wizard.

Select [Web] > [Dynamic Web Project] from the [New Project] wizard, then click [Next].

Check and enter the following setup items:

Setup Items	Setup Content
Project name	WebSample
Target Runtime	Interstage Application Server V11.1 IJServer Cluster (Java EE)
Dynamic Web Module version	2.5
Configuration	Default Configuration for Interstage Application Server V11.1 IJServer Cluster (Java EE)
EAR Membership	Do not select

Set the information, then click [Next].The Web Module page is displayed.

Check and enter the following setup items. After the following information is set, click [Finish].

Parameter	Setting
Context Root	WebSample
Content Directory	WebContent
Java Source Directory	src
Generate deployment descriptor	Select

2) Creating the Java Classes

2-1) Creating the Data Class

Create a data class that stores the country name and total population information, and gets information. To create the data class, select the created project, then right-click to display the context menu. Select [New] > [Class] from the context menu. The [New Java Class] wizard is displayed.

Check and enter the following setup items. After the following information is set, click [Finish].

Parameter	Setting
Source folder	WebSample/src
Package	sample
Name	CountryData

The following source file is generated:

Source file	Description
CountryData.java	Data class

Implement the processing for storing the country names and total populations. Add the places shown in **red** below to the source.

Data Class Implementation (CountryData.java)

```
package sample;

public class CountryData {

    private String countryName;
    private int totalPopulation;

    public CountryData(String name, int total) {
        setCountryName(name);
        setTotalPopulation(total);
    }

    public String getCountryName() {
        return countryName;
    }

    public void setCountryName(String countryName) {
        this.countryName = countryName;
    }

    public int getTotalPopulation() {
        return totalPopulation;
    }

    public void setTotalPopulation(int totalPopulation) {
        this.totalPopulation = totalPopulation;
    }
}
```



After the fields are added, and while the class is in the selected state, [Source] > [Generate Getters and Setters] can be selected from the menu to add getter/setter.

2-2) Creating the Logic Class

To create the logic class, select the created project, then right-click to display the context menu. Select [New] > [Class] from the context menu. The [New Java Class] wizard is displayed.

Check and enter the following setup items. After the following information is set, click [Finish].

Parameter	Setting
Source folder	WebSample/src
Package	sample
Name	PopulationRanking

The following source file is generated.

Source file	Description
PopulationRanking.java	Logic class

Implement the processing for entering the ranking and returning the country name and total population. Add the places shown in **red** below to the source.

Logic Class Implementation (PopulationRanking.java)

```
package sample;

public class PopulationRanking {

    private CountryData[] countries;

    public PopulationRanking(){

        countries = new CountryData[]{
            new CountryData("China",1330000000),
            new CountryData("India",1140000000),
            new CountryData("U.S.A.",300000000),
            new CountryData("Indonesia",230000000),
            new CountryData("Brazil",190000000),
            new CountryData("Pakistan",160000000),
            new CountryData("Bangladesh",150000000),
            new CountryData("Russia",140000000),
            new CountryData("Nigeria",140000000),
            new CountryData("Japan",130000000)
        };

    }

    public CountryData getCountryData(int rank) {
        --rank;
        if (rank < 0 || rank >= countries.length) {
            return null;
        }

        return countries[rank];
    }
}
```

```

    }
}

```

3) Creating the Servlet Class

3-1) Creating the Pattern for the Servlet Class

To create the servlet class pattern, select the created project, then right-click to display the context menu. Select [New] > [Servlet] from the context menu. The [Create Servlet] wizard is displayed.

Check and enter the following setup items. After the following information is set, click [Next].

Parameter	Setting
Web project	WebSample
Source folder	\WebSample\src
Java package	sample
Class name	ServletController
Superclass	javax.servlet.http.HttpServlet

Check the following setup items then, without changing anything, click [Next].

Parameter	Setting
Name	ServletController
URL Mappings	/ServletController

Check and enter the following setup items. After the following information is set, click [Finish].

Parameter	Setting
Modifiers	Public
Which method stubs would you like to create?	Constructors from superclass Inherited abstract methods doPost (remove the doGet check)

The following source file is generated.

Source file	Description
ServletController.java	Servlet class

Point

With the wizard, web.xml servlet mapping definitions are also added at the same time. As a result, editing of web.xml is not required in this application. If editing of web.xml is required for a reason other than servlet mapping definitions, refer to "[2.3.8 Editing the web.xml](#)".

3-2) Implementing the Servlet Class

Add the places shown in **red** in the created servlet class to the source.

Servlet Class (ServletController.java)

```

package sample;

```

```

import java.io.IOException;

import javax.servlet.RequestDispatcher;

import javax.servlet.ServletContext;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

/**
 * Servlet implementation class ServletController
 *
 */
public class ServletController extends HttpServlet {
    private static final long serialVersionUID = 1L;

    /**
     * @see HttpServlet#HttpServlet()
     */
    public ServletController() {
        super();
    }

    /**
     * @see javax.servlet.http.HttpServlet#doPost(HttpServletRequest request, HttpServletResponse
    response)
     */
    protected void doPost(HttpServletRequest request, HttpServletResponse response)
    throws ServletException, IOException {
        // TODO Auto-generated method stub
        request.setCharacterEncoding("ISO-8859-1");
        ServletContext sc = getServletContext();

        // Get the type of file to be invoked.
        String mode = request.getParameter("mode");
        if (mode != null && mode.equals("top")) {
            // Invoke the Input page HTML file.
            RequestDispatcher inRd = getServletContext().getRequestDispatcher("/
default.jsp");
            inRd.forward(request, response);
            return;
        }

        int rank;

        // Input check
        try {
            rank = Integer.parseInt(request.getParameter("rank"));
        } catch (NumberFormatException e){
            RequestDispatcher errRd = sc.getRequestDispatcher("/error.jsp");
            errRd.forward(request, response);
            return;
        }

        PopulationRanking pop = new PopulationRanking();
        CountryData country = pop.getCountryData(rank);

        // Check the obtained value.
        if(country == null){
            RequestDispatcher errRd = sc.getRequestDispatcher("/error.jsp");

```



```

        errRd.forward(request, response);
        return;
    }

    sc.setAttribute("ranking", country);

    RequestDispatcher outRd = sc.getRequestDispatcher("/result.jsp");
    outRd.forward(request, response);

}
}

```

 **Point**

After the required implementation is coded, right-click on the Java editor, then select [Source] > [Organize Imports] from the context menu. The import statement that is appropriate for the class path that was set in the project can then be inserted.

4) Creating the I/O Page

4-1) Creating the Input Page Pattern

Create the pattern for the JSP file that will be the input page. To create the JSP, select the created project, then right-click to display the context menu. Select [New] > [JSP] from the context menu. The [New JavaServer Page] wizard is displayed.

Check and enter the following setup items. After the following information is set, click [Next].

Parameter	Setting
Enter or select the parent folder	WebSample/WebContent
File name	default.jsp

Enter the following setup items, then click [Finish] - JSP is displayed.

Parameter	Setting
Use JSP Template	Check
Name	New JSP file (HTML)

4-2) Editing the Input Page

Modify the character strings shown in **red** in the created JSP file. For details on editing a JSP file, refer to "2.3.4.2 Editing the JSP File".

Input Page (default.jsp)

```

<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/
html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
<title>World ranking for total population</title>
</head>
<body>
<h1>Input page</h1>
<p>Enter a ranking from 1 to 10.</p>
<p></p>
<form action="ServletController" method="post">
    Rank:<br>

```

```

Rank<br>
<p></p>


```

Point

JSP files are associated with the JSP editor by default, but a Web Page editor that enables editing while checking the design and layout can also be used. To select an editor, select the JSP file, then select [Open With] > [Web Page Editor] from the context menu. For details on editing using a Web Page editor, refer to "2.3.5 Graphically Editing the HTML/JSP Files".

4-3) Creating the Output Page Pattern

Use the wizard, in the same way as for the input page, to create the JSP file.

Enter the following values, then click [Finish]. The initial values can be used for the items not shown in the table.

Parameter	Setting
Enter or select the parent folder	WebSample/WebContent
File name	result.jsp

4-4) Editing the Output Page

Modify the character strings shown in **red** in the created JSP file.

Output Page (result.jsp)

```

<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/
html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
<title>World ranking for total population</title>
</head>
<body>
<h1>Output page</h1>

Total population ranking${param.rank} rank is "$
{applicationScope.ranking.countryName}"<br>
Total population is ${applicationScope.ranking.totalPopulation} people.
<br>
<hr>
<form action="ServletController" method="post">
    <input type="submit" value="Back to input page">
    <input type="hidden" name="mode" value="top">
</form>

</body>
</html>

```

4-5) Creating the Error Page Pattern

Use the wizard, in the same way as for the input page, to create the JSP file.

Enter the following values, then click [Finish]. The initial values can be used for the items not shown in the table.

Parameter	Setting
Enter or select the parent folder	WebSample/WebContent
File name	error.jsp

4-6) Editing the Error Page

Modify the character strings shown in **red** in the created JSP file.

Error Page (error.jsp)

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/
html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
<title>World ranking for total population</title>
</head>
<body>

<h1>Error page</h1>
The entered ranking is not in the specified range, or a server error occurred.<br>
<br>
<hr>
<form action="ServletController" method="post">
    <input type="submit" value="Back to input page">
    <input type="hidden" name="mode" value="top">
</form>

</body>
</html>
```

5) Verifying the Application Behavior

5-1) Associate the project with a server

Select a server from the Servers view, then select [Add and Remove Projects] from the context menu.

Select the project from the available projects, then click [Add]. Check that the dynamic Web project has been added to the configured projects.

Check the following setup items, then click [Finish].

Parameter	Setting
Configured projects	WebSample
If server is started, publish changes immediately	Select

Point

.....
If the deployment destination server in the Servers view is not registered, then it must be added. For details on how to do that, refer to "6.2.6 Checking Application Behavior".
.....

5-2) Setting Breakpoints

Set a breakpoint at the first line of the doPost() method of the servlet class. To set or delete a breakpoint, double-click the vertical ruler on the left-hand edge of the editor.

Point

Similarly, breakpoints can be set in JSP files using the JSP editor. However, breakpoints can be set only in lines that have JSP tags (breakpoints cannot be set in parts that have HTML tags or similar).

Note

If breakpoints are set in a JSP file, execution is also interrupted for JSP files with the same name in different folders.

5-3) Launching the Server

Select a server from the Servers view, then select [Restart in Debug] from the context menu.

Point

Even if [Restart in Debug] or [Connect(Debug Launch)/Login] is implemented prior to deployment, debugging is not possible.

Check the following information in the Servers view:

Item to Check	Contents to Check
Server state	Debugging
Server status	Synchronized
WAR file (WebSample) state	Synchronized

5-4) Executing the Application

In the Servers view, select the deploying project, then from the context menu, select [Web browser]. The Web browser is started, and the input window is opened.

Parameter	Setting
URL of the input page	http://localhost/WebSample/

Enter a ranking in the [Rank:] input field, then click [OK].

Point

The user can select the project, then select [Run As] > [Run on Server] or [Debug As] > [Debug on Server] from the context menu to add the server, add the project, launch the server, and then launch the client all together.

In addition, the Web browser used as the client can be changed by selecting [Window] > [Web Browser] from the menu.

5-5) Debugging the Application

The applications runs and is interrupted at the breakpoints. Check the value of variable by Variables view. Select [Run] > [Step Over] from the menu and check the program state in the Variables view. For debug details, refer to "[6.2.6.1 Debugging](#)".



In addition to checking values, values can be changed from the Variables view.

Processing that has been interrupted by the debugger can be restarted by selecting [Run] > [Resume] from the menu. This allows the processing on the server side to complete so that the output page is displayed on the Web browser. Check that the country name and total population are displayed for the entered ranking.

6) Distributing the Application to the Operating Environment

A WAR file must be created in order to distribute the application to the operating environment. Use the server distribution function to distribute the created WAR file to the server.

6-1) Exporting the Application

The WAR file is created with the Export wizard. To launch the Export wizard, select [File] > [Export]. From [Export] wizard, select [Web] > [WAR file].

The WAR Export wizard is displayed.

Check and enter the following setup items. After the following information is set, click [Finish].

Parameter	Setting
Web project	Specify the dynamic Web application project.
Destination	Specify the creation destination for the WAR file.

6-2) Distributing to the Operating Environment

The application can be deployed from a remote environment to the operating environment by logging in to the Interstage Java EE Admin Console at the operating environment and by specifying a local WAR file.

2.3 Tasks

This section describes how to develop a Web application using Interstage Studio. It explains each of the development work topics (tasks) separately.

- [2.3.1 Preparing the Environment for Creating the Web Application](#)
- [2.3.2 Creating a Servlet](#)
- [2.3.3 Creating an HTML File](#)
- [2.3.4 Creating a JSP File](#)
- [2.3.5 Graphically Editing the HTML/JSP Files](#)
- [2.3.6 Creating the JavaScript](#)
- [2.3.7 Creating the CSS](#)
- [2.3.8 Editing the web.xml](#)
- [2.3.9 Verifying the Web Application Behavior](#)
- [2.3.10 Verifying JavaScript Operation](#)
- [2.3.11 Distributing the Web Application to the Operating Environment](#)

2.3.1 Preparing the Environment for Creating the Web Application

In order to create a Web application, first create a project. Then prepare an environment that allows the required libraries to be set in the project and enables the build.

Creating the Project

From the [New Project] wizard, select [Web] > [Dynamic Web Project] to create the Web application project. For details on the parts common to project wizards, refer to "[6.2.2.1 Creating New Projects](#)".

Refer below for the settings that are unique to the Dynamic Web Project wizard.

- Context Root
Specify the highest level folder used when deploying to the Web server.

- Content Directory
Specify the folder that stores all the resources deployed as the Web application.

- Java Source Directory
Specify the folder that stores the project source.

- Generate deployment descriptor
web.xml is required in Web applications.

Class Path Settings

Class paths must be set if there are libraries or similar that are required in order to create the application. Class paths are set in the project build path.

For details on build path settings, refer to "[6.2.2.3 Setting Classpaths](#)".



Do not store class files directly in WEB-INF/classes because this is deleted during the build.



Note the following points when using JSF in a Web application:

- Specify either "JavaServer Faces v1.1 Project" or "JavaServer Faces v1.2 Project" in [Configuration] of the Dynamic Web Project wizard.
- Add libraries using the following procedure in the [JSF Capabilities] page of the Dynamic Web Project wizard:
 1. Click [New].
 2. Enter "JSF" for the library name, click [Add], select the following jar file, and then click [Finish].
`<installation folder>\APS\F3FMisjee\lib\jsf-impl.jar`

2.3.2 Creating a Servlet

To create a servlet, use the Servlet wizard to create the servlet class source file and implement the required methods in the file. The method is described below.

Creating a New Servlet

From the [New] wizard, select [Web] > [Servlet], and the wizard creates the servlet. Refer to the following for the wizard settings:

- Web project
Specify the project that stores the servlet class source.

- Source folder
Specify the folder that stores the servlet class source.

- Java package
Specify the servlet class package name.

- Class name
Specify the class name of the servlet class.

- Superclass
Specify the class to be inherited.

- Servlet Name, Description, Initialization Parameters, and URL Mappings
Specify the servlet definition information. The content specified here is inserted in the deployment descriptor (web.xml) of the Web application.

- Modifiers
Specify the modifiers of the servlet class.

- Interfaces
Specify the interface to be implemented.

- Which method stubs would you like to create?
Select the method stubs to be implemented.

When the wizard is executed, the servlet class Java source is generated and the required description is added to web.xml.

Editing the Servlet

After the source is generated, implement the doPost() and doGet() servlet methods. These methods usually implement processing in the following sequence:

1. Get the parameter values from the request object.
2. Set headers in the response object.
3. Get the output stream from the response object and output response to the response object.

The way to implement these methods is the same as for an ordinary Java class. Use the Java editor to implement the methods.

2.3.3 Creating an HTML File

To create an HTML file, use the HTML wizard to generate an HTML file, then use an HTML editor or similar to set the page layout. The method is described below.

2.3.3.1 Creating a New HTML File

From the [New] wizard, select [Web] > [HTML], and the wizard creates the HTML file. Refer to the following for the wizard settings:

- Enter or select the parent folder
Select the folder in which the HTML file is to be created.

- File name
Specify the name of the HTML file being created. If the extension is omitted, ".html" is used as the extension.

- Use HTML Template
Select this item if a template is to be used to create the HTML file. New templates can also be added.

Point

The extension that is used if the extension is omitted, and the embedded encoding used during file creation, can be customized using [Web] > [HTML Files] on the Preferences page.

2.3.3.2 Editing the HTML File

Use the HTML editor to edit the HTML file. The HTML editor is a text editor that has the following features:

- Highlighted display of syntax
Element names, attribute names, attribute values, comments, and so on, are each highlighted in different colors. The highlighting method can be customized using preferences.
- Problem identification
If there are errors in the entered tags, attribute names, attribute values, or similar, errors and warnings are indicated by line markers or wavy lines under the text.
- Contents Assist
In accordance with the cursor location, suggestions concerning selectable element names, attribute names, attribute values and so on are displayed in Contents Assist. Press the [Ctrl+Space] keys to display the contents assist list, and enter characters to narrow down the possible choices.
- Tool tips display
When the cursor is moved over an element name or attribute name and the [F2] key is pressed, a description of that element or attribute is displayed.
- User-definable templates and snippets
The templates that can be used with Contents Assist can be registered.
In addition, fragments of code or similar that are used frequently can be registered in snippets, thus enabling drag-and-drop to be used to insert code easily.
- Tag selection
Depending on the cursor position, an indicator showing the range covered by a tag is displayed in the vertical ruler on the left-hand side of the page.

Use the appropriate view from the following when editing using the HTML editor:

- Outline view
Shows the HTML syntax outline. Elements and attributes can also be added using this view.
- Properties view
Shows the attributes of an element selected using the editor. The value can also be changed. Attribute values can also be selected from a list.
- Snippets view
Fragments of code that are used frequently can be registered, thus enabling drag-and-drop to be used to easily insert code in the source.

Entering Tags

Use one of the following methods to enter tags:

- Contents Assist
When the [Ctrl+Space] keys are pressed together, tag suggestions that suit the current cursor position are displayed in the Contents

Assist list. The Contents Assist list displays the tags that can be inserted in that syntax, and also displays templates registered in the HTML template.

- Using the Outline view
Tags can be inserted from the context menu of the Outline view. A tag can be inserted before, after or as a child of the tag currently selected in the Outline view.

Inserting Tables

Use Contents Assist to insert a table. Tables are registered in advance in the templates as a "table".

Inserting Lists

Use Contents Assist to insert a list. In the templates, "ol" is used for a sequential list, and "ul" for a non-sequential list.



.....

It is convenient to register frequently-used tags in the templates or the Snippets view. Use templates if Contents Assist is used from the keyboard. Use the Snippets view if mouse operations are used to insert tags. User-defined variables can also be used from the Snippets view, and the variable value can be specified during insertion. For details on registering templates, refer to "[Templates](#)", under Tips. For details on registering snippets, refer to "[Snippets](#)", under Tips.

.....

Entering Attributes/Attribute Values

Use either of the following methods to add attributes and to change attribute values:

- Using Contents Assist
When the [Ctrl+Space] keys are pressed in the middle of a tag, a list of the attributes that can be used with that tag is displayed in the Contents Assist list. Select from the list the attribute that you want to add. For attribute values that have fixed options, move the caret symbol to the start of the attribute values and launch Content Assist. This displays suggested attribute values in the Contents Assist list.
- Using the Properties view
The attribute list for the tag at which the caret is positioned in the editor is displayed in the Properties view. When a value is entered in the Value column of the Properties view, that value is reflected in the attribute value of the tag.

Previewing the Page

To preview a created HTML file, open that file using a Web browser.



.....

For convenience, arrange the HTML editor used for editing and the Web browser used for previewing the page to be side by side. To display the editor beside the browser, drag the editor tab.

.....

2.3.3.3 Validating the HTML Tags

The HTML syntax validator is used to verify HTML tags. For details on validation, refer to "[6.2.5.2 Validation](#)".

2.3.4 Creating a JSP File

To create a JSP file, use the JSP wizard to generate the JSP file. Then use the JSP editor or a similar editor to set the page layout, operation, and so on. The method is described below.

2.3.4.1 Creating a New JSP File

From the [New] wizard, select [Web] > [JSP], and the wizard generates the JSP file. Refer to the following for the wizard settings:

- Enter or select the parent folder
Select the folder in which the JSP file is to be created.
- File name
Specify the name of the JSP file to be created. If the extension is omitted, ".jsp" is used as the extension.
- Use JSP Template
Select the checkbox if you want to use a template to create the JSP file. New templates can also be added.

Point

- For files that are meant to be fetched by another JSP file using an include directive (`<%@ include file="..."%>`) or similar, rather than being invoked directly as JSP, do not use ".jsp" as the extension. Use an extension such as ".jspx" for such files.
- If the extension is omitted, the extension that is used, and the embedded encoding used during file creation, can be customized using [Web] > [JSP Files] on the preferences page.

2.3.4.2 Editing the JSP File

Use the JSP editor to edit a JSP file. The JSP editor is a text editor that has the following features:

- Highlighted display of syntax
Element names, attribute names, attribute values, comments, and so on, are each highlighted in different colors. The highlighting method can be customized using preferences.
- Problem identification
If there are errors in the entered tags, attribute names, attribute values, or similar, errors and warnings are indicated by lines markers or wavy lines in the text.
- Contents Assist
Depending on the cursor position, suggestions concerning selectable element names, attribute names, attribute values and so on are displayed in Contents Assist. Press the [Ctrl+Space] keys together to display the Contents Assist list, and enter characters to reduce the number of possible choices.
- Tool tips display
When the cursor is moved over an element name or attribute name and the [F2] key is pressed, a description of that element or attribute is displayed.
- User-definable templates and snippets
The templates that can be used with Contents Assist can be registered.
In addition, fragments of code or similar that are used frequently can be registered in snippets, thus enabling drag-and-drop to be used to easily insert code.
- Tag selection
Depending on the cursor position, an indicator showing the range covered by a tag is displayed in the vertical ruler on the left-hand side of the page.

Use the appropriate view from the following when editing using the JSP editor:

- Outline view
Shows the JSP syntax outline. Elements and attributes can also be added using this view.
- Properties view
Shows the attributes of an element selected using the editor. The value can also be changed. Attribute values can also be selected from

a list.

- Snippets view

Fragments of code that are used frequently can be registered, thus enabling drag-and-drop to be used to easily insert code in the source.

JSP editor operation is the same as for the HTML editor. Refer to "[2.3.3.2 Editing the HTML File](#)".

Using JSP Extended Tags

If you are setting JSP extended tags that will be needed during Web application operation, JSP extended tags are displayed in the Contents Assist list. For this, use the following procedure:

1. Specify the JSP tag library location.

Use one of the following methods to specify the tag library:

- Code the taglib tag in web.xml, and specify the .tld file location.
- Place the .tld file in /WEB-INF.
- Place the JAR file in WEB-INF/lib. The JAR file contains the tld file that is stored in the /META-INF.

2. Code a taglib directive in the JSP file.

To display JSP extended tags in the Contents Assist list, code a taglib directive in the JSP file.

To code a taglib directive, select the template "JSP taglib directive" from Contents Assist. Contents Assist can be used even when entering uri attribute values. In addition, if the taglib uri is specified in the uri attribute, Contents Assist can be used to input the prefix attribute value, and the default prefix specified in taglib is displayed in the Contents Assist list.



- The tag configuration may collapse and an error may occur if a JSP file meeting the condition below is opened using the JSP Editor and [Source] > [Format] is executed from the menu. Use a Web Page Editor to execute [Format] with a JSP file that uses JavaScript.
 - An attribute for JavaScript is described within a tag, and JavaScript is used (Example: onclick attributes, and so on).
- Attributes added using JSP2.1 (such as trimDirectiveWhitespaces) and specified in the JSP file will not be recognized (a warning will be displayed in the Problems view), but will not cause any errors.

2.3.4.3 Validating the JSP

The validators available for verifying JSP are the JSP syntax validator and the JSP content validator. For details on validation, refer to "[6.2.5.2 Validation](#)".



The validators do not check for correct correspondence between JSP start tags and end tags. Thus, errors and warnings are not output in the Problems view. However, lack of correspondence between start tags and end tags may result in errors being detected during compilation of Java code converted from the JSP file.

2.3.5 Graphically Editing the HTML/JSP Files

Use a Web Page editor to graphically edit an HTML file or JSP file while checking the design and layout.



To edit a file using a Web Page editor, select the file, then select [Open With] > [Web Page Editor] from the context menu.

The Web Page editor has the following features:

Graphical editing and text editing

The [Design] tab has a design page and a source page.

On the design page, editing tasks for the Web page output image can be performed directly on the image.

On the source page, source contents are displayed in different colors in accordance with the syntax, and Contents Assist or similar can be used to perform editing tasks.

The edited contents of the design page and source page are immediately reflected in another page.

Preview

The [Preview] tab can be used to check the Web page image displayed.

Use the appropriate view from the following when you are editing using the Web Page editor:

Palette view

The standard HTML and JSP tags and also the JSP extended tag items set in class paths are displayed. From the Palette view, drag-and-drop can be used to insert items in the design page.

Outline view

Shows the HTML/JSP syntax outline. Elements and attributes can also be added using this view.

Properties view

Shows the attributes of an element selected using the editor. The value can also be changed. Attribute values can also be selected from a list.

Source page operations are the same as for the HTML editor/JSP editor. Refer to "[2.3.3.2 Editing the HTML File](#)" and "[2.3.4.2 Editing the JSP File](#)".

Changing the Page Layout of Design Tab

The [Design] tab has a design page and a source page, but users can switch between the following four display patterns from the toolbar:

- The page split into top and bottom parts
- The page is split into left and right parts
- Display only the design page
- Display only the source page

In addition, part of the design page is used for the palette, but the Design view can be enlarged for easier use by displaying the Palette view.

Changing the Style

On the design page, the style can be changed by selecting sub-menu from the [Style] menu of the context menu.

2.3.6 Creating the JavaScript

To create a JavaScript file, use the JavaScript wizard to generate the JavaScript file, then use the JavaScript editor or similar to edit the file. The method is described below.

Creating a New JavaScript File

From the [New] wizard, select [JavaScript] > [JavaScript Source File], and the wizard creates the JavaScript file. Refer to the following for the wizard settings:

- Enter or select the parent folder
Select the folder where the JavaScript file is to be created.
- File name
Specify the name of the JavaScript file to be created. If the extension is omitted, ".js" is used as the extension.

Editing the JavaScript File

Use the JavaScript editor to edit a JavaScript file. The JavaScript editor has the following features:

- Highlighted display of syntax
Keywords, comments, character string literals, and so on, are each highlighted in different colors. The highlighting method can be customized by changing the preferences.
- Contents Assist
Depending on the cursor position, suggestions concerning objects, methods, arguments, properties and so on are displayed in the Contents Assist list. Press the [Ctrl+Space] keys together to display the Contents Assist list, and enter characters to reduce the list of possible choices.

Validating the JavaScript

Verification of the description contents of JavaScript files is performed by a JavaScript validator. JavaScript files must be stored in the folder specified as the JavaScript source folder.

To specify the JavaScript source folder, follow the procedure below:

1. In the Project Explorer, select the target project.
2. From the context menu, select [Properties].
3. The [Properties] dialog box is displayed. In the left trees view, select [JavaScript] > [JavaScript Libraries].
4. In the [JavaScript Libraries] page, select the [Source] tab, then add the JavaScript source folder.



Point

In the case of a JavaScript project, the folder is set as the JavaScript source folder by default.



Note

Semantic analysis of JavaScript is not performed. As a result, the settings on the [Preferences] page or the [JavaScript] > [Validator] > [Errors/Warnings] page of the project [Properties] page may not be enabled.

2.3.7 Creating the CSS

To create a CSS file, use the CSS wizard to generate the CSS file, then use the CSS editor or similar to edit the file. The method is described below.

Creating a new CSS File

From the [New] wizard, select [Web] > [CSS], and the wizard creates the CSS file. Refer to the following for the wizard settings:

- Enter or select the parent folder
Select the folder where the CSS file is to be created.
- File name
Specify the name of the CSS file to be created. If the extension is omitted, ".css" is used as the extension.
- Use CSS Template
Select the checkbox if you want to use a template to create the CSS file. New templates can also be added.

Editing the CSS File

Use the CSS editor to edit a CSS file. The CSS editor has the following features:

- Highlighted display of syntax
Selectors, comments, property names, property values, and so on, are each highlighted in different colors. The highlighting method can be customized using settings.
- Contents Assist
In accordance with the cursor location, suggestions concerning selectors, property names, and property values are displayed in the Contents Assist list. Press the [Ctrl+Space] keys to display the contents assist list, and enter characters to narrow down the possible choices.

Point

In order to check the CSS file design, use a Web browser to view the HTML file that applies the CSS file. To apply a CSS file in an HTML file, code a LINK tag like the following in the HTML file:

```
<LINK href="default.css" rel="stylesheet" type="text/css">
```

For convenience, arrange the Web browser used to check the design and the CSS editor to be side by side. After the CSS file is changed, save the file, then select "Refresh" at the Web browser.

2.3.8 Editing the web.xml

Use the XML editor to edit web.xml. The XML editor has a [Design] tab and a [Source] tab.

From the [Source] tab, the XML file can be edited directly, and Contents Assist can be used to add tags.

From the [Design] tab, the XML structure is displayed in a tree format, and operations such as addition of tags can be performed from the context menu.

For XML editor details, refer to "[Editing XML Files](#)".

Point

When using Contents Assist, tags must be added one at a time. However, if the user selects [Add Child] from the context menu when adding tags, the required child elements (tags) of the tag being added can also be added at the same time.

2.3.9 Verifying the Web Application Behavior

To verify the behavior of a Web application, it must be deployed to the execution environment, and then executed. Perform these operations from the Servers view.

For operation details, refer to "[6.2.6 Checking Application Behavior](#)".

Note

Any Web application that is monitored with the TCP/IP monitor may stop responding.

If this problem occurs, take one of the following actions:

- Reexecute processing of the request from the Web application.
- Restart the Web browser, and reexecute the Web application.

2.3.10 Verifying JavaScript Operation

When using the JavaScript debug support function, JavaScript global variable values can be referenced and changed without coding a print instruction for debugging.

Note

The JavaScript debug support function cannot set breakpoints to interrupt JavaScript execution, and cannot reference or change variable values at the time of an interruption. If that type of debugger is required, use a third party browser-embedded debugger. The Internet Explorer Developer Tools can be used for Internet Explorer 8, 9 or 10. Refer to the Internet Explorer Help for details.

Procedures prior to initiating debugging

The procedures prior to initiating debugging using the JavaScript debug support function are as follows:

1. Deploying the debug support Web application

Deploy the debug support Web application stored in the following location to the IIServer Cluster:

<Java integrated development environment installation folder>\etc\jsdebug\f5drjsdbg.war

2. Launching the IIServer Cluster

Launch the application to be debugged and the IIServer Cluster to which the debug support Web application was deployed.

3. Accessing the debug target Web application from a Web browser via the debug support Web application

When accessing the debug support Web application, specify the URL of the debug target Web application in the URL parameter

Example) `http://localhost/f5drjsdbg/?url=/(debug target context root)/index.html`

Implementing the above displays the debug target application window and the debug support function window in the Web browser.

Note

- The debug support function and the debug target application must be deployed on the same host.
- The debug support function window is displayed as a popup window. If the browser settings are configured to block popup windows, the debug support function window may not be displayed. In this case, set the browser to allow popup windows from the site on which the debug target application is deployed.

Referencing variable values

Refer to the following and add variables to the variable list and reference variable values:

- Variable name

Specify the name of the variable to be added to or deleted from the variable list. Multiple variable names can be specified, separated by commas or line feeds.

- Variable list

Variable names, types, and values are displayed in the following format:

Variable name (variable type): Variable value

If a variable is an object, the "+" icon is shown before the variable name. When this icon is clicked, a hierarchical information display shows the property name, type, and value of the object.

- Displaying the most recent values

The display of [Variable List] values is not updated automatically when the values are changed in an application. Click [Show Latest Values] to update the display. Locations with updated values are displayed in red.

Note

- Only global variables can be displayed. Local variables defined in functions cannot be displayed.
- Function type variable values cannot be referenced. Note that, if Internet Explorer is being used, "object" is displayed as the variable type even if the variable is a function type variable.

- JavaScript variables are enabled in only the page in which they are defined. When the display is switched to another page, the variables defined in the previous page are disabled.
- Variable value references and changes are possible only while the application window remains open. Once the window has been closed, the variable values can no longer be referenced or changed when the window is reopened.

Point

If an application window is split into frames, the variables defined in a page within a frame can be displayed by specifying "frame name.variable name" at [Variable Name].

Changing variable values

To change a variable value, click the variable value displayed in [Variable List]. This changes the area that shows the value to an input field. Enter the new value in the input field and press the [Enter] key to change the variable value. The changed location is displayed in green.

The value input method depends on the type of value to be entered, as shown below.

Value type	Input method
number	Enter a numeric value.
boolean	Enter "true" or "false".
string	Enter a character string enclosed in double quotation marks ("").
undefined	Enter "undefined" to set the variable value as undefined.
null	Enter "null" to set the variable value to null.
object	The value of an object type variable cannot be changed. If an object has a numeric type, logical value type, or string type property, the value can be changed for that property.
function	The value of a function type variable cannot be changed.

2.3.11 Distributing the Web Application to the Operating Environment

To distribute the created Web application to the operating environment, create an archive file, then deploy the application from the Interstage Java EE Admin Console.

Creating an Archive File

Use the Export wizard to create the archive file.

For details on operations using the Export wizard, refer to "[6.2.7 Distributing to the Operating Environment](#)".

Deploying from the Interstage Java EE Admin Console

The Interstage Java EE Admin Console can be operated from the remote environment. Log in to the Interstage Java EE Admin Console of the operating environment, and specify a local file to be deployed. This enables the file to be deployed from a remote environment to the operating environment.

Refer to the method for using the Java EE execution environment of Interstage Application Server for deployment and Interstage Java EE Admin Console details.

Chapter 3 Developing Enterprise JavaBeans (EJB)

This chapter presents an overview of Enterprise JavaBeans (hereafter abbreviated as EJB) and describes how to create enterprise beans and EJB client applications that run in a Java EE application execution environment.

3.1 Overview

This section presents an overview of EJB and the EJB development support functions.

3.1.1 What is an EJB?

An EJB is a server component model for Java, based on multi-tier (3-tier) distributed object-oriented programming. This is a framework for creating server components with a high degree of abstraction simply by coding the business logic processing. Low-level interfaces, such as management of the lifecycle of components required for server applications, transaction management, and so on, are hidden.

EJB implements the various management processes, such as the lifecycle management required for server applications, as server component receptacles known as containers. Containers shoulder the burden of complicated processes. Under EJB, server components that operate in containers are called enterprise beans.

Installing an enterprise bean in a container and making it executable is called deployment. Under EJB, the installation method known as deployment enables the creation of server components that have portability and that are not dependent on a specific container.

The following functions are regulated by EJB:

- Enterprise bean instance lifecycle management
- Transaction management
- Security management
- Session management
- Resource management

Enterprise Bean Types

Enterprise beans can be broadly divided into the following types:

- Session beans
An enterprise bean that performs interactive processing with clients. It codes mainly processes (business logic) required for business processing.
- Message-driven beans
An enterprise bean for performing asynchronous processing.

These classifications and their uses are described below.



Specifications up to EJB 2.1 also have entity beans for data handling in database systems or similar. However, from EJB 3.0, the Java Persistence API is used instead of entity beans, so an entity bean description is not included in this manual.

Session Beans

Session beans deploy application business logic to servers and provide services to multiple clients via a network. Session beans execute business logic by invoking other enterprise beans, controlling the flow of transactions and processes, and by implementing independent processes.

There are two types of session beans:

- Stateful
A stateful session bean has a one-to-one relationship with a client. It can maintain the transaction state and the variable values defined

in an enterprise bean across multiple methods invoked by the client. Accordingly, it is used if multiple procedures (methods) are required to provide a certain function.

- Stateless

A stateless session bean cannot maintain the transaction state and the variable values defined in an enterprise bean across multiple methods. Accordingly, it is used to provide functions that are completed by one method. In addition, since multiple clients can share the same session bean instance, the server load can be reduced.

For example, a stateless session bean should be used if a session bean implements processing of a simple arithmetic operation. Since there is no information that needs to be held, use of the stateless session bean reduces the load on the server. However, if a session bean implements the shopping cart of an online shopping site, a stateful session bean should be used. This is because the goods that the user has put into the shopping cart must be maintained across multiple pages.

Message-driven Beans

Message-driven beans provide the foundation for performing asynchronous processing in relation to a message sent from a client. A Message-driven bean can invoke a session bean or similar to execute that function asynchronously.

Message-driven beans receive and process JMS messages and resource adapter messages. For JMS messages, message processing methods can be classified as the following two types:

- Point-To-Point model

A specific recipient performs processing in relation to a single message sent from a sender. That is, this model is used if the message is allocated to a single recipient.

With this model, the sent messages are held in a queue on the JMS server. The queued messages are allocated to a specific recipient and processed.

- Publish/Subscribe model

Multiple recipients perform processing in relation to a single message sent from a sender. That is, this model is used if the same message needs to be allocated to multiple recipients.

With this model, the sent messages are queued in a topic on the JMS server. The queued messages are allocated to all recipients and processed by each recipient.

3.1.1.1 Modifications since J2EE 1.4

The EJB specification included in Java EE is EJB 3.0. In EJB 3.0, the specifications have been improved to enable easier development in response to complaints concerning the difficulty of EJB up to J2EE 1.4.

Simplification of Coding

One of the major features of EJB 3.0 is the simplification of EJB coding.

Under conventional EJB specifications, the various EJB definitions needed to be described in the deployment descriptor known as ejb-jar.xml. Under EJB 3.0, Java annotations can now be used to describe the various EJB definitions in the enterprise bean class. Thus, in most cases, EJB deployment descriptors do not need to be created.

Another improvement is that the various EJB definitions need only be defined when required. This reduces the number of definitions that the developer must describe. For example, when a single Java interface is implemented in a session bean class, it is automatically handled as an EJB local interface. In this case, the developer does not need to code information that indicates which interface is the Local interface.

In addition, the minimum number of files required for implementing an enterprise bean has now been reduced. The following table shows the elements required for developing an enterprise bean under conventional EJB and EJB 3.0 respectively.

Table 3.1 Mandatory Elements for Implementing Enterprise Beans

Enterprise Bean type	Earlier EJB (up to EJB 2.1)	EJB 3.0
Session bean	<ul style="list-style-type: none"> - Enterprise bean class - Home interface - Component interface - deployment descriptor 	<ul style="list-style-type: none"> - Enterprise bean class - Business interface
Message-driven bean	<ul style="list-style-type: none"> - Enterprise bean class 	<ul style="list-style-type: none"> - Enterprise bean class

Enterprise Bean type	Earlier EJB (up to EJB 2.1)	EJB 3.0
	- deployment descriptor	

The Introduction of Java Persistence API (JPA)

Another major feature of EJB 3.0 is the introduction of JPA. JPA is a persistence API used instead of entity beans. As with other enterprise beans, Java annotation can be used for JPA coding. In addition, under JPA, a query language to which EJB QL is enhanced is used for access to databases.

A major feature of JPA is that it is an ordinary persistence API. JPA is not only used from EJB but can also be used from Web applications and Java applications.

For details on JPA and development using JPA, refer to "[4.1.2 Developing Applications that use JPA](#)".

EJB 3.0 Main Annotations

The main EJB 3.0 annotations are introduced below with reference to simple examples.

javax.ejb.Stateless/javax.ejb.Stateful Annotation

This annotation defines a class as a session bean. The following properties can be specified.

Property name	Explanation
name	Defines the EJB name. If this property is not defined, the session bean class name is used as the EJB name.
mappedName	Use this property if the application server product requires a unique bean mapping name.
description	Defines the explanation of this session bean.

Example of Session Bean Creation using Stateless Annotation

```

package sample;

import javax.ejb.Stateless;

@Stateless
public class CalcBean implements Calc {
    public int add(int param1,int param2) {
        return param1 + param2;
    }
}

```

javax.ejb.MessageDriven Annotation

This annotation defines a class as a message-driven bean. The following properties can be specified.

Property name	Explanation
name	Defines the EJB name. If this property is not defined, the message-driven bean class name is used as the EJB name.
messageListenerInterface	Defines the message listener interface of the bean. This property needs to be defined only if the bean class does not implement a message listener interface or if multiple interfaces are implemented.
activationConfig	Defines the properties for the message-driven bean.
mappedName	Use this property if the application server product requires a unique bean mapping name.
description	Defines the explanation of this message-driven bean.

Example of Message Driven Bean Creation using Message Driven Annotation

```

package sample;

import javax.ejb.ActivationConfigProperty;
import javax.ejb.MessageDriven;
import javax.jms.MessageListener;
import javax.jms.Message;

@MessageDriven(mappedName = "jms/CalcBean" activationConfig = {
    @ActivationConfigProperty(
        propertyName = "acknowledgeMode", propertyValue = "Auto-acknowledge"),
    @ActivationConfigProperty(
        propertyName = "destinationType", propertyValue = "javax.jms.Queue")
})
public class CalcBean implements MessageListener {
    public void onMessage(Message msg) {
        ...
    }
}

```

javax.ejb.EJB Annotation

Annotation for performing Dependency Injection when invoking an enterprise bean. The following properties can be specified.

Property name	Explanation
name	JNDI name of bean to be referenced.
beanInterface	Business interface of bean to be referenced.
beanName	Bean name of bean to be referenced.
mappedName	Use this property if the application server product requires a unique bean mapping name.
description	Defines the explanation of this invocation.

Example of EJB Client Creation using EJB Annotation

```

@EJB
private static Calc calc;

public int callCalc(int param1,int param2) {
    return calc.add(param1,param2);
}

```

3.1.2 Developing EJBs

An overview of EJB development is given below.

Preparing for EJB Development

Before developing an EJB, an EJB project must be created. Create a project and set the required target runtime, classpath, and other project settings.

For details, refer to "[3.3.1 Preparing the Environment to create EJBs](#)".

Creating Enterprise Beans

Create enterprise beans using the Session Bean wizard or Message-Driven Bean wizard.

For details, refer to "[3.3.2 Creating Session Beans](#)" and "[3.3.3 Creating Message-driven Beans](#)".

Using Databases

Use the Java Persistence API to access databases from an enterprise bean.

For details, refer to "[3.3.4 Using Databases](#)".

Creating EJB Clients

Use the @EJB annotation as the EJB invocation method.
For details, refer to "[3.3.5 Creating EJB Clients](#)".

Checking EJB Operation

Use the Servers view to check the operation of the created enterprise bean application.
For details, refer to "[3.3.6 Checking EJB Operation](#)".

Deploying EJBs

The EJB must be archived in order for it to be deployed. Use the Export wizard to create an archive file.
For details, refer to "[3.3.7 Distributing EJBs to the Operating Environment](#)".

3.2 Introduction

This section introduces the procedures for creating an application that uses EJB from a Web application.

3.2.1 Application to be created

Create a session bean that returns the country name and total population from a ranking list of the populations of countries when the ranking is entered, and create an application that invokes that session bean from a Web application and displays the result on a Web page. The resultant application has the same theme as the application created as described in the Web application introduction, but the business logic part is replaced by EJB.

3.2.2 Development Flow

Development of the above EJB application proceeds as follows:

1. Create the project for the EJB

In order to create the enterprise bean, first create an EJB project. When the EJB project is created as directed by the wizard, the classpath and other settings required for the build are set automatically.

2. Create the Session Bean

Use the following procedure to create a session bean:

- Create the data class
Create the data class to be used to implement the session bean.
- Create the pattern for the session bean
Use the wizard to create the session bean pattern. Since the application completes using one method, create a stateless session bean.
- Implement the session bean
Declare the method in the session bean created using the wizard to implement processing.

3. Create an EAR project

Create a project in order to create an EAR file for deployment as one application including the EJB client. When creating the EAR project, specify to include the created EJB project in the EAR file. Since the EJB client is not yet created, specify to include the EJB client in the EAR file when creating the EJB client project.

4. Create the EJB client

Use the following procedure to create the EJB client:

- Create the client project
Create a dynamic Web project for creating a Web application as an EJB client.

- Set the build path
Set the build path for invoking the enterprise bean from the Web application.

- Create a servlet class
Use the wizard to create a servlet as a controller for receiving Web application requests.

- Create an I/O page
Use the wizard to create the Web application I/O page.

- Specify the Dependency Injection
Set the Dependency Injection in the field such that the session bean method is invoked from the servlet.

- Implement the session bean invocation processing
Implement the processing for invoking the session bean method using the field that declared the Dependency Injection.

5. Check the application operation

Use the following procedure to check the Application operation:

- Associate the project and the server
Set which server the application is deployed on. Since the applications are grouped as an EAR, add the EAR project to the server.

- Set the breakpoints
Set the breakpoints for the debugger to check the program operation during execution.

- Launch the server
Launch the server so that it can receive requests in relation to the application from Web browsers. Deployment is performed automatically before the server starts.

- Execute the application
Launch the Web browser and access the application URL to start checking the operation of the application.

- Debug the application
Debug the program and check that the application is operating correctly.

6. Deploy the application to the operating environment.

Use the following procedures to deploy the application to the operating environment:

- Export the application
Create an EAR file to deploy the application to the operating environment.

- Deploy the application to the operating environment
Deploy the EAR file from the Interstage Java EE Admin Console.

3.2.3 Development Procedures

The actual procedures used to develop the application are described below.

- 1) [Creating the EJB Project](#)
- 2) [Creating the Session Bean](#)
- 3) [Creating the EAR Project](#)
- 4) [Creating the EJB Client](#)
- 5) [Checking the Application Operation](#)
- 6) [Deploying the Application to the Operating Environment](#)

1) Creating the EJB Project

Select [File] > [New] > [Project] from the menu bar to display [New Project].

Select [EJB] > [EJB Project] from the [New Project] wizard, then click [Next].

Check and enter the following setup items:

Setup Items	Setup Content
Project name	EJBSample
Target Runtime	Interstage Application Server V11.1 IJServer Cluster (Java EE)
EJB Module version	3.0
Configuration	Default Configuration for Interstage Application Server V11.1 IJServer Cluster (Java EE)
Add project to an EAR	Do not select.

Set the information, then click [Next]. The EJB module page is displayed.

Check and enter the following setup items. After the following information is set, click [Finish].

Setup Items	Setup Content
Source Folder	ejbModule
Generate deployment descriptor	Do not check



Point

Under EJB 3.0, a deployment descriptor is not required. Enterprise beans can be created using only annotation.

2) Creating the Session Bean

2-1) Creating the Data Class

Create a data class that stores the country name and total population information, and gets information. To create the data class, select the created project, then right-click to display the context menu. Select [New] > [Class] from the context menu. The [New Java Class] wizard is displayed.

Check and enter the following setup items. After the following information is set, click [Finish].

Setup Items	Setup Content
Source folder	EJBSample/ejbModule
Package	sample
Name	CountryData

The following source file is generated:

Source File	Description
CountryData.java	Data class

Implement the processing for retaining the country names and total populations. Add the places shown in **red** below to the source.

Data Class Implementation (CountryData.java)

```

package sample;

public class CountryData {

    private String countryName;
    private int totalPopulation;

    public CountryData(String name, int total) {
        setCountryName(name);
        setTotalPopulation(total);
    }

    public String getCountryName() {
        return countryName;
    }

    public void setCountryName(String countryName) {
        this.countryName = countryName;
    }

    public int getTotalPopulation() {
        return totalPopulation;
    }

    public void setTotalPopulation(int totalPopulation) {
        this.totalPopulation = totalPopulation;
    }

}

```

 **Point**

After the fields are added, and while the class is in the selected state, [Source] > [Generate Getters and Setters] can be selected from the menu to add getter/setter.

2-2) Creating the Session Bean Pattern

To create the session bean pattern, select the created project, then right-click to display the context menu. Select [New] > [Session Bean] from the context menu. The [Create EJB 3.0 Session Bean] wizard is displayed.

Check and enter the following setup items.

Setup Items	Setup Content
EJB project	EJBSample
Source folder	/EJBSample/ejbModule
Java package	sample
Class name	PopulationRanking

Setup Items	Setup Content
State type	Stateless
Remote	Do not select.
Local	Select sample.PopulationRankingLocal

After the information is set, click [Next]. The information setup page of the session bean is displayed.

Check the following setup items. After the information is set and checked, click [Finish].

Setup Items	Setup Content
Bean name	PopulationRanking
Transaction type	Container
Interfaces	Sample.PopulationRankingLocal
Inherited abstract methods	Select
Constructors from superclass	Select

When [Finish] is clicked, the following source files are generated.

Source Files	Description
PopulationRanking.java	Session bean class
PopulationRankingLocal.java	Business interface

2-3) Implementing the Session Bean

Implement as a session bean method the processing for entering the ranking and returning the country name and total population. Add the places in **red** below to the source.

Business Interface Implementation (PopulationRankingLocal.java)

```
package sample;

import javax.ejb.Local;

@Local
public interface PopulationRankingLocal {
    public CountryData getCountryData(int rank);
}
```

Session Bean Class Implementation (PopulationRanking.java)

```
package sample;

import javax.ejb.Stateless;

/**
 * Session Bean implementation class PopulationRanking
 */
@Stateless
public class PopulationRanking implements PopulationRankingLocal {

    /**
     * Default constructor.
     */
    public PopulationRanking() {
```

```

    }

    private CountryData countries[] = new CountryData[] {
        new CountryData("China",1330000000),
        new CountryData("India",1140000000),
        new CountryData("America",3000000000),
        new CountryData("Indonesia",2300000000),
        new CountryData("Brazil",1900000000),
        new CountryData("Pakistan",1600000000),
        new CountryData("Bangladesh",1500000000),
        new CountryData("Russia",1400000000),
        new CountryData("Nigeria",1400000000),
        new CountryData("Japan",1300000000)
    };

    public CountryData getCountryData(int rank) {
        --rank;
        if (rank < 0 || rank >= countries.length) {
            return null;
        }
        return countries[rank];
    }
}

```

3) Creating the EAR Project

Select [File] > [New] > [Project] from the menu bar. The [New Project] is displayed.

Select [Java EE] > [Enterprise Application Project] from the [New Project] wizard, then click [Next].

Check and enter the following setup items:

Setup Items	Setup Content
Project name	EJBEARSample
Target Runtime	Interstage Application Server V11.1 IJServer Cluster (Java EE)
EAR version	5.0
Configuration	Default Configuration for Interstage Application Server V11.1 IJServer Cluster (Java EE)

Set the information, then click [Next]. The module page to be added to EAR is displayed.

Check and enter the following setup items. After the following information is set, click [Finish].

Setup Items	Setup Content
Java EE Module Dependencies	Select EJBSample.
Content Directory	EarContent
Generate deployment descriptor	Do not select.

4) Creating the EJB Client

If developing applications under normal circumstances, procedures to create a Web application as a client are needed, but in this introduction, a sample is imported and the important points as an EJB client are checked. If you would like to learn the detailed procedure for creating a Web application, refer to "2.2 Introduction" in the Web application chapter.

Use the following procedure to import a sample.

Select [File] > [Import] from the menu. Select [General] > [Existing Projects into Workspace] from the displayed Import wizard. Select [Select archive file], then click the [Browse] button and select the following files:

<Workbench installation folder>\sample\EJBSample.zip

Select [EJBClientSample] from [Projects], then click [Finish].

After the EJBClientSample project has been imported, select the EJBEARSample project. Right-click to display the context menu and select [Properties]. Select [Java EE Module Dependencies] from the Properties dialog box. Select EJBClientSample and add the EJB client to the EAR.

4-1) Creating the Client Project

Create a dynamic Web project as an EJB client. Since this can be completed by importing a sample, details are omitted. Refer to the following table for the settings.

Setup Items	Setup Content
Project name	EJBClientSample
Target Runtime	Interstage Application Server V11.1 IJServer Cluster (Java EE)
Configuration	Default Configuration for Interstage Application Server V11.1 IJServer Cluster (Java EE)
Dynamic Web Module version	2.5
Add project to an EAR	Select
EAR Project Name	EJBEARSample

4-2) Setting the Build Path

Set the build path (classpath) used to reference the business interface (PopulationRanking). This is already set in the imported sample project.

To check the settings, select the project (EJBClientSample), and right-click. Select [Properties] from the context menu. Select [Java EE Module Dependencies] from the Properties dialog box to display the [Java EE Modules] tab.

Check that the following setting is selected.

Setup Items	Setup Content
JAR/Module	EJBSample.jar

4-3) Creating the Servlet Class

Create the servlet class. The servlet class is already created in the imported sample project with the following settings.

Setup Items	Setup Content
Java package	sample
Class name	ServletController
Superclass	javax.servlet.http.HttpServlet
Which method stubs would you like to create?	Constructors from the superclass Inherited abstract methods doPost (remove the doGet check)

4-4) Creating the I/O Page

Create the I/O page JSP. A file like the following is already created in the imported sample project.

Input Page (default.jsp)

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/
html4/loose.dtd">
```

```

<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
<title>World ranking for total population</title>
</head>
<body>
<h1>Input page</h1>
<p> Enter a ranking from 1 to 10. </p>
<p></p>
<form action="ServletController" method="post">
    Rank:<br>
    <input name="rank" type="text" size="10">Rank<br>
    <p></p>
    <input type="submit" value="OK">
    <input type="reset" value="Clear">
</form>

</body>
</html>

```

Output Page (result.jsp)

```

<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/
html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
<title>World ranking for total population</title>
</head>
<body>
<h1>Output page</h1>

Total population ranking ${param.rank} rank is "$
{applicationScope.ranking.countryName}"<br>
Total population is ${applicationScope.ranking.totalPopulation} people.
<br>
<hr>
<form action="ServletController" method="post">
    <input type="submit" value=" Back to input page">
    <input type="hidden" name="mode" value="top">
</form>

</body>
</html>

```

Error Page (error.jsp)

```

<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/
html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
<title>World ranking for total population</title>
</head>
<body>

```

```
<h1>Error page</h1>
The entered ranking is not in the specified range, or a server error occurred.<br>
<br>
<hr>
<form action="ServletController" method="post">
  <input type="submit" value=" Back to input page ">
  <input type="hidden" name="mode" value="top">
</form>

</body>
</html>
```

4-5) Setting the Dependency Injection

The implementation for invoking the session bean is performed in the servlet class. This is already implemented in the servlet source of the imported sample project.

Create a business interface field, set the following EJB annotation, and specify the Dependency Injection.

EJB Annotation Definition

```
@EJB
private PopulationRankingLocal business;
```

Point

.....

Instead of the lookup method used up to J2EE1.4, Dependency Injection is used in Java EE and can be used to fetch and use an object. (During execution, the object is automatically set in a field by means of a Java EE container.)

.....

4-6) Implementing the Session Bean Invocation Processing

To invoke the session bean method, simply use the business interface type field that defined the EJB annotation to invoke the method without making any changes.

This is already implemented in the servlet source of the imported sample project.

Definition for the Session Bean Invocation Method

```
CountryData country = business.getCountryData(rank);
```

The servlet class in which session bean invocation is implemented is as shown below. (The imported project sample source is in this format). The parts in **red** are the added and changed places.

Servlet Class (ServletController.java)

```
package sample;

import java.io.IOException;

import javax.ejb.EJB;
import javax.servlet.RequestDispatcher;

import javax.servlet.ServletContext;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

/**
```

```

* Servlet implementation class ServletController
*/
public class ServletController extends HttpServlet {
    private static final long serialVersionUID = 1L;

    @EJB
    private PopulationRankingLocal business;

    /**
     * @see HttpServlet#HttpServlet()
     */
    public ServletController() {
        super();
    }

    /**
     * @see HttpServlet#doPost(HttpServletRequest request, HttpServletResponse response)
     */
    protected void doPost(HttpServletRequest request, HttpServletResponse response) throws
ServletException, IOException {
        // TODO Auto-generated method stub
        request.setCharacterEncoding("ISO-8859-1");
        ServletContext sc = getServletContext();

        // Get the type of file to be invoked.
        String mode = request.getParameter("mode");
        if (mode != null && mode.equals("top")) {
            // Invoke the Input page HTML file.
            RequestDispatcher inRd = getServletContext().getRequestDispatcher("/default.jsp");
            inRd.forward(request, response);
            return;
        }

        int rank;

        // Input check
        try {
            rank = Integer.parseInt(request.getParameter("rank"));
        } catch (NumberFormatException e) {
            RequestDispatcher errRd = sc.getRequestDispatcher("/error.jsp");
            errRd.forward(request, response);
            return;
        }

        CountryData country = business.getCountryData(rank);

        //Check the obtained value
        if (country == null) {
            RequestDispatcher errRd = sc.getRequestDispatcher("/error.jsp");
            errRd.forward(request, response);
            return;
        }

        sc.setAttribute("ranking", country);

        RequestDispatcher outRd = sc.getRequestDispatcher("/result.jsp");
        outRd.forward(request, response);
    }
}

```

Point

After the required implementation is coded, right-click on the Java editor, then select [Source] > [Organize Imports] from the context menu. The import statement that is appropriate for the classpath that was set in the project can then be inserted.

5) Checking the Application Operation

5-1) Associating the Project with a Server

Select a server in the Servers view, then select [Add and Remove Projects] from the context menu.

Select the created EAR project from the available projects, then click [Add]. Check that the EAR project has been added to the configured projects.

Check the following setup items, then click [Finish].

Setup Items	Setup Content
Configured projects	EJBEARSample

Point

If the deployment destination server in the Servers view is not registered, then it must be added. For details on how to do that, refer to "[6.2.6 Checking Application Behavior](#)".

5-2) Setting Breakpoints

Set a breakpoint at the first line of the `getCountryData()` method of the session bean class. To set or delete a breakpoint, double-click the ruler part on the left edge of the editor.

5-3) Launching the Server

Select a server in the Servers view, then select [Debug] or [Restart in Debug] or [Connect(Debug Launch)/Login] from the context menu.

Point

If you want to simply launch the application without using the debugger, select [Start] from the context menu.

The EAR file is deployed automatically before the server starts.

Check the following information in the Servers view:

Item to Check	Contents to Check
Server state	Started
Server status	Synchronized
EAR file (EJBEARSample) state	Synchronized
State of the EJB-JAR (EJBSample) file under the EAR file	Synchronized
State of the WAR file (EJBClientSample) under the EAR file	Synchronized

5-4) Executing the Application

In the Servers view, select the deploying Web project (EJBClientSample), then from the context menu, select [Web browser]. The Web browser is started, and the input window is opened.

Setup Items	Setup Content
Input page URL	http://localhost/EJBClientSample/

Enter a population ranking in the [Rank:] input field, then click [OK].

Point

The user can select the project, then select [Run As] > [Run on Server] or [Debug As] > [Debug on Server] from the context menu to add the server, add the project, launch the server, and launch the client all together.

In addition, the Web browser used as the client can be changed by selecting [Window] > [Web Browser] from the menu.

5-5) Debugging the Application

The application runs and is interrupted at the breakpoints. Check the Variables view display. Select [Run] > [Step Over] from the menu and check the program state in the Variables view. For debug details, refer to "[6.2.6.1 Debugging](#)".

Point

In addition to checking values, values can be changed in the Variables view.

Processing that has been interrupted by the debugger can be restarted by selecting [Run] > [Resume] from the menu. This allows the processing on the server side to complete so that the output page is displayed on the Web browser. Check that the country name and total population for the entered ranking is displayed.

6) Deploying the Application to the Operating Environment

An EAR file must be created in order to deploy the application to the operating environment. The created EAR file uses the server deployment function to deploy the application to the server.

6-1) Exporting the Application

The EAR file is created from the Export wizard. To launch the Export wizard, select [File] > [Export]. From [Export] wizard, select [Java EE] > [EAR file].

The EAR Export wizard is displayed.

Check and enter the following setup items. After the following information is set, click [Finish].

Setup Items	Setup Content
EAR project	Specify EJBEARSample.
Destination	Specify the creation destination for the EAR file.

6-2) Deploying to the Operating Environment

The application can be deployed from a remote environment to the operating environment by logging in to the Interstage Java EE Admin Console at the operating environment and by specifying a local EAR file.

3.3 Tasks

This section describes how to develop an EJB application using Interstage Studio. It explains each of the development work topics (tasks) separately.

- [3.3.1 Preparing the Environment to create EJBs](#)
- [3.3.2 Creating Session Beans](#)
- [3.3.3 Creating Message-driven Beans](#)

- [3.3.4 Using Databases](#)
- [3.3.5 Creating EJB Clients](#)
- [3.3.6 Checking EJB Operation](#)
- [3.3.7 Distributing EJBs to the Operating Environment](#)

3.3.1 Preparing the Environment to create EJBs

In order to create an EJB, first create a project. Then prepare an environment that allows the required libraries to be set in the project and enables the build.

Creating Projects

From the [New Project] wizard, select [EJB] > [EJB Project] to create the EJB project. For details on the parts common to project wizards, refer to "[6.2.2.1 Creating New Projects](#)".

Refer below for the settings that are unique to the EJB Project wizard.

- Source Folder
Specify the folder that stores the project sources.
- Generate deployment descriptor
Under EJB 3.0, a deployment descriptor is not required.
Select as required, not only for a Dependency Injection that uses annotation, but also if JNDI lookup processing is performed, if an environment entry is used, and so on.

Classpath Settings

Classpaths must be set if there are libraries or similar that are required in order to create the application. Classpaths are set in the project build path.

For details on build path settings, refer to "[6.2.2.3 Setting Classpaths](#)".

3.3.2 Creating Session Beans

To create session beans, use the Create EJB 3.0 Session Bean wizard to create the session bean class source file and implement the business methods in the file. The method is described below.

Creating Session Bean Classes with the Create EJB 3.0 Session Bean Wizard

From the [New] wizard, select [EJB] > [Session Bean], and the wizard creates the session bean. Refer to the following for the wizard settings:

- EJB project
Specify the EJB project that will generate the session bean.
- Source folder
Specify the folder that stores the session bean sources.
- Java package
Specify the package name of the session bean class and the business interface.
- Class name
Specify the bean class name of the session bean.
- State type
Select either "Stateful" or "Stateless" in accordance with the type of session bean you want to create.

- Create business interface
Select the business interface to be generated as "Remote" or "Local", then specify the interface name to be generated.

- Bean name
Specify the EJB name of the session bean.

- Mapped name
Specify the JNDI name mapped to the session bean.

- Transaction type
Select the transaction type as "Container" or "Bean".

- Interfaces
Specify the interface to be used by the session bean.

- Home and Components Interfaces (EJB 2.x)
If generating an EJB2.x format interface, select "Remote" or "Local", then specify the interface name to be generated.

When the wizard is executed, the Java source of the session bean class and the business interface is generated.

Implementing the Business Method

After the source is generated, implement the business method in the session bean. The procedure for implementing the business method is as follows:

1. Code the method declaration in the business interface.
2. Code the implementation of that method in the session bean class.

The method declared in the business interface is automatically handled as a session bean business method.

The methods for declaring and implementing these methods are the same as for ordinary Java interfaces and Java classes. Use the Java editor to declare and implement the methods.

3.3.3 Creating Message-driven Beans

To create a message-driven bean, use the Enterprise Bean wizard to create a message-driven bean class source file, then implement the message listener method (`onMessage()`). The method is described below.

Using the Create EJB 3.0 Message-Driven Bean Wizard to create Message-driven Bean Classes

Select [EJB] > [Message-Driven Bean] from the [New] wizard, and the message-driven bean is created by the wizard. Refer below for the settings which are set by the wizard.

- EJB project
Specify the EJB project that will generate the message-driven bean.

- Source folder
Specify the folder where the message-driven bean source is stored.

- Java package
Specify the name of the message-driven bean class package.

- Class name
Specify the bean class name of the message-driven bean.

- Destination name
Specify the JNDI name of the JMS resource.

- JMS
Check if using JMS messages. If this is checked, set the "Destination type".

- Destination type
If "JMS" is checked, select the destination type of the message-driven bean as "Queue" or "Topic".

- Bean name
Specify the EJB name of the message-driven bean.

- Transaction type
Select the transaction type as "Container" or "Bean".

- Interfaces
Specify the interface to be used by the message-driven bean.

When the wizard is executed, the Java source for the message-driven bean class is generated.

Implementing the Message Listener Method (onMessage())

An empty onMessage() method is defined in the Java source of the message-driven bean class generated by the wizard. Implement the processing for the received message in this method.

3.3.4 Using Databases

Under EJB 3.0, the Java Persistence API (JPA) has replaced the entity bean as the mechanism for handling databases. For details on using JPA to use databases, refer to ["4.1.2 Developing Applications that use JPA"](#).

3.3.5 Creating EJB Clients

This section describes how to create a client that invokes a session bean business method and a client that sends messages to a message-driven bean.

3.3.5.1 Creating Clients that invoke Session Beans

A special definition file is not required when creating a client that invokes a session bean. In the Java class (servlet or similar) that is used as the client, code the processing that gets the session bean business interface and the processing that uses the interface to invoke the business method.

Creating the Business Interface

The business interface can be fetched easily using the Dependency Injection. Use @EJB annotation as shown below to get the business interface.

Example of using EJB Annotation

```
@EJB
private Calc calc;
```

Coding Example	Explanation
ejb/Calc	Specify the JNDI name associated with the session bean.
Calc	This is the business interface. (It is a local interface.)
Calc#add(int, int)	This is the business method.



Note

The Dependency Injection can be used only in Java EE components managed in Java EE containers, such as servlets, EJB, and so on. For other classes, use JNDI lookup to get objects.

For details, refer to "[10.2.2 Using JNDI Lookup to Obtain Objects](#)".

Invoking the Business Method

To invoke the business method, use the obtained business interface to invoke the method.

Example of Invoking a Business Method

```
int result = calc.add(100,200);
```

3.3.5.2 Creating Clients that send Messages to Message-driven Beans

A special definition file is not required when creating a client that sends messages to a message-driven bean. In the Java class (servlet or similar) that is used as the client, code the processing that fetches the message-driven bean JMS connection factory or similar and the processing that uses this to send messages.

The explanation below is for sending messages to a point-to-point model message-driven bean.

Fetching the JMS Connection Factory and the JMS Destination

The JMS connection factory and the JMS Destination can be fetched easily using a Dependency Injection.

Use an @Resource annotation such as the following to fetch the JMS connection factory and the JMS Destination.

Example of Using Resource Annotation

```
@Resource(name="jms/QueueCF001")
private QueueConnectionFactory qcf;
@Resource(name="jms/Queue")
private Queue queue;
```

Coding Example	Explanation
jms/QueueCF001	JMS connection factory name
jms/Queue	JMS Destination name



Note

The Dependency Injection can be used only in Java EE components managed in Java EE containers, such as servlets, EJB, and so on. For other classes, use JNDI lookup to fetch objects.

For details, refer to "[10.2.2 Using JNDI Lookup to Obtain Objects](#)".

Sending Messages

After the JMS connection factory and the JMS Destination are fetched, the message sending procedure is the same as for up to EJB 2.1. An example is below.

Example of Sending a Message

```
// Create connection, session, and sender
QueueConnection connection = qcf.createQueueConnection();
QueueSession session = connection.createQueueSession(false,
Session.AUTO_ACKNOWLEDGE);
QueueSender sender = session.createSender(queue);
```

```
// Send a text message
TextMessage msg = session.createTestMessage();
msg.setText("Hello World!");
sender.send(msg);
// Close
connection.close();
```

3.3.6 Checking EJB Operation

To check the operation of an EJB, it must be deployed to the EJB execution environment, then executed. Perform these operations in the Servers view.

For operation details, refer to "[6.2.6 Checking Application Behavior](#)".

3.3.7 Distributing EJBs to the Operating Environment

To deploy the created EJB to the operating environment, create an archive file, then deploy the EJB from the Interstage Java EE Admin Console.

Creating Archive Files

Use the Export wizard to create an archive file.

For details on operations using the Export wizard, refer to "[6.2.7 Distributing to the Operating Environment](#)".

Deploying from the Interstage Java EE Admin Console

The Interstage Java EE Admin Console can be operated from the remote environment. Log in to the Interstage Java EE Admin Console of the operating environment, and specify a local file to be deployed. This enables deployment from a remote environment to the operating environment.

For details on deploying and the Interstage Java EE Admin Console, refer separately to the usage method of the Interstage Application Server Java EE execution environment.

Chapter 4 Developing Applications that use Java Persistence API (JPA)

This chapter presents an overview of Java Persistence API (hereafter abbreviated as JPA) and describes how to create applications that use JPA.

4.1 Overview

This section presents an overview of JPA and JPA development support functions.

4.1.1 What is JPA?

Java Persistence API (JPA) groups together the EJB 2.1 entity bean specifications as one independent specification from EJB 3.0. JPA provides object/reference (O/R) mapping mechanisms in order to operate relational databases (RDB) from Java applications. O/R mapping maps the Java persistence class and RDB tables, thereby making it easy to make data persistent. Since JPA handles databases as Java objects, the number of situations where SQL statements are issued to databases can be reduced.

Simple explanations are given below for the terminology required to understand JPA.

- Persistence unit
This is the logical grouping for achieving the persistence defined in persistence.xml. The persistence unit is defined by multiple persistence classes (Entity classes and so on) and O/R mapping files.
- Entity
An Entity is a Java class that is associated with a database table. An instance of this Java class corresponds to one database record.
- Persistence context
This is a set of Entity class instances. JPA performs operations in relation to persistence contexts, and achieves database synchronization by transaction segmentation.
- Java Persistence Query Language (JPQL)
This is a query language that is equivalent to database SQL.
- Entity manager
This is the interface for Entity and persistence context operations using JPQL.
- persistence.xml
The Entity class, O/R mapping file, transaction types, and other configuration information can be defined in persistence units.
- orm.xml
This is the O/R mapping file. The O/R mapping file codes the associations set between Entities and databases. This file is not mandatory, since these can also be coded using annotations.

4.1.1.1 Modifications since EJB2.1

The EJB 2.1 entity bean and the JPA Entity have the following major differences:

- File that configures the Entity
EJB 2.1 entity bean conventions require that an entity bean class, a HOME interface, and a Component interface be created, and the entity bean must be defined in the deployment descriptor.
With a JPA Entity, simply create a Java class for the Entity and set O/R mapping information of @Entity annotations, and so on, in that Java class.

- Entity invocation method

In Java EE, Dependency Injection can be used for JPA Entities and also to easily code conventional lookup processing.

Simple examples of the main annotations used by JPA are introduced below.

javax.persistence.Entity Annotation

This annotation defines a class as an Entity. The following element can be specified.

Element Name	Explanation
name	The Entity name. If omitted, the class name is used as the Entity name.

Example of using Entity annotation

```
package sample;

import javax.persistence.Entity;
import javax.persistence.Id;

@Entity
public class Employee {
    @Id
    private int id;
    private String name;
    private String address;
    private String tel;
}
```

javax.persistence.PersistenceContext Annotation

An Entity is managed by the Entity manager. The PersistenceContext annotation can obtain this Entity manager through Dependency Injection. The following elements can be specified.

Element Name	Explanation
name	This is the name used to access the Entity manager in an environment that references a context. This name is not used with Dependency Injection.
properties	Use this element if you want to specify properties for a container or a persistence provider.
type	Specify the transaction type.
unitName	Persistence unit name. Specify the name that can be accessed using JNDI.

Example of using PersistenceContext annotation

```
@PersistenceContext
EntityManager em;

public Employee getEmployeeByPrimarykey (int id) {
    return em.find(Employee.class, id);
}
```

4.1.2 Developing Applications that use JPA

An overview of development of applications that use JPA is given below.

Preparing for Development of Applications that use JPA

Programs that use JPA can be handled as ordinary libraries, or can be included in Web applications and EJB applications. If included in a Web application or an EJB application, a special-purpose project does not need to be created. However, create a JPA project if the

program is to be used as a library.

For details, refer to "[4.3.1 Preparing the Environment to create Applications that use JPA](#)".

Developing Persistence Units

To develop a persistence unit, an Entity class must be created and an XML file must be edited.

A function is provided for verifying that development has been performed in accordance with JPA specifications.

For details, refer to "[4.3.2 Developing Persistence Units](#)".

Creating Entity Classes

To create an Entity class, either perform Java class and database mapping, or create it from database tables.

For details, refer to "[4.3.3 Creating Entity Classes](#)" and "[4.3.3.4 Generating Entity Classes from Tables](#)".

Operating Database with JPA

Under JPA, database operations are performed by using the Entity manager to operate an Entity.

For details, refer to "[4.3.4 Operating Database with JPA](#)".

Checking Operation of Applications that use JPA

Use the Servers view to check the operation of the created application.

For details, refer to "[4.3.5 Checking Operation of Applications that use JPA](#)".

Distributing Applications that use JPA

Applications that use JPA must be archived in order to be distributed. Use the Export wizard to create an archive file for the application.

For details, refer to "[4.3.6 Distributing Applications that use JPA to the Operating Environment](#)".

4.2 Introduction

This section introduces the procedures for creating applications that use JPA.

4.2.1 Application to be created

Create a JPA library that gets data from a database that stores country name and total population data. Then, create a Web application that invokes this library. The Web application provides a page for entering the population ranking, and a page for output of the country name and total population corresponding to the entered rank.

Use the database created in "[8.2 Introduction](#)" of "Database Operation" and the Web application created in "[2.2 Introduction](#)" of "Developing Web Applications".

4.2.2 Development Flow

Development of applications that use JPA is described below.

1. Prepare the database

Prepare the environment for the database to be used with the JPA application.

2. Create the persistence unit

- Create a JPA project

In order to create a JPA application, first create a JPA project. When a JPA project is created as indicated by the wizard, the classpath and other settings required for the build are set automatically.

- Edit persistence.xml

Code the data source used by the persistence unit in persistence.xml.

3. Create the Entity

Use the following procedures to create the Entity:

- Create the Entity class
Use the wizard to create the Entity class.

- Set the Entity annotation
Use the JPA Structure view and the JPA Details view to set the Entity annotation in the created class.

- Set the association with the tables
Use the JPA Structure view and the JPA Details view to set the associations between the Entity and the database table.

- Add fields
Add the fields that are to be the Entity persistence items.

- Set associations with columns
Use the JPA Structure view and the JPA Details view to set the associations between persistence fields and the database.

- Add the get method
Add the get method to be used to access the persistence fields.

4. Create the logic class

- Create the class
Use the wizard to create a Java class.

- Implement the logic class
In the logic class, implement the processing that uses the Entity to access the database.

5. Create the EAR project

Create a project for creating an EAR file for distribution as one application including the client. When creating the EAR project, specify to include the created JPA project in the EAR file. Since the client is not yet created, specify to include the client in the EAR file when creating the client project.

6. Create the Web application

Use the following procedure to create the Web application that is to be used as the client:

- Create the dynamic Web project
Create a dynamic Web project for creating a Web application as a client.

- Set the build path
Set the build path for invoking the JPA project logic class from the Web application.

- Create the servlet class
Use the wizard to create a servlet as a controller for receiving Web application requests.

- Specify the Dependency Injection
An EntityManagerFactory is required because the logic class performs database access processing using an Entity. To fetch this object, specify the Dependency Injection in the field.

- Implement the servlet class
Perform implementation processing for the servlet that includes the logic class invocation.
 - Create an I/O page
Use the wizard to create the Web application I/O page.
7. Set the Interstage Application Server JDBC
- Use the following procedures to set up the environment so that the data source specified in the persistence unit can be used by a Java EE container.
- Launch the Interstage Java EE Admin Console
Launch the Interstage Java EE Admin Console in the Servers view.
 - Create a JDBC Connection Pool
Use the Interstage Java EE Admin Console to create a JDBC Connection Pool.
 - Create a JDBC Resource
Use the Interstage Java EE Admin Console to create a JDBC Resource.
 - Set the classpath for the JDBC driver
Add the JDBC driver to the classpath so that the JDBC driver can be referenced by applications that run in the JavaEE container.
8. Check the application operation
- Use the following procedure to check the application operation:
- Associate the project and the server
Set which server the application is deployed on. Since the applications are archived as an EAR, add the EAR project to the server.
 - Set the breakpoints
Set the breakpoints for the debugger to check the program operation during execution.
 - Launch the server
Launch the server so that it can receive requests in relation to the application from Web browsers. Deployment is performed automatically before the server starts.
 - Execute the application
Launch the Web browser and access the application URL to start checking the operation of the application.
 - Debug the application
Debug the program and check that the application is operating correctly.
9. Distribute the application to the operating environment
- Use the following procedures to distribute the application to the operating environment:
- Export the application
Create an EAR file to distribute the application to the operating environment.
 - Distribute the application to the operating environment
Deploy the EAR file from the Interstage Java EE Admin Console.

4.2.3 Development Procedures

The actual procedures used to develop the application are described below.

- 1) [Preparing the Database](#)
- 2) [Creating the Persistence Unit](#)
- 3) [Creating the Entity](#)
- 4) [Creating the Logic Class](#)
- 5) [Creating the EAR Project](#)
- 6) [Creating the Web Application](#)
- 7) [Setting the Interstage Application Server JDBC](#)
- 8) [Checking Application Operation](#)
- 9) [Distributing the Application to the Operating Environment](#)

1) Preparing the Database

The database uses Oracle. Create the database table that stores the country name and total population data and insert the data required for execution for the application. In addition, prepare an environment that enables connection from the workbench to the database. For details, refer to "[8.2 Introduction](#)" in "Database Operation", where these tasks are described.

2) Creating the Persistence Unit

2-1) Creating the JPA Project

Select [File] > [New] > [Project] from the menu bar. The [New Project] wizard is displayed.

Select [JPA] > [JPA Project] from the [New Project] wizard, then click [Next].

Check and enter the following setup items:

Setup items	Setup content
Project name	JPASample
Target Runtime	Interstage Application Server V11.1 IJServer Cluster (Java EE)
Configuration	Default Configuration for Interstage Application Server V11.1 IJServer Cluster (Java EE)

Set the information, then click [Next]. The [JPA Facet] page is displayed.

Check and enter the following setup items. After the following information is set, click [Finish].

Setup items	Setup content
Platform	Generic
Connection	Oracle_Studio
Override default schema from connection	Do not select.
Use implementation provided by server runtime	Select
Discover annotated classes automatically	Select
Create orm.xml	Do not select.



.....
By specifying the connection profile connected to the database at [Connection], the database table and column information can be used in the JPA Details view or similar.
.....

2-2) Editing the persistence.xml File

In persistence.xml, code the JDBC Resource of the database associated with the Entity.

Select the created project (JPASample), then double-click [src] > [META-INF] > [persistence.xml] for the Persistence XML editor to open the file.

In the Persistence XML Editor, select the [Connection] tab, then configure the JDBC Resource as displayed below.

Setup items	Setup content
Transaction Type	JTA
JTA Data Source Name	jdbc/Oracle

3) Creating the Entity

Create the Entity that performs database table mapping.



.....
 Since the method of using the JPA Structure view and the JPA Details view is described below, the method for creating a Java class and associating the database with that is described. However, an Entity can also be generated from a database table. For details, refer to "[4.3.3.4 Generating Entity Classes from Tables](#)".

3-1) Creating the Entity

Select the created project, then right-click to display the context menu. Select [New] > [Other] from the context menu. The [New] wizard is displayed.

In the [New] wizard, select [JPA] > [Entity], then click [Next].

Check and enter the following setup items.

Setup items	Setup content
Project	JPASample
Source folder	/JPASample/src
Java package	sample
Class name	CountryData
Inheritance	Entity
Add to entity mappings in XML	Do not select.

After the information is set, click [Next]. The [Entity Properties] page is displayed.

Check and enter the following setup items. After the following information is set, click [Finish].

Setup items	Setup content
Entity Name	CountryData
Use default	Do not select.
Table Name	C_DATA
Entity Fields	Add the following field
Access Type	Field-based

Key	Name	Type
Select	id	int

Key	Name	Type
Do not select	countryName	java.lang.String
Do not select	totalPopulation	int

3-2) Setting the Association with the Table

Set an association between the Entity CountryData class and the database C_DATA table. Check that the CountryData class is selected in the [JPA Structure] view. In the [JPA Details] view, check that "C_DATA" is set in [Name] of the [Table] group.

Point

When not connected to the database, the combo box does not display suggestions but the table name can be specified directly.

Check and select the following setup items.

Setup items	Setup content
Name	Default (CountryData)
Table: Name	C_DATA
Table: Catalog	Default ()
Table: Scheme	STUDIO

3-3) Setting Associations with Columns

Set the associations between the Entity fields and the table columns.

Each field corresponds to a column, as shown in the table below.

Field	Association	Column: name
id	Id	ID
countryName	Basic (Default)	C_NAME
totalPopulation	Basic (Default)	T_POP

The preceding procedures create the following source.

CountryData class

```

package sample;

import java.io.Serializable;
import java.lang.String;
import javax.persistence.*;

/**
 * Entity implementation class for Entity: CountryData
 *
 */
@Entity
@Table(name="C_DATA", schema = "STUDIO")
public class CountryData implements Serializable {

    @Id
    @Column(name="ID")
    private int id;

    @Column(name="C_NAME")

```

```

private String countryName;

@Column(name="T_POP")
private int totalPopulation;
private static final long serialVersionUID = 1L;

public CountryData() {
    super();
}
public int getId() {
    return this.id;
}
public void setId(int id) {
    this.id = id;
}

public String getCountryName() {
    return this.countryName;
}
public void setCountryName(String countryName) {
    this.countryName = countryName;
}
public int getTotalPopulation() {
    return this.totalPopulation;
}
public void setTotalPopulation(int totalPopulation) {
    this.totalPopulation = totalPopulation;
}
}

```

4) Creating the Logic Class

Create the PopulationRanking class that performs queries to the database and so on.

4-1) Creating the Class

Select the created project and right-click to display the context menu. Select [New] > [Class] from the context menu. The [New Java Class] wizard is displayed.

Check and enter the following setup items. After the following information is set, click [Finish].

Setup items	Setup content
Source folder	JPASample/src
Package	sample
Name	PopulationRanking

4-2) Implementing the Logic Class

The Entity is managed by the javax.persistence.EntityManager class. Add the constructor set in the argument to the javax.persistence.EntityManagerFactory class that creates this class. Since the PopulationRanking class is used by the Web application servlet class, this argument is passed from the servlet class. Add the following code to the PopulationRanking class.

Constructor Addition

```

package sample;

import javax.persistence.EntityManager;
import javax.persistence.EntityManagerFactory;

```

```

public class PopulationRanking {
    private EntityManager em;

    public PopulationRanking(EntityManagerFactory factory) {
        this.em = factory.createEntityManager();
    }
}

```

Point

After the required implementation is coded, right-click on the Java editor, then select [Source] > [Organize Imports] from the context menu. The import statement that is appropriate for the classpath that was set in the project can then be inserted.

Add the method that fetches from the C_DATA table the total population ranking that corresponds to the ID. To fetch the CountryData class that is mapped to the C_DATA table, use the find method of the EntityManager instance. Since the table data is stored in order, starting with the largest total population, the rank and ID are the same value.

Add the following code to the PopulationRanking class.

Addition of Method for getting Total Population Ranking

```

public CountryData getCountryData(int rank) {
    if (rank < 1 || rank > 10) {
        return null;
    }
    CountryData country = em.find(CountryData.class, rank);
    return country;
}

```

5) Creating the EAR Project

Select [File] > [New] > [Project] from the menu bar. The [New Project] is displayed.

Select [Java EE] > [Enterprise Application Project] from the New Project wizard, then click [Next].

Check and enter the following setup items:

Setup items	Setup content
Project name	JPAEARSample
Target Runtime	Interstage Application Server V11.1 IJServer Cluster (Java EE)
EAR version	5.0
Configuration	Default Configuration for Interstage Application Server V11.1 IJServer Cluster (Java EE)

Set the information, then click [Next]. The [Enterprise Application] page is displayed.

Check and enter the following setup items. After the following information is set, click [Finish].

Setup Items	Setup Content
Java EE Module Dependencies	JPASample
Content Directory	EarContent
Generate deployment descriptor	Do not select.

6) Creating the Web Application

Conventionally, procedures to create a Web application as a client are needed, but with Interstage Studio a sample is imported and, if JPA is used, the important points are checked. If you want to know the detailed procedure for creating a Web application, refer to "2.2 Introduction" in "Developing Web Applications".

Use the following procedure to import a sample.

Select [File] > [Import] from the menu. Select [General] > [Existing Projects into Workspace] from the displayed Import wizard. Select [Select archive file], then click the [Browse] button and select the following files:

<Workbench installation folder>\sample\JPASample.zip

Select [JPAClientSample] from [Projects], then click [Finish].

After the JPAClientSample project has been imported, select the JAPEARSample project. Right-click to display the context menu and select [Properties]. Select [Java EE Module Dependencies] from the Properties dialog box. Select JPAClientSample and add the client to the EAR.

6-1) Creating the Dynamic Web Project

Create a dynamic Web project as a client. Since this can be completed by importing a sample, details are omitted. Refer to the following setup items.

Setup items	Setup content
Project name	JPAClientSample
Target Runtime	Interstage Application Server V11.1 IJServer Cluster (Java EE)
Configuration	Default Configuration for Interstage Application Server V11.1 IJServer Cluster (Java EE)
Add project to an EAR	Select
EAR Project name	JPAEARSample

6-2) Setting the Build Path

Set the build path (classpath) used to reference the Entity class (CountryData) and the logic class (PopulationRanking). This is already set in the imported sample project.

To check the settings, select the created project (JPAClientSample), and right-click. Select [Properties] from the context menu. Select [Java EE Module Dependencies] from the Properties dialog box to display the [Java EE Modules] tab.

Check that the following setting is selected.

Setup item	Setup content
JAR/Module	JPASample.jar

6-3) Creating the Servlet Class

Create the servlet class. The servlet class is already created in the imported sample project with the following setup items.

Setup items	Setup content
Java package	sample
Class name	ServletController
Superclass	javax.servlet.http.HttpServlet
Which method stubs would you like to create?	Constructors from superclass Inherited abstract methods doPost (remove the doGet check)

6-4) Specifying the Dependency Injection

The implementation for invoking the logic class is performed in the servlet class. This is already implemented in the servlet source of the imported sample project.

To create the PopulationRanking class, javax.persistence.EntityManagerFactory is required in the constructor. Create the EntityManagerFactory fields, then set the PersistenceUnit annotation and specify the Dependency Injection as shown below.

PersistenceUnit Annotation Definition

```
@PersistenceUnit
private EntityManagerFactory emf;
```

Point

.....

Instead of the lookup method used up to J2EE1.4, Dependency Injection is used in Java EE and can be used to fetch and use an object. (During execution, the object is automatically set in a field by means of a Java EE container.)

.....

6-5) Implementing the Server Class

Create a logic class instance like the one shown below and invoke the method. This is already implemented in the servlet source of the imported project.

Entity Invocation Method Definition

```
PopulationRanking ranking = new PopulationRanking(emf);
CountryData country = ranking.getCountryData(rank);
```

The implemented servlet class is as shown below. (The imported project servlet source is in this format).

The parts in **red** are the added definitions.

Servlet Class (ServletController.java)

```
package sample;

import java.io.IOException;

import javax.persistence.EntityManagerFactory;
import javax.persistence.PersistenceUnit;
import javax.servlet.RequestDispatcher;

import javax.servlet.ServletContext;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

/**
 * Servlet implementation class ServletController
 */
public class ServletController extends javax.servlet.http.HttpServlet {

    private static final long serialVersionUID = 1L;

    //@PersistenceUnit
    @PersistenceUnit
    private EntityManagerFactory emf;

    /**
     * @see HttpServlet#HttpServlet()
     */
}
```

```

*/
public ServletController() {
    super();
    // TODO Auto-generated constructor stub
}

/**
 * @see HttpServlet#doPost(HttpServletRequest request, HttpServletResponse response)
 */
protected void doPost(HttpServletRequest request, HttpServletResponse response) throws
ServletException, IOException {
    // TODO Auto-generated method stub
    request.setCharacterEncoding("ISO-8859-1");
    ServletContext sc = getServletContext();

    // Get the type of file to be invoked.
    String mode = request.getParameter("mode");
    if (mode != null && mode.equals("top")) {
        // Invoke the Input page HTML file.
        RequestDispatcher inRd = getServletContext().getRequestDispatcher("/default.jsp");
        inRd.forward(request, response);
        return;
    }

    int rank;

    // Input check
    try{
        rank = Integer.parseInt(request.getParameter("rank"));
    } catch (NumberFormatException e){
        RequestDispatcher errRd = sc.getRequestDispatcher("/error.jsp");
        errRd.forward(request, response);
        return;
    }

    PopulationRanking ranking = new PopulationRanking(emf);
    CountryData country = ranking.getCountryData(rank);

    // Check the obtained value.
    if(country == null){
        RequestDispatcher errRd = sc.getRequestDispatcher("/error.jsp");
        errRd.forward(request, response);
        return;
    }

    sc.setAttribute("ranking", country);

    RequestDispatcher outRd = sc.getRequestDispatcher("/result.jsp");
    outRd.forward(request, response);
}
}

```

Point

After the required implementation is coded, right-click on the Java editor, then select [Source] > [Organize Imports] from the context menu. The import statement that is appropriate for the classpath that was set in the project can then be inserted.

6-6) Creating the I/O Page

Create the I/O page JSP. A file like the following is already created in the imported sample project.

Input Page (default.jsp)

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/
html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
<title>World ranking for total population</title>
</head>
<body>
<h1>Input page</h1>
<p> Enter a ranking from 1 to 10. </p>
<p></p>
<form action="ServletController" method="post">
    Rank :<br>
    <input name="rank" type="text" size="10">Rank<br>
    <p></p>
    <input type="submit" value="OK">
    <input type="reset" value="Clear">
</form>

</body>
</html>
```

Output Page (result.jsp)

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/
html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
<title>World ranking for total population</title>
</head>
<body>
<h1>Output page</h1>

Total population ranking ${param.rank}rank is "$
{applicationScope.ranking.countryName}"<br>
Total population is ${applicationScope.ranking.totalPopulation} people.
<br>
<hr>
<form action="ServletController" method="post">
    <input type="submit" value="Back to input page">
    <input type="hidden" name="mode" value="top">
</form>

</body>
</html>
```

Error Page (error.jsp)

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>
```

```

<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/
html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
<title>World ranking for total population</title>
</head>
<body>

<h1>Error page</h1>
The entered ranking is not in the specified range, or a server error occurred. <br>
<br>
<hr>
<form action="ServletController" method="post">
  <input type="submit" value="Back to input page">
  <input type="hidden" name="mode" value="top">
</form>

</body>
</html>

```

7) Setting the Interstage Application Server JDBC

From the Interstage Java EE Admin Console, set the JDBC so that the Oracle database can be used from the Java EE container.

Check separate documentation concerning how to use the Java EE execution environment of Interstage Application Server for details of how to launch and use the Interstage Java EE Admin Console.

7-1) Creating the JDBC Console Pool

Create an Oracle JDBC Connection Pool. Select [Resources] > [JDBC] > [Connection Pools] from the left frame to display the [Connection Pools] page.

Click [New...], then enter the following setup items.

Setup items	Setup content
Name	OraclePool
Resource Type	javax.sql.ConnectionPoolDataSource
Database Vender	Blank

Set the information, then click [Next].

Enter the following setup item in the next page.

Setup item	Setup content
Datasource Classname	oracle.jdbc.pool.OracleConnectionPoolDataSource

Add properties in the last line at [Additional Properties]. To add the properties, click [Add Property]. When input of the following setup items is finished, click [Finish].

Setup items	Setup content
User	Specify the User ID used to access the database.
Password	Specify the password that corresponds to the user name.
PortNumber	1521
networkProtocol	tcp
DatabaseName	COUNTRYDATA

Setup items	Setup content
serverName	myhost
DriverType	thin

7-2) Creating the JDBC Resource

Create the JDBC Resource that corresponds to the created JDBC Console Pool. From the left frame, select, [Resources] > [JDBC] > [JDBC Resource] to display the [JDBC Resources] page. Click [New] to display the [New JDBC Resource] page.

Enter the following setup items.

Setup items	Setup content
JNDI Name	jdbc/Oracle
Pool Name	OraclePool
Targets(Selected Targets)	MyDebugJEE

7-3) Setting the JDBC Driver Classpath

Set the Oracle JDBC driver in the classpath. From the left frame, select [Configurations] > [MyDebugJEE-config] > [JVM Settings] to display the [JVM General Settings] page. Click the [Path Settings] tab and enter the following settings.

Enter the following setup items.

Setup items	Setup content
Server Classpath	<JDBC driver storage destination directory>\ojdbc6.jar
Native Library Path Suffix	<JDBC driver storage destination directory>

8) Checking Application Operation

8-1) Associating the Project with a Server

Select a server in the Servers view, then select [Add and Remove Projects] from the context menu.

Select the created EAR project from the usable projects, then click [Add]. Check that the EAR project has been added to the configured projects.

Check the following setup item, then click [Finish].

Setup item	Setup content
Configured projects	JPAEARSample



.....
 If the deployment destination server in the Servers view is not registered, then it must be added. For details on how to do that, refer to "6.2.6 Checking Application Behavior".

8-2) Setting Breakpoints

Set a breakpoint at the first line of the getCountryData() method of the logic class. To set or delete a breakpoint, double-click the ruler part on the left edge of the editor.

8-3) Launching the Server

Select a server in the Servers view, then select [Debug] from the context menu.

Point

If you want to simply launch the application without using the debugger, select [Start] from the context menu.

The EAR file is deployed automatically before the server starts.

Check the following information in the Servers view:

Item to check	Contents to check
Server state	Started
Server status	Synchronized
EAR file (JPAEARSample) state	Synchronized
State of the JAR file (JPASample) under the EAR file	Synchronized
State of the WAR file (JPAClientSample) under the EAR file	Synchronized

8-4) Executing the Application

In the Servers view, select the deploying Web project (JPAClientSample), then from the context menu, select [Web browser]. The Web browser is started, and the input window is opened.

Setup item	Setup content
URL of input page	http://localhost/JPAClientSample/

Enter a population ranking in the [Rank] input field, then click [OK].

Point

The user can select the project, then select [Run As] > [Run on Server] or [Debug As] > [Debug on Server] from the context menu to add the server, add the project, launch the server, and launch the client all together.

In addition, the Web browser used as the client can be changed by selecting [Window] > [Web Browser] from the menu.

8-5) Debugging the Application

The applications runs and is interrupted at the breakpoints. Check the Variables view display. Select [Run] > [Step Over] from the menu and check the program state in the Variables view. For details, refer to "6.2.6.1 Debugging" for debugging details.

Point

In addition to checking values, values can be changed in the Variables view.

Processing that has been interrupted by the debugger can be restarted by selecting [Run] > [Resume] from the menu. This allows the processing on the server side to complete so that the output page is displayed on the Web browser. Check that the country name and total population for the entered ranking is displayed.

9) Distributing the Application to the Operating Environment

An EAR file must be created in order to distribute the application to the operating environment. The created EAR file uses the server distribution function to distribute the application to the server.

9-1) Exporting the Application

The EAR file is created from the Export wizard. To launch the Export wizard, select [File] > [Export]. From [Export] wizard, select [Java EE] > [EAR file].

The EAR file Export wizard is displayed.

Check and enter the following setup items. After the following information is set, click [Finish].

Setup items	Setup content
EAR project	Specify the enterprise application project.
Destination	Specify the creation destination for the EAR file.

9-2) Deploying to the Operating Environment

The application can be deployed from a remote environment to the operating environment by logging in to the Interstage Java EE Admin Console at the operating environment and by specifying a local EAR file.

4.3 Tasks

This section describes how to develop an application that uses JPA using Interstage Studio. It explains each of the development work topics (tasks) separately.

- [4.3.1 Preparing the Environment to create Applications that use JPA](#)
- [4.3.2 Developing Persistence Units](#)
- [4.3.3 Creating Entity Classes](#)
- [4.3.4 Operating Database with JPA](#)
- [4.3.5 Checking Operation of Applications that use JPA](#)
- [4.3.6 Distributing Applications that use JPA to the Operating Environment](#)

4.3.1 Preparing the Environment to create Applications that use JPA

Applications that use JPA can be created in the following ways, depending on the provision format:

- Library
Select this method if the program that uses JPA is provided as a component.
- Embedded in EJB or a Web application
Select this method if used only by EJB or a Web application and if the interface of the program that uses JPA is not strictly prescribed.

After the provision format is determined, create the project and set the required library in the project, then prepare an environment that enables the build.



Point

Support functions for constructing a database connection environment, for generating an Entity class from a database tables, and for other tasks can be used when developing applications that use JPA.

For details, refer to "[8.3.1 Connecting to the Database](#)" for details of constructing a database connection environment. The connection information for the constructed database is used in the setup of information concerning JPA. For details, refer to "[4.3.1.1 Setting the JPA Facet](#)" for details.

Creating the Project

Use the Project wizard to create the project that corresponds to the application that uses JPA. The Project wizard includes a [Project Facets] page. On this page, select the required [Java Persistence] project facet.

Projects are described below.

- Web application
If the JPA is to be used in a Web application, create a dynamic Web project. Select [Web] > [Dynamic Web Project] from the [New

Project] wizard.

For details, refer to "[2.3.1 Preparing the Environment for Creating the Web Application](#)" for details of the Dynamic Web project wizard.

- EJB application

If the JPA is to be used in an EJB application, create an EJB project. Select [EJB] > [EJB Project] from the [New Project] wizard.

For details, refer to "[3.3.1 Preparing the Environment to create EJBs](#)" for details of the EJB project wizard.

- Library

If the program that uses JPA is to be provided as a library, create a JPA project. Select [JPA] > [JPA Project] from the [New Project] wizard.

For details, refer to "[6.2.2.1 Creating New Projects](#)" for details of the common parts of the Project wizards.

If the [Java Persistence] project facet was selected, information related to JPA can be set. For details, refer to "[4.3.1.1 Setting the JPA Facet](#)" for details of the settings for JPA-related information.



.....
If you want to use JPA in an existing EJB project or dynamic Web project, add [Java Persistence] from [Project Facets] in the project properties, then set the JPA-related information.

For details, refer to "[4.3.1.1 Setting the JPA Facet](#)" for details of the settings for JPA-related information.
.....

Classpath Settings

Classpaths must be set if there are libraries or similar that are required in order to create the application. Classpaths are set in the project build path.

For details, refer to "[6.2.2.3 Setting Classpaths](#)" for details of build path settings.

4.3.1.1 Setting the JPA Facet

If the [Java Persistence] project facet is added to a project, use the Project wizard or the Facet Addition wizard to set the following JPA-related information:

- Connection

Database connection can be set in the project. However, an application that uses JPA can be created even if the database connection is not set.

For details, refer to "[8.3.1 Connecting to the Database](#)" for details of database connection.

- JPA implementation

Specify the JPA implementation class. Select whether the implementation provided by the server runtime or the implementation library is used.

- Persistent class management

Select whether management is achieved by coding the Entity class in persistence.xml, or whether the annotation class is automatically detected without the Entity class being coded in persistence.xml.

- Create orm.xml

Select whether or not an O/R mapping file is created.



.....
Connection and persistent class management can also be specified from [JPA] under the project properties.
.....

4.3.2 Developing Persistence Units

To develop a persistence unit, define persistence.xml, multiple persistence classes (Entity classes and so on), and O/R mapping files. Code the data source used by the persistence unit and explicitly declare persistence classes and O/R mapping files in persistence.xml. An O/R mapping file is used to code information for persistence classes that do not use annotation, or to overwrite information that was coded in Java source using annotations.

Point

The orm.xml file created during project creation is an O/R mapping file that does not need to be explicitly declared in persistence.xml. A persistence class declaration is also not usually required in persistence.xml if annotation is used for development, unless the class is stored in a different JAR file, for example.

Due to the above specifications, persistence unit development consists of using annotation to create the persistence class, and editing persistence.xml and O/R mapping files.

For details, refer to "4.3.3 Creating Entity Classes" for the method for using annotation to create a persistence class.

persistence.xml Editing

To edit persistence.xml, use a Persistence XML editor.

The Persistence XML editor has [General], [Connection], [Properties] and [Source] tabs. From the [Source] tab, direct editing of XML files is possible and Contents Assist can be used to add tags.

In the [General], [Connection] and [Properties] tabs, XML content can be edited using GUI.

For example, code persistence.xml as shown below in order to declare the data source used for the persistence unit.

persistence.xml Usage Example

```
<?xml version="1.0" encoding="UTF-8"?>
<persistence version="1.0" xmlns="http://java.sun.com/xml/ns/persistence"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="http://
java.sun.com/xml/ns/persistence http://java.sun.com/xml/ns/persistence/
persistence_1_0.xsd">
  <persistence-unit name="JPASample">
    <jta-data-source>jdbc/Oracle</jta-data-source>
  </persistence-unit>
</persistence>
```

O/R mapping File Editing

The XML editor is also used to edit O/R mapping files.

In addition, while the XML editor is open, if EntityMappings is selected in the [JPA Structure] view, the [JPA Details] view can be used to enter settings for a number of elements. For details, refer to below for settings details:

- Package
Specify the appropriate package for classes not specified by a full modifier in the mapping information coded in the XML file.

- Schema and Catalog
Specify the schema and catalog used as the default.

- Access
Specify whether the default for whether the persistence provider uses fields or uses properties to access the Entity.

Point

Elements under a persistence unit are suitable for persistence classes coded using annotation. Other items are suited to persistence classes coded using an O/R mapping file (orm.xml).

Validating as a JPA Application

Use the JPA application validator to validate the persistence unit.
For details, refer to "[6.2.5.2 Validation](#)" for details of validation.

4.3.3 Creating Entity Classes

To create an entity class, a class with an @Entity annotation, etc. set can be generated using the New JPA Entity wizard.

Point

An Entity class can also be generated from a database table.
For details, refer to "[4.3.3.4 Generating Entity Classes from Tables](#)" for details.

Opening the JPA Perspective

First open the JPA perspective.

Creating Entity Classes

Create entity classes using the [New] wizard, by selecting [JPA] > [Entity]. For the wizard settings, see below.

- Project
Specify the project that generates the entity class.

- Source folder
Specify the folder that stores the entity class source.

- Java package
Specify the package name of the entity class.

- Class name
Specify the name of the entity class.

- Inheritance
Select the class that is inherited by the entity class from "Entity" or "Mapped Superclass".

- XML Entity Mappings
If adding an entity to an O/R mapping file, select "Add to entity mappings in XML".

- Entity Name
Specify the entity name.

- Table Name
If associating with an existing table, deselect "Use default" and specify the table name.

- Entity Fields
Add an entity field. Specify and add the field name and type. For the field to be made the primary key, select "Key".
- Access Type
Specify the access type of the entity as "Field-based" or "Property-based".

Setting the Annotation

Activate the [JPA Structure] view and select the created Java class. In the [JPA Details] view, click the mapping type displayed in [Type]. In the [Mapping Type Selection] dialog box, when the mapping type is selected, the annotation is set in the entity class. An Entity suited to complicated data structures can also be created using inheritance, embedding, and so on. For details, refer to "[4.3.3.1 Creating an Entity taking Data Structures into Account](#)" for details.

Point

Java classes set as entity are not displayed in the JPA Structure view. Therefore, add the annotation definitions to the Java Source directly.

Entity Class and Database Mapping

To perform Entity class and database mapping, specify the database table name in the [Name] combo box under the [Table] group in the [JPA Details] view. The default can be used when mapping if the table has the same name as the class name.

Mapping can be performed for the Entity class fields and the database columns. Create fields in the Entity class, then select the fields in the [JPA Structure] view. Specify the database table name in the [Name] combo box under the [Column] group in the [JPA Details] view. The defaults can be used when mapping if the columns have the same names as the field names.

Point

When connected to the database that is set in the project, the [JPA Details] view combo box lists the mapping possibilities.

Database mapping can also be performed for things other than setting simple table or column associations. For details, refer to "[4.3.3.2 Mapping the Entity and Database](#)" for details of database mapping. In addition, relationships between Entities can be mapped by combining them with database information. For details, refer to "[4.3.3.3 Defining Relationships between Entities](#)" for details.

Point

Database mapping can also be performed using an O/R mapping file, not just by using annotation. Use an O/R mapping file to improve application portability if you want to use it in multiple environments, for example.

4.3.3.1 Creating an Entity taking Data Structures into Account

An Entity can be created by inheriting an Entity. In addition to an Entity, common persistence items can also be managed in a superclass, and these also can be created by inheritance. To create this superclass, set the @MappedSuperclass annotation. If an Entity has been inherited and created, the attributes of the inherited persistent items can be changed by overwriting, and the mapping method can be specified for the database tables of each of the inherited Entities. Use the @AttributeOverride annotation and the @Inheritance annotation for these purposes.

In addition, relevant persistence items can be grouped together and managed as a special class, and the special class can be embedded in the Entity by using it as the Entity field type. To create this special class, set the @Embeddable annotation and, if you want to embed this in the Entity, set either the @Embedded annotation or the @EmbeddedId annotation in the Entity field. If embedding is performed, the attributes of the persistence items in the embedded class can be changed by overwriting.

Setting MappedSuperclass Annotation

Select the superclass in the [JPA Structure] view. In the [JPA Details] view, click the mapping type displayed in [Type]. In the [Mapping Type Selection] dialog box, when "Mapped Superclass" is selected, the @MappedSuperclass annotation is set.

If using a Java Class, open the class file in a Java editor and define the annotations directly.

Setting AttributeOverride Annotation

Select the Entity class in the [JPA Structure] view. At [Attribute Overrides] in the [JPA Details] view, select the persistence item that you want to overwrite and overwrite the attribute.

Setting Inheritance Annotation

Select the hierarchy root Entity class in the [JPA Structure] view. For [Inheritance] in the [JPA Details] view, specify the method of mapping to the Entity database table.

For details, refer to below for the settings:

- Strategy
Select from the following table mapping methods:
 - Single Table
All Entities that have an inheritance relationship are mapped in a single table. For this method, the table requires a column (discriminator column) for the purpose of indicating which data of the Entity class that has an inheritance relationship is in each row of the database table.
 - JOINED
Each Entity class unit is mapped to a table. For each table, the column that corresponds to the persistence items (not including inherited persistence items) specific to that Entity class, and the join column, are defined. The purpose of the join column is to set the association with the primary key of the table (the primary table) corresponding to the hierarchy root Entity.
 - Table per Class
Each Entity class unit is mapped to a table. The column corresponding to the persistence items (including inherited persistence items) specific to that Entity class is defined.
- Discriminator Column, Discriminator Type, and Discriminator Value
Specify these items if "Single Table" is selected as the method. The discriminator value sets the value used to identify the Entity classes that have an inheritance relationship.
These items are equivalent to the @DiscriminatorColumn and the @DiscriminatorValue annotations.
- Primary Key Join Columns
Specify this item if "JOINED" is selected as the method. Specify the pair of columns referenced from the primary key column of the current Entity and the primary key column of the primary table.
This item is equivalent to the @PrimaryKeyJoinColumn annotation.

Setting Embeddable Annotation

Select the Java class in the [JPA Structure] view. In the [JPA Details] view, click the mapping type displayed in [Type]. In the [Mapping Type Selection] dialog box, when "Embeddable" is selected, the @Embeddable annotation is set.

If using a Java Class, open the class file in a Java editor and define the annotations directly.

Setting Embedded or EmbeddedId Annotation

In the [JPA Structure] view, select the persistence items of the class type for which Entity class @Embeddable annotation has been set. In the [JPA Details] view, click the mapping type displayed in [Attribute]. In the [Mapping Type Selection] dialog box, when "Embedded" or "Embedded Id" is selected, the @Embedded annotation or @EmbeddedId annotation is set in the persistence item.

If "Embedded" is selected, the attributes of the class for which the @Embeddable annotation was set can be changed by overwriting using [Attribute Overrides] in the [JPA Details] view.

4.3.3.2 Mapping the Entity and Database

Use the `@Table` annotation to perform mapping between the Entity and the database table. In addition, an Entity can be mapped to multiple databases by using the `@SecondaryTable` annotation.

When an Entity and a table are mapped, names are used to set the associations between the persistence items and the columns. However, database columns can also be mapped to persistence item units. The persistence item to be used as the primary key must be declared.

In addition to specifying information for mapped columns, users can also specify the data read (fetch) method, type-related options, non-persistence items, and items used for Optimistic Locking.

Setting Table Annotation

In the [JPA Structure] view, select the Entity class. Under [Table] in the [JPA Details] view, specify the table name, catalog, and schema.

Setting SecondaryTable Annotation

In the [JPA Structure] view, select the Entity class. Under [Secondary Tables] in the [JPA Details] view, add tables other than the primary table (the table specified by the `@Table` annotation).

For [Primary Key Join Columns], specify which column of the table added as a secondary table is associated with the primary key of the primary table.

Mapping Persistence Items and Columns

In the [JPA Structure] view, select the Entity class persistence item. At [Column] in the [JPA Details] view, specify the column name and table.

The [Insertable] and [Updatable] items specify whether the column corresponding to that persistence item is an insert target or an update target.

Setting the Primary Key

In the [JPA Structure] view, select the persistence item to be used as the primary key of the Entity class. In the [JPA Details] view, select "Id" in the mapping type displayed in [Attribute]. The `@Id` annotation is set in the field.

If you want the primary key value to be generated automatically rather than being explicitly set within the program, select [Primary Key Generation] in the above [JPA Details] view. For details, refer to below for the settings.

This item corresponds to the `@GeneratedValue`, `@TableGenerator`, and `@SequenceGenerator` annotations.

- Strategy
Select one of the following primary key generation methods:
 - Auto
The execution environment selects the generation method that is appropriate for the database.
 - Identity
The database IDENTITY column is used.
 - Sequence
The database sequence is used.
 - Table
The database table is used.
- Generator Name
Specify the name specified for the table generator or sequence generator.
- Table Generator
The name is the name specified for the generator name. A value column is a column that stores values generated by the generator.

The value of the primary key column specifies the value used to identify which generator generated the table when this table is used to generate the primary keys of multiple tables. (This value and the last generated primary key value are stored as a pair in the primary key column and the value column.)

- Sequence Generator

The name is the name specified for the generator name. For the sequence, specify the database sequence name.

Specifying Fetch Method

While the persistence item is in the selected state at the [JPA Structure] view, specify [Fetch] in the [JPA Details] view to specify the fetch method.

For details, refer to below for details:

- EAGER

Specify Eager if the item must be fetched when the Entity is fetched.

- LAZY

Specify Lazy if it is okay for the item to be fetched the first time the persistence item is accessed.

Specifying Date and Time Options

For time-related type persistence items, users can specify whether the association is with Date, Time, or the Timestamp. Use the @Temporal to specify which is associated.

To set the @Temporal annotation, select the time-related type persistence value in the [JPA Structure] view, then select the association method in the [Temporal] combo box of the [JPA Details] view.

Specifying Enumeration-related Options

For enumeration persistence items, users can specify whether the value is handled as an Ordinal or a String. Use the @Enumerated annotation to specify how it is handled.

To set the @Enumerated annotation, select the handling method in the [Enumerated] combo box of the [JPA Details] view.

Specifying Large Object Options

If the persistence item type is the type associated with the database large object type (LOB), users can specify whether the item is handled as a large object using the @Lob annotation.

To set the @Lob annotation, select the persistence item associated with the large object type in the [JPA Structure] view, then select [Lob] in the [JPA Details] view.

Specifying Non-persistence Items

The fields and properties declared in an Entity class are handled as persistence items by default. For non-persistence items, the @Transient annotation must be set.

To set the @Transient annotation, in the [JPA Structure] view, select the items not for persistence, and in the [JPA Details] view, for the mapping type displayed in [Attribute], select "Transient".

Specifying Items that Apply Optimistic Locking

The exclusion method used to check that a value has not changed when data is read and when values are updated is known as Optimistic Locking. The @Version annotation must be set for items that apply this format.

To set the @Version annotation, in the [JPA Structure] view, select the item, and in the [JPA Details] view, for the mapping type displayed in [Attribute], select "Version".

4.3.3.3 Defining Relationships between Entities

The relationship between Entities can be either one-to-one, many-to-one, one-to-many, or many-to-many. The relationships have directionality, and can be either bidirectional enabling referencing from either direction, or unidirectional allowing referencing from only one side.

To define the relationship between Entities, the Entity type of the side that is referenced, its collection type field, and its properties are defined in the Entity, and the relationship type is set using either the `@OneToOne`, the `@ManyToOne`, the `@OneToMany`, or the `@ManyToMany` annotation.

Defining a One-to-one or a Many-to-one Relationship

To the referencing side Entity, add the Entity type field or property of the referenced side Entity. Select the items added in the [JPA Structure] view, and in the [JPA Details] view, for the mapping type displayed in [Attribute], select "One to One" or "Many to One", then define the relationship. For details, refer to below for the relationship definitions:

- Target Entity
Specification is not required because the target can be discerned from the field or property type.

- Fetch
Select one of the following as the timing for fetching the reference destination Entity:
 - EAGER
Specifies that fetch is essential when the Entity is fetched.

 - LAZY
Specifies that it is okay to fetch at the time of first access.

- Mapped By
If the relationship is bidirectional, specify the owning side persistence item in the owned side definitions.
In a one-to-one relationship, the side that has the external key is the owning side.
In a many-to-one relationship, the multi side is the owning side, so this definition cannot be specified.

- Cascade
Select the persistence processing (method) that cascades to the reference destination. When making the selection, take into account, for example, when an Entity itself is deleted, whether or not the deletion is performed as far as the reference destination.

- Join Columns
For one-to-one and many-to-one relationships, an external key is used to define the relationship. Therefore, specify the external key column name and the reference destination column name that it references, or similar, as the join column.
This item is equivalent to the `@JoinColumn` annotation.

Defining a One-to-many or a Many-to-many Relationship

To the referencing side Entity, add the Entity collection type field or property of the referenced side Entity. Select the items added in the [JPA Structure] view, and in the [JPA Details] view, for the mapping type displayed in [Attribute], select "One to Many" or "Many to Many", then define the relationship.

For details, refer to below for the relationship definitions. (The definitions that are the same as for one-to-one or many-to-one are omitted.)

- Target Entity
If the collection type is declared and a generic name is used, specification is not required because the target can be discerned from the field or property type. If a generic name is not used, specify the reference destination Entity type.

- Mapped By
If the relationship is one-to-many, the many side is the owning side. Therefore, specify this definition if the directionality is bidirectional.
For a bidirectional many-to-many relationship, decide which is the owning side and specify this definition at the owned side.

- Cascade
In the reference destination entity, select the persistence process (method) to be cascaded. For example, select with consideration to

whether deletion is to be extended to the reference destination if the entity itself is deleted.

- Order By

Specify the reference destination Entity sequence to use when fetch is performed.
This item is equivalent to the @OrderBy annotation.

- Join Table

For one-to-many and many-to-many relationships, the join table is used to define the relationship. The join table must have corresponding columns at the owning side and the owned side. Specify the table columns corresponding to the Entity referenced from these column names and columns.
This item is equivalent to the @JoinTable annotation.

4.3.3.4 Generating Entity Classes from Tables

If the database connection has been set in the project, the following procedure can be used to generate an Entity class from the database table:

1. Use the [Data Source Explorer] view to connect to the database that was set in the project.
For details, refer to "8.3.1 Connecting to the Database" for details of connecting to the database.
2. Right-click the project, then select [JPA Tools] > [Generate Entities].
3. Select the table from the displayed [Generate Entities] dialog box, then enter the source folder or package.

4.3.4 Operating Database with JPA

With JPA, an Entity is operated by the Entity manager, and this performs database operations. Thus, an Entity manager is required. Operations equivalent to database searches, insertions, updates, and deletions can be performed by using the fetched Entity manager to operate the Entity.



.....
Strictly speaking, the Entity manager performs operations in relation to the persistence context (the accumulation of Entities constructed within memory), and synchronization with the database is performed during breaks in transactions.
.....

Fetching the Entity Manager

Use the following type of @PersistenceContext annotation to fetch the Entity manager.

Example of using the PersistenceContext annotation

```
@PersistenceContext
private EntityManager em;
```

Persistence Context Operations Performed by the Entity Manager

An example of persistence context operations performed by the Entity manager is shown below.

Searching using the Entity Primary Key

In this example, the ID (primary key) from the EMPLOYEE table is specified to search for an employee. The Entity class and the primary key value are specified in the Entity manager find method to perform the search.

Example of a Search using the Entity Primary Key

```
public Employee getEmployeeByPrimarykey (int id) {
    return em.find(Employee.class, id);
}
```


Searching using the Entity Java Persistence Query Language

In this example, a search is performed for the employee that matches a name from the EMPLOYEE table. A query is created by the Entity manager createQuery method and the parameters are set, then the query is executed and the search results are fetched.

Example of a Search using the Entity Java Persistence Query Language

```
public Collection<Employee> getEmployeeByName (String name) {
    Query query = em.createQuery("SELECT employee FROM Employee employee
WHERE employee.name = :name");
    query.setParameter("name" ,name);
    return query.getResultList();
}
```

Inserting an Entity

In this example, a new employee is added to the EMPLOYEE table. An Entity class (Employee) instance is created, and the Entity manager persist method adds it to the persistence context.

Example of inserting an Entity

```
public void insertEmployee (int id, String name, String address, String
tel) {
    Employee newEmployee = new Employee(id, name, address, tel);
    em.persist(newEmployee);
}
```

Updating an Entity

In this example, the ID (primary key) from the EMPLOYEE table is specified to search for an employee, and the address and phone number are changed. The Entity manager find method performs the search, then the Entity class values are changed.

Example of updating an Entity

```
public void updateEmployee (int id, String address, String tel) {
    Employee employee = em.find(Employee.class, id);
    employee.setAddress(address);
    employee.setTel(tel);
}
```

Point

In order to make a change within the persistence context, simply change the Entity instance and the change will be detected automatically. If an instance is outside the persistence context (for example, if the return value instance returned by the above getEmployeeByPrimarykey method is to be updated), the Entity manager merge method must be invoked to post the change to the persistence context.

Deleting an Entity

In this example, the ID (primary key) from the EMPLOYEE table is specified to search for an employee, and the information of that employee is deleted. The Entity class instance searched for using the Entity manager find method is specified in an argument and the remove method is invoked to delete it from the persistence context.

Example of Deletion using the Entity Primary Key

```
public void deleteEmployee (int id) {
    Employee employee = em.find(Employee.class, id);
    em.remove(employee);
}
```

4.3.5 Checking Operation of Applications that use JPA

To check the operation of an application that uses JPA, it must be deployed to the execution environment, then executed. Perform these operations in the Servers view.

For details, refer to "[6.2.6 Checking Application Behavior](#)" for operation details.

4.3.6 Distributing Applications that use JPA to the Operating Environment

To distribute the created Java EE application to the operating environment, create an archive file, then deploy the application from the Interstage Java EE Admin Console.

Creating an Archive File

Use the Export wizard to create the archive file.

For details, refer to "[6.2.7 Distributing to the Operating Environment](#)" for details of operations using the Export wizard.

Deploying from the Interstage Java EE Admin Console

The Interstage Java EE Admin Console can be operated from the remote environment. Log in to the Interstage Java EE Admin Console of the operating environment, and specify a local file to be deployed. This enables deployment from a remote environment to the operating environment.

Refer to separate documentation concerning the method for using the Java EE execution environment of Interstage Application Server for deployment and Interstage Java EE Admin Console details.

Chapter 5 Developing Web Service Applications

This chapter presents an overview of Web services and describes how to create Web service applications and Web service client applications that run in a Java EE application execution environment.

5.1 Overview

This section presents an overview of Web services and the Web service development support functions.

5.1.1 What is a Web Service?

A Web service is technology that enables public access to system functions as a service via a network. Web services are being acclaimed as a method that promotes efficient utilization of existing systems and reuse of software.

Use of SOAP, using HTTP for low-level protocol, for communications and use of WSDL to define the interfaces of published services have become the standards for implementing such Web services.

In addition, interoperability is an important aspect of Web services, and ambiguous constraints and Web service specifications (SOAP/WSDL/UDDI and so on) are clarified by being prescribed by WS-I Basic Profile.

In practice, Web service provision is implemented by providing a Service Endpoint, and the Service Endpoint Interface is determined on the basis of the following conventions:

- SOAP

SOAP is a protocol for communication between objects via a network. Data structures that use XML for coding are prescribed, but the protocol used to send data is not prescribed and a variety of protocols, including HTTP and SMTP, can be used. However, in many cases, HTTP is generally used.

- JAX-WS

JAX-WS is a successor of the JAX-RPC prescribed for implementing application remote invocation (Remote Procedure Call) using Java. To code the functions published as Web services using Java, create them within the range of this convention.

- WSDL

WSDL is the coding method used to publish Web service specifications to users. Coding is in XML.

- WS-I Basic Profile 1.1

In terms of interoperability improvements, the WS-I Basic Profile 1.1 conventions prescribe more detailed restrictions concerning the standard Web service specifications of SOAP/WSDL/UDDI and so on. Users generally do not need to be aware of the content of conventions during processes involved in creating a Web service application using the workbench. However, users do need to be aware of conventions when directly creating and publishing WSDL.

- WS-I Attachments Profile 1.0

The WS-I Attachments Profile 1.0 conventions are based on SOAP Messages with Attachments, and clarify the SOAP and WSDL definition specifications and set limits to the usage range in order to improve the interoperability of Web services when file attachments are used. Web services that conform to WS-I Attachments Profile provide improved interoperability between systems when sending and receiving SOAP messages with attachments.



.....
The development method can be either top-down development, in which the Service Endpoint Interface is designed using WSDL, or bottom-up development, in which the Java class that will be the Service Endpoint is created and the Service Endpoint Interface is determined as a result of that.
.....

5.1.1.1 Modifications since J2EE1.4

The following specification differences exist between J2EE1.4 Web services and Java EE Web services:

- JAX-RPC and JAX-WS

Under J2EE1.4, Web services conformed to JAX-RPC, but Web service development that conforms to JAX-WS is recommended under Java EE.

- Files that configure a Web service

Under J2EE1.4, numerous files, such as an SEI (Service Endpoint Interface), implementation class, WSDL, mapping file, deployment descriptor, and so on, are required in order to create a Web service. However, under Java EE, a Web service can be created simply by declaring the @WebService annotation in the Java class.

- Web service invocation method

Under Java EE, Dependency Injection can be used for Web services and also to easily code conventional lookup processing.

Simple examples of the main annotations used during Web service development are shown below while introducing Web services under Java EE.

javax.jws.WebService Annotation

This annotation defines a class as a Web service. The following properties can be specified.

Property Name	Explanation
endpointInterface	Define the Service Endpoint Interface if you do not want to make all the class methods public, and use this property to declare it.
name	Name of the WSDL portType
portName	Name of the WSDL port
serviceName	Name of the WSDL service
targetNamespace	WSDL namespace
wsdlLocation	Use this property if a Web service is published based on an already defined WSDL.

Example of using the WebService Annotation to create a Web Service

```

package sample;

import javax.jws.WebService;

@WebService(endpointInterface="sample.Calc")
public class CalcImpl implements Calc {
    public int add(int param1,int param2) {
        return param1 + param2;
    }
}

```

javax.xml.ws.WebServiceRef Annotation

This annotation sets a Dependency Injection when invoking a Web service. The following properties can be specified.

Property Name	Explanation
mappedName	Mapped resource name
name	JNDI name
type	Java type of the resource
value	Service class Normally, javax.xml.ws.Service is inherited.
wsdlLocation	Specify the WSDL location.

Example of using the WebServiceRef Annotation to create a Web Service Client

```
@WebServiceRef(name="service/Calc")
private CalcService service;

public int callCalc(int param1,int param2) {
    Calc port = service.getCalcPort();
    return port.add(param1,param2);
}
```

5.1.2 Web Services Development

An overview of the development of the Web service is shown below.

Preparing for Web Service Development

A Web service can be developed by either of two methods: by creating it as a Web application or by creating it as an EJB application. Create a project in accordance with the selected method, and set the required target runtime, classpath, and other projects.

For details, refer to ["5.3.1 Preparing the Environment to create the Web Service"](#).

Creating Web Services

The method used to create a Web service can be either a bottom-up method or a top-down method.

With the bottom-up method, use the Java class wizard to create the Web service. With the top-down method, use the WSDL wizard, WSDL editor, and WSDL validator to create the Web service. After this, use the wizard to create the Service Endpoint Interface, and the Java class wizard to create the class that implements these.

For details, refer to ["5.3.2 Creating the Web Service"](#).

If creating the Web service as an EJB application, refer to ["5.3.3 Exposing a Stateless Session Bean as a Web Service"](#) for details.

Creating Web Service Clients

A wizard is provided for creating from WSDL the Service Endpoint Interface and other requirements for invoking a Web service.

For details, refer to ["5.3.6 Creating a Web Service Client"](#).

Validating Web Services

The Interstage Java EE validation function is provided in order to verify whether the development conforms to Java EE Web service specifications and whether the Web service will operate under Interstage Application Server.

For details, refer to ["6.2.5.3 Interstage Java EE Validator"](#).

Checking Web Service Operation

Use the Servers view to check the operation of the created application. Also provided are a Web service explorer that can invoke a Web server, and a TCP/IP monitor that can check Web service messages.

For details, refer to ["5.3.7 Checking Web Service Operation"](#).

Distributing Web Services

The application must be archived in order for the Web service to be distributed. Use the Export wizard to create an archive file for the created application.

For details, refer to ["5.3.8 Distributing Web Services to the Operating Environment"](#).

5.2 Introduction

This section introduces the procedures for creating an application that invokes a Web service from a Web application.

5.2.1 Application to be created

Create a Web service application that returns the country name and total population from a ranking list of the populations of countries when the ranking is entered, and create an application that invokes that Web service from a Web application and displays the result on a

Web page.

In this case, the application created as described in the introduction of Web Applications is used as the Web service client application.

5.2.2 Development Flow

Development of the above Web service application proceeds as follows:

1. Create the project for the Web service

In order to create the Web service application, first create a dynamic Web application project. When the dynamic Web application project is created as directed by the wizard, the classpath and other settings required for the build are set automatically.

2. Create the Web service

Use the following procedure to create the Web service:

- Create the data class
Create the data class to be used to implement the Web service.

- Create the Web service pattern
Use the wizard to create the Java class.

- Declare the WebService annotation
Declare the annotation in the Java class created by the wizard.

- Implement the Web service
Declare the method in the class that declared the annotation to implement processing.

3. Create the EAR project

Create a project for creating an EAR file for distribution as one application including the EJB client. When creating the EAR project, specify to include the created dynamic Web project in the EAR file. Since the client is not yet created, specify to include the client in the EAR file when creating the client project.

4. Prepare to fetch the WSDL

Use the following procedure to prepare to fetch the WSDL required for client creation:

- Set the relation between the EAR project and the server
Set the relation with the server to enable deployment to the server so that the WSDL will be able to be fetched.

- Deploy to the server
Actually deploy to the server to make it possible to fetch the WSDL from the server.

- Check the URL of the WSDL
Check the URL to be used to fetch the WSDL.

5. Create the Web service client

Use the following procedure to create the Web application that will be the client:

- Create the client project
Create a dynamic Web project for creating a Web application as a client.

- Create the Service Endpoint Interface from the WSDL
Use the wizard to create from the WSDL the Service Endpoint Interface and other required files.

- Create the servlet class
Use the wizard to create a servlet as a controller for receiving Web application requests.
 - Create the I/O page
Use the wizard to create the Web application I/O page.
 - Specify the Dependency Injection
Set the Dependency Injection in the field such that the Web service method is invoked from the servlet.
 - Implement the Web service invocation processing
Implement the processing for invoking the Web service method using the field that declared the Dependency Injection.
6. Check the operation of the application
Use the following procedure to check the application operation:
- Set the relation between the project and a server
Set which server the application is deployed on. Since the settings are already completed in order to fetch the WSDL, nothing need be done here.
 - Set the breakpoints
Set the breakpoints for the debugger to check the program operation during execution.
 - Launch the server
Launch the server so that it can receive requests in relation to the application from Web browsers. Deployment is performed automatically before the server starts.
 - Execute the application
Launch the Web browser and access the application URL to start checking the operation of the application.
 - Debug the application
Debug the program and check that the application is operating correctly.
7. Distribute the application to the operating environment
Use the following procedures to distribute the application to the operating environment:
- Export the application
Create an EAR file to distribute the application to the operating environment.
 - Distribute to the operating environment
Deploy the EAR file from the Interstage Java EE Admin Console.

5.2.3 Development Procedures

The actual procedures used to develop the application are described below.

- 1) [Creating the Web Service Project](#)
- 2) [Creating the Web Service](#)
- 3) [Creating the EAR Project](#)
- 4) [Preparing to Get the WSDL](#)
- 5) [Creating the Web Service Client](#)

6) Checking the Application Operation

7) Distributing the Application to the Operating Environment

1) Creating the Web Service Project

Select [File] > [New] > [Project] from the menu bar. [New Project] is displayed.

Select [Web] > [Dynamic Web Project] from the [New Project] wizard, then click [Next].

Check and enter the following setup items:

Setup Items	Setup Content
Project name	WebServiceSample
Target Runtime	Interstage Application Server V11.1 IJServer Cluster (Java EE)
Dynamic Web Module version	2.5
Configuration	Default Configuration for Interstage Application Server V11.1 IJServer Cluster (Java EE)
Add project to an EAR	Do not select.

Set the information, then click [Next]. The [Web Module] page is displayed.

Check and enter the following setup items. After the following information is set, click [Finish].

Setup Items	Setup Content
Context Root	WebServiceSample
Content Directory	WebContent
Java Source Directory	src
Generate deployment descriptor	Check.

2) Creating the Web Service

2-1) Creating the Data Class

Hold the country name and total population information, and create a data class that gets information from there. To create the data class, select the created project, then right-click to display the context menu. Select [New] > [Class] from the context menu. The [New Java Class] wizard is displayed.

Check and enter the following setup items. After the following information is set, click [Finish].

Setup Items	Setup Content
Source folder	WebServiceSample/src
Package	sample
Name	CountryData
Constructors from superclass	Check

The following source file is generated:

Source file	Description
CountryData.java	Data class

Implement the processing for holding the country names and total populations. Add the places shown in **red** below to the source.

Data Class Implementation (CountryData.java)


```

package sample;

public class CountryData {

    private String countryName;
    private int totalPopulation;

    public CountryData() {
        // TODO Auto-generated constructor stub
    }

    public CountryData(String name, int total) {
        setCountryName(name);
        setTotalPopulation(total);
    }

    public String getCountryName() {
        return countryName;
    }

    public void setCountryName(String countryName) {
        this.countryName = countryName;
    }

    public int getTotalPopulation() {
        return totalPopulation;
    }

    public void setTotalPopulation(int totalPopulation) {
        this.totalPopulation = totalPopulation;
    }

}

```

Point

After the fields are added, and while the class is in the selected state, [Source] > [Generate Getters and Setters] can be selected from the menu to add getter/setter.

2-2) Creating the Web Service Pattern

The Web service is performed from creating a Java class. Select the created project, then right-click to display the context menu. Select [New] > [Class] from the context menu. Creation of a special Java class is not required. Specify the source folder, package, and name to create the class.

Check and enter the following setup items. After the following information is set, click [Finish].

Setup Items	Setup Content
Source folder	WebServiceSample/src
Package	sample
Name	PopulationRanking

2-3) Declaring the WebService Annotation

In the created Java class, define the WebService annotation as shown below. The parts in **red** are the WebService annotation definition.

WebService Annotation Definition

```

package sample;

import javax.jws.WebService;

@WebService
public class PopulationRanking {
}

```

2-4) Implementing the Web Service

Implement the method of the function you want to publish as a Web service. Add the places shown in **red** below to the source.

Implementation of the Method published as the Web Service

```

package sample;

import javax.jws.WebService;

@WebService
public class PopulationRanking {

    private CountryData[] countries;

    public PopulationRanking(){
        countries = new CountryData[]{
            new CountryData("China",1330000000),
            new CountryData("India",1140000000),
            new CountryData("The United States",3000000000),
            new CountryData("Indonesia",2300000000),
            new CountryData("Brazil",1900000000),
            new CountryData("Pakistan",1600000000),
            new CountryData("Bangladesh",1500000000),
            new CountryData("Russia",1400000000),
            new CountryData("Nigeria",1400000000),
            new CountryData("Japan",1300000000)
        };
    }

    public CountryData getCountryData(int rank){

        --rank;
        if(9 < rank || rank < 0 ){
            return null;
        }

        return countries[rank];
    }
}

```

3) Creating the EAR Project

Select [File] > [New] > [Project] from the menu bar. The [New Project] is displayed.

Select [Java EE] > [Enterprise Application Project] from the New Project wizard, then click [Next].

Check and enter the following setup items:

Setup Items	Setup Content
Project name	WebServiceEARSample

Setup Items	Setup Content
Target Runtime	Interstage Application Server V11.1 IJServer Cluster (Java EE)
EAR version	5.0
Configuration	Default Configuration for Interstage Application Server V11.1 IJServer Cluster (Java EE)

Set the information, then click [Next]. The [Enterprise Application] page is displayed.

Check and enter the following setup items. After the following information is set, click [Finish].

Setup Items	Setup Content
Java EE Module Dependencies	WebServiceSample
Content Directory	EarContent
Generate deployment descriptor	Not Checked

4) Preparing to Get the WSDL

4-1) Associating the EAR Project and Server

Select the server in the Servers view, then select [Add and Remove Projects] from the context menu.

Check and enter the following setup items. After the following information is set, click [Finish].

Setup Items	Setup Content
Configured projects	WebServiceEARSample

4-2) Deploying to the Server

Select the server in the Servers view, then select [Start] from the context menu.

The EAR file is automatically deployed when the server starts.

Check the following information in the Servers view.

Items to Check	Contents to Check
Server state	Started
Server status	Synchronized
Status of the EAR file (WebServiceEARSample)	Synchronized
Status of the WAR file under the EAR file (WebServiceSample)	Synchronized

4-3) Checking the URL for the WSDL

Use the Interstage Java EE Admin Console to check the URL of the WSDL.

In the Servers view, select a server, then from the context menu, select [Admin Console] to start the Interstage Java EE Admin Console.

For details on how to use the Interstage Java EE Admin Console, separately check the details on how to use the Interstage Application Server Java EE execution environment.

From the tree on the left side of the Interstage Java EE Admin Console, click [Web Service] > [Web Service Class Name (PopulationRanking)]. Next, click the [View] of [WSDL] button displayed on the right side of the window. The WSDL created using the Interstage Application Server is displayed.

Check the URL of the displayed WSDL. Use this URL when creating the client.

URL of the WSDL

`http://localhost/WebServiceSample/PopulationRankingService?wsdl`

5) Creating the Web Service Client

Conventionally, procedures to create a Web application as a client are needed, but with this product a sample is imported and the important points for a Web service client are checked. If you want to know the detailed procedure for creating a Web application, refer to "2.2 Introduction" in the Web application chapter.

Use the following procedure to import a sample.

Select [File] > [Import] from the menu. Select [General] > [Existing Projects into Workspace] from the displayed Import wizard. Select [Select archive file], then click the [Browse] button and select the following files:

<Workbench installation folder>\sample\WebServiceSample.zip

Select [WebServiceClientSample] from [Project], then click [Finish].

After the WebServiceClientSample project has been imported, select the WebServiceEARSample project. Right-click to display the context menu and select [Properties]. Select [Java EE Module Dependencies] from the Properties dialog box. Select WebServiceClientSample.war and add the client to the EAR.

5-1) Creating the Client Project

Create a dynamic Web project as a Web service client. Since this can be completed by importing a sample, details are omitted. Refer to the following table for the settings.

Setup Items	Setup Content
Project name	WebServiceClientSample
Target Runtime	Interstage Application Server V11.1 IJServer Cluster (Java EE)
Dynamic Web Module version	2.5
Configuration	Default Configuration for Interstage Application Server V11.1 IJServer Cluster (Java EE)
Add project to an EAR	Check
EAR Project Name	WebServiceEARSample

5-2) Creating the Service Endpoint Interface from the WSDL

The Service Endpoint Interface and other files required to invoke the Web service must be created from the WSDL. These are created by the imported sample project as shown below.

Select [Web Services] > [Web Service(JAX-WS)] from the [New] wizard. The [New Web Service(JAX-WS)] wizard creates the files. Set the source folder for importing files and the URL of the WSDL file fetched from the Interstage Application Server in Step 4).

If deploying a Web service as an EAR, the package of the service endpoint interface must be modified.

Check and enter the following setup items. After the following information is set, click [Finish].

Setup Items	Setup Content
Source folder	WebServiceClientSample/src
WSDL file name	Set the URL fetched from the Interstage Application Server.
It changes from the package for which the package of the source generated automatically is specified in WSDL	Check
Package	stub

Check that the following files that are in the file name list have been created under the source folder.

File Name	Notes
PopulationRanking.java	Service Endpoint Interface
PopulationRankingService.java	Service interface
CountryData.java	Data definition class
GetCountryData.java	Data binding class
GetCountryDataResponse.java	Data binding class
ObjectFactory.java	Object generation class
package-info.java	Package information class



In order to refer the WSDL definitions and so on, the WSDL file specified in the Web service (JAX-WS) wizard and XML Schema file referred by WSDL file are automatically obtained from the reference destination, and copied to "wsdl" folder under the project.

5-3) Creating the Servlet Class

Create the servlet class. Creation is completed simply by setting the following setup items in the imported sample project.

Setup Items	Setup Content
Java package	sample
Class name	ServletController
Superclass	javax.servlet.http.HttpServlet
Which method stubs would you like to create?	Constructor from the superclass Inherited abstract methods doPost (remove the doGet check)

5-4) Creating the I/O Page

Create the I/O page JSP. Files like the following are already created in the imported sample project.

Input Page (default.jsp)

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/
html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
<title>World ranking for total population</title>
</head>
<body>
<h1>Input page</h1>
<p> Enter a ranking from 1 to 10.</p>
<p></p>
<form action="ServletController" method="post">
    Rank:<br>
    <input name="rank" type="text" size="10">Rank<br>
    <p></p>
    <input type="submit" value="OK">
    <input type="reset" value="Clear">
</form>
```

```
</body>
</html>
```

Output Page (result.jsp)

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/
html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
<title>World ranking for total population</title>
</head>
<body>
<h1>Output page</h1>

Total population ranking ${param.rank} rank is "$
{applicationScope.ranking.countryName}".<br>
Total population is ${applicationScope.ranking.totalPopulation} people.
<br>
<hr>
<form action="ServletController" method="post">
    <input type="submit" value="Back to input page">
    <input type="hidden" name="mode" value="top">
</form>

</body>
</html>
```

Error Page (error.jsp)

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/
html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
<title>World ranking for total population</title>
</head>
<body>

<h1>Error page</h1>
The entered ranking is not in the specified range, or a server error occurred.<br>
<br>
<hr>
<form action="ServletController" method="post">
    <input type="submit" value="Back to input page">
    <input type="hidden" name="mode" value="top">
</form>

</body>
</html>
```

5-5) Specifying the Dependency Injection

The implementation for invoking the Web service is performed in the servlet class. This is already implemented in the servlet source of the imported sample project.

Create a business interface field, set the following WebServiceRef annotation, and set the Dependency Injection specifications.

WebServiceRef Annotation Definition

```
@WebServiceRef(name="service/PopulationRanking")
private PopulationRankingService service;
```

Point

.....

Instead of the lookup method used up to J2EE1.4, Dependency Injection is used in Java EE and can be used to fetch and use an object. (During execution, the object is automatically set in a field by means of a Java EE container.)

.....

5-6) Implementing the Web service Invocation Processing

To invoke the Web service, fetch the object to the Service Endpoint Interface as shown below, then invoke the method via the interface. The Web service invocation processing is defined as shown below in the created servlet class. This is already implemented in the servlet source of the imported sample project.

Web Service Invocation Method Definition

```
PopulationRanking business = service.getPopulationRankingPort();
CountryData country = business.getCountryData(rank);
```

The servlet class in which Web service invocation is implemented is as shown below. (The imported project sample source is in this format). The parts in **red** are the added and changed places.

Servlet Class (ServletController.java)

```
package sample;

import java.io.IOException;
import javax.servlet.RequestDispatcher;
import javax.servlet.ServletContext;

import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import javax.xml.ws.WebServiceRef;

import stub.CountryData;
import stub.PopulationRanking;
import stub.PopulationRankingService;

/**
 * Servlet implementation class ServletController
 */
public class ServletController extends HttpServlet {

    static final long serialVersionUID = 1L;

    @WebServiceRef(name="service/PopulationRanking")
    private PopulationRankingService service;

    /**
     * @see HttpServlet#HttpServlet()
     */
    public ServletController() {
        super();
        // TODO Auto-generated constructor stub
    }
}
```

```

/**
 * @see HttpServlet#doPost(HttpServletRequest request, HttpServletResponse response)
 */
protected void doPost(HttpServletRequest request, HttpServletResponse response) throws
ServletException, IOException {

    request.setCharacterEncoding("ISO-8859-1");
    ServletContext sc = getServletContext();

    // Get the type of file to be invoked.
    String mode = request.getParameter("mode");
    if (mode != null && mode.equals("top")) {
        // Invoke the Input page HTML file.
        RequestDispatcher inRd = getServletContext().getRequestDispatcher("/default.jsp");
        inRd.forward(request, response);
        return;
    }

    int rank;

    // Input check
    try{
        rank = Integer.parseInt(request.getParameter("rank"));
    } catch (NumberFormatException e){
        RequestDispatcher errRd = sc.getRequestDispatcher("/error.jsp");
        errRd.forward(request, response);
        return;
    }

    PopulationRanking business = service.getPopulationRankingPort();
    CountryData country = business.getCountryData(rank);

    // Check the obtained value.
    if(country == null){
        RequestDispatcher errRd = sc.getRequestDispatcher("/error.jsp");
        errRd.forward(request, response);
        return;
    }

    sc.setAttribute("ranking", country);

    RequestDispatcher outRd = sc.getRequestDispatcher("/result.jsp");
    outRd.forward(request, response);
}
}

```

6) Checking the Application Operation

6-1) Associating the Project with a Server

No action is required because this has already been performed during preparations to create the client.

6-2) Setting Breakpoints

Set a breakpoint at the first line of the `getCountryData()` method of the Web service class. To set or delete a breakpoint, double-click the ruler part on the left edge of the editor.

6-3) Launching the Server

Select a server in the Servers view, then select [Debug] from the context menu.

Point

- If you want to simply launch the application without using the debugger, select [Start] from the context menu.
- If the deployment destination server in the Servers view is not registered, then it must be added. For details on how to do that, refer to "6.2.6 Checking Application Behavior".

The EAR file is deployed automatically before the server starts.

Check the following information in the Servers view:

Items to Check	Contents to Check
Server state	Started
Server status	Synchronized
EAR file (WebServiceEARSample) status	Synchronized
Status of the WAR file (WebServiceSample) under the EAR file	Synchronized
Status of the WAR file (WebServiceClientSample) under the EAR file	Synchronized

6-4) Executing the Application

In the Servers view, select the deploying Web project (WebServiceClientSample), then from the context menu, select [Web browser]. The Web browser is started, and the input window is opened.

Setup Items	Setup Content
URL of input screen	http://localhost/WebServiceClientSample/

Enter a population ranking in the [Rank:] input field, then click [OK].

Point

The user can select the project, then select [Run As] > [Run on Server] or [Debug As] > [Debug on Server] from the context menu to add the server, add the project, launch the server, and launch the client all together.

In addition, the Web browser used as the client can be changed by selecting [Window] > [Web Browser] from the menu.

6-5) Debugging the Application

The applications runs and is interrupted at the breakpoints. Check the Variables view display. Select [Run] > [Step Over] from the menu and check the program state in the Variables view.

For debug details, refer to "6.2.6.1 Debugging".

Point

In addition to checking values, values can be changed in the Variables view.

Processing that has been interrupted by the debugger can be restarted by selecting [Run] > [Resume] from the menu. This allows the processing on the server side to complete so that the output page is displayed on the Web browser. Check that the country name and total population for the entered ranking is displayed.

7) Distributing the Application to the Operating Environment

An EAR file must be created in order to deploy the application to the operating environment. The created EAR file uses the server deployment function to deploy the application to the server.

7-1) Exporting the Application

The EAR file is created from the Export wizard. To launch the Export wizard, select [File] > [Export]. From [Export] wizard, select [Java EE] > [EAR file].

The [EAR Export] wizard is displayed.

Check and enter the following setup items. After the following information is set, click [Finish].

Setup Items	Setup Content
EAR project	Specify the enterprise application project.
Destination	Specify the creation destination for the EAR file.

7-2) Distributing to the Operating Environment

The application can be deployed from a remote environment to the operating environment by logging in to the Interstage Java EE Admin Console at the operating environment and by specifying a local EAR file.

5.3 Tasks

This section describes how to develop a Web service application using Interstage Studio. It explains each of the development work topics (tasks) separately.

- [5.3.1 Preparing the Environment to create the Web Service](#)
- [5.3.2 Creating the Web Service](#)
- [5.3.3 Exposing a Stateless Session Bean as a Web Service](#)
- [5.3.4 Creating WSDL](#)
- [5.3.5 Creating a Service Endpoint Interface from WSDL](#)
- [5.3.6 Creating a Web Service Client](#)
- [5.3.7 Checking Web Service Operation](#)
- [5.3.8 Distributing Web Services to the Operating Environment](#)

5.3.1 Preparing the Environment to create the Web Service

Two methods can be used to create a Web service. Use either the method for creating it as a Web application, or the method for creating it as an EJB application, as shown below.

- Providing a Web service using a Web application
Select this method if you want to create a simple Web service and if the interface of the Web service must be strictly defined and WSDL must be used for the interface design.
For details, refer to "[5.3.2 Creating the Web Service](#)".
- Providing a Web service using an EJB application
Select this method if you want to also use the provided function as EJB and if it is easier to develop and maintain a Web service if the application structure is an EJB module.
For details, refer to "[5.3.3 Exposing a Stateless Session Bean as a Web Service](#)".

If the provision format has been determined, create the project, then prepare an environment that allows the required libraries to be set in the project and enables the build.

Creating the Project

If the Web service is to be provided as a Web application, select [Web] > [Dynamic Web Project] from the [New Project] wizard and create a dynamic Web project.

If the Web service is to be provided as an EJB application, select [EJB] > [EJB Project] from the [New Project] wizard and create an EJB project.

For details, refer to "[2.3.1 Preparing the Environment for Creating the Web Application](#)" or "[3.3.1 Preparing the Environment to create EJBs](#)".

Classpath Settings

Classpaths must be set if there are libraries or similar that are required in order to create the application. Classpaths are set in the project build path.

For details on build path settings, refer to "[6.2.2.3 Setting Classpaths](#)".

5.3.2 Creating the Web Service

The following approaches can be used to create a Web service as a Web application:

- Expose a Java class as a Web service
This is the simplest way to create a Web service. This method is easy if you simply want to create a Web service and there are no additional constraints.
- Creating a Web service from WSDL
Select this method if, for example, the interface of the Web service must be strictly defined and WSDL must be used for the interface design.

5.3.2.1 Exposing a Java Class as a Web Service

In order to expose a Java class as a Web service, the Java class must first be created. Naturally, an existing Java class can also be made into a Web service.

The Web service can then be created by declaring the `@WebService` annotation in the Java class and implementing the service you want to publish.

Creating the Java Class

To create the Java class, select [Class] from the [New] wizard. Creation of a special Java class is not required. Specify the source folder, package, and name to create the class.

Declaring the WebService Annotation

Code the `@WebService` annotation in the created Java class as shown below.

Example of using the WebService Annotation

```
package sample;

import javax.jws.WebService;

@WebService
public class Calc {
    public int add(int param1,int param2) {
        return param1 + param2;
    }
}
```

Implementing the Service

Define the method of the function you want to publish as a Web service, and implement the method.

5.3.2.2 Creating a Web Service from WSDL

In order to create a Web service from WSDL, the WSDL must first be created. Naturally, existing WSDL can also be used to create a Web service.

Then, the Web server can be created by creating a Web server that conforms to the WSDL and implementing the declared method.

Creating the WSDL

Select [Web Services] > [WSDL] from the [New] wizard and the WSDL can be created by the wizard. In addition, a special-purpose editor can be used to edit the WSDL.

For details on how to create and edit WSDL, refer to "[5.3.4 Creating WSDL](#)".

Creating a Web Service that conforms to the WSDL

In order to create a Web service that conforms to the WSDL, a Service Endpoint Interface must first be created from the WSDL. The Service Endpoint Interface can be created by selecting [Web Services] > [Web Service(JAX-WS)] from the [New] wizard.

For details on creating the interface using the wizard, refer to "[5.3.5 Creating a Service Endpoint Interface from WSDL](#)".

If the Service Endpoint Interface has been created, use the Java class wizard to create the class that implements the Service Endpoint Interface.

Select [Class] from the [New] wizard. To create the class that implements the Service Endpoint Interface, specify the following to generate it.

- Package and name
Specify the package and the class name of the Web service to be created.

- Interfaces
Specify the Service Endpoint Interface.

Code the @WebService annotation in the created Java class as shown below.

Example of using the WebService Annotation

```
package sample;

import javax.jws.WebService;

@WebService(endpointInterface="sample.Calc", wsdlLocation="CalcService.wsdl")
public class CalcImpl implements Calc {
    public int add(int param1,int param2) {
        return param1 + param2;
    }
}
```

Implementing the Method (Service)

Implement the declared method.

5.3.3 Exposing a Stateless Session Bean as a Web Service

In order to expose a stateless session bean as a Web service, the stateless session bean must first be created in an EJB project. Naturally, an existing stateless session bean coded according to EJB 3.0 specifications can also be exposed as a Web service.

Then, the Web service can be exposed by declaring the @WebService annotation in the stateless session bean.

Creating the Stateless Session Bean

Create the stateless session bean in an EJB project.

For details on the creation method, refer to "[3.3.1 Preparing the Environment to create EJBs](#)" and "[3.3.2 Creating Session Beans](#)".

Declaring the WebService Annotation

Code the @WebService annotation in the stateless session bean as shown below.

Example of using the WebService Annotation

```
package sample;

import javax.jws.WebService;
```

```

import javax.ejb.Stateless;

@WebService
@Stateless
public class Calc {
    public int add(int param1,int param2) {
        return param1 + param2;
    }
}

```

5.3.4 Creating WSDL

The New wizard and a special-purpose editor are provided for creating WSDL. A WSDL validator is also provided and can check the validity of a WSDL file when, for example, it is saved.

WSDL Wizard

When [Web Services] > [WSDL] is selected from the [New] wizard, the wizard can create the WSDL. Refer to the following for the wizard settings:

- Target namespace and Prefix
Specify the specifying namespace of the WSDL and its prefix.
- Protocol and SOAP Binding Options
To use a form that conforms to WS-I Basic Profile, specify SOAP as the protocol and either document literal or RPC literal as the soap binding option.
The document literal option is better suited to program communication. Selection of the widely used document literal option is recommended unless special reasons apply.
Encoded RPC was the widely used form before WS-I Basic Profile 1.0 was established. Select this option if it is necessary to connect using encoded RPC, for example, when connecting to systems that use earlier SOAP services.

WSDL Editor

The WSDL editor has a [Design] tab and a [Source] tab.

At the [Source] tab, the WSDL can be edited directly as an XML file.

At the [Design] tab, the context menu, the properties views, and the outline view can be used to graphically edit the following elements that comprise the WSDL:

- Services
When a service is added and while in the initial state, a port is added.
- Port
When the service is selected, a port can be added. When a port is added, specify the binding and the protocol.
Select the port to set the binding. An existing binding can be used, or a new binding can be created and set.
- Bindings
A binding can be created in advance, regardless of the port.
Select the binding to set the port type. An existing port type can be used, or a new port type can be created and set.
- Port type
A port type can be created in advance, regardless of the binding.
When a port type is created and while in the initial state, one operation is added.
- Operation
When the port type is selected, an operation can be added. Also, when an operation is selected, fault can be added.
Messages can be set in relation to input, output, and fault. Existing message can be used, or a new message can be created and set.

- Messages
Messages can be created in advance regardless of the operation.
When a message is created and while in the initial state, one part is added.
- Parts
When a message is selected, a part can be added.
Select a part to set the type or element settings. An existing type or element can be used or new ones can be created and set.
- Imports
Import can be added in the outline view.
- Types
Editing can also be performed from the parts settings. In the outline view, a schema under a type can be opened and the schema can be edited.

WSDL Validator

The validator can check whether or not the WSDL conforms to WS-I Basic Profile 1.1. The results of the validity check are displayed in the Problems view. Or, when the editor is used for editing, marks are attached on the editor. For validator details, refer to "[6.2.5.2 Validation](#)".



Note

The Web service-related wizards provided by Interstage Studio do not perform WSDL validation even if the WSDL validation option ([Select whether WSDL validation should be performed on wizards that consume WSDL files]) is enabled at the above setup page.

5.3.5 Creating a Service Endpoint Interface from WSDL

A Service Endpoint Interface is required if a Web service is invoked or if a Web service is created from WSDL. Create the Service Endpoint Interface from WSDL.

Creating the Service Endpoint Interface

From the [New] wizard, select [Web Services] > [Web Service(JAX-WS)]. The wizard creates the Service Endpoint Interface and so on. Refer to the following concerning the Web service (JAX-WS) wizard.

- WSDL file name
Specify the WSDL file. Either a URL or a local file location can be specified.
When a client that invokes a Web service is being created, and if the WSDL does not need to be held as a development resource, the URL of a published WSDL can also be specified. Conversely, if the Web service is created from the WSDL, and if the WSDL is held as a development resource, a local file can also be specified.



Point

The WSDL file specified in the Web service (JAX-WS) wizard and XML Schema file referred by WSDL file are automatically obtained from the reference destination, and copied to "wsdl" folder under the project. The copied WSDL file is for reference use, therefore do not define it in the Web service (JAX-WS) wizard.



Note

In the case below, a WSDL inspection error may occur on the WSDL file which was obtained through Web service (JAX-WS) wizard and copied in the project.

- Obtaining a WSDL file by executing Web service (JAX-WS) wizard, updating Web service application (such as adding a service), and then obtaining the updated WSDL file by executing Web service (JAX-WS) wizard again.

In this case, the XML Schema file referred by the WSDL file is registered in the workbench cache when obtaining the first WSDL file. If the updated WSDL file is obtained by executing Web service (JAX-WS) wizard again, then the inspection is performed using the cached XML Schema in this WSDL inspection, and therefore an error occurs due to mismatching between the WSDL file and XML Schema.

To solve this issue, follow the procedure below:

1. From workbench menu, select [Window] > [Preferences].
2. The [Preferences] dialog box is displayed. Select [General] > [Network Connections] > [Cache].
3. The [Cache] page is displayed. From [Cache entries], select the XML Schema file referred by the WSDL file, and click [Remove].
4. From workbench menu, select [Project] > [Clean].
5. The [Clean] dialog box is displayed. Select the project which the error is happening, then click [OK].

5.3.6 Creating a Web Service Client

To create a client that invokes a Web service, a Service Endpoint Interface and a service interface are required. These required files can be created from WSDL.

After the required files are prepared, the Dependency Injection can be defined and so on, objects can be associated with the interfaces, and methods can be invoked via the interfaces.

Creating the required Files

The Service Endpoint Interface and so on can be created by the wizard. Select [Web Services] > [Web Service(JAX-WS)] from the [New] wizard.

For details on the creation method, refer to "[5.3.5 Creating a Service Endpoint Interface from WSDL](#)".

Getting the Object to the Interface

Objects can be obtained easily by means of a Dependency Injection.

To invoke a Web service, use the @WebServiceRef annotation as shown below.

Example of using the WebServiceRef Annotation

```
@WebServiceRef(name="service/Calc")
private CalcService service;
```

Coding Example	Explanation
service/Calc	Specify the name associated with the service.
CalcService	This is the service interface.



Note

The Dependency Injection can be used only in Java EE components managed in Java EE containers, such as servlets, EJB, and so on.

For other classes, use JNDI lookup to get objects.

For details, refer to "[10.2.2 Using JNDI Lookup to Obtain Objects](#)".

Invoking the Web Service

To invoke the Web service, get the object to the Service Endpoint Interface as shown below, then invoke the method via the interface.

Example of Invoking the Web Service

```
Calc port = service.getCalcPort();
int result = port.add(100,200);
```

Explanation Item	Explanation
Calc	This is the Service Endpoint Interface.
service	The object fetched by the Dependency Injection or JNDI lookup is set in the service interface field.
port.add(100,200)	The method published by the Service Endpoint Interface is invoked. The add method is published by the Calc interface.

5.3.7 Checking Web Service Operation

To check the operation of the Web service, the Web service application must be deployed to the execution environment, then executed. Perform these operations in the Servers view.

For details on operations, refer to "6.2.6 Checking Application Behavior".

In addition, the following support functions are available for checking operation:

- Web Service Explorer
If the arguments of the method published as a Web server and the return values are the primitive type or other simple type, rather than creating a client for invoking the Web service, the Web service can be invoked from the Web service explorer and the results can be checked.
For details, refer to "5.3.7.1 Using the Web Service Explorer as a Web Service Client".
- TCP/IP Monitor
A TCP/IP Monitor can be used to monitor the exchange of SOAP data between the Web service client and the Web service.
For details, refer to "5.3.7.2 Using the TCP/IP Monitor to check Web Service Messages".

5.3.7.1 Using the Web Service Explorer as a Web Service Client

If the arguments of the method published as a Web server and the return values are the primitive type or other simple type, the Web service can be invoked from the Web service explorer and the results can be checked.

Launching the Web Service Explorer

To launch the Web service explorer, select [Run] > [Launch the Web Services Explorer] from the menu.

Displaying the WSDL Page

The Web service explorer has UDDI, WSIL, and WSDL pages, and users can switch between these pages using the right icon button. Click the [WSDL Page] button to display the WSDL page.

Invoking the Web Service

Use the following procedures to invoke the Web service from the WSDL page of the Web service explorer:

1. Select WSDL Main from the Navigator on the left side.
2. Specify the URL of the WSDL in the Actions at upper-right, then click the [Go] button.
3. The WSDL structure is displayed in the Navigator on the left side. Select the operation that you want to check.
4. Enter the operation argument in the Actions at upper-right, then click the [Go] button.
5. The result of invoking the Web service is displayed in the Status at lower-right.



Web service messages can be checked directly by selecting the source in the Actions at upper-right or the Status at lower-right.



Note

The UDDI registry can be referred to on the UDDI page of the Web service explorer. However, this is not normally used because the registry function is not supported by Interstage Application Server.

5.3.7.2 Using the TCP/IP Monitor to check Web Service Messages

The TCP/IP monitor is placed between the Web service and the Web service client, and can be used to monitor SOAP data exchanges.

TCP/IP Monitor Setup

TCP/IP monitor setup is performed from [Run/Debug] > [TCP/IP Monitor] on the Preferences page.

Refer to the following for the settings:

- Local monitoring port
Specify the port number used for access to the TCP/IP monitor from the client.

- Host name
Specify the name of the host operating on the server.

- Port
Specify the port number operating on the server.

Launching the TCP/IP Monitor

To launch the TCP/IP monitor, select the monitor on the [TCP/IP Monitor] preferences page, then click the start button.

Changing the Client Side Connection URL

Since the client side needs to access the Web service via the TCP/IP monitor, the connection URL may need to be changed. If the @WebServiceRef annotation is being used, use wsdlLocation as shown below to specify the TCP/IP monitor URL.

Example of using the WebServiceRef Annotation

```
@WebServiceRef(wsdlLocation="http://localhost:8888/websv/CalculateService")  
private CalcService service;
```

Using the TCP/IP Monitor to Check Messages

When messages are sent from the client to the TCP/IP monitor while the TCP/IP monitor is in the started state, these request messages and the corresponding response messages are displayed on the TCP/IP monitor.

To clear the messages, click the clear button on the right of the TCP/IP monitor.

Validating the WS-I Message Log File

The following procedures can be used to check whether or not the messages monitored by the TCP/IP monitor are valid as WS-I Basic Profile messages:

1. Select send request at the TCP/IP monitor.
2. Click the Validate WS-I Message Log File button on the right of the TCP/IP monitor.
3. Specify the message save destination in the displayed wizard.
4. If required, specify include for the WSDL document.
5. Use the WS-I message validator to check the validity of the messages.

For validator details, refer to "[6.2.5.2 Validation](#)".

5.3.8 Distributing Web Services to the Operating Environment

To distribute the created Java EE application to the operating environment, create an archive file, then deploy the application from the Interstage Java EE Admin Console.

Creating an Archive File

Use the Export wizard to create the archive file.

For details on operations using the Export wizard, refer to "[6.2.7 Distributing to the Operating Environment](#)".

Deploying from the Interstage Java EE Admin Console

The Interstage Java EE Admin Console can be operated from the remote environment. Log in to the Interstage Java EE Admin Console of the operating environment, and specify a local file to be deployed. This enables deployment from a remote environment to the operating environment.

Refer to separate documentation concerning the method for using the Java EE execution environment of Interstage Application Server for deployment and Interstage Java EE Admin Console details.



Note

.....

The Web service client uses a URL to access the Web service. After the Web service has been deployed to the operating environment, the access URL must be changed at the Web service client side. Refer to separate documentation concerning the method for using the Java EE execution environment of Interstage Application Server for details of changing the URL.

.....

Chapter 6 Items Common to Java EE 5 Applications

This chapter describes the common items involved in creating applications that run in a Java EE 5 application execution environment.

6.1 Overview

This section provides an overview of Java EE 5 and of development functions that are common to Java EE 5 applications.

6.1.1 What is Java EE 5?

Java EE 5 is an abbreviation of "Java Platform, Enterprise Edition 5". It is a standard proposed for enterprise systems by Sun Microsystems, Inc. (now Oracle Corporation) related to Java platforms. To the Java SE set of standard functions, Java EE adds a set of functions intended for Servlets, EJB, and other server applications used for the development of business systems. Application servers implement handling as components, various services, communication methods, and so on, in accordance with these conventions as a Java EE 5 platform, thereby providing technology for creating multi-tiered applications that combine reusable components.

Under Java EE 5, applications are provided as EAR (Enterprise ARchive) files. Web applications, EJB, and other Java EE 5 modules are archived and included in the EAR file. A mutual reference relationship can be set between the modules within an EAR file. By deploying a Java EE 5 application included in an EAR file to a Java EE 5 platform, it can operate without the need to set classpaths and so on.

Modifications since J2EE1.4

The application development method centering on EJB and Web service applications has changed considerably. In addition, innovations such as the ability to use Dependency Injection using annotation have greatly changed resource and object referencing methods. Deployment as an EAR file means that deployment descriptors (application.xml) are no longer required.

These changes have made it simple to develop applications.

6.1.2 Developing Java EE 5 Applications

An overview of Java EE 5 application development is given below.

Preparing for Java EE 5 Application Development

To develop a Java EE 5 application, first an enterprise application project must be created. Create the project, then add the Java EE 5 module to the project. For details, refer to "[6.2.2.2 Creating Enterprise Applications](#)".

Validating Java EE 5 Applications

Users can verify whether or not an application has been developed in accordance with Java EE 5 specifications. For details, refer to "[6.2.5.3 Interstage Java EE Validator](#)".

Checking Java EE 5 Application Operation

Use the Servers view to check the operation of the created Java EE 5 application.

For details, refer to "[6.2.6 Checking Application Behavior](#)".

Distributing Java EE 5 Applications

In order to distribute a Java EE 5 application file, it must be archived in an EAR file. Use the Export wizard to create the archive file.

For details, refer to "[6.2.7 Distributing to the Operating Environment](#)".

6.2 Tasks

This section describes the common items involved in developing Java EE 5 applications using Interstage Studio. It explains each of the development work topics (tasks) separately.

- [6.2.1 Preparing the Deployment Destination for Verifying Application Operation](#)
- [6.2.2 Preparing to Create Applications](#)

- [6.2.3 Creating the Java Class and the Interface](#)
- [6.2.4 Creating XML Files](#)
- [6.2.5 Detecting and Correcting Problems](#)
- [6.2.6 Checking Application Behavior](#)
- [6.2.7 Distributing to the Operating Environment](#)
- [6.2.8 Developing Java EE Applications Using Entity Beans](#)

6.2.1 Preparing the Deployment Destination for Verifying Application Operation

There are two deployment destinations that verify application operation: IJServer Cluster and the Interstage Java EE DAS service. The advantages of each are explained below, and notes are provided. Perform operation verification at the deployment destination that suits your environment and requirements.



Note

The use of both IJServer Cluster and the Interstage Java EE DAS service for operation verification is not recommended because care is required as indicated below. Use one or the other to perform operation verification.

- The Interstage Java EE DAS service is a mandatory service for launching IJServer Cluster. Therefore, operations such as stopping and restarting the Interstage Java EE DAS service also affect IJServer Cluster operation.
- IJServer Cluster advantages
 - Operation verification can be performed from a remote machine and under conditions that closely match the operating environment.
 - If the Interstage Management Service has not been started, a user account control dialog box is displayed to start the service when operation is first attempted, but is not displayed for subsequent operations.



Note

To use IJServer Cluster, refer to "[6.2.1.1 Creating an IJServer Cluster \(MyDebugJEE\)](#)" and create an IJServer Cluster.

- Interstage Java EE DAS service advantages
 - If the Interstage Application Server functions are installed, the Interstage Java EE DAS service is already created so, unlike IJServer Cluster, no work is required to create it.
 - Operation can be verified with reduced memory usage because the Interstage Java EE Node Agent service and IJServer Cluster need not be launched.



Note

- The functions of the operating system being used display the user account control dialog box at the time of server view operations such as [Connect(Debug Launch)/Login], [Connect/Login], and [Stop].
- Operation cannot be verified using the Interstage Java EE DAS service of a remote machine.
- If using the Interstage Java EE DAS service, select "server" when selecting an IJServer Cluster.

6.2.1.1 Creating an IJServer Cluster (MyDebugJEE)

The Interstage Java EE Admin Console is usually used to create IJServer Clusters. However, an IJServer Cluster to be used for debugging can be created by executing the command shown below. Refer to the "Interstage Application Server Java EE Operator's Guide" for details.

```
asadmin create-cluster --newinstances MyDebugJEEInstance --user %Admin_User% --passwordfile
%File_Name% MyDebugJEE
```

For details on the variable information, see below.

Variable Information	Description
%Admin_User%	Specify the admin user ID name.
%File_Name%	Specify the file where the admin password was described. <ul style="list-style-type: none"> - "/" or "\" characters can be used as the file path separator of the password file Example: c:/password.txt - The format of the password file is as follows: AS_ADMIN_PASSWORD=password



Note

To execute commands, the following is required:

- A path must be set to the "bin" folder under the "F3FMisjee" folder of the Interstage Application Server folder.
Example: c:\Interstage\APS\F3FMisjee\bin
- Interstage Management Service must be started.

6.2.2 Preparing to Create Applications

In order to create the application, projects must be created to suit the modules being developed.

If an application is to comprise multiple modules, these can be grouped together as an enterprise application.

6.2.2.1 Creating New Projects

To launch the New project wizard, select [File] > [New] > [Project] from the menu. Select the project wizard that is appropriate for the module, as shown below.

Category	Project	Explanation
EJB	EJB Project	Used to create an EJB module.
Java EE	Application Client Project	Used to create an application client module.
	Enterprise Application Project	Used to create an EAR file. EAR files can group together EJB, Web applications, and libraries.
	Utility Project	Used to create libraries that are shared by multiple modules.
JPA	JPA Project	Used to create libraries that use JPA.
Web	Dynamic Web Project	Used to create Web application modules.

For the contents to specify in the project wizard, refer to the following:

- Project name
Specify the project name to be generated.
- Project contents
Specify the storage destination of the project resources. The default location is under the workspace folder.
- Target Runtime
Select the runtime that runs the Java EE application. For an Interstage Application Server Java EE container (IIServer Cluster), select

[Interstage Application Server V11.1 IJServer Cluster (Java EE)].

- Module version
For an EJB or Web application, etc., specify the module version.
- Configuration
Set the conventions and versions that the created modules and libraries conform to. Operate the support functions such as the wizard and editor, so that they conform to the conventions and versions. For an Interstage Application Server Java EE container (IJServer Cluster), select [Default Configuration for Interstage Application Server V11.1 IJServer Cluster (Java EE)].
- EAR Membership
If the modules and libraries are grouped together in an EAR file, select the enterprise application project. If the enterprise application is to be created later, it need not be selected here.

Point

When [Default Configuration for Interstage Application Server V11.1 IJServer Cluster (Java EE)] is specified in Configuration, an Interstage Application Server-specific deployment descriptor is generated when the project is created. For details on this file, refer to separate documentation concerning Interstage Application Server Java EE execution environment specifications.

Note

- When an EJB project is created without [Add project to an EAR] being selected under [EAR Membership], the Session Bean wizard business interface is created in the EJB project.
- When an EJB project is created with [Add project to an EAR] selected under [EAR Membership], the Session Bean wizard business interface is created in an EJB client project.
- When an EJB project is created with [Add project to an EAR] selected under [EAR Membership] and without [Create an EJB Client JAR module to hold the client interfaces and classes.] being selected, the Session Bean wizard business interface is created in the EJB project.

6.2.2.2 Creating Enterprise Applications

The enterprise application project can group together modules and libraries and create an EAR file.

Refer below for specifying contents specific to the enterprise application project wizard:

- Content Directory
Specify the name of the folder used to store files, apart from the Java EE module created as the project, that you want to include in the EAR file.

After the enterprise application project is created, if you want to change the Java EE module to be added to the EAR file, edit the project [Java EE Module Dependencies] property.

Point

- If an EAR file is used, in addition to enabling applications to be deployed as a group, the dependency relationship between the Java EE modules in the EAR file can be defined and the work involved in setting classpaths and other tasks at the operating environment can be reduced.
- When an application client is included in an enterprise application, the modules and libraries to be included in the enterprise application can be downloaded as client stubs from the Interstage Java EE Admin Console. If EJB application development and EJB client development are performed in separate environments, this function can be used to download the jar file containing the EJB interface required for EJB client development.

6.2.2.3 Setting Classpaths

In order to perform build for the Java application, set the build path (classpath) in the project. Use any of the following methods to set the build path for Java EE modules and libraries:

Target runtime

When the runtime that runs the Java EE application is selected, that library is added to the build path. This can be set in the [Targeted Runtimes] property of the project.

The library is added by the runtime name in the build path.

Manifest classpath of the enterprise application

The group of modules and libraries in the EAR file can reference each other's classes. This manifest classpath can be set in the [Java EE Module Dependencies] property. Set the reference target with the [Java EE Module Dependencies] property for each project.

The target is added to the build path EAR library.

WEB-INF/lib of the Web application

In the Web application module, the library can be put under WEB-INF/lib. This specification can be set in the [Java EE Module Dependencies] property. Set the reference target with the [Java EE Module Dependencies] property for each project.

The target is added to the build path Web application library.

Java build path

The build path can be set using the project [Java Build Path] property. A library added using this setting is not reflected in the operating environment. Therefore, before deployment to the application server, classpath settings may be required at the operating environment.



When a dynamic Web project is selected in the [Java EE Module Dependencies] property of an enterprise application project, the two tabs [Java EE Modules] and [Web Libraries] are displayed in the [Java EE Module Dependencies] property of the dynamic Web project. The manifest classpath of the enterprise application can be set at the [Java EE Modules] tab. The WEB-INF/lib of a Web application can be set at the [Web Libraries] tab.

6.2.2.4 Developing Using a Java EE Module that has no Development Resources

A Java EE application may be developed using a Java EE module that was developed by another person.

If an EAR file is to be created by an enterprise application project, the Java EE module must be imported and a Java EE module project must be created.

Select [File] > [Import] from the workbench menu, then select the file format to be imported from the [Import] wizard.



If a source file is included in the imported Java EE module, the source file is stored in the project source folder created by the Import wizard. If a source file is not included in the imported Java EE module, an ImportedClasses folder is created in the project created by the Import wizard and the class file is stored in the ImportedClasses folder.

6.2.3 Creating the Java Class and the Interface

The code for Java classes and interfaces is in source files which have an extension of '.java'. This is called a Java source file. This section describes how to create and edit Java class and interface source files.

Creating New Java Class Source Files

To create a new Java class source file, select [Java] > [Class] from the [New] wizard. This launches the New Java Class wizard. Specify the package name, class name, superclass, interface to be implemented, and so on, and create the Java class.

Creating New Java Interface Source Files

To create a new Java interface source file, select [Java] > [Interface] from the [New] wizard. This launches the New Java Interface wizard. Specify the package name, class name, extension interface, and so on, and create the Java interface.

Editing Java Source Files

Use the Java editor to edit the Java source file. When the Java source file is opened using the Project Explorer or Navigator view, it is actually opened in the Java Editor. The Java editor provides the following editing support:

- Content Assist

The [Ctrl+Space] keys can be pressed from the editor to display a list of possible elements that can be inserted at that location. For example, if the first few characters of a class name are entered and the [Ctrl+Space] keys are pressed, a list of class names starting with those characters is displayed. Input assist also offers suggestions for other method names, variable names, and so on.

- Organize Imports

If other classes or interfaces are referenced within the source, the full name (the name including the package name) of that class must be written in the import statement. The Java editor assists with creating this import statement. When [Source] > [Organize Import] is executed, any import statements that are missing from the source are added, and any import statements that are unnecessary to the source are removed.

- Refactoring

Refactoring is the changing of the internal structure of source code without changing the way the program behaves. The purpose of this is to arrange the program in a more concise and easily understood way and to make subsequent specification changes more flexible. For example, if a class name, method name, variable name, or similar has been changed, locations in other source files that reference that name are also changed at the same time. Many other refactoring techniques, such as extracting superclass or method, introducing factory or introducing parameter object, and so on, are also provided.

- Quick Fix

On the Java editor, any places that have compile errors in the source or warnings are displayed with a wavy line. When the [Ctrl+1] keys are pressed at a wavy line, suggestions on how to correct the error or warning are displayed. For example, if a variable name was entered incorrectly, suggestions to either correct the name to the correct variable name or to add a new variable declaration are displayed. When one of the suggestions is selected, that correction is applied.

6.2.4 Creating XML Files

This section describes how to create and edit XML files.

Creating XML Files

To create a new XML file, select [Other] > [XML] > [XML] from [New] under the [File] menu. This launches the [New XML File] wizard. Specify the file name, creation method, and other information required to create the XML file as indicated in the wizard pages.

On the [Create XML File From] page of the [New XML File] wizard, select from the following XML file creation methods:

- Create XML file from a DTD file

Create the XML file on the basis of the definitions in a DTD file selected from the workspace or XML catalog.

- Create XML file from an XML schema file

Create the XML file on the basis of the definitions in an XML schema file selected from the workspace or XML catalog.

- Create XML file from an XML template

Create the XML file on the basis of a registered template.

Editing XML Files

Use the XML editor to edit XML files. When the XML file is opened and operated in the Project Explorer or the Navigator view, the file opens in the XML editor. The XML editor provides the following editing support:

- Content Assist

The [Ctrl+Space] keys can be pressed from the editor to display a list of possible values that can be inserted at that location. If DTD and XML schemas are specified in the XML that is being edited, the element names, attribute names, and attribute values that can be entered in accordance with those specification are displayed as suggestions. If DTD and XML schemas are not specified, element names, attribute names, and attribute values extracted from the XML being edited are displayed as suggestions.

- Editing using the Design view

The [Design] view can be used to edit XML data and attribute values using a grid-format editor. Alternatively, the following editing operations can be performed using the context menu of the [Design] view:

- At the processing instruction context menu, [Edit Processing Instruction] can be selected to edit the processing instruction attributes.
- At the DOCTYPE context menu, [Edit DOCTYPE] can be selected to edit the DOCTYPE attributes.
- At the element context menu, [Add Child] can be selected to add elements using a tree format that includes child elements and further child elements.

- Cleanup

When [Cleanup Document] is selected from the [Source] menu, the following operations can be applied as a batch to the file being edited. The format is in accordance with the format set using [XML]>[XML Files]>[Editor] under [Preferences] in the [Window] menu.

- Compress empty element tags
- Insert required attributes
- Insert missing tags
- Quote attribute values
- Format source
- Convert line delimiters

- Quick fix

On the XML editor, any places that have compile errors in the source or warnings are displayed with a wavy line. When the [Ctrl+1] keys are pressed at a wavy line, suggestions on how to correct the error or warning are displayed. For example, if there is no closing tag, suggestions to either insert an end tag before the next child element or to insert an end tag at the end of the element are displayed. When one of the suggestions is selected, that correction is applied.

6.2.5 Detecting and Correcting Problems

If there are errors, warnings, or other problems in Java files, JSP files or other source files, markers are set at those locations. Thus, problems can be checked from the icons displayed in the editor rulers and the highlighted displays in the source. A list of problems is also displayed in the Problems view. The problem content is displayed as marker information. Correct these locations as required.

Errors, warnings, and other problems are detected by various functions. The following functions detect problems:

- Java compiler

The Java compiler not only checks for coding errors that violate Java specifications, but can also check for coding that may cause problems despite conforming to Java specifications.

- Validation

This function checks whether or not a file or application unit conforms to conventions and so on.

- Interstage Java EE validator

This function investigates an application's validity as a Java EE application.

- FindBugs
This function identifies code that may contain bugs.

6.2.5.1 Java Compiler

The Java compiler not only checks for coding errors that violate Java specifications, but can also check for coding that may cause problems despite conforming to Java specifications.

The following categories can be checked:

- Code style
- Potential programming problems
- Name shadowing and conflicts
- Deprecated and restricted API
- Unnecessary code
- Generic types
- Annotations

More detailed check items belong to the above categories, and the check level can be customized. Customized settings can be entered from [Java] > [Compiler] on the preferences page, or from other preferences page under the preferences page. To customize settings for a specific project, use [Java Compiler] under the project properties.

6.2.5.2 Validation

The validators shown below are provided to validate files and applications. The validators can check in file units and project units.

Validator	Contents to be checked
HTML Syntax Validator	Checks the basic syntax of HTML files.
JavaScript Syntax Validator	Checks the JavaScript syntax described in HTML/JSP files.
JPA Validator	Checks the validity of an application as a JPA application.
JSP Syntax Validator	Converts a JSP file to Java code then checks for compile errors in that Java code, thereby checking the syntax of the JSP file.
JSP Content Validator	Checks the EL, directives, and so on of the JSP.
XML Validator	Checks that an XML file is in the correct format and checks its validity.

The validators run at the following times:

- When a file is saved
Executed as a builder if ON is set for automatic build.
- When editing using an editor
Validators that can check in file units also check during editing using an editor.
- When executing from a menu
Validation can be performed by selecting a resource (file, project, or similar), then selecting [Validate] from the context menu.

Validation Settings

The validation operation can be customized using [Validation] on the Preferences page. To customize settings for a specific project, use [Validation] under the project properties.

Refer to the following concerning customization:

- Suspend all validators
Specify this if you want to stop all validators temporarily.
- Manual/Build
You can enable/disable each validator for both manual operation (executed by menu) and build.

6.2.5.3 Interstage Java EE Validator

The validity of an application as a Java EE application can be validated in archive file units, as shown below.

- Before deployment (default)
The validity is checked before the application is deployed from Interstage Studio to the Interstage Application Server in order to check operation of the application, and so on.
- Execution from the menu
The validity is checked when the project is selected and [Interstage Java EE Validator] > [Validate] is selected from the context menu. Use this method if you want to check the validity before application creation is completed.
- Build
The Java EE validation function can be executed at build time. However, if ON is set for automatic build, all archive files are checked for validity as a Java EE application every time a file is saved, which is inefficient. Therefore, it is not recommended that developers use this method when developing separate Java EE modules. Use this method for group validation and execution when, for example, application build is performed as a batch.



When an enterprise application project is checked, the modules it contains are also checked. If the workspace is used to create an enterprise application and also its constituent modules, and if the validity is checked at build time, avoid duplication by selecting only one or the other.

Interstage Java EE Validator Settings

The Interstage Java EE Validator operation can be customized using [Interstage Java EE Validator] on the preferences page. To customize settings for a specific project, use [Interstage Java EE Validator] under the project properties. Perform customization with reference to the following:

- Check JSP file
Checking the JSP duplicates the validation validator, and identifying problem locations is easier using the validation validator. Therefore, the default is to not check the JSP. However, if you want to perform a strict check for an application operating on an Interstage Application Server in order to use and check the Interstage Application Server JSP compiler, the JSP check can be used instead of the HTML/JSP syntax validator to validate the Java EE application.
- Execute pre-deploy
This checks the validity before the application is deployed from the Servers view. Once the application structure is determined, and while at the stage of only correcting faults in the implementation part, for example, omit this check if there is little need to check the validity before checking application operation.
- Don't check at incremental build
If the check is performed at build time, the default setting is to not perform the check for a incremental build. Deselect this item if you want to check in archive file units every time even for differential builds.

6.2.5.4 FindBugs

FindBugs can be used to identify code which may contain bugs as shown below.

- Execute from the menu
To execute FindBugs, select the resource (project or file), then select [Find Bugs] > [Find Bugs] from the context menu.
- Build
FindBugs can be executed at build time. (Execution of FindBugs at build time is not set as the default.)

Point

It is also possible to change the settings collectively to execute as builders by selecting multiple project and [Find Bugs] > [Turn on "Run FindBugs automatically"] from the context menu.

The detected problem will be displayed in the Problems View in the following format.

```
<Warning priority> <Bug category> <Bug pattern> : <Message>
```

- Warning priority
Represented by the initials of High, Medium, Low. Detection level can be customized in [Minimum priority to report].
- Bug category
Represented by an abbreviation of bug category (one English letter). The bug category unit is a unit that can be specified to check or not check. For details on mapping of bug category and abbreviation, refer to "[Check Contents](#)".
- Bug pattern
Represented by an abbreviation of bug pattern (a combination of numbers and letters). The operation check can be customized by listing in the filter file.

In addition, as listed below, views and perspectives for the purpose of displaying problems detected by FindBugs are also provided.

- FindBugs perspective
This perspective is for correcting code for which the possibility of a bug has been detected. This perspective is comprised of the Bug Details view, the Bug Tree view, the Bug User Annotations view, and so on.
- Bug Details view
This view displays detailed information concerning detected problems.
The warning priority, bug category, problem location class, method name, field, source file name and line number, an overview of the problem, and detailed contents are displayed as detailed information.
- Bug Tree view
This view displays the detected problems in a tree format.
Tabs are assigned in project units in the Bug Tree view, and the detected problems are displayed in tree format within the tabs in bug pattern units. Details of a problem selected in the Bug Tree view are displayed in the Bug Details view. Alternatively, if the problem in the Bug Tree is double-clicked, the location that caused the problem is displayed in the Editor area in the same way as in the Problems view.
- Bug User Annotations view
User annotations can be entered concerning each of the problems detected by FindBugs.
As an annotation, a classification can be selected or any comment can be specified. In addition, the view displays the date and time the problem was detected.

Check Contents

In FindBugs, the checks shown below are performed.

Bug Category	Abbreviation	Overview
Performance	P	<p>It is not necessarily a mistake, but it is possible that execution efficiency will worsen. For example, the following points are detected.</p> <ul style="list-style-type: none"> - Inefficient use of URL class - Inefficient use of String class - Inefficient use of valueOf method - Inefficient use of boxing/ unboxing - Field or method declaration that is not in use
Correctness	C	<p>Obvious coding errors that resulted in unintended code by the developers. For example, the following points are detected.</p> <ul style="list-style-type: none"> - Misuse of equals method - Obvious infinite loops - Naming issues in fields, methods and so on - Null pointers
Internationalization	I	<p>Flaws exist in code that deals with Internationalization or locale. For example, the following points are detected.</p> <ul style="list-style-type: none"> - A method that keeps the Locale in the argument is not used
Multithreaded correctness	M	<p>Flaws exist in code that deals with Threads, locks and volatiles.</p>
Bad practice	B	<p>Code that deviates from recommended or fundamental coding patterns. For example, the following points are detected.</p> <ul style="list-style-type: none"> - Exception handling - Incorrect comparison of character strings - Misuse of finalizer - Use of equals method - Confusing names of fields, methods and so on - Serialization problems
Malicious code vulnerability	V	<p>Code that is vulnerable to attacks from untrusted code. For example, the following points are detected.</p> <ul style="list-style-type: none"> - Possibility of internal expressions spilling - Related to private or final (and so on) modifiers
Dodgy	D	<p>Code that is easily misinterpreted, improper, or invites errors. For example, the following points are detected.</p> <ul style="list-style-type: none"> - Related to settings of values to variables - Related to cast

The bug category units above can be customized to check or not check.

FindBugs Settings

The FindBugs operation can be customized using [FindBugs] on the preferences page. To customize settings for a specific project, use [FindBugs] under the project properties.

Perform customization with reference to the following:

- Minimum priority to report
Problems with the specified detection level or greater are detected.

- Exclude filter files
Bug detection is excluded by the settings of an added filter file.

For setup items that are specific to a project, refer to the following:

- Run FindBugs automatically
Specify this option when performing FindBugs at build time.



It is also possible to change the settings collectively to execute as builders by selecting multiple project and [Find Bugs] > [Turn on "Run FindBugs automatically"] from the context menu.

FindBugs Filter File Format

With FindBugs it is possible to control whether or not checked content is treated as a problem using the filter file. For example, if with FindBugs detects a problem but a code check proves that it is not actually a problem, the exclude filter file can be set so that in future it will not detect that as a problem.

The tags used by Filter File are explained below.

Tag	Attribute	Description
FindBugsFilter	-	Top level element of filter file.
Match	-	Element used to identify class.
	class	Specifies class name.
	classregex	Specifies class name as a regular expression.
BugCode	-	Element used to specify bug pattern abbreviations. The bug pattern abbreviation is the alphabet combination displayed third in the problem message.
	name	Separate each bug pattern abbreviation with a comma.
Priority	-	Element used to specify priority.
	value	Specify one of the following values. 1: High 2: Normal 3: Low
Method	-	Element used to specify method. The params attribute and returns attribute are not mandatory, but if specifying, both must be specified.
	name	Specifies method name.
	params	Separate each full qualified argument type name with a comma.
	returns	Separate each full qualified return value type name with a comma.
Or	-	Element that acts as an OR logical operator.

The following is an actual exclude filter file.

```
<?xml version="1.0" encoding="UTF-8"?>
<FindBugsFilter>
  <!-- Do not detect a problem related to ClassNotToBeAnalyzed class -->
  <Match class="com.foobar.ClassNotToBeAnalyzed" />
</FindBugsFilter>
```

```

<!-- Do not detect the following problems related to ClassWithSomeBugsMatched class
  DE : Exception is overlooked or ignored.
  UrF: A field that will not be referenced exists.
-->
<Match class="com.foobar.ClassWithSomeBugsMatched">
  <BugCode name="DE,UrF" />
</Match>

<!-- Do not detect SQL problems in any class -->
<Match classregex=".*" >
  <BugCode name="SQL" />
</Match>

<!-- Do not detect field double-check problems for nonOverloadedMethod, frob or blat method of
AnotherClass class -->
<Match class="com.foobar.AnotherClass">
  <Or>
    <Method name="nonOverloadedMethod" />
    <Method name="frob" params="int,java.lang.String" returns="void" />
    <Method name="blat" params="" returns="boolean" />
  </Or>
  <BugCode name="DC" />
</Match>

<!-- Do not detect priority 2 (normal) problems in check item "substitution of meaningless local
variables" for the someMethod of the MyClass class-->
<Match class="com.foobar.MyClass">
  <Method name="someMethod"/>
  <BugCode name="DLS" />
  <Priority value="2"/>
</Match>

</FindBugsFilter>

```

6.2.6 Checking Application Behavior

To check the behavior of a Java EE application, deploy the Java EE application to the server, then start the server. Perform these operations in the Servers view. When the server starts, the Java EE application waits to be invoked from the client. The behavior is checked by invoking Java EE application from the client.

In addition, the application can be debugged by interrupting program execution and executing code one line at a time and checking variable values.

Preparations for Server Operation

In order to start and stop the server from the workbench, the server must be added to the Servers view. Once the server is added you will need to specify things such as the applications to be deployed to it.

Adding Servers

Add to the Servers view the server that is the application deployment destination for checking the application behavior. Use the New Server wizard to add the server. Right-click the Servers view, then select [New] > [Server] from the context menu. For wizard setup items, refer to the following:

- Server's host name
Set the name of the host where the server exists. For a local machine, specify "localhost".

- Server type
Select the server type. For an Interstage Java EE container, select [FUJITSU LIMITED] > [Interstage Application Server V11.1 IJServer Cluster (Java EE)].

- Server name
The server name can be customized, but modifications made here are not reflected in Interstage Studio, and use the format "IJSERVER Cluster name [server type name] (host name)"
- Server runtime environment
Specify the runtime settings in accordance with the server type. For an Interstage Java EE container, select [Interstage Application Server V11.1 IJSERVER Cluster (Java EE)].

If the server is an Interstage Application Server, the following setup items are also displayed:

- Use HTTPS communication to connect to the target
Select this item if SSL encryption is used. It must match the server settings.
For the server settings, refer to the security property value of the HTTP listener in the Interstage Java EE Admin Console.
- Interstage JMX Service Port Number
Set the port number to a value that is the same as the port number of the Interstage Java EE Admin Console. After specifying the port number, click [Login] to check whether login to the server is possible. If login succeeds, [Next] is enabled. If the user does not have administrator permissions, an authentication dialog box will be displayed. Set a user ID and password with administrator permissions.
- Admin Console port number
Specify the Interstage Java EE Admin Console port number. The default port number is 12001. It must match the server settings.
For the server settings, refer to the Admin Server port number in the Interstage Java EE Admin Console.
- Select IJSERVER Cluster
In the Servers view, select the IJSERVER Cluster for operations.
- HTTP Port Number and Debug Port Number
The default values are displayed, and can be changed if required.
- Stop the IJSERVER Cluster on shutdown
This checkbox changes to selectable if the IJSERVER Cluster is on the local machine (not selectable if "server" is selected as the IJSERVER Cluster). If selected, the IJSERVER Cluster is stopped on workspace shutdown.
Note that the IJSERVER Cluster is not stopped for remote machine, when the workspace shuts down. Thus, this checkbox cannot be selected if a remote machine was specified at the wizard.

Point

- The Interstage Java EE DAS service can be used to perform operation verification on the local machine. To use the Interstage Java EE DAS service, select "server" when selecting the IJSERVER Cluster.
- A server can also be added using the New wizard.

Note

- Refer to "[6.2.1 Preparing the Deployment Destination for Verifying Application Operation](#)" for information on the advantages of the Interstage Java EE DAS service and of IJSERVER Cluster, and for related notes.
- For IJSERVER(J2EE), [Use HTTPS communication to connect to the target] is replaced by [HTTPS is used for the connection to the management console] and, if specified, SSL communication targets only connections to the Admin Console.
- Wizard input items may be masked depending on the specified host and IJSERVER Cluster type.

- If the server type was set to a value prior to Interstage Application Server V9.3, set the following values.
 - Interstage JMX Service Port Number
Specify the Interstage Java EE Admin Console port number. The default port number is 8919.
 - Admin Console port number
Specify the Interstage Admin Console port number. The default port number is 12000.

.....

Specifying the Project to be Deployed to the Server

The project to be deployed to the server is specified by adding the project to the Servers view server. Use the Add and Remove Projects wizard to add the project to be deployed to the server. To launch the Add and Remove Projects wizard, select the server at the Servers view, then select [Add and Remove Projects] from the context menu.
A project can also be added using the last page of the New Server wizard.

Point

.....

To deploy an application manually, select the server, then select [Publish] from the context menu. If [Clean] is used, all deployed applications are temporarily discarded and then redeployed. If the application is synchronized with the server, "synchronized" is displayed as the status. If not synchronized, "Republish" is displayed.

.....

Starting Servers

In the Servers view, select the server, then select either [Start] or [Debug] from the context menu. The application can be deployed automatically when the server starts. The default setting is that applications are deployed automatically when the server starts.

Point

-
- If the connection to the IJServer Cluster is not yet established, nothing is displayed in the [Status] column for the IJServer Cluster at the Servers view, and start and stop operations are not performed. In this case, select the IJServer Cluster at the Servers view, then select [Connect/Login] from the context menu to connect to the IJServer Cluster.
 - If the Interstage Management Service of the server is stopped while connected to the IJServer Cluster, connection to the IJServer Cluster will be interrupted, causing failure of operations of the IJServer Cluster in the Servers view. If operations fail, in the Servers view, select IJServer Cluster, and from the context menu, select [Refresh]. After updating the status of the IJServer Cluster, from the context menu, select [Connect/Login] to reconnect to the IJServer Cluster.
 - The user can select the project, then select [Run As] > [Run on Server] or [Debug As] > [Debug on Server] from the context menu to add the server, add the project, launch the server, and launch the client all together.
 - To set such that the application is not deployed automatically when the server starts, select [Server] > [Launching] from the Preferences page, then set [Automatically publish when starting servers] to off.
-

Note

.....

If starting the server in debug mode, and from the project context menu, selecting [Run As] > [Run on Server] or [Debug As] > [Debug on Server], take note of the following points.

- If selecting [Run As] > [Run on Server], debug mode must be cancelled. Before execution, in the Servers view, restart or stop the server.
 - When [Debug As] > [Debug on Server] is selected, the [Debug On Server] dialog box is displayed. Select [Continue in the current mode], then select [OK].
-

Stopping Servers

To stop the server, select the server at the Servers view, then select [Stop] from the toolbar.

6.2.6.1 Debugging

This section describes the debugging that is used to detect logical errors in programs.

To debug an application, set breakpoints to interrupt the launched program, and then execute the code one line at a time and check the variable content.

Breakpoints

Program execution is interrupted at the lines where breakpoints have been set. When execution is interrupted, a confirmation dialog box asking whether or not to open the Debug perspective is displayed. The Debug perspective is comprised of layouts used during debugging, such as the Debug view, the Variables view, and the Breakpoints view.

- Setting and removing breakpoints

Double-click the ruler part of the editor to toggle between setting and removing breakpoints. In addition, the Breakpoints view can be used to check breakpoints that have been added and to remove breakpoints.

- Disabling breakpoints

To temporarily disable a breakpoint, select [Disable Breakpoint] from the ruler context menu, or deselect it from the Breakpoints view. Alternatively, to temporarily skip all breakpoints, select [Skip All Breakpoints] under [Run] from either the Breakpoints view toolbar or the menu bar.



If breakpoints have been set in a JSP file, execution is also interrupted for JSP files with the same name in different folders.

Execution Control

Various methods are available for restarting execution of execution-interrupted programs, including step over, step into, step return, and resume. To execute these commands, select stack frame from the Debug view, then select a command from [Run] in the Debug view toolbar, context menu, or menu bar.

- Step Over

The currently selected line is executed, then execution is interrupted at the next executable line.

- Step Into

The next expression that needs to be executed in the currently selected line is invoked, then execution is interrupted at the next executable line of the invoked method.

- Step Return

Execution up to the next return statement of the current method is restarted, then execution is interrupted at the next executable line.

- Resume

Execution continues until the next breakpoint is reached.

Checking and Changing Variable Values

When a stack frame is selected, the variables that are visible in that stack frame can be displayed in the Variables view.

Primitive type values are displayed in the Variables view. If complex variables are expanded and their members are displayed, these can be inspected. Values can be changed using [Change Value] under the context menu.



Debug information must be added to a compiled class file in order to reference variable values and perform debugging. The default setting is that the debug information required for debugging is added to the class file. If debugging is not required, for example, when deploying to an operating environment, the size of the class file can be reduced by not adding the debug information.

The debug information is used to display source file names and line numbers in the stack trace that is output when, for example, an exception is thrown. It is recommended to add source file names and line numbers to a class file.

6.2.6.2 Starting the Interstage Java EE Admin Console


In the Servers view, from the context menu, when [Admin Console] is selected, the Interstage Java EE Admin Console is displayed in the editor area.

To display the Interstage Java EE Admin Console, an Internal Web Viewer is used. This is a built-in browser used to display pages on the Web server.

An address bar and toolbar are provided with the Internal Web Viewer.






Address Bar

The address bar shows the [Address] field for entering of the URL of the page to be displayed, along with the following button.

Button	Command	Description
	Go	Moves to the page belonging to the entered URL.

Tool Bar

The toolbar shows the following buttons.

Button	Command	Description
	Go Back	Returns to the previous page that was displayed in the Internal Web Viewer before the current page.
	Go Forward	Redisplays the page that was displayed before [Back] was clicked in the Internal Web Viewer.
	Stop	Stops loading of the current page in the Internal Web Viewer.
	Refresh	Reloads the current page displayed in the Internal Web Viewer.
	Text Size	Changes the size of displayed characters in the Internal Web Viewer. You can select from the following five sizes: [Largest], [Larger], [Medium], [Smaller], and [Smallest].

Note

- If displaying the Interstage Java EE Admin Console in the Internal Web Viewer, the address bar is not displayed.
- If the Interstage Management Service of the server is stopped, even if [Admin Console] is selected from the context menu, the Interstage Java EE Admin Console will fail to display. In such cases, connection to the IJServer Cluster will be interrupted, therefore reconnection to the IJServer Cluster is required. To reconnect to the IJServer Cluster, in the Servers view, select IJServer Cluster, and from the context menu, select [Refresh]. After updating the status of the IJServer Cluster, from the context menu, select [Connect/Login]. After reconnection is finished, execute [Admin Console] again.

6.2.7 Distributing to the Operating Environment

To distribute a Java EE application to the operating environment, an archive file must be created. Use the server deployment function to deploy the created archive file to the server.

Creating an Archive File

Use the Export wizard to create the archive file.

Export

1. Launching the Export wizard
Select [File] > [Export].
2. Entering settings items
 - Export destination

Project	Export destination
EJB Project	[EJB] > [EJB JAR file]
Enterprise Application Project	[Java EE] > [EAR file]
Application Client Project	Export the project as an EAR, because the project is included in the application created in enterprise application project. Use the following if exporting it on a standalone basis: [Java EE] > [App Client JAR file]
Dynamic Web Project	[Web] > [WAR file]
JPA Project	Export the project as an EAR or a WAR because the project is included in the application created in enterprise application project or in dynamic web project. Use the following if exporting it on a standalone basis: [Java] > [JAR file]

- Module
Specify the project to be exported.
- Destination
Specify the creation destination of the archive file.

 **Point**

- When an enterprise application project is exported, an EAR file containing the archive files of each of the projects included in the enterprise application project is created. Therefore, individual archive files need not be created.
- Of the JAR files included in the build path, those that cannot be included in the EAR must be set in the server classpath.

Deploying to the Server

Use the server deployment function to deploy the application to the server.
For details of deployment, refer to separate documentation concerning the Java EE execution environment of Interstage Application Server.

6.2.8 Developing Java EE Applications Using Entity Beans

Use the Java Persistence API, available since EJB 3.0, for the development of Java EE applications. Refer to "[D.6.1 Migration from J2EE Container to Java EE Container](#)" to migrate from Entity Beans used up until EJB 2.1 and migrate the CMP extension information files and DB definitions.

Refer to "[D.6.1 Migration from J2EE Container to Java EE Container](#)" and the manual for Interstage Application Server for information on modifying the CMP2.0 relation definitions for Java EE applications.

Chapter 7 Developing Java Applications

This section provides an overview of Java applications, and explains how to create Java applications.

7.1 Overview

This section provides an overview of Java applications.

7.1.1 What is a Java Application?

A Java application is an application that runs on a Java platform. Java applications can be developed using the Java programming language.

Interstage Studio provides Java application projects in which the Java build paths and builders required for application development are already set. The Java application projects can be used to develop the applications shown below.

Client applications

Client applications are programs that use java commands and javaw commands and can execute them independently.

Applets

Applets are programs that can be executed by Web browsers by being embedded in HTML. HTML is only a static expression, but applets can perform dynamic expressions.

Libraries

A library is a program that groups together as one component various functions that can be used by multiple applications. Since a library cannot be executed independently, it is used from a client application or an applet. It can also be used from other J2EE applications, such as Web applications and EJB applications.

Forms

Forms can be broadly classified as frames, dialogs, and panels. Inherited forms are provided for all three types.

JavaBeans

JavaBeans are conventions that componentize Java classes. JavaBeans is also a generic name for components that conform to these conventions. Individual componentized component are referred to as Beans.



Point

There are two container and Bean types; the heavyweight component (AWT) and the lightweight component (Swing). When creating Java Forms and applets, be as consistent as possible with the use of heavyweight and lightweight components. When heavyweight and lightweight components are mixed, there may be problems such as overlapping (with heavyweight components always appearing on top), and menu Beans submerging.

7.1.2 Developing Java Applications

This section provides an overview of Java application development.

Preparing for Java Application Development

Before developing a Java application, a Java application project must be created. Create a project and set the required classpath and other project settings. For details, refer to "[7.3.1 Preparing the Environment for Creating Java Applications](#)".

Creating classes/applets/forms/JavaBeans

Use the source wizards and editors to create the resources required for Java applications.

Refer to "[7.3.2 Creating a Class](#)", "[7.3.3 Creating an Applet](#)", "[7.3.4 Creating a Form](#)", and "[7.3.5 Creating JavaBeans](#)" for details.

Building and Validating a Java Application

By default, a build is performed automatically when resources are saved. During a build, various validators check that there are no compile errors or resource problems.

For details, refer to "[Build](#)" and "[6.2.5.2 Validation](#)".

Verifying Java Application Operation

Execute and debug the created Java application.

For details, refer to "[7.3.11 Verifying Java Application Operation](#)".

Distributing a Java Application

Distribute the execution resources to the operating environment.

For details, refer to "[7.3.12 Distributing a Java Application to the Operational Environment](#)".

7.2 Introduction

This section introduces the procedures for creating Java applications that use applets.

There are also tutorials for creating applets that transition through several windows and for developing JavaBeans. Refer to "[F.1 Java Application](#)" for details.

7.2.1 Applet that is Created

Create an applet that returns the country name and total population from a ranking list of the populations of countries when the ranking is entered, and displays the result on a message box.

7.2.2 Development Flow

The development of the applet proceeds as follows:

1. Creating the Project

First create a Java Application Project to create applets. Classpath settings necessary for the build are made automatically by creating the Java Application Project with the wizard.

2. Creating the Java Classes

Create the two classes required for the applet:

- Create the data class
- Create the logic class

3. Creating the input window

Create the applet to be used as the input window and edit it with the Graphical Editor:

- Create the input window template
- Edit the input window

4. Checking behavior of the applet

Use the following procedure to confirm the application behavior:

- Setting breakpoints
Set breakpoints to confirm the behavior of the program in the debugger when the application is executed.

- Execute the applet
Start the applet viewer to start the confirmation of the applet behavior.
 - Debugging the applet
Debug the program to confirm whether the applet is behaving correctly.
5. Distribute the applet to the operating environment
Use the following procedure to distribute the applet to the operating environment:
- Export the applet
Create a JAR file to distribute the applet to the operating environment.
 - Create the HTML files
Create the HTML files to be displayed in the Web browser and enter the applet definition.
 - Distribute to the operating environment
Deploy the HTML and JAR files.

7.2.3 Development Procedures

The actual procedures used to develop the applet are described below.

- 1) [Creating the Project](#)
- 2) [Creating the Java Classes](#)
- 3) [Creating the I/O Page](#)
- 4) [Verifying the Applet Behavior](#)
- 5) [Distributing the Applet to the Operating Environment](#)

1) Creating the Project

Select [File] > [New] > [Project] from the menu bar to display the [New Project] wizard.

Select [Java] > [Java Application Project] from the [New Project] wizard, then click [Next].

Check and enter the following setup items. After the following information is set, click [Finish].

Setup Items	Setup Content
Project name	AppletSample
Contents	Create new project in workspace
JRE	Use default JRE
Add project to working sets	Do not select

2) Creating the Java Classes

2-1) Creating the Data Class

Create a data class that stores the country name and total population information, and gets information. To create the data class, select the created project, then right-click to display the context menu. Select [New] > [Class] from the context menu. The [New Java Class] wizard is displayed.

Check and enter the following setup items. After the following information is set, click [Finish].

Setup Items	Setup Content
Source folder	AppletSample/src

Setup Items	Setup Content
Package	sample
Name	CountryData

The following source file is generated:

Source file	Description
CountryData.java	Data class

Implement the processing for storing the country names and total populations. Add the places shown in **red** below to the source.

Data Class Implementation (CountryData.java)

```

package sample;

public class CountryData {

    private String countryName;
    private int totalPopulation;

    public CountryData(String name, int total) {
        setCountryName(name);
        setTotalPopulation(total);
    }

    public String getCountryName() {
        return countryName;
    }

    public void setCountryName(String countryName) {
        this.countryName = countryName;
    }

    public int getTotalPopulation() {
        return totalPopulation;
    }

    public void setTotalPopulation(int totalPopulation) {
        this.totalPopulation = totalPopulation;
    }

}

```

Point

After the fields are added, and while the class is in the selected state, [Source] > [Generate Getters and Setters] can be selected from the menu to add getter/setter.

2-2) Creating the Logic Class

To create the logic class, select the created project, then right-click to display the context menu. Select [New] > [Class] from the context menu. The [New Java Class] wizard is displayed.

Check and enter the following setup items. After the following information is set, click [Finish].

Setup Items	Setup Content
Source folder	AppletSample/src
Package	sample
Name	PopulationRanking

The following source file is generated.

Source file	Description
PopulationRanking.java	Logic class

Implement the processing for entering the ranking and returning the country name and total population. Add the places shown in **red** below to the source.

Logic Class Implementation (PopulationRanking.java)

```

package sample;

public class PopulationRanking {

    private CountryData[] countries;

    public PopulationRanking(){

        countries = new CountryData[]{
            new CountryData("China",1330000000),
            new CountryData("India",1140000000),
            new CountryData("U.S.A.",300000000),
            new CountryData("Indonesia",230000000),
            new CountryData("Brazil",190000000),
            new CountryData("Pakistan",160000000),
            new CountryData("Bangladesh",150000000),
            new CountryData("Russia",140000000),
            new CountryData("Nigeria",140000000),
            new CountryData("Japan",130000000)
        };
    }

    public CountryData getCountryData(int rank) {
        --rank;
        if (rank < 0 || rank >= countries.length) {
            return null;
        }

        return countries[rank];
    }
}

```

3) Creating the I/O Page

3-1) Creating the Input Page Pattern

Create the applet to be used as the input window. To create the applet, select the project that was created, then select [New] > [Other] from the right-click context menu. From the wizard list, select [Java] > [GUI] > [Applet], then click [Next]. The [New Applet] wizard is displayed.

Check and enter the following setup items. After the following information is set, click [Next].

Setup Items	Setup Content
Source folder	AppletSample/src
Package	sample

The [New Applet] dialog box is displayed. Select [Applet] in the Java tab, then click [OK]. The applet wizard is displayed.

Check and enter the following setup items. After the following information is set, click [Next].

Setup Items	Setup Content
Package name	sample
Applet Name	PopulationRankingApplet
Base class	com.fujitsu.jbk.gui.JFApplet
Bounds	(80,170,400,400)
Font	Dialog,12
Foreground	Black
Background	White

To create the HTML, confirm and enter the following setup items. After the following information is set, click [Next].

Setup Items	Setup Content
Generate HTML	Select
Generate HTML for JBK plug-in	Select
Page title	World ranking for total population
Width of applet	400
Height of applet	400

Parameters are not used in this applet. Select "myItem0" in the parameter information, then click [Delete]. After deleting the parameter, click [Create].

3-2) Editing the Input Page

Use the Graphical Editor to edit the PopulationRankingApplet.java file that was created. Refer to ["7.3.6 Editing a Java Form"](#) for information on editing with the Graphical Editor.

Use the following procedure to paste Beans into the applet:

1. From the list of objects displayed at the top of the properties window, select the applet.
2. From the list of properties displayed at the bottom of the properties window, set the following values for the applet's properties. It is possible to change the setting according to property type by clicking or double-clicking the mouse on the property settings section.

Property name	Value
layout	<NONE>

3. In the object palette of the Java Form Designer window, click [AWT].
4. From the object palette, select the AWT label.
5. Position it in the applet. Click on the location where the Bean is to be pasted to. Drag until it is big enough and then release the mouse button to paste the Bean.
6. Similarly, paste two AWT labels (a total of three including the Bean pasted in the previous step), an AWT text field, and an AWT button.
7. In the properties window, set the values shown below for the properties of the pasted Bean. Switch the [Standard]/[Original] tabs of the properties window, select the property, then change the value.

label1 [AWT Label]

Property name	Value
bounds	(24,24,256,24)
text	Enter a ranking from 1 to 10.

label2 [AWT Label]

Property name	Value
bounds	(24,80,48,24)
text	Rank:

textField1 [AWT TextField]

Property name	Value
bounds	(80,80,120,24)
text	

label3 [AWT Label]

Property name	Value
bounds	(208,80,48,24)
text	Rank

button1 [AWT Button]

Property name	Value
bounds	(40,136,200,32)
label	OK

3-3) Entering the event process for the input window

In the Graphical Editor applet window, select Button1, then select [Event], [action], [actionPerformed] from the context menu. The event process "button1_action_actionPerformed" is added to the PopulationRankingApplet.java file. Add the places shown in **red** below to the source.

Event process "action_actionPerformed" of button1 (PopulationRankingApplet.java)

```
private void button1_action_actionPerformed$(java.awt.event.ActionEvent e) {
    if (!defaultEventProc$(e)) {
        // The process when the event is generated is described below.
        // Get input value
        int rank;
        try {
            rank = Integer.parseInt(textField1.getText());
        } catch (NumberFormatException ex) {
            rank = 0;
        }

        // Check input
        if (rank < 1 || rank > 10) {
            javax.swing.JOptionPane.showMessageDialog(
                this,
                "The input ranking is not within the specified range.",
                "Error",
```

```

        javax.swing.JOptionPane.ERROR_MESSAGE);
    return;
}

// Output of the result
PopulationRanking pop = new PopulationRanking();
CountryData country = pop.getCountryData(rank);
StringBuilder message = new StringBuilder();
message.append("The country with the ");
message.append(rank);
message.append("rank is\");
message.append(country.getCountryName());
message.append("\");
message.append(System.getProperty("line.separator"));
message.append("The population is");
message.append(country.getTotalPopulation());
message.append("\");
javax.swing.JOptionPane.showMessageDialog(
    this,
    message,
    "Result",
    javax.swing.JOptionPane.INFORMATION_MESSAGE);
}
}

```

4) Verifying the Applet Behavior

4-1) Setting Breakpoints

Set a breakpoint at the first line of the `button1_action_actionPerformed$()` method of the applet class. To set or delete a breakpoint, double-click the vertical ruler on the left-hand edge of the editor.

4-2) Executing the Applet

In the Package Explorer view, select the `PopulationRankingApplet.java` file then select `[Debug As] > [Java Applet]` from the context menu. The applet viewer starts, and the input screen opens.

Enter the population ranking in the `[Rank]` input field, then click `[OK]`.

The applet runs and is interrupted at the breakpoints. Check the value of variable by Variables view. Select `[Run] > [Step Over]` from the menu and check the program state in the Variables view. For debug details, refer to ["6.2.6.1 Debugging"](#).



Point

.....

If the `[Java Applet]` launch configuration has not been created yet, create it from `[Run] > [Debug Configurations]`.

In addition to checking values, values can be changed from the Variables view.

.....

Processing that has been interrupted by the debugger can be restarted by selecting `[Run] > [Resume]` from the menu. The result is displayed in the message box. Check that the country name and total population are displayed for the entered ranking.

5) Distributing the Applet to the Operating Environment

To distribute this applet to the operating environment, create HTML and JAR files.

5-1) Exporting the Applet

The JAR file is created with the Export wizard. To launch the Export wizard, select `[File] > [Export]`. From `[Export]` wizard, select `[Java] > [JAR file]`.

The JAR Export wizard is displayed.

Check and enter the following setup items. After the following information is set, click [Finish].

Setup Items	Setup Content
Select the resources to export	Select only the following files in the AppletSample/src/sample folder: <ul style="list-style-type: none"> - CountryData.java - PopulationRanking.java - PopulationRankingApplet.java Deselect any other files that are selected.
Export generated class files and resources	Select
JAR file	AppletSample\AppletSample.jar
Compress the contents of the JAR file	Select

5-2) Creating the HTML file

Create the HTML file and define the applet. This applet uses the template HTML file that was generated automatically when the project was created without any changes. Use an editor to open the PopulationRankingApplet-JBKPlugin.html that was generated automatically in the project and confirm that the following definitions are there:

Property name	Value
code	sample.PopulationRankingApplet.class
archive	AppletSample.jar

Refer to the "J Business Kit Online Manual" for information on how to create a new HTML file.

5-3) Distributing to the Operating Environment

Copy the following files (created above) to the same folder on the Web server of the operating environment:

- PopulationRankingApplet-JBKPlugin.html
- AppletSample.jar



Point

.....

The HTML file and applet can be deployed to different locations by adding the codebase attribute to the applet definition that is defined in the HTML file. Refer to the "J Business Kit Online Manual" for details.

.....

7.3 Tasks

This section explains, for each development task, how to use Interstage Studio to develop Java applications.

- [7.3.1 Preparing the Environment for Creating Java Applications](#)
- [7.3.2 Creating a Class](#)
- [7.3.3 Creating an Applet](#)
- [7.3.4 Creating a Form](#)
- [7.3.5 Creating JavaBeans](#)
- [7.3.6 Editing a Java Form](#)
- [7.3.7 Editing the Screen Control Panel and Window Control Panel](#)

- [7.3.8 Defining BeanInfo](#)
- [7.3.9 Describing User Interface Processing](#)
- [7.3.10 Performing Environment Setup for Java Applications](#)
- [7.3.11 Verifying Java Application Operation](#)
- [7.3.12 Distributing a Java Application to the Operational Environment](#)
- [7.3.13 Developing Java Applications with JDK 6](#)
- [7.3.14 Notes](#)

7.3.1 Preparing the Environment for Creating Java Applications

To create Java applications, first create a project and prepare an environment in which the build can occur.

Creating the Project

From the [New] wizard, select [Java] > [Java Application Project] to create the Java application project.

Setting the Classpaths

If there are libraries for creating applications, set the classpaths. Set the classpaths in the [Java Build Path] property of the project.

7.3.2 Creating a Class

Use the Java Class Wizard to create a Java class file and then use an editor such as the Java editor to edit it. The methods for doing this are explained below.

Creating a new class file

Select [Class] from the New Wizard and then use the Wizard to create a class file.

Editing a class file

Use the Java editor to edit a class file. The Java editor is a text editor with the following features:

- Syntax highlighting
Uses colors and font styles (for example bold type) to highlight keywords
- Content Assist or Code Assist
When code is being created, displays Content Assist (also referred to as Code Assist)
- Code format
Formats source code according to the code format settings
- Import support
Allows a suitable import statement to be inserted according to the classpath set in a project
- Protection of an immutable source of a Java Form
Protects sections of source code that invoke information or event processing of a Java Form screen (Bean) in order to prevent it from being modified
- BeanInfo definition
Defines BeanInfo for JavaBeans and adds BeanInfo information as a comment

Refer to "Java editor" in "Concepts" in the "Java Development User Guide" for details on the Java editor.

7.3.3 Creating an Applet

Use the Applet Wizard to create an applet source and then use the Graphical Editor or the Java editor to edit it. The methods for doing this are explained below.

Creating a new applet

This is a Java Form with `java.applet.Applet/javax.swing.JApplet` and its extended class as a superclass.

Select [Java] > [GUI] > [Applet] from the [New] wizard and then use the Wizard to create a new applet.

The settings in the Wizard include the following:

- Package name
Specify the package name of the source to be generated.

- Applet name
Specify the class name of the applet source.

- Config
Specify the bounds, font, foreground color, and background color of the applet.

- Extended class details
Specify the base class.

- Generate HTML and Generate HTML for JBK plug-in
Specify the generation of HTML for launching an applet or for launching an applet for JBK Plugin, as well as the HTML page title and the width and height of the applet to be described in the HTML.

- Parameter information
This is parameter information that is to be reflected in the automatically generated HTML file. Pairing the names and values of applet parameters allows a number of pairs to be specified.



If another project class or library class has been specified in the superclass of the applet, the automatically generated HTML template must be corrected. This is because a superclass will be required in order to execute the applet.

For example, if the `MyApplet` class of the applet is in "project1.jar" and if the superclass is in `baseclass.jar`, add "baseclass.jar" to the archive attribute of the applet tag.

```
<applet code=MyApplet.class width=400 height=400 archive="project1.jar, baseclass.jar">
</applet>
```

When the applet is being built, an error message may be displayed stating that an abstract method has not been listed. If this happens, start an editor such as the Java editor and then describe the abstract method that was displayed in the error message.

7.3.3.1 Applet Information

Item	Description
Source Folder	Select the source folder to store the applet source.
Package	Enter the package name if the package name of the class to be generated is to be specified.

7.3.4 Creating a Form

To create a form, use the Form Wizard to create a source and then use the Graphical Editor or the Java editor to edit it. The methods for doing this are explained below.

7.3.4.1 Creating a Frame, Dialog, Panel

Forms can be classified into the following types. A form can be made up with any frame, dialog box, or panel class as a superclass. A superclass is assumed to be located above a classpath.

Select [Java] > [GUI] > [Form] from the [New] wizard and then use the Wizard to create a form.

Item	Description
Frame	This is a Java Form with java.awt.Frame/javax.swing.JFrame and its extended class as a superclass. This form contains frames of ordinary windows made up of elements such as pull-down menus and title bars.
Dialog	This is a Java Form with java.awt.Dialog/javax.swing.JDialog and its extended class as a superclass. This form is similar to a frame in appearance but it has functions that differ from a frame, including the fact that it can be modal and that title bar icons cannot be specified.
Panel	This is a Java Form with java.awt.Container/java.awt.Window/java.awt.Panel/javax.swing.JPanel and its extended class as a superclass. A superclass that is handled by another class extended form will not be a panel.

Point

To extend a specific class, including a class created by the user, select a frame, dialog, or panel.

Note

When a frame, dialog, or panel is being built, an error message may be displayed stating that an abstract method has not been listed. If this happens, start an editor such as the Java editor and then describe the abstract method that was displayed in the error message.

7.3.4.2 Creating a Screen Control Panel, Screen Control Tabbed Panel, Window Control, and Object Control

This section describes how to create a new Screen Control Panel, Screen Control Tabbed Panel, Window Control, and Object Control.

Select [Java] > [GUI] > [Form] from the [New] wizard and then use the Wizard for creating.

Each Wizard and the points of difference in editing are shown below.

	Screen Control Panel	Screen Control Tabbed Panel	Object Control	Window Control
Superclass	JFLCardPanel	JFCTabbedPanel	JFCObjectLoaderManager	JFCWindowLoaderManager
Superclass of object being controlled	JFLEntryPanel	JFLEntryPanel	java.lang.Object	JFCFrame or JFCDialog
Name on window for editing object being controlled	Screen name	Screen name	Object name	Window name
Edit start comment	//@@JFLCardPanel createPanels Method start	//@@JFCTabbedPanel createPanels Method start	// @@JFCObjectLoaderManager createObjects Method start	// @@JFCWindowLoaderManager createWindows Method start

	Screen Control Panel	Screen Control Tabbed Panel	Object Control	Window Control
Edit end comment	//@@JFLCardPanel createPanels Method end	//@@JFCTabbedPanel createPanels Method end	// @@JFCObjectLoaderM anager createObjects Method end	// @@JFCWindowLoa derManager createWindows Method end

Note

Unless otherwise noted, the package name for each class is com.fujitsu.jbk.gui.ctrl. The part between the edit start comment and the edit end comment will be the target of editing by the editing tool.

The Object Control Wizard generates a createObjects method so that an add method will be invoked from the constructor. Note that a createObjects method will not be generated if creation is performed by specifying an existing resource.

In Object Control editing, there is no function for creating or editing an object on the side being controlled.

In Window Control editing, if a dialog has been specified as a class on the side being controlled, rather than specifying the timing of generation or deletion, specify modal or modeless. If a frame has been specified, specify the timing of generation and the timing of deletion in the same way as for other classes. In Window Control editing, if resources are selected from project resources, they will be automatically classified as dialogs and frames.

7.3.4.3 Java Form Information

Item	Description
Source Folder	Select the source folder to store the Java Form source.
Package	Enter the package name if the package name of the class to be generated is to be specified.

7.3.5 Creating JavaBeans

Many types of JavaBeans exist and each has a different creation method.

- Uni Bean

A Uni Bean is a Bean that is made up of a single component such as a text field or a button. An existing Uni Bean is created as a new Uni Bean by being extended and given an additional function.

- Compound Bean

A Compound Bean is a Bean that is made up of a number of Beans. A Compound Bean is created by using the Java Form Designer to paste Beans on a panel (JPanel or Panel), in the manner of creating the layout of a form.

- Invisible Bean

If a class that has no view has been specified in a superclass, an Invisible Bean will result. As an Invisible Bean has no view, it is edited using the Java editor in the same way as an ordinary Java source.

Select [Java] > [GUI] > [JavaBeans] from the New Wizard and then use the Wizard to create a Uni Bean or an Invisible Bean.

The settings in the Wizard include the following:

- Package name

Specify the package name of the source to be generated.

- Bean name

Specify the class name of the Bean.

- Generate Beaninfo

This automatically creates BeanInfo information. The properties, methods, and events of the parent of an extending class are included

as default BeanInfo information. If you do not select this option, the wizard will create empty BeanInfo information.

- Config
Specify the rectangle, font, foreground color, and background color of the Bean.
- Extended class details
Specify the base class.

After creating a form (panel), perform BeanInfo definition by selecting [Edit] > [Define BeanInfo] > [BeanInfo Definition] from the workbench menu bar, in order to create a Compound Bean. Refer to "[7.3.8 Defining BeanInfo](#)" for details on BeanInfo definition.

7.3.5.1 JavaBeans Information

Item	Description
Source Folder	Select the source folder to store the Bean source.
Package	Enter the package name if the package name of the class to be generated is to be specified.

7.3.6 Editing a Java Form

Editing a Java Form

To edit a Java Form, the Java Form Designer can be used to perform the following operations:

- Setting Beans
From the object palette, select and lay out the Beans and then set the attributes.
- Defining an exclusive selection group
Select [Tools] > [Button Groups] from the Java Form Designer menu bar.
- Decorating a Java Form
Draw elements such as lines and squares to decorate the Java Form.

Screen configuration of the Java Form Designer

The Java Form Designer screens are made up of the following five windows, editors, and views. Use these screens to edit Java Forms:

- Java Form Designer window
Allows a Java Form to be saved and provides editing functionality, such as copying a Bean
- Properties window
Displays a list of Beans and allows properties to be viewed and modified
- Java Form window
Allows beans to be positioned on a Java Form and the layout edited
- Java editor
Allows event processing and user-defined methods for Beans to be described
- Bean List view
Displays a list of Beans and allows methods and events to be inserted



Note

If the Java editor has been used to open a Java Form, sections of the source displayed with a different background color cannot be edited. The source in these sections invokes screen (Bean) information or event processing. Modification is also prohibited for "`// Graphical Editor Form`" at the beginning and for the part enclosed within "`//@@Form Design Information start`" and "`//@@Form Design Information end`". Once these parts are edited, they cannot be opened by the Graphical Editor.

7.3.6.1 Setting a Bean

Use a mouse to perform operations such as laying out a new Bean, modifying the size, shifting, and copying.

- Laying out a new Bean
Allows a new Bean to be easily dragged and dropped from the Java Form Designer
- Laying out a new Bean
Allows a new Bean to be easily dragged and dropped from the Java Form Designer

7.3.6.2 Laying out a Bean

The layout manager is provided as a method of laying out a Bean using a Java application. When the layout manager is used, the layout will be corrected automatically according to the modifications made to the screen size but will be subject to the constraints of the screen formats supported by the layout manager.

From the viewpoint of screen extendibility, handling modifications made to the screen size is desirable, so using the layout manager in the screen design of Java applications is recommended.

Layout characteristics

- FlowLayout
Lays out Beans horizontally
- BorderLayout
Lays out Beans from top to bottom, left to right
- GridLayout
Lays out Beans in a grid with equal vertical and horizontal sizes
- GridBagLayout
Lays out Beans in a grid in the same way as GridLayout, but allows detailed setting of sizing rules
- CardLayout
Used for switching panels



.....

If Bean drawings overlap, a Heavyweight will be drawn over a Lightweight. In addition, if the drawings overlap among components, they will be determined according to the order in which the components were added to a container. The order in which they were added is their order on the component tree. If the layout manager specification is other than "none", there will be no overlapping among drawings within the same component.

.....

7.3.6.3 Referencing or Setting Bean Properties

Bean properties can be classified into the following types:

- Standard Properties
- Original Properties

List of Standard Properties

Standard Properties are typical properties of a Java Form or Bean. The list of Standard Properties is as follows:

Property
beanName
layout
bounds
border

Property
constraints
addIndex
visible
background
foreground
font
AWTName
toolTipText
attention

Note

The bounds property displays the values of the width and height of the client area of a window, but the values of the width and height of the window itself will be displayed if [Edit as Execution Image] has been selected.

Original Properties

Original Properties are displayed based on information that has been specified in the BeanInfo information provided by various Beans.

If a Original Properties is a read-only property, a write-only property, or an invisible property, it will not be displayed.

In addition, some properties will not be displayed if [Show expert property] has not been selected in the properties of the Environment Setup options.

Note

For projects in which the Peculiar Property name begins with an asterisk (*), the initial value of [Value] cannot be changed.

Information in the properties sheet

The properties sheet of the properties window is used to reference or set properties. The following information will be displayed in the properties sheet:

- Standard tab
Displays the Standard Properties of a Bean
- Original tab
Displays the Peculiar Properties of a Bean
- Property Name
Displays the names of the properties
- Value
Displays the values of the properties being set. When these values are rewritten, the properties of a Bean can be modified. Depending on the property type, values are selected either from a list or by selecting a numeric value using a spin button.
- Description
Descriptions of the selected properties are displayed in the lower part of the properties sheet

Functions of the properties sheet

The buttons in the properties window have the following functions:

- Show Customizer
If the Beans being selected are being provided by the Customizer, displays the Customizer

- Select from the list
If the properties being selected are being provided by the properties editor, displays the properties editor
- Set null value
Sets nulls in the properties being selected. These can be set for properties that are not basic data types in Java.
- Reset the value
Reverts the values of the properties to the standard values
- Undo
Reverts the values of the properties to the status they had before the last modification

Point

The Java Form Designer has a function for editing the main properties of a Bean on a form, which is referred to as the spot edit function. To use the spot function, from the Java Form Designer menu bar, go to [Settings] > [Options] > [Form] tab and then select [Enable spot editing].

When you click on the Bean being selected or when you press the [Ctrl+E] keys simultaneously, the spot edit status will result, and a label indicating that spot editing is in progress will be displayed in the upper part of the Bean.

To end the spot edit, press the [Enter] key, or for a spot edit that allows multiple line input, press the [Ctrl+Enter] keys.

7.3.6.4 Creating Bean Relationships

The Java Form Designer allows Bean Relationships to be created in an interactive format in Java Forms, applets, and Java programs and allows these Bean Relationships to be visualized.

Creating Bean Relationships

[Bean Relationships Wizard] can be launched using the method shown below. When you follow the instructions of the Wizard to set a source Bean event and a target Bean method or property, a Bean Relationships source will be created in the description of the event processing of the specified Bean.

- Launch the menu.
Select [Tools] > [Bean Relationships Wizard] from the Java Form Designer menu bar.
- Launch wiring.
 1. Right-click the Bean that is to perform event processing of the Java Form (source Bean).
 2. While holding down the right mouse button, drag the mouse. While you are dragging, a line (wire) will be displayed from the position at which the mouse was right-clicked to the current position of the mouse cursor.
 3. While still holding down the mouse button, move the mouse and then release the right mouse button on the Bean that is to invoke a method, obtain a property, or set a property (target Bean).

Editing Bean Relationships

When you edit the values that have been specified in [Bean Relationships Wizard] and click [Create], the Bean Relationship source will be updated in the description of the event processing that corresponds to the Visualized Bean Relationships (directional lines).

To start the Wizard:

- Double-click the Visualized Bean Relationships (directional lines) of the Java Form.
- Right-click the Visualized Bean Relationships (directional lines) of the Java Form and display the context menu. Select [Edit by Wizard].
- Select [Tools] > [Check Bean Relationships] from the Java Form Designer menu bar. Select the Bean Relationships to be edited from the Bean Relationships list on the [Check Bean Relationships] screen and then click [Edit by Wizard].

Deleting Bean Relationships

Bean Relationships can be deleted by right-clicking the Visualized Bean Relationships (directional lines) of the Java Form and clicking [Delete] from the context menu.



Deleting Bean Relationships can also be performed from the Bean Relationships confirmation screen.

Displaying Bean Relationships

To confirm Bean Relationships, select [Tools] > [Check Bean Relationships] from the Java Form Designer menu bar and launch the [Check Bean Relationships] screen.

The [Check Bean Relationships] screen displays a list of all the Bean Relationships and allows the creation, edit, and deletion of Bean Relationships.

Analyzing Bean Relationships

Bean Relationships are analyzed automatically, but it is also possible to perform optional analysis by selecting [Edit] > [Re-analyze Bean Relationships] from the Java Form Designer menu bar.

7.3.6.5 Defining a Menu

Menu definition

If pull-down menus are to be used, select [Tools] > [Menus] from the Java Form Designer menu bar and launch menu definition. In menu definition, define the configuration of the menus and information such as the character strings to be displayed in the menus.

Menu definition

Menu objects that have been defined in Java Forms will become the class fields of the Java Forms. There is usually no need to use these fields to operate menus directly, but objects can be operated via class fields in order to perform special processing.

An example of operating a menu object is shown below.

Example

If an "APPLY" attention occurs, a "cancel" menu object is masked.

```
public boolean processAttention_APPLY() {
    cancel.setEnabled(false);
    return true;
}
```

Attentions

Attentions are used to define the actions for menu operations. Attentions are the names given to describe the general operations of both the screens and menus. They are used to define the operation of menus, buttons, or keys.



It is essential for an action event to have been defined in order for an attention to be caused, so this is no different from an ordinary action event. Therefore, as attentions may require more time and effort, their use is not recommended.

7.3.6.6 Defining an Exclusive Selection Group

An exclusive selection group is a function that groups exclusive selection type Beans (radio buttons) and automatically sets the selection status. Only one Bean in a single exclusive selection group will have a status of selected; the other Beans will be set with an unselected status.

Method of defining an exclusive selection group

1. Displaying the setup dialog box of the exclusive selection group
Select [Tools] > [Button Groups] from the Java Form Designer menu bar and start defining the exclusive selection group.
2. Creating a new exclusive selection group
Click [Add JFButtonGroup(JBK)], [Add CheckboxGroup(AWT)], or [Add ButtonGroup(Swing)] to create a new selection group.
3. Setting group members
Select a created group name that from the [Groups] list and then click [Edit Member]. The [Exclusive Selection Members] screen will be displayed. Set the group members in this screen.



Note

Default exclusive selection groups (group names: __cdDefaultJFButtonGroup, __cdDefaultCheckboxGroup, and __cdDefaultButtonGroup) have been defined in advance. If an exclusive selection type Bean has been pasted onto a Java Form, it will be automatically added to the default exclusive selection groups. The default exclusive selection groups cannot be deleted.

Direct operation of exclusive selection groups

To perform special processing, objects can be operated using class fields.

[Class field name]

Name specified by "group object name" when a group has been added using the exclusive selection group definition dialog box

[Class field type]

Type of exclusive selection group that has been created	Class field type of exclusive selection group object
Group for AWT	java.awt.CheckboxGroup
Group for Swing	javax.swing.ButtonGroup

Example

If an "APPLY" attention occurs, red will be set as the background color of the object that has been exclusively selected in the "select" exclusive selection group for AWT.

```
public boolean processAttention_APPLY() {
    select.getSelectedCheckbox().setBackground(java.awt.Color.red);
    return true;
}
```

7.3.7 Editing the Screen Control Panel and Window Control Panel

7.3.7.1 Editing a Screen Control Panel

Creating a new Screen Control Panel using existing resources

To open a Screen Control Panel using existing resources, select [Java] > [GUI] > [Form] from the [New] wizard and perform the following procedure:

1. Use the [New Form] Wizard to specify a source folder and a package.
2. Select the [New Java Form] dialog box > [Java] tab > [Screen Control Panel] icon and then click [OK].
3. Select [Convert source to which Screen Control Panel Editor can edit.] and then click [Next].
4. Enter a Java source name in the [Original source] column and then click [Convert].
5. Alternatively, click [Select] to select a Java source to be converted.

6. Click [Create]. The Screen Control Panel edit function screen will display and a backup of the source to be converted will be taken.

Saving a Screen Control Panel

To save a Screen Control Panel that is being edited, use the [File] menu of the Screen Control Panel.

Editing a Screen Control Panel

In the following items, editing can be performed on elements such as the order, name, timing of generation, and timing of deletion of the screen parts to be set. Refer to "UI Screen Control Library" in the "JBK GUI Library User's Guide" for details.

Item	Description
Name	Screen name to be registered
Class name	Class name to be registered
Create timing	Time at which the screen part is generated <ul style="list-style-type: none"> - At initialization: In response to JFCPanelLoader.BUFFERED_INSTANCE - Background: In response to JFCPanelLoader.BACKGROUND_CREATION - At display: In response to JFCPanelLoader.DELAYED_CREATION
Release timing	Time at which the screen part is deleted <ul style="list-style-type: none"> - Timekeeping of nondisplay: In response to JFCPanelLoader.BUFFERED_INSTANCE - Timekeeping: In response to JFCPanelLoader.PERMANENT_INSTANCE - At nondisplay: In response to JFCPanelLoader.TEMPORARY_INSTANCE

7.3.7.2 Editing a Window Control Panel

Saving a Window Control Panel

To save a Window Control Panel that is being edited, use the [File] menu of the Window Control Panel.

Editing a Window Control Panel

In the following items, editing can be performed on elements such as the order, name, timing of generation, and timing of deletion of the screen parts to be set. Refer to "UI Screen Control Library" in the "JBK GUI Library User's Guide" for details.

Item	Description
Classification	Frame or dialog box.
Window Name	Window name to be registered.
Class name	Class name to be registered.
Create timing(For the frame)	Time at which the window part is generated: <ul style="list-style-type: none"> - At initialization: In response to JFCPanelLoader.BUFFERED_INSTANCE - Background: In response to JFCPanelLoader.BACKGROUND_CREATION - At display: In response to JFCPanelLoader.DELAYED_CREATION
Release timing(For the frame)	Time at which the window part is deleted: <ul style="list-style-type: none"> - Timekeeping of nondisplay: In response to JFCPanelLoader.BUFFERED_INSTANCE - Timekeeping: In response to JFCPanelLoader.PERMANENT_INSTANCE

Item	Description
	- At nondisplay: In response to JFCPanelLoader.TEMPORARY_INSTANCE
Modal or not(For the frame)	Switch between modal and modeless when the dialog box is displayed.

7.3.8 Defining BeanInfo

BeanInfo is interface information for programs that use Beans as components. Use BeanInfo definition to define BeanInfo information. Open a Bean file and select [Edit] > [Define BeanInfo] > [BeanInfo Definition] from the workbench menu bar to invoke BeanInfo definition.

Point

The BeanInfo definition menu will only be enabled if the Java editor of the Bean file is focused. Open the menu after checking that the focus is on the editor.

Defining BeanInfo includes the following:

- Defining properties
To add Bean properties, enter information using the [Add Property] dialog box that is displayed by clicking [Add Property].
- Definition fields
Bean properties include the following special information:

Item	Description
Edit class name	Specify the dedicated class name for editing properties, with the package name attached. The edit class name can be omitted.
Invisible for internal use	Select this to make a property that cannot be used by ordinary users.
The feature for the expert	Select this to make a property that is for experts.
Attribute information	Allows a simple description of the property to be described

Point

[Invisible for Internal use], [The feature for the expert], and [Attribute information] are common to each definition field of a property, method, and event.

- Defining a method
To enable a method, select the method name in the tree display area and then click [Enable]. To make the method of an extending class the method of a Bean, give [Show Public Methods of Extended Classes] a selected status, select the method name of the extending class in the tree display area, and then click [Enable].
To disable a method of a Bean, select the method name in the tree display area and then click [Disable].
- Defining an event
To add a Bean event, click [Add Event]. Enter the required information in the [Add Event] dialog box that is always displayed and then click [OK].
To make a superclass event a Bean event, select an event name from the tree display area and then click [Enable].
To disable a Bean event, select the event name from the tree display area and then click [Disable].
When generating event object classes and event listener interface source, select [Edit] > [Define BeanInfo] > [Generate event source] from the workbench menu bar. Edit the generated source just as you would normal Java source in a Java editor.

Note

The properties, methods, and events generated using BeanInfo definitions remain in the source even if they are disabled. Delete with a Java editor to delete them from the source.

7.3.9 Describing User Interface Processing

This section explains the method of creating user interface processing. This is relevant for forms, applets, and the visible Beans of JavaBeans.

7.3.9.1 Describing Processing of an Application with a User Interface

This section explains the processing of an application that displays a frame, as an example of an application with a user interface.

The processing includes the following:

- Main program: Initialization of the application and display of a form
- Initialization processing: Initialization processing of the form
- Event processing: Processing of events generated by actions such as user operations

Example of main program that displays a frame

```
public class MyJavaApp {
    //Constructor
    public MyJavaApp() {
    }
    //Method that displays a frame
    public void run() {
        //Create a frame instance.
        Frame1 form = new Frame1();
        //Use setVisible method of the frame instance and enable display.
        form.setVisible(true);
    }
    //Main processing
    public static void main(String[] args) {
        //Create a MyJavaApp class instance.
        MyJavaApp object = new MyJavaApp();
        //Invoke a run method of the MyJavaApp class instance.
        object.run();
    }
}
```

7.3.9.2 Operating a Form

Forms are made up of frames, dialogs, and panels.

This section explains the description of the processing, using frames as an example.

Frame operation

This section explains frame operation as an example of operation.

- Sharing information

If data is to be shared between the main program and the Java Form or among Java Forms, either define the data in the main program and reference that data in each form, or add a constructor and pass the data that is to be shared as a parameter.

- Frame display method

An example of a frame display method is shown below.

```
If a frame called Frame1 is to be invoked from an application or existing frame
    Frame1 form = new Frame1();    //Create a frame instance.
    form.init();                  //Initialize the frame.
    form.setVisible(true);        //Display the frame.
```

- Frame close method

To close a frame, describe "this.dispose();".

7.3.9.3 Operating a Bean

This section explains how to operate a Bean.

A Bean that has been pasted into a Java Form will be generated as a class field of a Java Form class. The class field will have the name specified in the "Bean name" property when the Java Form was defined.

Bean operation is performed by setting and referencing Bean properties and invoking Bean methods.

Referencing and setting properties

An example of referencing and setting properties is shown below.

A focus_focusGained event to be generated when the text field Bean "textField1" gains focus is used to modify the background color (background) property to blue. In addition, a focus_focusLost event to be generated when focus is lost is used to return the background color (background) property to white.

```
[Processing of TextField1]
public void textField1_focus_focusGained(FocusEvent e) {
    if (!defaultEventProc(e)) {
        textField1.setBackground(java.awt.Color.blue);
    }
}
public void textField1_focus_focusLost(FocusEvent e) {
    if (!defaultEventProc(e)) {
        textField1.setBackground(java.awt.Color.white);
    }
}
```

Invoking a method

An example of invoking a method is shown below.

When the push button "button1" is pressed, the characters that have been entered in the "textField1" text field are added to the "list1" list box.

```
[Processing of button1]
public void button1_action_actionPerformed(ActionEvent e) {
    if (!defaultEventProc(e)) {
        list1.addItem(textField1.getText());
    }
}
```

7.3.9.4 Describing Event Processing

There are three methods of adding event processing:

- Using the Java editor to describe event processing directly
Use the Java editor to describe a direct method for each event.
- Using the Bean Relationships Creation Wizard to generate event processing
When this wizard is used, processing can be generated interactively rather than via programming. Refer to "[7.3.6.4 Creating Bean Relationships](#)" for details.
- Using the Bean List view to insert event processing
The Bean List view displays a list of the Java Forms and Beans that have been edited by the Java Form Designer and has a function to insert methods and events. The Bean List view has the following functions:
 - Displaying a list of the Beans that have been edited by the Java Form Designer
Allows the Beans that have been edited in a Java Form to be displayed in tree format
 - Displaying a list of the properties, methods, and events of each Bean
Expands the properties, methods, and events to allow these details to be listed for each Bean

- Inserting methods in the source
Allows the [Method Insert] dialog box to be used to easily insert methods in the source
- Inserting events in the source
Allows events to be easily inserted in the source

7.3.10 Performing Environment Setup for Java Applications

7.3.10.1 Registering a Bean

Registering a created Bean in the Java Form Designer will allow it to be pasted on a Java Form, in the same way as for functions such as J Business Kit. Register a Bean by performing the following procedure from the [Settings] > [Bean Registration] dialog box, accessed via the Java Form Designer menu bar:

1. Click [Add Bean] in the [Edit of palette] tab. The [Add Bean] dialog box will be displayed.
2. Enter a JAR file name or a Bean class name. The specified Bean will be added to "Bean which can be displayed in the palette".
3. Add the Bean that has been added to "Bean which can be displayed in the palette" to "Bean displayed in the palette".

7.3.10.2 Custom Wizards

The Wizards for Java Forms, applets, and JavaBeans can be customized in the following ways:

Select [Settings] > [Edit Wizard] from the Java Form Designer menu bar to configure each item.

- Editing a Wizard
Allows the layout, tabs, name, and icons to be set
- User-set forms Wizard
If the superclass of a class extended form has been predetermined, a custom Wizard with a fixed superclass can be created by using the [Add or Edit Wizard] dialog box.
Select [Customize Panel], [Customize Frame], [Customize Dialog], or [Customize Applet] from the [Available Wizard] list.



Note

java.applet.Applet and java.awt.Frame cannot be specified in a user-set panel. Use a user-set frame or a user-set applet to specify these.

7.3.10.3 Setting Options

In the Java Form Designer, it is possible to set various options by selecting [Settings] > [Options] from the Java Form Designer menu bar. The special items are described below.

- [Form] tab

Item	Description
Adjust to Grid	Specifies whether or not a newly pasted Bean is to be positioned to fit the boundaries of a grid. When the Bean is shifted, the upper left of the Bean will be positioned along the boundaries of the grid.
Create with preferred size	A new Bean on a form is created using the "preferred size", regardless of the rectangle that was specified using the mouse

- [Properties] tab

Item	Description
Show expert property	Displays properties for experts in the properties sheet
Settings	Displays the [Select Font] dialog box

Note

For the properties option, the size and style cannot be set for the font.

- [Editor] tab

Item	Description
Deletes the event description when Bean is deleted	When a Bean is deleted, the event descriptions that correspond to the Bean are deleted from the source
Deletes target Bean Relationships when Bean is deleted	When a Bean is deleted, the Bean Relationships in which the Bean is a target are deleted from the source
Replace the unconditionally when the Bean is renamed	When the name of a Bean is changed, replaces the new name for the Bean name character string in the source

Note

If [Deletes the event description when Bean is deleted] or [Deletes target Bean Relationships when Bean is deleted] has been specified, the event processing descriptions and Bean Relationships cannot revert to their original state by means of a revert operation. Take note of this when deleting a Bean.

If [Replace the unconditionally when the Bean is renamed] has been specified, because the character string that corresponds to the Bean name in the source will be replaced unconditionally, unintended replaces may occur. Take note of this when changing the name of a Bean.

- [Environment] tab

Item	Description
Search path of base form	Displays a search path list for Base forms. If a Base form is read, searching is performed using the order of this search path list.

- [Wizard] tab

This tab sets information on forms generated by a Wizard and sets initial values for each property when a new Java Form is created.

Item	Description
Class Details	Sets initial values for Swing or AWT
From active form	Obtains values for each item from active forms that are currently being displayed

- [Visualize Bean Relationships] tab

This tab sets the visualization method for Bean Relationships and the color at visualization.

Item	Description
Way of Visualizing Bean Relationships	Selects a visualization method for Bean Relationships. The [Detailed setting] button can be used to configure detailed settings for the visualization method: <ul style="list-style-type: none">- Visualize all Bean Relationships: Select when Bean Relationships are to be visualized for all Beans.- Visualize Bean Relationships about last selected Bean: Select when Bean Relationships are to be visualized for the last selected Bean.

Item	Description
	<ul style="list-style-type: none"> - Don't visualize all Bean Relationships: Select when none of the Bean Relationships are to be visualized.
Colors	<p>Sets the color used when Bean Relationships are visualized. When the [Setting] button is clicked, the [Select Color] dialog box is displayed:</p> <ul style="list-style-type: none"> - A color at Visualizing by Bean Relationships: Sets the color used when Bean Relationships are visualized - A color at Visualizing by Bean Relationships set as a parameter: Sets the color used when Bean Relationships that are to be set as parameters are visualized. Bean Relationships that are to be set as parameters are indicated by the "jTextField1.getText()" part in the following example: jLabel1.setText(jTextField1.getText()); - A color of selected Bean Relationships: Sets the color used when visualized Bean Relationships are selected

- [Detail of a way of Visualizing about Bean Relationships] dialog box
This dialog box sets the details of the visualization method for a Bean.

Item	Description
Only visualize Bean Relationships in specified events	<p>Select when Bean Relationships are to be visualized for specific events. Standard events: Select the event name for which Bean Relationships are to be visualized. Other events: Specify event names that are not standard events using commas, semicolons, or blank spaces as delimiters.</p>
Visualize Bean Relationships set as a parameter	<p>Select when Bean Relationships that have been set as parameters are to be visualized. Bean Relationships that have been set as parameters are indicated by the "jTextField1.getText()" part in the following example: jLabel1.setText(jTextField1.getText());</p>

- [Other] tab
Set other items.

Item	Description
Display non-recommended function	<p>The deprecated items shown below are valid. These features are likely to be removed in the future and are introduced here in the interests of compatibility. Do not use if they can be avoided.</p> <ul style="list-style-type: none"> - The class non-inheriting wizard, and the tabs for the items (applet, frame, dialog box, panel) - Bean serialization - Java Form attributes - Saving and restoring Bean status - Focus traversal order definition

7.3.10.4 List of Forms

This displays a list of forms opened with a Java Form Designer. Either double-click the form selected in the list, or click [Show] to display that form at the front.

7.3.11 Verifying Java Application Operation

Debug using the method below that suits the developed Java application. The [Debug Configurations] dialog box can be opened from the workbench menu bar by selecting [Run] > [Debug Configurations].

- Use the [Java Application] launch configuration.
If debugging a created Java application, use the Java application launch configuration. In the [Debug Configurations] dialog box, select [Java Application].
- Use the [Java Applet] launch configuration.
An applet can be debugged using the Java applet launch configuration. When debug is started, the applet view starts and debug operations can be performed. In the [Debug Configurations] dialog box, select [Java Applet].

Note

When debugging swing-base applets (applets inheriting the javax.swing.JApplet class), the javax.swing.TimerQueue class opens in the class editor when debug ends. To prevent this action, from the workbench menu bar, select [Window] > [Preferences] > [Java] > [Debug] and deselect [Suspend execution on uncaught exceptions].

- Use the [Remote Java Application] launch configuration.
An application can be debugged remotely by starting the application in debug mode and using the remote Java application launch configuration. In the [Debug Configurations] dialog box, select [Remote Java Application].
- Debugging JavaBeans
If the JavaBeans were generated using the workbench wizard, the main method is provided as the test driver. This is commented out immediately after generation. The JavaBeans can be easily debugged as Java applications by enabling the main method.
- Debugging with applet displayed in a browser
To perform debugging by running an applet in Internet Explorer or another browser, use the JBK plug-in and perform remote debugging. The applet can be run in debug mode by adding coding similar to the example below to jbkplugin.properties in the classes folder located in the folder where JBK runtime is installed.

```
Example :  
jbk.plugin.vmooption=-Xrs -agentlib:jdwp=transport=dt_socket,server=y,suspend=n,address=nnnn
```

Information

nnnn is the port number used for communication between the debugger and the program being debugged. The value specified in the jbk.plugin.vmooption properties must be set as the port number used on the debugger side for communication during remote debugging.

Point

A created launch configuration does not need to be re-created every time debugging is performed. To perform debugging using the same launch configuration, select the created launch configuration from [Debug Configurations], and then click [Debug] to start debugging.

7.3.12 Distributing a Java Application to the Operational Environment

Distribute the Java project to the operating environment by exporting a JAR file.

JAR File Package (JAR Export)

Use the export wizard to create the JAR file. To start the export wizard, select [File] > [Export].

Deploying applets

Include the description of the JAR file in the HTML file used to deploy the applet.

7.3.13 Developing Java Applications with JDK 6

Java applications can be developed in the Java EE 6 workbench using JDK 6.

The following describes the procedure for developing Java applications using the Java EE 6 workbench, focusing on the differences from the standard workbench.

7.3.13.1 Preparing the Environment for Creating Java Applications

Creating the Project

From the [New] wizard, select [Java] > [Java Application Project] to create the Java application project.

In [JRE], specify [Use an execution environment JRE] and select [JavaSE-1.6].

Setting the Classpaths

The build path for the project must be set if libraries, such as the JBK library, need to be added.

Refer to "[9.6.1.4 Setting a Classpath](#)" for details.

7.3.14 Notes

Points to consider on applets

This section explains the points to consider when developing and operating applets.

- Selecting a Web browser
Depending on which Web browser is used, there will be differences in areas such as the size in which an applet will be displayed on the screen. In addition, depending on the browser options, some applet operations may be able to be modified. Note should also be taken of the levels of support provided by the browser for JavaScript and HTML.
- Java Plugin
A Java Plugin is a mechanism for executing applets by using an external Java Virtual Machine (VM) (for example, a JDK Java VM), without using a Java VM (execution environment) provided by the Web browser. If a Java Plugin is to be used, the plugin must be installed in the client machine.
- Limiting server access
In order to prevent damage due to malicious applets, access from applets to local files or other servers is prohibited.
If a function that performs communication from applets to a database server is to be used, a database server environment must be created on the Web server on which the applets exist.
In order to alleviate the load on the Web server, place server applications such as servlets and Enterprise JavaBeans (abbreviated to EJB from now on) on the Web server so that the applets can communicate with the server applications and access other servers (such as database servers) from the server applications.
- Using EJB
EJB on the Web server can be used from applets.
When developing applets, add an Enterprise Bean stub to a project and join it to the JAR file of the project. Also, during operation, a library that is to be used by the EJB must be installed in a server.
- Using databases
If databases on the server are to be used from applets, a JDBC driver suited to the databases must be installed in the client machine.

Points to consider on Beans

The following points are specific to Beans:

- Split panes (JSplitPane)
In split panes, panels must be pasted onto the split parts. Therefore, when a split pane is pasted, a panel will be pasted automatically. This panel cannot be deleted.
- Scrolling (JScrollPane)
In items such as the Swing list box (JList), even if the contents exceed the display area of the Bean, a scroll bar will not be added automatically. To add a scroll bar, first paste a scroll pane (JScrollPane), and then paste an item such as a Swing list box (JList) above

it. This will ensure that a scroll bar will be displayed when it is needed. Note that only one Bean can be pasted into a scroll pane (JScrollPane).

For an AWT Bean, even if it is pasted into a scroll pane (JScrollPane), this scrolling action cannot be performed. AWT Beans have their own scroll function so there is no need to use a scroll pane.

- Heavyweights and panels

If a Heavyweight is to be pasted onto something, it is recommended that the pasting destination also be made a Heavyweight. For example, if an AWT button or an AWT text field is to be pasted onto a panel, use an AWT panel for the panel as well.

- Heavyweights and Lightweight

When Heavyweight Beans and Lightweight Beans overlap, the Heavyweight Bean will always be displayed over the top.

- Invisible Beans

For pasting a new Invisible Bean without the GUI, the pasting operation is performed on the container or Java Form in the same way as for a Visible Bean. However, once it has been pasted, the Invisible Bean will not be displayed on the Java Form but will be displayed in gray on the component tree. Operate the properties by selecting the Bean on the component tree.

- Tabs (JTabbedPane)

Tabs are made up of a number of pages. The following items explain the operations of adding and deleting pages and specifying page titles:

- Adding a page

A Bean that has been pasted directly onto a tab becomes one page of the tab. In general, one page is made up of a number of Beans. Therefore, for each page, first paste a panel (JPanel) (a panel can paste a number of Beans internally), and then paste a number of required Beans over the top. At this time, select the layout manager of the panel as required. Note that when a new tab has been pasted, a number of pages that are pasted with that panel will be generated automatically.

To actually add a page, select a panel from the object palette and start dragging from the position of the tab strip (location at which the title of the page being clicked is displayed at page switching). A page will be added when dragging a pasted Bean from the position of the tab strip. In the same way as when the position of the tab strip is below or adjacent to the tab, when dragging is started from below or adjacent to the tab strip, a page will be added. Note that if no pages are present, a page will be added regardless of the position from which dragging starts.

- Deleting a page

To delete a page from a tab, select the panel to which the page was pasted to perform a deletion operation for the Bean.

- Specifying a page title

To specify a page title, select the panel of the relevant page and specify the title character string in the "constraints" in the Standard Properties of the panel.

- Modifying the page order

To modify the page sequence, select the panel of the page to be modified and specify the order of addition to the Parent container.

- Differences in Bean design as a result of changes in the JDK version

If the JDK version is changed, differences may occur in the Bean design.

Notes on Java forms

The following describes some points to note regarding Java forms:

- Do not edit change-prohibited source from the package explorer, outline, or source menu.

Note: Change-prohibited source is between lines starting with `/// Graphical Editor Form` and `///@Form Design Information start` and ending with `///@Form Design Information end`.

- The background color of change-prohibited source cannot be changed.

- Java Form Designers are started separately for each project.

- The source for Java forms, applets, and JavaBeans cannot be refactored.

- Do not change the file name in the Navigator view. Use the Java Form Designer to save the Java form source with a different name. If the name is changed inadvertently, restore the original name and then open it again.

- If Java form class declarations get corrupted and cannot be parsed, restore the source for the part that cannot be parsed. The source from before restoration remains as is, without being commented out.

- The Graphical Editor cannot open link files or files that are in link folders.

- If an editor other than the Graphical Editor is used to edit a Java form, applet, or JavaBeans file, use a function such as the local history to restore the file to the state it was in before it was edited, then re-open with the Graphical Editor.
- The "Reference" menu in the Bean List view is not displayed in the standard workbench. Refer to the appropriate manuals for the API references.
- Regarding the properties window Help display feature: The Help button remains enabled even if there is no manual for the selected item. Additionally, items related to the Peculiar Property JBK may not always be displayed. These are limitations.

Notes on Editing the Screen Control Panel

The following describes some points to note regarding the editing of the Screen Control Panel:

- If the name of the Screen Control Panel file is changed using refactoring, the file is refactored at that point, even if it is being edited. Refactoring is not reflected in the source being edited. Additionally, when the source that is being edited is saved, the file is saved using the file name of the file before it was changed by refactoring.

Common Notes

The following describes some other points to note:

- If a file or project that is being edited in the Graphical Editor is moved, has its name changed, or is deleted, the editing in the Graphical Editor cannot be continued.
- Do not use non-Windows line separators (from UNIX or Macintosh, for example) in the source.
- Do not edit a file that is being edited by a Graphical Editor from multiple workbenches simultaneously.
- The source that can be edited with the Graphical Editor cannot include the following in its class and method declarations:
 - Generics
 - Varargs
 - Annotations (Metadata)

Chapter 8 Database Operation

This chapter describes the functions used to operate databases.

Note

- The correct behavior of the functions described in this document depends on the JDBC drivers used by the various products. Use the tools and applications provided by the various products for database operations such as creating, updating, and deleting content.
- There are limitations on some of the database operations in HA Database Ready(NativeSQL).

8.1 Overview

This section presents an overview of databases and the functions that operate databases.

8.1.1 What is a Database?

A database is software that manages access to data so that it can be shared by multiple users. Databases have various data models, such as hierarchical, network, and object-oriented types, but relational databases are the most widely used type. Relational databases manage data in two-dimensional tables in rows and columns and manage the relationships between tables.

Workbench provides functions for operating these relational databases.

Terminology related to relational databases is explained below.

SQL

SQL is a language that can operate relational databases. SQL includes Data Definition Language (DDL) for creating, changing, and deleting tables, Data Manipulate Language (DML) for updating, deleting and searching data, and Data Control Language (DCL) for operating, determining, and cancelling data.

SQL is a standards specification but conformance varies according to the vendor.

Schema

A schema expresses the element structure of tables and views.

View

A virtual table formed by extracting columns from one or more tables.

Index

Intended to improve the efficiency of data searches in databases.

Constraints

A constraint can define the conditions for data that can be entered in a table. Constraints include primary keys and external keys.

8.1.2 Functions that Operate Databases

The functions that operate databases are included in the workbench.

Connecting to the Database

In order to connect to a database, create a connection profile. When connected, the database contents can be referenced.

For details, refer to "[8.3.1 Connecting to the Database](#)".

Referencing Database Contents

The tables, views, and other elements defined in a database can be referenced. Database contents can be referenced in the [Data Source Explorer] view.

For details, refer to "[8.3.2 Referencing Database Contents](#)".

Executing SQL

The SQL coded in the SQL file can be executed. The execution results can be checked in the [SQL Results] view.

For details, refer to "[8.3.3 Executing SQL Statements](#)".

Creating Tables

In the standard workbench, the SQL statement that creates tables can be created in a dialog box. The table can be created by executing this SQL statement.

For details, refer to "[8.3.4 Creating Tables](#)".

Deleting Tables

In the standard workbench, the SQL statement that deletes tables can be created. The table can be deleted by executing this SQL statement.

For details, refer to "[8.3.5 Deleting Tables](#)".

Updating Table Data

In the standard workbench, the table data can be edited using a table data editor.

For details, refer to "[8.3.6 Updating Table Data](#)".

8.1.3 Limitations

There are limitations on some of the functions used to manipulate databases that are described in this manual.

No.	Limitations
1	Use the JDBC Driver corresponding to the JDK/JRE Version.

8.2 Introduction

This section introduces the procedures for performing database operations from the workbench.

8.2.1 Database to be Created

In the database, create a table that stores the total populations of countries. The table columns are ID, country name, and total population, and the ID is the primary key. The data is stored in sequence according to the size of the total population. This section introduces the procedures to use to create this table.

8.2.2 Development Flow

Operations for the database proceed as follows:

1. Prepare for the database

- Create the database environment

A database server is required. The server side environment for the server used for the database is described here.

- Install the JDBC driver

In order to access the database from the workbench, the JDBC driver must be installed.

2. Connect to the database

Use the following procedures to connect to the database:

- Switch perspectives
Switch to the perspective used for database development.
- Create connection profile
Create the connection profile for the prepared database environment.
- Connect to the database
Use the defined connection profile to connect to the database.

3. Create a table

Use the following procedure to create a database table:

- Switch perspectives
Switch to the perspective for creating the SQL file.
- Create a project
Create a project for storing the SQL file.
- Create an SQL file
Use the wizard to create the SQL file.
- Edit the SQL file
In the SQL file, code the SQL statement that creates a table.
- Execute the SQL
Execute the SQL statement that is coded in the SQL file.

4. Check the table

- Switch perspectives
Switch the perspective in order to check the database definition information.
- Check the table definitions
Use the Data Source Explorer view to check the definition information of the created table.
- Reference the table data
Use the Table Data editor to check the table data.

8.2.3 Development Procedures

The actual procedures for creating a database are described below.

1) [Preparing Databases](#)

2) [Connecting to the Database](#)

3) [Creating Tables](#)

4) [Checking Tables](#)

1) Preparing Databases

Create the database that stores the country name and total population data. Use Oracle for the database.

1-1) Creating the Database Environment

Create the environment for the Oracle using settings like those shown below. If using an environment where the setup items are different, substitute those setup items in the explanation below. For details on the setup method, refer to the Oracle manual.

Item	Content
Product	Oracle Database 11g
Server name	myhost
SID	COUNTRYDATA
Port number for remote connection	1521
User name	STUDIO
Password	Password corresponding to the user name
Schema	STUDIO

1-2) Installing the JDBC Driver

Install the Oracle JDBC driver in order to connect to the Oracle from the workbench. For details on the installation method, refer to the Oracle manual.

2) Connecting to the Database

2-1) Switching Perspectives

From the menu bar, select [Window] > [Open Perspective] > [Other]. The [Open Perspective] dialog box is displayed. Select [Database Development] from the [Open Perspective] dialog box, then click [OK].

2-2) Creating the Connection Profile


Select [Databases] in the Data Source Explorer view, then right-click to display the context menu. Select [New] from the context menu to display the [New Connection Profile] wizard.

From the [Connection Profile Types] list on the [Connection Profile] page, select [Oracle].

Set the following information, then click [Next].

Setup Items	Setup Content
Name	Oracle_Studio

The [Specify a Driver and Connection Details] page is displayed.

Click  (New Driver Definition) to the right of [Drivers] to display the [New Driver Definition] dialog box. In the [Name/Type] tab, from the [Available driver templates] list, select [Database] > [Oracle Thin Driver]. Check and enter the other items as shown below. After the following information is set, click [OK].

[Name/Type] tab

Setup Items	Setup Content
Name	Oracle Thin Driver
System Vendor	Oracle
System Version	11

[Jar List] tab

Setup Items	Setup Content
Driver files	<JDBC driver storage destination directory>\ojdbc6.jar

In the [Specify a Driver and Connection Details] page, set the [Properties] group items as shown below. Click [Next] to display the [Summary] page.

[General] tab of [Properties] group

Setup Items	Setup Content
SID	COUNTRYDATA
Host	myhost
Port number	1521
User name	STUDIO
Password	Password corresponding to the user name

The entered information can be checked on the [Summary] page.

Check the following information, then click [Finish].

Item to Check	Contents to Check
Name	Oracle_Studio
Description	None
Auto connect at startup	false
Auto connect on finish	true
SID	COUNTRYDATA
Host	myhost
Prot number	1521
User name	STUDIO
Save password	false
URL	jdbc:oracle:thin:@myhost:1521:COUNTRYDATA

2-3) Connecting to the Database

In the [New Connection Profile] wizard, on the [Specify a Driver and Connection Details] page, [Connect when the wizard completes] is selected by default. In this state the database is automatically connected to when the wizard completes.

If connecting manually, select the connection profile [Oracle_Studio] created in the Data Source Explorer view. Right-click to display the context menu, and select [Connect].

3) Creating Tables

3-1) Switching Perspectives

Select [Resource] from the [Open Perspective] dialog box, then click [OK].

3-2) Creating the Project

Create a project to place the SQL file in. If the SQL file is to be placed in an existing project, a project does not need to be created. Select [File] > [New] > [Project] from the menu bar to display the [New Project] wizard. Select [General] > [Project] from the New Project wizard, then click [Next].

Check and enter the following setup item. After the following information is set, click [Finish].

Setup Items	Setup Content
Project name	DBSample

3-3) Creating the SQL File

Directly under the project, create the SQL file that creates the table. Select the [DBSample] project in the Project Explorer view, then right-click to display the context menu. Select [New] > [Other] from the context menu. Select [SQL Development] > [SQL File] from the [New] wizard, then select [Next]. The [New SQL File] wizard is displayed.

Check and enter the following setup items. After the following information is set, click [Finish].

Setup Items	Setup Content
File name	countrydata.sql
Database server type	Oracle_11
Connection profile name	Oracle_Studio
Database name	COUNTRYDATA

3-4) Editing the SQL File

In the created SQL file, code the table definitions and data insertion as shown below.

SQL File Definitions

```
CREATE TABLE C_DATA (
  ID      NUMERIC PRIMARY KEY NOT NULL,
  C_NAME  VARCHAR2(80),
  T_POP   NUMERIC(10)
);

INSERT INTO C_DATA VALUES(1, 'China', 1330000000);
INSERT INTO C_DATA VALUES(2, 'India', 1140000000);
INSERT INTO C_DATA VALUES(3, 'U.S.A.', 300000000);
INSERT INTO C_DATA VALUES(4, 'Indonesia', 230000000);
INSERT INTO C_DATA VALUES(5, 'Brazil', 190000000);
INSERT INTO C_DATA VALUES(6, 'Pakistan', 160000000);
INSERT INTO C_DATA VALUES(7, 'Bangladesh', 150000000);
INSERT INTO C_DATA VALUES(8, 'Russia', 140000000);
INSERT INTO C_DATA VALUES(9, 'Nigeria', 140000000);
INSERT INTO C_DATA VALUES(10, 'Japan', 130000000);
```

3-5) Executing SQL

Open the SQL file created in the SQL File Editor and right-click in the editor area. Select [Execute All] from the context menu.

Check the SQL execution results in the [SQL Results] view.

4) Checking Tables

4-1) Switching Perspectives

Select [Database Development] in the [Open Perspective] dialog box, then click [OK].

4-2) Checking Table Definitions

The created table can be checked in the Data Source Explorer view. To change the contents, select the connection profile [Oracle_Studio], then right-click to display the context menu. Select [Refresh].

4-3) Referencing Table Data

The created table data can be referenced using the Table Data editor. In the Data Source Explorer view, select [Databases] > [Oracle_Studio] > [COUNTRYDATA] > [Schemas] > [STUDIO] > [Tables] > [C_DATA], then right-click to display the context menu. Select [Data] > [Edit] from the context menu to launch the Table Data editor.

Point

Values can be changed by editing the editor cells. In addition, editing can be performed by selecting a cell, then selecting operations such as row deletion, row addition, and so on, from the context menu.

8.3 Tasks

This section describes how to perform database operations from the workbench. It explains each purpose (task) separately.

- [8.3.1 Connecting to the Database](#)
- [8.3.2 Referencing Database Contents](#)
- [8.3.3 Executing SQL Statements](#)
- [8.3.4 Creating Tables](#)
- [8.3.5 Deleting Tables](#)
- [8.3.6 Updating Table Data](#)
- [8.3.7 Supported Database Information](#)
- [8.3.8 Using the DB Access Class Wizard to Automatically Generate JDBC Processes](#)

8.3.1 Connecting to the Database

If you want to connect to the database, create a connection profile. The method for creating a connection profile and connecting to the database is described below.

Opening the Database Development Perspective


Open the [Database Development] perspective in order to display the [Data Source Explorer] view.

Creating the Connection Profile

Select [Databases] in the [Data Source Explorer] view, then select [New] menu from the context menu. The [New Connection Profile] wizard is displayed. In this wizard, enter the settings for connecting to the database. Refer below for the settings to be specified in the wizard:

Note

When connecting with Symfoware Server(Open interface) V12, specify the "Symfoware Server(Open interface) 12 JDBC Driver" in the [Name/Type] tab of the [New Driver Definition] dialog box. The "9(Open interface)" shown in the version is the JDBC driver version.

- Connection profile type
Select a Connection profile type to match the database you want to connect to.
- Driver
Select a driver from the drop-down menu. If this has not been defined, click  (new Driver Definitions) to the right, and add definitions as shown below. For details on the JDBC drivers for various databases, refer to "8.3.7.1 JDBC Drivers".

1. In the [New Driver Definition] dialog box, on the [Name/Type] tab, select the driver template to be used in connection.

2. In the [Jar List] tab, at driver file, specify where the JDBC driver file is stored. (If there is a driver file that was added provisionally, specify the correct path.)
3. If necessary, in the [Properties] tab, modify the property used as the default.
4. When [OK] is clicked, the [New Driver Definition] dialog box closes.

- Connection profile properties

Set the connection profile properties according to the selected driver type. For the connection properties profile information of each database, refer to "[8.3.7.2 Connection Profile Properties](#)".

- Connection test

After the driver and properties are set, click [Test Connection] to test the connection to the database.

Connecting to the Database

Select the connection profile under [Databases] in the [Data Source Explorer] view, then select [Connect] from the context menu. This connects to the database and the database contents are displayed.

8.3.2 Referencing Database Contents

The tables, views, and other elements defined for the database can be referenced in the [Data Source Explorer] view.

Connecting to the Database

Connect to the database. For details on database connection, refer to "[8.3.1 Connecting to the Database](#)".

Referencing the Database Contents

The database contents are displayed as a tree structure under the connection profile shown in the [Data Source Explorer] view. The following elements defined for the database can be referenced:

- Schemas
- Tables
- Columns
- Indexes
- Constraints
- Views



.....
If a schema exists in a database, and the tables, views, etc. belonging to that schema do not exist, schema elements will not display in a tree structure.
.....

Operating Database Elements

The [Data Source Explorer] view is the starting point for database element operations. For example, if you want to edit the data in a table, select the table, then select [Data] > [Edit] from the context menu.

8.3.3 Executing SQL Statements

To issue SQL statements to a database, create an SQL file, then execute that SQL statements.

Creating the SQL File

From the [New] wizard, select [SQL Development] > [SQL File]. Refer below for the contents to specify in the wizard:

- Database server type
Select a database product and version combination.

- Connection profile name
The connection profiles for the database server type are listed. Select from the list.

- Database name
The databases to which the connection profile connect are listed. Select from the list.

- [Create] button
A connection profile can be created. For details on connection profiles, refer to "[8.3.1 Connecting to the Database](#)".

Point

.....

An SQL file can be created without selecting a database server type, connection profile name, and database name. However, if these are not set, some editor support functions may not be available.

.....

Editing the SQL File

Use the SQL File editor to open the SQL file. Code SQL statements in the SQL file.
If connected to the database, content assist can be used to enter tables and columns.

Point

.....

Specify a semicolon (;) as the delimiter between SQL statements. For some SQL statements, this delimiter cannot be parsed. In those cases, enter a line feed at the end of the SQL statement and specify a semicolon at the start of the next line. Alternatively, code the SQL statement in one line.

.....

Executing SQL Files

Check the [Connection profile] group shown in the SQL File editor. Check that values are set in the type, name, and database combo boxes and that the database is connected.

From the context menu of the SQL File Editor, select one of the items below. The SQL statement described in the SQL file is executed, and the result is displayed in the [SQL Results] view.

- [Execute All]
Splits all text with a ";" (semicolon) or GO delimiter, and then executes the SQL statement.

- [Execute Selected Text]
Splits the selected text with a ";" (semicolon) or GO delimiter, and then executes the SQL statement.

- [Execute Selected Text As One Statement]
Executes the selected text as one SQL statement.

- [Execute Current Text]
Executes the SQL statement where the cursor is currently positioned.

Point

.....

The operation of [Execute Current Text] follows the settings in the workbench settings window, [Data Management] > [SQL Development] > [SQL Editor] under [Execute current text]. The default setting is [Execute current line].

- Execute SQLs between delimiters
Executes the SQL between the delimiters before and after where the cursor is currently positioned.
 - Execute current line
Executes the text in the line where the cursor is positioned.
 - Execute SQLs between blank lines
Executes the SQL between the blank lines before and after where the cursor is currently positioned.
-

8.3.4 Creating Tables

In the standard workbench, the SQL statement that creates tables can be created in a dialog box.

Connecting to the Database

Connect to the database. For details on database connection, refer to "[8.3.1 Connecting to the Database](#)".

Generating SQL Statements

Generate the SQL statements for creating the table. Select [Tables] in the [Data Source Explorer] view, then select [New Table] from the context menu. Refer below for the contents to specify in the dialog box:

- Options
The SQL (DDL) statement corresponding to the selected option is generated.
- Columns
Set the table name. Use the [Add Column] button to add a column, and then set the column name and data type.
- Primary key
Select the primary key from within the column. Alternatively, set a name as the primary key.

Executing SQL Files

SQL statements are generated to the SQL File editor, and executed from the SQL File editor. For details on executing an SQL file, refer to "[8.3.3 Executing SQL Statements](#)".

When SQL is executed, the results are displayed in the [SQL Results] view. Refresh the [Data Source Explorer] view to check the created table.

8.3.5 Deleting Tables

In the standard workbench, the SQL statement that deletes tables can be created.

Connecting to the Database

Connect to the database. For details on database connection, refer to "[8.3.1 Connecting to the Database](#)".

Generating SQL Statements

Generate the SQL statements that delete the table. Select the table you want to display under [Tables] in the [Data Source Explorer] view, then select [Delete] from the context menu.

Executing SQL Files

SQL statements are generated to the SQL File editor, and executed from the SQL File editor. For details on executing an SQL file, refer to "[8.3.3 Executing SQL Statements](#)".

When SQL is executed, the results are displayed in the [SQL Results] view. Refresh the [Data Source Explorer] view to check the deleted table.

8.3.6 Updating Table Data

In the standard workbench, the table data can be edited. For details on the data types that can be edited in various databases, refer to "8.3.7.3 Editable Data Types".

Connecting to the Database

Connect to the database. For details on database connection, refer to "8.3.1 Connecting to the Database".

Editing the Table Data

Use the following procedure to edit data in a table:

1. Select the table you want to display under [Tables] in the [Data Source Explorer] view, then select [Data] > [Edit] from the context menu.
2. Use the Table Data editor to edit the data. Data values can be changed, and the context menu can be used to insert rows and so on.
3. Select [File] > [Save] from the menu to save the data. Users can check if the data save succeeded in the [SQL Results] view.



Point

In some cases, rows can be inserted but rows cannot be deleted or updated.

For row deletion and update, a WHERE clause that specifies the target row is needed for the SQL statement DELETE or UPDATE. If the table has a primary key, the primary key column is specified in the WHERE clause. However, if the table does not have a primary key, all columns are specified in the WHERE clause. Therefore, if a table column is a data type that cannot be specified in a WHERE clause, an error occurs and row deletion or update fails.

8.3.6.1 Editing the Data

The Table Data editor can be used to edit data in tables. The following data can be edited using this editor:

- Character

If the data type of the column is a type that stores characters, data can be set using characters.

- Numeral

If the data type of the column is a type that stores numerals, data can be set using numerals.

- Floating point number

If the data type of the column is a type that stores floating point numbers, data can be set using floating point numbers.

- Date and time

If the data type of the column is a type that stores dates and times, data can be set using specific formats. For the DATE type, set data in the "yyyy-mm-dd" format. For the TIME type, set data in the "hh:mm:ss" format. For the TIMESTAMP type, set data in the "yyyy-mm-dd hh:mm:ss.ffffff" format.

- Binary

If the data type of the column is a type that stores binary, data can be set using two-digit hexadecimal digits.



Point

For data that cannot be edited, null is the only value that can be set.

8.3.6.2 Extracting and Loading Data

Data in tables can be extracted to a file, and file data can be loaded into a table.

Extracting Data from a Table

Table data can be saved in a file. Use the following procedure to extract the data:

1. Select the table you want to display under [Tables] in the [Data Source Explorer] view, then select [Data] > [Extract] from the context menu.
2. In the [Extract Data] dialog box, specify the file to be saved and the format. This file is a text file encoded in UTF-8.

Loading Data to a Table

File data can be loaded into a table. Use the following procedure to load the data:

1. Select the table you want to display under [Tables] in the [Data Source Explorer] view, then select [Data] > [Load] from the context menu.
2. In the [Load Data] dialog box, specify the file to be loaded and the format. The file must be a text file encoded in UTF-8.



When loading the NCHAR type and NCHAR VARYING type of Symfoware Server data in the Java EE 6 workbench, the encoding of the file being loaded must be the same as the encoding of the text file of the workspace.

Confirm the encoding of the text file in [Window] > [Preferences] > [General] > [Workspace] in the Java EE 6 workbench.

8.3.7 Supported Database Information

This section presents information concerning the databases supported by this function. For details, refer to the manual for your particular database.

- JDBC drivers
- Connection profile properties
- Editable data types

8.3.7.1 JDBC Drivers

Connection to a database uses the JDBC driver DriverManager class. Information concerning the various databases is given below.



Do not set multiple Driver Definitions for one driver file. Connection to the database will succeed with one set of Driver Definitions, but will fail if there is one more set of Driver Definitions.

Symfoware Server

When connecting to a Symfoware Server, use the native bridge (type 2) driver type as the connection format.

The driver definitions for remote access (RDB2_TCP linkage) are shown below.

Property	Setting
Driver file (Standard workbench)	[JDBC2.X Driver] <Installation folder>\fjjdbc\lib\fjsymjdbc2.jar [JDBC4.X Driver] <Installation folder>\fjjdbc\lib\fjsymjdbc4.jar
Driver file (Java EE 6 workbench)	[JDBC4.X Driver] <Installation folder>\fjjdbc\lib\fjsymjdbc4.jar

Property	Setting
Driver class	com.fujitsu.symfoware.jdbc.SYMDriver
Connection URL	jdbc:symford://hostname:2050/dbname

Oracle

The driver definitions for the Oracle Thin driver are shown below.

Property	Setting
Driver file (Standard workbench)	[Oracle Database 11g JDBC Driver] <JDBC driver storage destination directory>\ojdbc6.jar
Driver file (Java EE 6 workbench)	[Oracle Database 11g Release 2 JDBC Driver] <JDBC driver storage destination directory>\ojdbc6.jar
Driver class	oracle.jdbc.OracleDriver
Connection URL	jdbc:oracle:thin:@server:1521:db

SQL Server

Microsoft(R) JDBC drivers are not bundled with Microsoft(R) SQL Server(TM). Download the SQL Server(TM) JDBC Driver 2.0 or later from the Microsoft Corporation website, then install and use the driver.

An example of the driver definitions is shown below.

Property	Setting
Driver file (Standard workbench)	<Installation folder>\sqljdbc_<version>\<location>\sqljdbc4.jar <version>: For Microsoft(R) SQL Server(TM) JDBC Driver 2.0, use "2.0". <version>: For Microsoft(R) SQL Server(TM) JDBC Driver 3.0, use "3.0". <version>: For Microsoft(R) JDBC Driver 4.0 for SQL Server(TM), use "4.0". <location>: For the English edition, use "enu" ; For the Japanese edition: use "jpn".
Driver file (Java EE 6 workbench)	<Installation folder>\sqljdbc_<version>\<location>\sqljdbc4.jar <version>: For Microsoft(R) SQL Server(TM) JDBC Driver 3.0, use "3.0" <version>: For Microsoft(R) JDBC Driver 4.0 for SQL Server(TM), use "4.0". <location>: For the English edition, use "enu" ; For the Japanese edition: use "jpn".
Driver class	com.microsoft.sqlserver.jdbc.SQLServerDriver
Connection URL	jdbc:sqlserver://localhost:1433;databaseName=pubs

PowerGres Plus

When connecting to PowerGres Plus, copy the driver that is bundled with PowerGres Plus to a local environment and use this driver.

An example of the driver definitions is shown below.

Property	Setting
Driver file (Standard workbench)	Copy the drivers below to the local environment and use those drivers. [PowerGres Plus installation destination driver location]

Property	Setting
	/usr/local/pgsqlplus/share/java/postgresql_jdbc4.jar
Driver file (Java EE 6 workbench)	Copy the drivers below to the local environment and use those drivers. [PowerGres Plus installation destination driver location] /usr/local/pgsqlplus/share/java/postgresql_jdbc4.jar
Driver class	org.postgresql.Driver
Connection URL	jdbc:postgresql://hostname:5432/<Database name>

Derby (Java DB)

When connecting to Derby (Java DB), the drivers bundled with this product can be used. The standard workbench can connect to Derby (Java DB) V10.4, and the Java EE 6 workbench can connect to Derby (Java DB) V10.8.

An example of the driver definitions is shown below.

Property	Setting
Driver file (Standard workbench)	[Derby 10.4 Derby embedded JDBC driver] <Interstage Studio Installation folder>\APS\F3FMisjee\javadb\lib\derby.jar [Derby client JDBC driver] <Interstage Studio Installation folder>\APS\F3FMisjee\javadb\lib\derbyclient.jar
Driver file (Java EE 6 workbench)	[Derby 10.8 Derby embedded JDBC driver] <Interstage Studio Installation folder>\APS\F3FMisje6\javadb\lib\derby.jar [Derby client JDBC driver] <Interstage Studio Installation folder>\APS\F3FMisje6\javadb\lib\derbyclient.jar
Driver class	[Derby 10.4/10.8 Derby embedded JDBC driver] org.apache.derby.jdbc.EmbeddedDriver [Derby client JDBC driver] org.apache.derby.jdbc.ClientDriver
Connection URL	jdbc:derby:<database location>;create=true

HA Database Ready (NativeSQL)

When connecting to HA Database Ready (NativeSQL), use the native bridge (type 2) driver type as the connection format.

The driver definitions for remote access (RDB2_TCP linkage) are shown below.

Property	Setting
Driver file (Standard workbench)	[JDBC2.X Driver] <Client Installation folder>\fjjdbc\lib\fjsymjdbc2.jar [JDBC4.X Driver] <Client Installation folder>\fjjdbc\lib\fjsymjdbc4.jar
Driver file (Java EE 6 workbench)	[JDBC4.X Driver] <Client Installation folder>\fjjdbc\lib\fjsymjdbc4.jar

Property	Setting
Driver class	com.fujitsu.symfoware.jdbc.SYMDriver
Connection URL	jdbc:symford://hostname:26551/dbname

HA Database Ready (OpenSQL)

When connecting to HA Database Ready (OpenSQL), copy the driver that is bundled with HA Database Ready (OpenSQL) to a local environment and use this driver.

An example of the driver definitions is shown below.

Property	Setting
Driver file (Standard workbench)	[JDBC4.X Driver] <Client Installation folder \OICL32\JDBC\lib\postgresql-jdbc4.jar Note: For 64-bit systems, the file is in the OICL64 folder
Driver file (Java EE 6 workbench)	[JDBC4.X Driver] <Client Installation folder>\OICL32\JDBC\lib\postgresql-jdbc4.jar Note: For 64-bit systems, the file is in the OICL64 folder
Driver class	org.postgresql.Driver
Connection URL	jdbc:postgresql://hostname:5432/<Database name>

8.3.7.2 Connection Profile Properties

This section describes the properties to be set when creating a new connection profile.

Symfoware Server

Property	Content
Database	Database name
Host	Database server name or IP address
Port number	Port number used in database connection
User name	Username used in database connection
Password	Password used in database connection

Oracle

Property	Content
SID	Database SID
Host	Database server name or IP address
Port number	Port number used in database connection
User name	Username used in database connection
Password	Password used in database connection

SQL Server

Property	Content
Database	Database name
Host	Database server name or IP address

Property	Content
Port number	Port number used in database connection
Use integrated authentication	Select if using Windows authentication in database connection.
User name	Username used in database connection. Set this property if not using integrated authentication
Password	Password used in database connection. Set this property if not using integrated authentication

Note

To use [Use integrated authentication] in the SQL Server connection profile, the sqljdbc_auth.dll(x86) provided with the SQL Server JDBC driver is required. Add the folder storing sqljdbc_auth.dll(x86) (example: <SQL Server JDBC driver installation folder> \sqljdbc_1.2\jpn\auth\x86) to the environment variable PATH in advance.

PowerGres Plus

Property	Content
Database	Database name
URL	URL used in database connection
User name	Username used in database connection
Password	Password used in database connection

Derby (Java DB)

Property	Content
Database location	Name of folder in which the database is deployed
User name	Username used in database connection (optional)
Password	Password used in database connection (optional)

HA Database Ready (NativeSQL)

Property	Content
Database	Database name
Host	Database server name or IP address
Port number	Port number used in database connection
User name	Username used in database connection
Password	Password used in database connection

HA Database Ready (OpenSQL)

Property	Content
Database	Database name
URL	URL used in database connection
User name	Username used in database connection
Password	Password used in database connection

8.3.7.3 Editable Data Types

The following table shows the editable data types for the Table Data editor, data extraction, and data loading.

Symfoware Server

Data Type	Explanation
CHARACTER	Fixed-length character string type
CHARACTER VARYING	Variable-length character string type
NATIONAL CHARACTER	Fixed-length national character string type
NATIONAL CHARACTER VARYING	Variable-length national character string type
NUMERIC	Exact numeric type
DECIMAL	Exact numeric type
INTEGER	Exact numeric type
SMALLINT	Exact numeric type
REAL	Approximate numeric type
DOUBLE PRECISION	Approximate numeric type
DATE	Date-time type (date)
TIME	Date-time type (time)
TIMESTAMP	Date-time type
BINARY LARGE OBJECT	Binary type

Oracle

Data Type	Explanation
CHAR	Fixed-length character string type
VARCHAR2	Variable-length character string type
NCHAR	Fixed-length national character string type
NVARCHAR2	Variable-length national character string type
NUMBER	Numeric data type
FLOAT	Floating point data type
LONG	Variable-length character string type
LONG RAW	Variable-length binary type
RAW	Variable-length binary type
DATE	Date-time type, but the time cannot be edited
BLOB	Binary type
CLOB	Character string type
NCLOB	National character string type



Note

Data cannot be modified or deleted in tables that have BLOB, CLOB, LONG, LONG RAW and NCLOB type columns, and that do not have a primary key.

SQL Server

Data Type	Explanation
bit	Truth value type
bigint	int type
int	int type
smallint	int type
tinyint	int type
decimal	decimal type
numeric	numeric type
money	money type
smallmoney	smallmoney type
float	Approximate numeric type
real	Approximate numeric type
datetime	datetime type
smalldatetime	smalldatetime type
char	Fixed-length character string type
varchar	Variable-length character string type
nchar	Fixed-length national character string type
nvarchar	Variable-length national character string type
binary	Fixed-length binary type
varbinary	Variable-length binary type
uniqueidentifier	GUID

PowerGres Plus

Data Type	Explanation
smallint	Small-range integer
integer	Usual choice for integer
bigint	Large-range integer
decimal	User-specified precision, exact
numeric	User-specified precision, exact
real	Variable-precision, inexact
double precision	Variable-precision, inexact
serial	Autoincrementing integer
bigserial	Large autoincrementing integer
character varying, varchar	Variable-length with limit
character, char	Fixed-length, blank padded
text	Variable unlimited length
bytea	Variable-length binary string
date	Date only
timestamp	Both date and time

Data Type	Explanation
timestamp with time zone	Both date and time, with time zone
time	Times of day only
time with time zone	Times of day only, with time zone
boolean	boolean type
oid	Object identifier

Derby(Java DB)

Data Type	Explanation
CHAR	Fixed-length character string type
CHAR FOR BIT DATA	Fixed-length bit string type
VARCHAR	Variable-length character string type
VARCHAR FOR BIT DATA	Variable-length bit string type
NUMERIC	Exact numeric type
DECIMAL	Exact numeric type
INTEGER	Exact numeric type
SMALLINT	Exact numeric type
BIGINT	Exact numeric type
REAL	Approximate numeric type
DOUBLE PRECISION,DOUBLE	Approximate numeric type
DATE	Date-time type (date)
TIME	Date-time type (time)
TIMESTAMP	Date-time type

HA Database Ready (NativeSQL)

Data Type	Explanation
CHARACTER	Fixed-length character string type
CHARACTER VARYING	Variable-length character string type
NATIONAL CHARACTER	Fixed-length national character string type
NATIONAL CHARACTER VARYING	Variable-length national character string type
NUMERIC	Exact numeric type
DECIMAL	Exact numeric type
INTEGER	Exact numeric type
SMALLINT	Exact numeric type
REAL	Approximate numeric type
DOUBLE PRECISION	Approximate numeric type
DATE	Date-time type (date)
TIME	Date-time type (time)
TIMESTAMP	Date-time type
BINARY LARGE OBJECT	Binary type

HA Database Ready (OpenSQL)

Data Type	Explanation
smallint	Small-range integer
integer	Usual choice for integer
bigint	Large-range integer
decimal	User-specified precision, exact
numeric	User-specified precision, exact
real	Variable-precision, inexact
double precision	Variable-precision, inexact
serial	Autoincrementing integer
bigserial	Large autoincrementing integer
character varying, varchar	Variable-length with limit
character, char	Fixed-length, blank padded
text	Variable unlimited length
bytea	Variable-length binary string
date	Date only
timestamp	Both date and time
timestamp with time zone	Both date and time, with time zone
time	Times of day only
time with time zone	Times of day only, with time zone
boolean	boolean type
oid	Object identifier



In the Java EE 6 workbench, tables of decimal and numeric types cannot be operated.

8.3.8 Using the DB Access Class Wizard to Automatically Generate JDBC Processes

By using the DB access class wizard in the standard workbench, DB access processing using JDBC can be generated automatically as a DB access class.

DB access classes deploy search, insertion, deletion, and update processes as methods based on the information specified using the wizard, and deploy the processing target database columns as class fields. Thus, when a DB access class is used, DB access processes are performed by invoking methods while setting values in and fetching values from class fields.

Creating a DB access class

From the [New] wizard, select [Java] > [Source] > [DB Access Class] and use the wizard to create the DB access class. See below for the wizard settings.

- Source folder
Specify the folder in which the DB access class source is stored.

- Package
Specify the name of the DB access class package.

- Name
Specify the DB access class name.

- Generate methods for manual commit
Specify whether or not to generate a method for manual commit for the DB access class. If selected, a commit method and rollback method are generated for controlling transactions.

- Catch the exception thrown by the DB access
Specify whether or not to catch exceptions (SQLException and similar) within DB access methods.

- Connection list box
A list of already created DB connections is displayed. Select one from the list.

- Add a connection
This opens a new DB connection wizard.

- Connect
Connects to the DB selected in the connection list box.

- [Information DB] page
Use the [Add], [Edit], and [Delete] buttons to specify information related to the DB access method definitions.

- [DB Access Method] dialog box
Specify the DB access method name, return value type, arguments, and so on.
A list of schemas, tables, and column names in the connected database is displayed at [Database objects]. Specify the processing target table and column.
At [Access type], select the type of database access performed by the method.
At [Condition], specify the conditions to be used when accessing the database. For the conditions, specify the character string after the SQL statement WHERE clause as in the example shown below. Instead of the "?" used to dynamically change a condition, specify the DB access method parameter name with "?" at the start and the end.
(Example) WHERE NAME = ?param1? AND ID = ?param2?

- [Field information definition] page
Confirm or change the contents of the DB access method definition.
The [Type] and [Field Name] can be changed.
[Getter] and [Setter] are enabled if [Generate getter/setter methods] was selected.

Information

If the JDBC driver 2.x is being used, the connect method of the automatically generated class must be replaced with the following comment content that is coded in the same source:

```
// This is the connect method for JDBC2.x connections.
// Substitute this for the existing connect method and enter the required items. This enables access
// to be executed from a data source.
//public void connect(java.lang.String userName, java.lang.String passWord) throws
//javax.naming.NamingException, java.sql.SQLException {
//    //Connect to the database.
//    java.util.Hashtable env = new java.util.Hashtable();
```

```
// env.put(javax.naming.Context.INITIAL_CONTEXT_FACTORY, "{ initial context factory}");
// env.put(javax.naming.Context.PROVIDER_URL, "{provider URL}");
// javax.naming.InitialContext ctx = new javax.naming.InitialContext(env);
// javax.sql.DataSource ds = (javax.sql.DataSource)ctx.lookup("jdbc/{ data source name}");
// con = ds.getConnection(userName, passWord);
// con.setAutoCommit(true);
//}
```

When replacing the comment contents, also correct the coding for the database connection items below.

Item name	Coding example
Initial context factory	com.fujitsu.symfoware.jdbc2.jndisp.SYMContextFactory (Register the JDBC driver in the project build path in advance.)
Provider URL	SYM://<connection destination address>:<port number>
Data source name	JDBC/<name registered by data source registration tool>



Chapter 9 Developing Java EE 6 Applications

This chapter explains how to develop Java EE 6 applications using the Java EE 6 workbench.

Also refer to "[How to Read this Manual](#)" for information on how to read the instructions.

9.1 Overview

This section provides an overview of the development of Java EE 6 applications.

9.1.1 What is Java EE 6?

Java EE 6 is an abbreviation of "Java Platform, Enterprise Edition 6". It is a standard proposed for enterprise systems by Sun Microsystems, Inc. (now Oracle Corporation) related to Java platforms. To the Java SE set of standard functions, Java EE adds a set of functions intended for Servlets, EJB, and other server applications used for the development of business systems. Application servers implement handling as components, various services, communication methods, and so on, in accordance with these conventions as a Java EE 6 platform, thereby providing technology for creating multi-tiered applications that combine reusable components.

Refer to the following specifications for details on Java EE 6:

JSR 316: Java(TM) Platform, Enterprise Edition 6 (Java EE 6) Specification



- There are APIs that are old and unused such as JAX-RPC and the Entity Beans of EJB 2.x, and also those for which alternative functions have been developed. The removal of these APIs from the specifications is being examined. Consider migrating to the Java EE 5 or Java EE 6 APIs.
- Some of the features for developing Java EE 6 applications in the Java EE 6 workbench are based on GlassFish plugin v3.1.2, an open source software application that maximizes the portability of the applications. JAX-RS supports the Java EE 6 standard feature range. Features implemented solely by GlassFish plugin are not supported.

9.1.2 Developing Java EE 6 Applications

The flow of Java EE 6 application development is shown below.

Preparation for Java EE 6 Application Development

When developing Java EE 6 applications, make preparations, then create the enterprise application project. Create the project and add the Java EE 6 module to the project. For details, refer to "[9.6.1 Preparing to Create Applications](#)" and "[6.2.2.2 Creating Enterprise Applications](#)".

Developing Java EE 6 Applications

Develop Java EE 6 applications such as Web applications and EJB.

For details, refer to "[9.2 Developing Web Applications](#)", "[9.3 Developing Enterprise JavaBeans \(EJB\)](#)", "[9.4 Developing Applications using Java Persistence API](#)" and "[9.5 Developing Web Service Applications](#)".

Verifying the Behavior of the Java EE 6 Application

Use the Servers view to verify the behavior of the Java EE 6 application that was created.

For details, refer to "[6.2.6 Checking Application Behavior](#)".

Distributing the Java EE 6 Application

To distribute Java EE 6 application files, archive into an EAR file. Use the Export wizard to create the archive file.

For details, refer to "[6.2.7 Distributing to the Operating Environment](#)".



Note

A Management Console feature has not been provided with the Java EE 6 application operating environment. Use commands for operations such as settings and deployment.

9.2 Developing Web Applications

This section provides an overview of Web applications and explains how to create Web applications to run on the Java EE 6 application operating environment.

9.2.1 Overview

9.2.1.1 What are Web Application?

Refer to "[2.1.1 What are Web Applications?](#)" for an outline of Web applications.

9.2.1.2 Changes from Java EE 5

The specification for the Web application contained in Java EE 6 is Servlet 3.0.

The following main features have been added in Servlet 3.0:

- Definitions for servlet and filter settings using annotations
Servlet or servlet filter settings can be inserted directly into classes using annotations. It is possible to avoid using web.xml.
- Settings for each framework
Pluggability is improved by using Web fragment descriptors to define the settings for each framework and including them jar files.
- File uploading
Multipart data is supported, so files can now be uploaded easily.
- Asynchronous processing
Asynchronous processes can be started on another thread, so push applications can be developed.

Definitions for Servlet and Filter Settings using Annotations

In Servlet 3.0, it is now possible to define servlet and servlet filter settings with annotations. Accordingly, the web.xml settings can be overwritten using the content defined in the annotation, so web.xml does not have to be used. The following shows the annotations used for the main servlet settings:

- @WebServlet
This defines the servlet class. This behaves the same as the web.xml elements <servlet> and <servlet-mapping>.
- @WebFilter
This defines the servlet filter. This behaves the same as the web.xml elements <filter> and <filter-mapping>.
- @WebListener
This defines the listener class. This behaves the same as the web.xml element <listener>.

The servlet settings can be defined by setting these annotations in the class that implements the servlet, servlet filter, and listener interface.



Point

Refer to the following specifications for details on Servlet 3.0:

- JSR-315: Java(TM) Servlet 3.0 Specification

Refer to the following specifications for details on JSP 2.2:

- JSR-245: JavaServer(TM) Pages 2.2

Refer to the following specifications for details on JSF 2.1:

- JSR-314: JavaServer Faces 2.1
-

9.2.1.3 Developing Web Applications

Refer to "[2.2.2 Development Flow](#)" and "[2.2.3 Development Procedures](#)" for information on the flow and procedures for developing Web applications. Refer to "[9.6.1 Preparing to Create Applications](#)" for information on preparations for creating annotations.



Specify "JavaServer Faces v2.0 Project" in [Configuration] in the Dynamic Web Project wizard to use JSF in the Web application.

9.3 Developing Enterprise JavaBeans (EJB)

This section provides an overview of EJBs and explains how to create EJBs that run in the Java EE 6 application operating environment.

9.3.1 Overview

9.3.1.1 What is an EJB?

Refer to "[3.1.1 What is an EJB?](#)" for an outline of EJB.

9.3.1.2 Changes from Java EE 5

The EJB specification in Java EE 6 is EJB 3.1. The following EJB 3.1 specifications have been improved to achieve even greater simplification of development:

- There is no need to create EJB-JAR files
Up until now it was necessary to create EJB-JAR files to use EJB, but this is no longer necessary in EJB 3.1.
Classes defined as components with annotations are turned into EJB components by packaging them as jar files in the WEB-INF/classes directory or in the WEB-INF/lib directory.
- The local business interface has been omitted
Up until now it was necessary to define the local business interface to use Enterprise Beans locally. With EJB 3.1, however, it is possible to omit the local business interface definition.

Example (existing definition)

```
Local business interface definition:

public interface Calc {
    public int add(int param1, int param2);
}

EJB class definition:
@Stateless
public class CalcBean implements Calc {
    public int add(int param1, int param2) {
        return param1 + param2;
    }
}

Used from the client:
@EJB Calc a;
```

```
a.add(1,1);
```

Example (definition in EJB 3.1)

```
EJB class definition:
@Stateless
public class CalcBean {
    public int add(int param1, int param2) {
        return param1 + param2;
    }
}

Used from the client:
@EJB CalcBean a;

a.add(1,1);
```

Point

Refer to the following specifications for details on EJB 3.1:

- JSR 318: Enterprise JavaBeans(TM) 3.1

9.3.1.3 Developing EJB

Refer to "[3.2.2 Development Flow](#)" and "[3.2.3 Development Procedures](#)" for information on the flow and procedures for developing EJB. Refer to "[9.6.1 Preparing to Create Applications](#)" for information on preparations for creating annotations.

Note

The wizard names in the standard workbench and the Java EE 6 workbench may be different. For example: The [Session Bean] wizard becomes the [Session Bean (EJB 3.x)] wizard.

9.4 Developing Applications using Java Persistence API

This section provides an overview of applications that use JPA and explains how to create them to run on the Java EE 6 application operating environment.

9.4.1 Overview

9.4.1.1 What is JPA?

Refer to "[4.1.1 What is JPA?](#)" for an outline of JPA.

9.4.1.2 Changes from Java EE 5

The specification for JPA in Java EE 6 is JPA 2.0. The following features have been added in JPA 2.0:

- Mapping of object collections that can be embedded
Annotations can be used for the mapping of object collections that can be embedded.
- Extension of Java Persistence Query Language (JPQL)
New operators such as CASE expression can be used.

- Criteria API
API controlled by Java objects can be used for query control.

Point

Refer to the following specifications for details on JPA 2.0:

- JSR 317: Java(TM) Persistence 2.0
-

9.4.1.3 Developing Applications using JPA

Refer to "[4.2.2 Development Flow](#)" and "[4.2.3 Development Procedures](#)" for information on the flow and procedures for developing JPA. Refer to "[9.6.1 Preparing to Create Applications](#)" for information on preparations for creating annotations.

Note

- To create connection profiles in the Java EE 6 workbench, in the [New] wizard, select [Connection Profile] > [Connection Profile].
 - When "Development Procedures" in "Chapter 4 Developing Applications that use Java Persistence API (JPA)" is performed using the Java EE 6 workbench, there are the following differences:
In "2-1) Creating the JPA Project", specify "Generic 2.0" under [Platform] in the [JPA Project] wizard and "Disable Library Configuration" for the JPA implementation.
In "6-2) Setting the Build Path", select the EAR project, then select [Properties] from the context menu and check the [Deployment Assembly].
 - To create the JDBC Connection Pool, use the create-jdbc-connection-pool command. To create the JDBC resource, use the create-jdbc-resource command. Refer to "Setting the Database Environment" in the "Interstage Application Server Java EE Operator's Guide (Java EE 6 Edition)" for information on Interstage Application Server JDBC settings such as the JDBC driver classpath settings.
-

9.5 Developing Web Service Applications

This section provides an overview of Web service applications and explains how to create Web service applications that run on the Java EE 6 application operating environment.

9.5.1 Overview

9.5.1.1 What is a Web Service?

Refer to "[5.1.1 What is a Web Service?](#)" for an outline of Web services.

9.5.1.2 Changes from Java EE 5

JAX-RS1.1 is included in Java EE 6.

By using JAX-RS1.1, Representational State Transfer (REST) type Web applications can easily be developed.

By using annotations in JAX-RS1.1, the Java resource path can be specified to easily bind the Java method to the HTTP request method.

Annotations

Use annotations to define classes to run as REST type Web applications.

The following shows annotations example:

- @Path
This is used to define the root resource or sub resource Java method path.

- @GET
This annotation specifies the "GET" HTTP request method. The root source or sub resource Java method can be bound to the "GET" HTTP request method.
- @POST
This annotation specifies the "POST" HTTP request method. The root source or sub resource Java method can be bound to the "POST" HTTP request method.
- @Produces
This generates the Java method of the root resource or sub resource, and shows the MIME media types that can be returned to the client.
- @Consumes
The Java method of the root resource or sub resource indicates the MIME media types that can be received from the client.

Example of creating a REST type Web application using annotations

```

@Path(value="/root")
public class RootResource {
    @GET
    public String getName() {
        return "name";
    }
}

```

Point

Refer to the following specifications for details on JAX-RS 1.1:

- JSR-311: JAX-RS: The Java(TM) API for RESTful Web Services

Refer to the following specifications for details on JAX-WS 2.2:

- JSR 224: Java(TM) API for XML-Based Web Services (JAX-WS) 2.2

9.5.1.3 Developing Web Service Applications

Refer to "[5.2.2 Development Flow](#)" and "[5.2.3 Development Procedures](#)" for information on the flow and procedures for developing Web service applications. Refer to "[9.6.1 Preparing to Create Applications](#)" for information on preparations for creating annotations.

Note

In the Java EE 6 workbench, the Service Endpoint Interface is generated from WSDL using the ijwsimport command.

1. Open the command prompt create the working folder.
Example: mkdir temp
2. Change to the working folder.
Example: cd temp
3. Create the source output destination folder.
Example: mkdir src
4. Use the ijwsimport command to generate the required source for the Web service client to the working folder.
ijwsimport -p <packagename> -s <sourceoutputfolder> -keep <WSDL URL (Uniform Resource Locator)>
Example: ijwsimport -p stub -s src -keep http://localhost:28282/WebServiceSample6/PopulationRankingService?wsdl
5. Copy the source files of the <source output folder> in 4 to each package folder in the Web service client project source folder. The WSDL files are unnecessary.

6. Select the Web service client project, then press the F5 key.
7. Delete the working folder.

If an error was detected in the source files that were added in 5 and 6 above, use the following procedure to set the build path:

1. Select the Web service client project, then select [Properties] from the context menu.
2. Click the [Java Build Path] > [Libraries] tabs, then add the following JAR files in [Add External JARs]:
<installation folder>\APS\F3FMisje6\glassfish\modules\endorsed
<installation folder>\APS\F3FMisje6\glassfish\lib\endorsed
3. In the [Order and Export] tab, move the JAR files that were added in 2 above [JRE System Library] then click [OK].

Refer to "Deploying and Obtaining/Storing WSDL" in "Developing Web Service Applications" in the "Interstage Application Server Java EE Operator's Guide (Java EE 6 Edition)" for details on the ijwsimport command and WSDL URL.

Point

Refer to "How to Create JAX-RS Applications" in the "Interstage Application Server Java EE Operator's Guide (Java EE 6 Edition)" for details on creating JAX-RS applications.

9.6 Common Subject Matter Regarding Java EE 6 Applications

This section describes the common items involved in developing Java EE 6 applications using Interstage Studio. It explains each of the development work topics (tasks) separately.

- [9.6.1 Preparing to Create Applications](#)
 - [9.6.1.1 Preparing to Operate Servers](#)
 - [9.6.1.2 Creating a New Project](#)
 - [9.6.1.3 Creating Enterprise Applications](#)
 - [9.6.1.4 Setting a Classpath](#)
 - [9.6.1.5 Development using a Java EE 6 Module that is not a Development Resource](#)
- [9.6.2 Creating the Java Classes and Interfaces](#)
- [9.6.3 Creating XML Files](#)
- [9.6.4 Detecting and Correcting Problems](#)
- [9.6.5 Checking Application Behavior](#)
 - [9.6.5.1 Debugging](#)
- [9.6.6 Distribute the Files to the Operating Environment](#)

9.6.1 Preparing to Create Applications

In order to create the application, projects must be created to suit the modules being developed.

If an application is to comprise multiple modules, these can be grouped together as an enterprise application.

9.6.1.1 Preparing to Operate Servers

To perform operations such as starting or stopping the server in the Java EE 6 workbench, add the server to be operated to the Servers view. The runtime specified when adding the server to the Servers view is also specified as the target runtime in the wizard when the Java EE 6 application is developed. Add the server before developing Java EE 6 applications.

Adding Servers

Add to the Servers view the server that is the application deployment destination for checking the application behavior. Use the New Server wizard to add the server. Right-click the Servers view, then select [New] > [Server] from the context menu. For wizard setup items, refer to the following:

- Server type
Select the server type. For an Interstage Java EE 6 operating environment, select [FUJITSU LIMITED] > [Interstage Application Server V11.1 (Java EE 6)].
- Server's host name
Set the name of the host where the server exists. For a local machine, specify "localhost".
- Server name
Specify the server name. The name that is specified here is displayed in the Server view.
- Server runtime environment
Specify the runtime setting that corresponds to the server type. In the Interstage Java EE 6 operating environment, select [Interstage Application Server V11.1 IJServer Cluster (Java EE)].

If the server is an Interstage Application Server, the following setup items are also displayed:

- PreserveSessions Across Redeployment
Specify to continue sessions when redeploying.
- HTTP listener port for operation management
Specify the HTTP listener port for operation management. The default port number is 12011. Set it based on the server setting.
- HTTP listener port
The default port number is 28282. Set it based on the server setting. When connecting to a remote server, specify the HTTPS listener port of the server.
- Debugging port number
The default number is displayed. Change it if necessary.
- HTTP port number
Specify the port number of the HTTP Server used during connection confirmation. The default number is displayed. Change it based on the server setting.
 - When the Web Server connector (used in Interstage HTTP Server 2.2) is used
Specify the port number of the Web Server (Interstage HTTP Server 2.2).
 - When the Web Server connector (used in Interstage HTTP Server 2.2) is not used
 - When Interstage Java EE 6 DAS service is specified as the deployment target, set its value to be the same as the HTTP listener port number or HTTPS listener port number in the Java EE 6 operating environment.
 - When IJServer cluster is specified as the deployment target, set its value to be the same as the HTTP listener port number or HTTPS listener port number of the IJServer cluster.

After the port number is specified, click [Login] to confirm that the server can be logged into. Specify the same administrator name and password as those in the Interstage Java EE 6 operating environment. [Next] is enabled after the server can be logged into.



-
- Servers can also be added from the New wizard.
 - Refer to "Interstage Application Server Java EE Operator's Guide (Java EE 6 Edition)" for details on the Interstage Application Server Java EE 6 operating environment, such as administrator name and password.

- The added server information can be changed in the server page (double-click the target server in [Servers] view).

Note

- When the host name of a server is set to a value other than "localhost", it is deemed a remote server. The remote server is connected using https communication.
- The wizard input items will vary depending on the content that is specified for the host.
- When the server operation used in the Server view and the server operation used from the command are used together, an error may sometimes occur at the time of the deployment, for example.

9.6.1.2 Creating a New Project

To launch the [New project] wizard, select [File] > [New] > [Project] from the menu. Select the project wizard that is appropriate for the module, as shown below.

Category	Project	Explanation
EJB	EJB Project	Used to create an EJB module.
Java EE	Application Client Project	Used to create an application client module.
	Enterprise Application Project	Used to create an EAR file. EAR files can group together EJB, Web applications, and libraries.
	Utility Project	Used to create libraries that are shared by multiple modules.
JPA	JPA Project	Used to create libraries that use JPA.
Web	Dynamic Web Project	Used to create Web application modules.

For the contents to specify in the project wizard, refer to the following:

- Project name
Specify the project name to be generated.
- Project location
Specify the storage destination of the project resources. The default location is under the workspace folder.
- Target runtime
Select the runtime that runs the Java EE 6 application. This sets the runtime library in the classpath. For an Interstage Application Server Java EE 6 operating environment, select [Interstage Application Server V11.1 (Java EE 6)].
- Module version
For an EJB or Web application, etc., specify the module version.
- Configuration
Set the conventions and versions that the created modules and libraries conform to. Operate the support functions such as the wizard and editor, so that they conform to the conventions and versions. For an Interstage Application Server Java EE 6 container, select [Default Configuration for Interstage Application Server V11.1 (Java EE 6)].
- EAR membership
If the modules and libraries are grouped together in an EAR file, select the enterprise application project. If the enterprise application is to be created later, it need not be selected here.

- Working sets
To add a working set to the project, select the working set.

Note

- When an EJB project is created without [Add project to an EAR] being selected under [EAR membership], the Session Bean wizard business interface is created in the EJB project.
- When an EJB project is created with [Add project to an EAR] selected under [EAR membership], the Session Bean wizard business interface is created in an EJB client project.
- When an EJB project is created with [Add project to an EAR] selected under [EAR membership] and without [Create an EJB Client JAR module to hold the client interfaces and classes] being selected, the Session Bean wizard business interface is created in the EJB project.

9.6.1.3 Creating Enterprise Applications

The enterprise application project can group together modules and libraries and create an EAR file.

Refer below for specifying contents specific to the enterprise application project wizard:

- Content directory
Specify the name of the folder used to store files, apart from the Java EE module created as the project, that you want to include in the EAR file.

To change the Java EE module that will be added to the EAR file after the enterprise application project has been created, edit using the project [Deployment Assembly] property.

Point

- If an EAR file is used, in addition to enabling applications to be deployed as a group, the dependency relationship between the Java EE modules in the EAR file can be defined and the work involved in setting classpaths and other tasks at the operating environment can be reduced.
- Modules or libraries contained in enterprise applications can be downloaded as client stubs by including application clients in the enterprise applications. Refer to "Download the Client Stub JAR File" in "Java EE Application Client Operation" in the "Interstage Application Server Java EE Operator's Guide (Java EE 6 Edition)" for details.

9.6.1.4 Setting a Classpath

In order to perform build for the Java application, set the build path (classpath) in the project. Use any of the following methods to set the build path for Java EE modules and libraries:

Targeted Runtimes

When the runtime that runs the Java EE 6 application is selected, that library is added to the build path. This can be set in the [Targeted Runtimes] property of the project.

The library is added by the runtime name in the build path.

Enterprise application Deployment Assembly

The modules and libraries in the packages (EAR/WAR files) can reference the classes in the package. This is specified in the EAR project [Deployment Assembly] property. This can also be set in the [Deployment Assembly] property [Deployment Assembly] tab of dynamic Web projects and EJB projects.

Java build path

The build path can be set using the project [Java Build Path] property. A library added using this setting is not reflected in the operating environment. Therefore, before deployment to the application server, classpath settings may be required at the operating environment.

9.6.1.5 Development using a Java EE 6 Module that is not a Development Resource

A Java EE 6 application may be developed using a Java EE 6 module that was developed by another person.

If an EAR file is to be created by an enterprise application project, the Java EE 6 module must be imported and a Java EE 6 module project must be created.

Select [File] > [Import] from the workbench menu, then select the file format to be imported from the [Import] wizard.

Point

.....

If a source file is included in the imported Java EE 6 module, the source file is stored in the project source folder created by the Import wizard. If a source file is not included in the imported Java EE 6 module, an ImportedClasses folder is created in the project created by the Import wizard and the class file is stored in the ImportedClasses folder.

.....

9.6.2 Creating the Java Classes and Interfaces

Refer to "[6.2.3 Creating the Java Class and the Interface](#)".

9.6.3 Creating XML Files

Refer to "[6.2.4 Creating XML Files](#)".

9.6.4 Detecting and Correcting Problems

Refer to "[6.2.5 Detecting and Correcting Problems](#)".

Note

.....

There are differences in the verification feature that is provided in the standard workbench and the Java EE 6 workbench.

.....

9.6.5 Checking Application Behavior

To check the behavior of a Java EE 6 application, deploy the Java EE 6 application to the server, then start the server. Perform these operations in the Servers view. When the server starts, the Java EE 6 application waits to be invoked from the client. The behavior is checked by invoking Java EE 6 application from the client.

In addition, the application can be debugged by interrupting program execution and executing code one line at a time and checking variable values.

Note

-
- After the application is deployed in the following environments, an archive file (such as EAR/WAR) will be created in the workspace folder. Please confirm that there is enough available space on the hard disk.
 - When the application is deployed to a remote server.
 - When the application is deployed to a localhost where [User Real Jar Archives for Deployment] of the server page is selected.
 - When WebServer(Interstage HTTP Server 2.2) and WebServer connector (used in Interstage HTTP Server 2.2) are used to confirm the application actions, refer to "Web Server Linkage Restrictions" in the "Interstage Application Server Java EE Operator's Guide (Java EE 6 Edition)".

During the setting of WebServer integration, the following operations are needed if the asadmin command is not available.

- After the application is deployed, it is necessary to add the assignment target application through the add-application-ref sub command of the wscadmin command.
 - After the application deployment is cancelled, it is necessary to delete the assignment target application through the delete-application-ref sub command of the wscadmin command.
-

Specifying the Project to be Deployed to the Server

The project to be deployed to the server is specified by adding the project to the Servers view server. Use the Add and Remove Projects wizard to add the project to be deployed to the server. To launch the Add and Remove Projects wizard, select the server at the Servers view, then select [Add and Remove] from the context menu.

A project can also be added using the last page of the New Server wizard.

Point

To deploy an application manually, select the server, then select [Publish] from the context menu. If [Clean] is used, all deployed applications are temporarily discarded and then redeployed. If the application is synchronized with the server, "synchronized" is displayed as the status. If not synchronized, "Republish" is displayed.

Starting Servers

In the Servers view, select the server, then select either [Start] or [Debug] from the context menu. The application can be deployed automatically when the server starts. The default setting is that applications are deployed automatically when the server starts.

Point

- The user can select the project, then select [Run As] > [Run on Server] or [Debug As] > [Debug on Server] from the context menu to add the server, add the project, launch the server, and launch the client all together.
- To set such that the application is not deployed automatically when the server starts, select [Server] > [Launching] from the Preferences page, then set [Automatically publish when starting servers] to off.

Stopping Servers

To stop the server, select the server at the Servers view, then select [Stop] from the context menu.

9.6.5.1 Debugging

This section describes the debugging that is used to detect logical errors in programs.

To debug an application, set breakpoints to interrupt the launched program, and then execute the code one line at a time and check the variable content.

Breakpoints

Program execution is interrupted at the lines where breakpoints have been set. When execution is interrupted, a confirmation dialog box asking whether or not to open the Debug perspective is displayed. The Debug perspective is comprised of layouts used during debugging, such as the Debug view, the Variables view, and the Breakpoints view.

- Setting and removing breakpoints
Double-click the ruler part of the editor to toggle between setting and removing breakpoints. In addition, the Breakpoints view can be used to check breakpoints that have been added and to remove breakpoints.
- Disabling breakpoints
To temporarily disable a breakpoint, select [Disable Breakpoint] from the ruler context menu, or deselect it from the Breakpoints view. Alternatively, to temporarily skip all breakpoints, select [Skip All Breakpoints] under [Run] from either the Breakpoints view toolbar or the menu bar.

Note

If breakpoints have been set in a JSP file, execution is also interrupted for JSP files with the same name in different folders.

Execution Control

Various methods are available for restarting execution of execution-interrupted programs, including step over, step into, step return, and resume. To execute these commands, select stack frame from the Debug view, then select a command from [Run] in the Debug view toolbar, context menu, or menu bar.

- Step Over
The currently selected line is executed, then execution is interrupted at the next executable line.

- Step Into
The next expression that needs to be executed in the currently selected line is invoked, then execution is interrupted at the next executable line of the invoked method.

- Step Return
Execution up to the next return statement of the current method is restarted, then execution is interrupted at the next executable line.

- Resume
Execution continues until the next breakpoint is reached.

Checking and Changing Variable Values

When a stack frame is selected, the variables that are visible in that stack frame can be displayed in the Variables view. Primitive type values are displayed in the Variables view. If complex variables are expanded and their members are displayed, these can be inspected. Values can be changed using [Change Value] under the context menu.



Point

Debug information must be added to a compiled class file in order to reference variable values and perform debugging. The default setting is that the debug information required for debugging is added to the class file. If debugging is not required, for example, when deploying to an operating environment, the size of the class file can be reduced by not adding the debug information.

The debug information is used to display source file names and line numbers in the stack trace that is output when, for example, an exception is thrown. It is recommended to add source file names and line numbers to a class file.

9.6.6 Distribute the Files to the Operating Environment

To distribute a Java EE 6 application to the operating environment, an archive file must be created. Use the server deployment function to deploy the created archive file to the server.

Creating an Archive File

Use the Export wizard to create the archive file.

Export

1. Launching the Export wizard
Select [File] > [Export].
2. Entering settings items
 - Export destination

Project	Export destination
EJB Project	[EJB] > [EJB JAR file]
Enterprise Application Project	[Java EE] > [EAR file]
Application Client Project	Export the project as an EAR, because the project is included in the application created in enterprise application

Project	Export destination
	project. Use the following if exporting it on a standalone basis: [Java EE] > [App Client JAR file]
Dynamic Web Project	[Web] > [WAR file]
JPA Project	Export the project as an EAR or a WAR because the project is included in the application created in enterprise application project or in dynamic web project. Use the following if exporting it on a standalone basis: [Java] > [JAR file]

- Module
Specify the project to be exported.
- Destination
Specify the creation destination of the archive file.

 **Point**

- When an enterprise application project is exported, an EAR file containing the archive files of each of the projects included in the enterprise application project is created. Therefore, individual archive files need not be created.
- Of the JAR files included in the build path, those that cannot be included in the EAR must be set in the server classpath.

Deploying to the Server

Use the server deployment function to deploy the application to the server.

Refer to "Deploying the Application" in the "Interstage Application Server Java EE Operator's Guide (Java EE 6 Edition)" for details on deployment.

Chapter 10 Tips

This chapter describes information that is useful to know in advance, separated into the following sections: Tool Use, Programming Techniques, Eclipse Plug-in Use, and Thin Client Environment Edition.

10.1 Tool Use

This section introduces useful workbench functions under the following categories:

- Settings
This section introduces environment setup, workbench customization, and so on.

- Editors
This section introduces functions that are useful to know when using the editors.

- Coding support
This section introduces functions that are useful to know when entering coding.

- Searches
This section introduces useful functions related to searches.

- Build and debugging
This section introduces useful functions related to build and debugging.

- Help
This section introduces useful functions related to Help.

- Other
This section introduces useful functions related to basic workbench operations, Javadoc, refactoring, JUnit, and so on.

10.1.1 Settings

Using Proxies

If the workbench is used in an environment that uses proxies, setup is required using [General] > [Network Connections] on the preference page.

Key Binding

Specific key strikes and key sequences can be assigned to specific commands according to the situation. The assignment method for these can be customized using [General] > [Keys] on the preference page.

Refer to the following when performing customization:

- Schema
This is the key binding set. The default and Emacs are available.

- Binding
Actual key strokes and key sequences can be entered by typing from the keyboard.

- When
Select the situation for which the key binding is being set.

Colors and Fonts

The fonts and screen colors used within the workbench can be set from [General] > [Appearance] > [Colors and Fonts] on the preference page.

When the item to be changed is selected in the Tree view, change buttons are displayed for fonts and colors. Use these buttons to change the font or color.

Encoding

File encoding is identified from the character code entered in a file (HTML, JSP, XML, and so on). However, the encoding for Java source and other files that do not contain a character code are determined by the content of the following settings:

- Resource properties

The encoding for a file or for the files under a folder can be specified by selecting [Resource] then [Text file encoding] under the properties of the selected resource.

- Contents type default encoding

The default encoding can be set in contents type units using [General] > [Content Types] on the preference page. The Workspace default is specified using [General] > [Workspace].

HTML, JSP, or another initial value for the encoding embedded by the New wizard can be specified in the preference page, as follows:

File Type	Preference Page
CSS	[Web] > [CSS Files]
HTML	[Web] > [HTML Files]
JSP	[Web] > [JSP Files]
XML file	[XML] > [XML Files]

Enabling Functions not Visible by Default

Functions that are not used by ordinary users are not made visible under Interstage Studio. One example is the plug-in development function provided for developing individual Eclipse plug-ins or similar.

In order to enable these functions, select [General] > [Capabilities] from the preference page, then perform the following:

1. Click [Advanced] to display the [Advanced Capabilities Settings] dialog box.
2. Select the function that you want to enable, then click [OK].

10.1.2 Editors

Changing the Default Editor

When an editor is to be used to open a file, [Open With] can be selected from the context menu to select the editor that is to be used.

From [General] > [Editors] > [File Associations] on the preference page, selectable editors can be added and default editors can be set for specific file types.

Maximizing the Editor Area

In the ordinary perspective, the editor area is displayed in the centre of the workbench. If you want to enlarge the editor area during coding, you can toggle between maximizing the editor area and returning it to the normal size by double-clicking the tab displaying the editor file name.

Contents Assist

When the [Ctrl+Space] keys are pressed, input suggestions relevant to that situation are displayed. In order to narrow down the input suggestions, enter the first few characters and then press the [Ctrl+Space] keys. Alternatively, once the suggestions are displayed, the suggestions can be narrowed down by inputting the first few characters.

Searching for a Corresponding Bracket

To search for a corresponding bracket from the Java editor, select either the opening bracket or the closing bracket, then select [Navigate] > [Go To] > [Matching Bracket] from the menu.

In addition, users can double-click after the opening bracket or before the closing bracket to select the text enclosed by these two brackets.

Quick Diff

In the text editor, using the Quick Diff function, the differences between the current value and the status that was saved previously or the resource-managed file can be displayed as color coding on the left side of the editor.

The changed content can be checked from a display balloon by holding the mouse cursor over the changed location.

Spell Check

The text editor Spell Check function can be customized in [General] > [Editors] > [Text Editors] > [Spelling] on the preference page.

Format

A format function is provided for the following file types, and the format type can be customized in the preference page.

File Type	Preference Page
Ant script	[Ant] > [Editor] > [Formatter]
Java	[Java] > [Code Style] > [Formatter]
CSS	[Web] > [CSS Files] > [Editor]
HTML	[Web] > [HTML Files] > [Editor]
JSP	[Web] > [JSP Files] > [Editor]
XML file	[XML] > [XML Files] > [Editor]

When the editor is open or when the file is selected in a view, the format can be executed by selecting [Source] > [Format] from the menu.

Cleanup

The editors shown below provide a more powerful cleanup function than format. Open the file from an editor, then select cleanup from the following menus to use this function.

Editor	Menu	Typical Added Functions
Java Editor	[Source] > [Clean Up]	Change of static member access method Deletion of unused import Addition of @Override and @Deprecated
HTML/JSP Editor	[Source] > [Cleanup Document]	Uppercase and lowercase unification of tags and attributes Completion of mandatory attributes and missing tags Conversion of line delimiters
CSS Editor	[Source] > [Cleanup Document]	Uppercase and lowercase unification
XML Editor	[Source] > [Cleanup Document]	Compression of empty element tags Insertion of mandatory attributes and missing tags Conversion of line delimiters

Specifying Syntax Color

The syntax color can be customized in the editors for the following file types from the preference page.

File Type	Preference Page
Ant script	[Ant] > [Editor] [Syntax] tab
Java	[Java] > [Editor] > [Syntax Coloring]
CSS	[Web] > [CSS Files] > [Editor] > [Syntax Coloring]
DTD	[XML] > [DTD Files] > [Syntax Coloring]
HTML	[Web] > [HTML Files] > [Editor] > [Syntax Coloring]
JavaScript	[JavaScript] > [Editor] > [Syntax Coloring]
JSP	[Web] > [JSP Files] > [Editor] > [Syntax Coloring]
XML file	[XML] > [XML Files] > [Editor] > [Syntax Coloring]

Setting Automatic Execution at Save Time

The Java editor can execute actions automatically at save time. Specify these actions from [Java] > [Editor] > [Save Actions] on the preference page.

Formats and import organization can be executed.

Synchronizing Editor and View

If you want to check the location in package explorer or similar, of a file being edited with the Java editor, select [Show In] > [<Select view>] from the context menu. The file that is being edited is made active in the view.

In addition, the file being edited and the view selection can be synchronized by setting [Link Editor] in the upper-right menu of the view.

Switching Files in the Editor

If many files are open in editors, not all file names are displayed in the editor area tabs. The number of undisplayed files is shown by a number at the right edge of the editor tab display area. Click the number to display a list of these files. Users can switch the file being edited by selecting a file from this list.

Back Function

When referencing or editing multiple files using editors, users may want to go back to the location referenced immediately before.

In these cases, the [Back], [Forward], [Last Edit Location] functions are useful. From the menu, select [Navigate] > [Back] or similar. These functions can also be accessed from the toolbar.

10.1.3 Coding Support

Adding a Try/Catch Block

In Java, if exceptions must be processed, a try/catch block can be added easily, as follows:

1. In the Java editor, select the range in which you want to process exceptions.
2. From the menu, select [Source] > [Surround With] > [Try/catch Block].

Generating a Getter and Setter

When using an editor to edit a Java class, you can specify a field and generate the Getter and Setter by selecting [Source] > [Generate Getters and Setters] from the context menu.

Adding Import Statements

In Java, when coding using only the class name without specifying the package name, the editor displays an error indicating an unresolvable type until an import statement is added.

In this situation, the required import statement can be added automatically by selecting [Source] > [Organize Imports] from the context menu.

Templates

Code fragments that are used numerous times can be saved as templates. These can be inserted when you are using Contents Assist to edit a file, or when you are using the New wizard to create a file.

Templates can be edited from the preference pages shown below.

File type	Preference Page
Ant script	[Ant] > [Editor] > [Templates]
Java	[Java] > [Editor] > [Templates]
SQL file	[SQL Development] > [SQL Editor] > [Templates]
CSS	[Web] > [CSS Files] > [Editor] > [Templates]
DTD	[XML] > [DTD Files] > [Templates]
HTML	[Web] > [HTML Files] > [Editor] > [Templates]
JSP	[Web] > [JSP Files] > [Editor] > [Templates]
XML file	[XML] > [XML Files] > [Editor] > [Templates]

Variables (user, date, selection resource, and so on) previously prepared in the workbench can be used in templates.

Snippets

Fragments used in coding can be registered in advance as snippets, as shown below. They can then be inserted easily by using the mouse to drag and drop.

1. Display the Snippets view.
2. Select [Customize] from the context menu.
3. Enter the name, variables, and template pattern in the [Customize Palette] dialog box.

If a template that uses variables is inserted, variable values can be specified during template insertion.

Note that templates cannot be added, deleted, or changed for category JSP that is already defined.

File Comments and Type Comments

File comments and type comments can be set from the preference page using [Java] > [Code Style] > [Code Templates]. This enables comments to be unified when creating a Java class.

Variable Name Suggestions

When declaring a field or local variable, enter the type name and a blank space, then enter the [Ctrl+Space] keys. This displays field name or variable name suggestions.

Variable and Return Highlighting

When a field or local variable name is selected, the places where that field or local variable is used are highlighted.

In addition, when a return value type is selected in a method, the return location is highlighted.

TODO Comments

If things that must be done later are coded in the source using a TODO comment, that information is collected during build and listed in the Tasks view. The user can jump from the Tasks view to the place where the comment is coded.

This function can be used with Java, XML, HTML, JSP, and so on. Task tags other than TODO can be added in the preference pages shown below for a variety of purposes.

File type	Preference Page
Java	[Java] > [Compiler] > [Task Tags]

File type	Preference Page
XML,HTML,JSP and so on	[General] > [Editors] > [Structured Text Editors] > [Task Tags]

String Externalization

Character string constants that have been coded directly in a Java source can be extracted, gathered together and managed in a property file, as shown below.

1. Select the package, source folder, or similar from the view.
2. From the menu, select [Source] > [Externalize Strings].
3. Select the source to be targeted in the [Externalize Strings] dialog box, then click the [Externalize] button.
4. The String Externalization wizard is displayed. Specify the property file to be used when externalizing strings, the key, the accessor class, and so on, and execute the string externalization.

10.1.4 Searching

Opening a Java Declaration

When coding in Java, you may want to check type, variable, method, and other declarations. If so, the declaration can be checked by selecting the variable, method, or similar, that you want check in the code, and selecting [Open Declaration] from the context menu.

Searching Java Reference Locations

When coding is in Java, you may want to search for the affected locations. If so, select the part for which you want to find affected locations in the code, then select [References] > [<Select range>] from the context menu. This displays a list of reference locations in the Search view.

Restricting the Search Range

When executing a search, the search range can be specified as shown below:

- Workspace
All resources are searched.
- Selected resources
Resources under the resource selected in the view are searched.
- Enclosing project
The project to which the resource selected in the view belongs is searched.
- Working set
The search targets can be customized by setting a working set.

Search Results History

Search results are displayed in the Search view. The Search view toolbar has a [Show Previous Searches] button. This allows users to select from the search history to switch the contents displayed in the Search view.

10.1.5 Build and Debugging

Problems View Filter Function

When many problems are displayed in the Problems view, users can select [Configure Filters] from the upper-right menu of the view to customize the contents displayed.

The following filter functions are available:

- Range specification
The display contents can be restricted to projects that include the selected resource, for example, by specifying [On any element in same project].
- Severity specification
Users can specify whether to display errors, warnings, or information.
- Problem type specification
The display contents can be restricted by specifying builder, validator, or other problem types.
- Other
The display contents can be narrowed down by specifying containing or not containing specific coding.

Ant Invocation During Build

Ant can be invoked during a build by coding an Ant script and setting it as follows:

1. Select [Builders] from the project properties.
2. Click [New], then select [Ant Builder] in the [Choose configuration type] dialog box.
3. In [Build file] under [Main], specify the Ant script file to be used.
4. If updating files that are in the Workspace with an Ant script, in the [Refresh] tab, specify the resources that should be updated after the Ant script has executed.
5. Under the [Targets] tab, the target can be set to suit the manual build, automatic build, or other build mechanism.
6. The properties required for Ant execution can be specified from the [Properties] tab.



Point

The Ant script can be debugged by selecting the Ant script, then selecting [Debug As] > [Ant Build] in the context menu.

Adding Classpaths

When adding a JAR file that is not in the Workspace to a classpath, the environment-dependent part can be absorbed using either of the methods below. This improves project resource portability.

Classpath variable

On the preference page, use [Java] > [Build Path] > [Classpath Variables] to associate and manage names and paths.

To add to the classpath, select project properties, then [Java Build Path], then the [Libraries] tab, then use the [Add Variable] button.

In the displayed dialog box, select the classpath variable then, from the [Extend] button, select the resource from under the path that is associated with the classpath variable.

User library

On the preference page, use [Java] > [Build Path] > [User Libraries] to associate and manage a name and multiple JAR files. This is useful when specifying a library group.

To add to the classpath of a specified library, select project properties, then [Java Build Path], then the [Libraries] tab, then use the [Add Library] button.

Performing Build using isstudiobld.exe

Use the isstudiobld.exe tool to build a project from the command line without launching the workbench. The isstudiobld.exe tool is in the following location:

```
<Workbench installation folder>\eclipse\isstudiobld.exe
```

The method of using the isstudiobld tool is described below.

Format:	
isstudiobl -data <workspace> [options] [<target> [<target2>...]]	
Note: [] can be omitted.<> are used to specify the respective values.	
Parameters:	
-data <workspace>	Specify the workspace folder.
<target>	Specify the target of the Ant being executed.
Options:	
-f <buildfile>	Specify the build file (Ant script). When this option is omitted, use the build file at the following address. <Installation folder>\IDE\1101\etc\build\buildAll.xml
-verbose or -v	Displays details.
-D<property>=<value>	Specify a property.
-propertyfile <name>	Load from a file where properties are specified.
-vm <JDK installation folder> <jre\bin	Specify a folder in which there is a JDK java.exe that executes the build file (Ant script). When this option is omitted, use the JDK specified in the environment variable PATH.



Note

- Use the eclipse.incrementalBuild ant task to build the project.
- In the eclipse.incrementalBuild Ant task, build the project according to the [JavaCompiler] and [Java Build Path] information specified in the project.
In the Ant tasks except eclipse.incrementalBuild, use the JDK that -vm option specifies.

The method for using the eclipse.incrementalBuild Ant task is shown below.

Attributes	Description
kind	The build type. Specify either incremental, full, or clean. The default value is incremental.
project	The project to be built. If omitted, a build of the Workspace is performed.

Example: Perform a full build of the Workspace.

```
<eclipse.incrementalBuild kind="full" />
```

Example: Clean the project project1.

```
<eclipse.incrementalBuild project="project1" kind="clean" />
```

Temporary Disablement of Breakpoints

When many breakpoints are set, execution is interrupted at too many places and debug efficiency may deteriorate.

In this case, breakpoints can be disabled temporarily without releasing the breakpoints. When [Group By] > [<Grouping Method>] is selected from the upper-right menu of the Breakpoint view, the breakpoints are grouped and therefore can be disabled together as a group. Alternatively, [Skip All Breakpoints] can be selected from the Breakpoint view toolbar to disable all breakpoints.

Exception Breakpoints

Users may want to interrupt program execution if an exception has occurred. If so, use an exception breakpoint. From the menu, use [Run] > [Add Java Exception Breakpoint] to specify an exception class. This sets a breakpoint that interrupts execution when that exception has occurred.

Step-in to selected Method

During debugging, when step-in execution is performed for a statement such as,

```
addValue(obj.getName(), obj.getValue());
```

step-in is performed in the following sequence when a simple step-in (F5) is executed:

1. Step-in to obj.getName()
2. Step-in to obj.getValue()
3. Step-in to addValue()

If you want to step-in only to addValue(), select "addValue" on the editor, then select [Step Into Selection] from the menu. This performs 3. immediately without performing 1. and 2.

Expression Evaluation

In addition to checking the object contents, users may want to test method execution during debugging.

If so, enter the expression in the Display view or the details frame of the Variables view and select the expression, then select either [Display], [Inspect], or [Execute] from the context menu.

Specifying the Java Version Used for Build and Debugging

The JRE System Library specified in the project build path is used to determine the Java version used for build and debugging. You can specify the Java version by configuring the JRE System Library of the project.

Specify the Java version with the JRE System Library as follows:

- Default JRE of the workspace
Use the default JRE of the workspace. The default JRE of the workspace is the installed JRE that is checked in [Java] > [Installed JREs] page in the Preferences dialog box that appears when [Window] > [Preferences] is selected from the menu bar. If the default JRE of the workspace is changed, the JRE System Library used in each project is also changed.
- Alternate JRE
Use the Alternate JRE to specify the Java version for a project. Select one of the installed JREs as that for the Java version used for build and debugging.

To configure the JRE System Library of a project, follow the procedure below.

1. Select a project from the Package Explorer view or the other views.
2. Select [Properties] from the context menu, or select [File] > [Properties] from the menu bar. The [Properties] dialog box appears.
3. Select [Java Build Path] in the left pane. The [Java Build Path] page is displayed.
4. Select the [Libraries] tab.
5. Select [JRE System Library] from the [JARs and class folders on the build path] list, and click [Edit]. The [Edit Library] dialog box appears.
6. Select [Workspace default JRE] or [Alternate JRE] from the System Libraries. If [Alternate JRE] is selected, select the Installed JRE to be used from the combo box.

To add JDK to the installed JRE, follow the procedure below:

1. From the workbench menu, select [Window] > [Preferences].
2. In the [Preferences] dialog box on the left pane, select [Java] > [Installed JREs].
3. In the right [Installed JREs] window, click [Add].
4. In the [JRE Type] window, select "Standard VM", then click [Next].

5. In [JRE home] of the [JRE Definition] window, specify the JDK installation folder. A name in accordance with the folder name selected in [JRE name] is displayed, and a list of JAR files for the JDK library specified in [JRE system libraries] is displayed. Click [Finish] to close the window.

Note

- Though [Execution Environment] is included as a System Library option, you cannot use it in Interstage Studio.
- Do not set a compiler compliance level that is higher than that of the Java version specified in the JRE System Library used.

Point

In Java compilation, you can specify Compiler compliance level, such as class files compatibility and Source compatibility, etc. You can specify compiler compliance level on the [Java Compiler] page of the project properties or on the [Java] > [Compiler] page of the workspace preferences.

Stack Trace View

The Stack Trace View can be used to read FJVM logs or thread dumps collected by the thread dump tool and to produce a hierarchical display of the list of registered threads and stack traces.

Use the following procedure to display the Stack Trace View:

1. Select [Window] > [Show View] > [Other] from the menu bar.
2. Select [Java] > [Stack Trace] in [Show View] dialog box.

See

For details on FJVM logs, refer to the "Interstage Application Server Tuning Guide" > "JDK/JRE Tuning".

Note










- The Stack Trace View cannot be used in the Java EE 6 workbench.
- The thread dump files that can be displayed by the Stack Trace View are those that were output with the thread dump tool "-f" option specified (thread dump output destination specified). Standard output thread dump files and thread dumps registered in application server log files cannot be displayed.

The Stack Trace View has the following functions:

- Open FJVM Log/Thread Dump
Opens a FJVM log file or thread dump file and displays its data in the view. The contents of the opened file are also displayed in the Console view.
- Close File
Closes the file currently displayed in the view.
- Display Selected File
Used to select the file to be displayed when multiple files are open.
- Display Console
Displays the contents of the file currently displayed in the Stack Trace view, in the Console view.
- Sort
Displays threads after sorting them. Clicking this button toggles between on and off for sorting. Sorting is off by default. If sorting is on, threads other than system threads are displayed first in ascending order of thread names, and system threads are then displayed in ascending order of thread names. If sorting is off, threads are displayed in the order that they were recorded in the file.

- Show System Threads
Displays system threads. Selecting this command item toggles between showing system threads and hiding system threads.
- Show Qualified Names
Displays qualified names (package names) together with the class names displayed for stack trace and monitor elements.
- Show Monitors
Displays monitors. If some threads and monitors in the thread dump are in the deadlock state, the entries for them are highlighted and the icons for them are also changed to indicate the deadlock.
Selecting this command item toggles between turning on and turning off monitors.
- Set Encoding
Specifies the type of encoding used when an FJVM log or thread dump file is loaded. The default is UTF-8. Use this command only if the file to be loaded is encoded in a format other than UTF-8.

If a thread dump contains monitor information and [Show Monitors] is turned on, the following elements are displayed.

Element	Icon	Description
Monitor owned by the thread	 	Displayed as a child of the thread. It represents a monitor owned by the thread. The class name and ID of the object acting as the monitor are displayed to the right of the icon. If the thread owns multiple monitors, as many instances of this element as the number of these monitors are displayed. If the monitor is in the deadlock state, the right icon is displayed.
Thread that is waiting for the monitor	 	Displayed as a child of a monitor owned by the thread. It represents a thread that is waiting for the monitor. The thread name and information indicating whether the thread is a system thread are displayed to the right of the icon. If multiple threads are waiting for the monitor, as many instances of this element as the number of these threads are displayed. If the thread is in the deadlock state, the right icon is displayed.
Monitor for which the thread is waiting	 	Displayed as a child of the thread. It represents a monitor for which the thread is waiting. The class name and ID of the object acting as the monitor are displayed to the right of the icon. If the monitor is in the deadlock state, the right icon is displayed.
Thread that owns the monitor	 	Displayed as a child of a monitor for which the thread is waiting. It represents a thread that owns the monitor. The thread name and information indicating whether the thread is a system thread are displayed to the right of the icon. If the thread is in the deadlock state, the right icon is displayed.
Monitor locked		Displayed between stack trace entries to identify the process where a monitor became locked. The class name and ID of the object acting as the monitor are displayed to the right of the icon.

10.1.6 Help

Adding Documents to Help

A customer-specific document that is referenced during development work can be added to the Help system. To add a document, from the [Window] menu, select [Preferences], then [Help] > [Documentation].

Specify the following values in the [User documentations] page that is displayed when a document is added or edited:

Item	Settings Value
Documentation name	Specify the document name.
Documentation URL	Select this item if the document is not archived.

Item		Settings Value
	Location path	Specify the first page of the document in URL format. If it is a local file, specify as "file:".
Documentation in archive		Select this item if the document is archived in a ZIP file or a JAR file.
	Archive path	Specify the path to the archive file.
	Path within archive	Specify the first page within the archive file.
Contents are automatically generated recognizing the link of the initial page		When this option is selected, the links within the HTML page specified in the location path or the path within the archive are extracted and a table of contents is created. If the document is not archived, this option is enabled only if the location path begins with "file:".
Searches the documents in the same folder		When this option is selected, the following HTML pages within the document can be added as search targets: - *.html, *.htm - *.xhtml, *.xhtm However, for a Javadoc file, only the following HTML pages can be added as search targets: - overview-summary.html - package-summary.html If this option is not selected, the only search targets are the HTML pages registered in the table of contents.



Note

Do not include national language characters in location paths, archive paths, or paths within archives. HTML pages do not display correctly if national language characters are included in these paths.

Infopop Help

If the method for using a dialog box, window, or similar is not understood, press the [F1] key while that GUI is displayed. This displays the Help associated with that GUI.

10.1.7 Other

Multiple Workbench Display

If you want to reference different places in a view or editor at the same time, or if it is inconvenient to switch between perspectives, you can select [Window] > [New Window] from the menu to enable multiple displays of the workbench.

View Display Method

As with editors, a view can be maximized to temporarily facilitate viewing by double-clicking on the title bar. The following methods can also be used to display a view to suit user preferences:

- Detaching views

A view can be detached from the workbench window and displayed separately by right-clicking on the view title bar, then selecting [Detached] from the context menu.

This enables the detached view to be displayed in the front at any position and at any size. To return the view to the inside of the workbench window, select [Detached] from the context menu again.

- Fast view

Users can toggle between displaying and hiding a view by right-clicking on the title bar and selecting [Fast View] from the context menu. The user can then use the button on the workbench status bar to display or hide the view. To stop using the quick view display, select [Fast View] from the context menu again.

Preferences and Properties Page Filters

Keywords can be entered at the upper-left of the project property and preferences dialog boxes to narrow down the pages displayed at the left edge.

Refactoring (Renaming)

If you want to change the name of a variable, field, method, class, package, or other item, the refactoring function can be used as shown below to make the required correction in all places, included invoked locations, at once.

1. Select the target to be changed in the editor or view.
2. Select [Refactor] > [Rename] from the menu.
3. Change the name as indicated in the displayed contents.

Checking Differences

Use the Project Explorer view or similar to select the resources to be compared, then select [CompareWith] > [Each Other] from the context menu. This enables users to check differences in folder or file units.

Javadoc

The Javadoc of a selected class, method, field, or other element can be displayed in the Javadoc view. Alternatively, the Javadoc can be checked in a balloon by holding the mouse cursor over the element in the editor.

The Javadoc to be displayed is either the item coded in Javadoc in the source, or the item in which the JAR file property [Javadoc Location] is coded.

In the latter case, the Javadoc can also be displayed on a browser by selecting [Navigate] > [Open External Javadoc] from the menu.



.....
A Javadoc document can be created, if desired, by selecting [Java] > [Javadoc] from the Export wizard.
.....

10.2 Programming Techniques

This section introduces notes concerning application development.

10.2.1 Notes on Encoding in Web Applications

When fetching parameter values from a request object, processing must take into account the input form encoding. Therefore, uniform encoding in Web applications is recommended.

Set HTML, JSP, and servlet encoding as shown below.

Web Component	Setting Method	Coding Example
HTML	Specify using the meta tag Content-Type (charset) value.	<code><meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1" /></code>
JSP	In the page directive contentType attribute, specify the HTTP header Content-Type (charset) value.	<code><%@ page language="java" contentType="text/html; charset= ISO-8859-1"%></code>
Servlet	In the HttpServletResponse interface setContentType method, specify the HTTP header Content-Type (charset) value.	<code>res.setContentType("text/html; charset=ISO-8859-1");</code>

Point

- With JSP, encoding for when the file is saved can be specified separately. Code this specification in the page directive pageEncoding attribute.
 - The HTML and JSP New wizards provide a method for embedding the encoding when a file is generated. Embedded encoding can be specified in Workspace units. Specify as follows in their respective preference dialog boxes:
 - HTML: [Web] > [HTML Files]
 - JSP: [Web] > [JSP Files]
-

If filtering takes the encoding into account, for example, when the class shown below is created, the filter mapping must be defined in web.xml.

In this example, the encoding is specified in the web.xml initial parameters and can be customized to suit the environment in which this filter class is used. Thus, the initial parameters also need to be defined.

Example of Taking Encoding into Account in a Filter

```
package filter;

import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;

public class flt_request implements Filter {
    private FilterConfig config=null;
    public void init(FilterConfig conf){
        this.config=conf;
    }
    public void destroy(){}
    public void doFilter(ServletRequest request, ServletResponse response, FilterChain
chain)
throws ServletException, IOException {
        request.setCharacterEncoding(config.getInitParameter("encoding"));
        chain.doFilter(request,response);
    }
}
```

Note

The encoding specification in setCharacterEncoding is enabled for the POST method, but is not enabled for the GET method because specification in setCharacterEncoding does not apply for the part of the URL for which the GET method passes the parameters.

10.2.2 Using JNDI Lookup to Obtain Objects

Under Java EE, resources and objects can be obtained by means of a Dependency Injection. However, Dependency Injection has the following problems:

- It can only be used with components managed by Java EE containers
Dependency Injection injects dependent resources and objects by means of a Java EE container. Therefore, it can be used only with components managed by Java EE containers. This is not a method that can be used with any class.
- Properties coded directly into code
When Dependency Injection is used, properties are coded in annotation, and this coding is coded directly into the source. Therefore, if a property is changed, the Java source must be compiled.
The lookup can be used without taking into account the actual environment because the association between a reference name and a JNDI name can be changed using the deployment descriptor.

The above problems can be resolved by using JNDI lookup to obtain resources and objects in the conventional manner. Various examples are shown below.

EJB Local Interface Lookup

Perform EJB local interface lookup as shown below.

Lookup Example

```
InitialContext ic = new InitialContext();
Calc calc = (Calc)ic.lookup("java:comp/env/Calc");
```

Example of Coding a deployment descriptor for Lookup

```
<ejb-local-ref>
  <ejb-local-ref-name>Calc</ejb-local-ref-name >
  <local>sample.Calc</local>
  <ejb-link>ejb1.jar#CalcBean</ejb-link>
</ejb-local-ref >
```

JMS Destination Lookup

Perform JMS Destination lookup as shown below.

Lookup Example

```
InitialContext ic = new InitialContext();
Queue queue = (Queue)ic.lookup("java:comp/env/jms/myQueue");
```

Example of Coding a Deployment Descriptor for Lookup

```
<message-destination-ref>
  <message-destination-ref-name>jms/myQueue</message-destination-ref-name>
  <message-destination-type>javax.jms.Queue</message-destination-type>
  <message-destination-usage>Produces</message-destination-usage>
  <message-destination-link>myDestination</message-destination-link>
</message-destination-ref >
...
<message-destination>
  <message-destination-name>myDestination</message-destination-name>
  <mapped-name>testQueue</mapped-name>
</message-destination>
```

JDBC Resource Lookup

Perform JDBC resource lookup as shown below.

Lookup Example

```
InitialContext ic = new InitialContext();
DataSource oracle = (DataSource)ic.lookup("java:comp/env/jdbc/Oracle");
```

Example of Coding a Deployment Descriptor for Lookup

```
<resource-ref>
  <res-ref-name>jdbc/Oracle</res-ref-name>
  <res-type>javax.sql.DataSource</res-type>
  <res-auth>Container</res-auth>
```

```
<mapped-name>OracleTestServer</mapped-name>
</resource-ref>
```

10.3 Eclipse Plug-in Use

This section describes how to use the Eclipse plug-ins and provides notes on them.

10.3.1 Installing and Uninstalling Plug-ins

This section describes how to install and uninstall Eclipse plug-ins.

Plug-ins can be added to the workbench by storing them in the "dropins" folder under the installation folder.



Point

.....
In a standard installation, the installation folder is <Workbench folder>eclipse.
.....

Installation procedure

1. Stop the workbench.
2. Store the plug-in to be installed in the "dropins" folder in the installation folder.
The plug-in storage methods below are supported:

Store the plug-in in the "dropins" folder

```
eclipse/
  dropins/
    com.fujitsu.*_1.0.0.jar
    com.fujitsu.*_1.0.0/
      plugin.xml
      tools.jar
      ... etc ...
    ...
```

Store the "eclipse" folder in the "dropins" folder

```
eclipse/
  dropins/
    eclipse/
      features/
      plugins/
```

Store the folder of each component in the "dropins" folder

```
eclipse/
  dropins/
    gef/
      eclipse/
        features/
        plugins/
    emf/
      eclipse/
        features/
        plugins/
    ... etc ...
```

Store the "link" file in the "dropins" folder

```
eclipse/  
  dropins/  
    fujitsu.link
```

3. From [Run] or [Command Prompt], start the workbench with the "-clean" option.
In a standard installation, use "<Workbench folder>\eclipse\isstudio.exe -clean" to start the workbench.
4. Select [Help] > [About Interstage Studio] > [Feature Details] or [Plug-in Details]. If the installed features and plug-ins are displayed, then installation was successful.

Uninstall procedure

1. Stop the workbench.
2. Delete the plug-in stored in the "dropins" folder of the installation folder.
3. From [Run] or [Command Prompt], start the workbench with the "-clean" option.
In a standard installation, use "<Workbench folder>\eclipse\isstudio.exe -clean" to start the workbench.
4. Select [Help] > [About Interstage Studio] > [Feature Details] or [Plug-in Details]. If the installed features and plug-ins are not displayed, then uninstall was successful.



If a link file was used in the plug-in storage method, also delete the plug-in specified in the link file.

10.3.2 Notes on Installing Eclipse Plug-ins

- Refer to the documentation for the Eclipse plug-in being installed and check whether it is compatible with workbench Eclipse version.
- The customer must confirm operation of the installed Eclipse plug-ins.
- The operation of installed Eclipse plug-ins is not guaranteed.
- Support is not provided for problems arising from installation of Eclipse plug-ins.
- Fujitsu has performed fixes for the Eclipse plug-ins listed in the table below. Therefore, these plug-ins are not the same as the Eclipse plug-ins published by the Eclipse Foundation.

<Standard workbench>

Plugins	Version
Eclipse	3.4.1
GEF	3.4.1
EMF	2.4.1
WTP	3.0.1
DTP	1.6.1

<Java EE 6 workbench>

Plugins	Version
Eclipse	3.6.2
GEF	3.6.2
EMF	2.6.1
WTP	3.2.3
DTP	1.8.2

- In principle, these plug-ins are compatible with the Eclipse Foundation Eclipse plug-ins, but the fixes performed by Fujitsu might affect the Eclipse plug-ins being installed.

10.4 Thin Client Environment Edition

This section provides notes on use in a thin client environment.

- Separate Workspaces must be created for each user.

A Workspace cannot be shared with another user. It is essential to create one or more Workspaces for each user.

- Projects outside linked folders, files, and Workspaces can only be used for reference.

A user cannot work on projects outside linked folders, files and Workspaces at the same time as another user. Only reference is possible.

- Environment settings are saved separately for each Workspace.

Settings that are configured in the settings page displayed from [Window] > [Preferences] cannot be shared by other users. If the same settings are required, configure them separately.

- Separate IJServer Clusters must be created for each user.

The same IJServer Cluster cannot be used by different users. It is essential to create a separate IJServer Cluster for each user. The Interstage Java EE DAS cannot be used.

- Use of the Interstage Management Service Operation Tool to stop services is not recommended.

If a service stop cannot be avoided, check that no other users are using Interstage or the Interstage Management Console before stopping services.



To use the development environment products or client operating environment products in the terminal services of Windows Server 2003/Windows Server 2008 or in the remote desktop services of Windows Server 2012, it is necessary to purchase licenses equal to the number of clients that simultaneously use the terminal services or remote desktop services.

10.5 Developing in an IPv6 Environment

This section provides notes for use in an IPv6 environment.



This product supports only IPv6/IPv4 dual stacks. Operation with IPv4 disabled is not supported.

Note: The following functions are only applicable to the IPv4 environment.

- [Servers] view function in the workbench
- All debugging functions

10.5.1 Specifying IPv6 Addresses

When using the standard workbench and the Java EE 6 workbench, add the following IPv6 unicast address and hostname declaration to the hosts file below to use IPv6 addresses. Specify the hostname after adding the declaration.

```
<Windows installation folder>\system32\drivers\etc\hosts
```

10.5.2 Notes on Application Development in an IPv6 Environment

The following section describes some points to note when developing applications in an IPv6 environment.

Developing Java EE 5 and J2EE1.4 applications

- Create IJServer cluster and IJServer in the localhost if they are to be created with "[6.2.1 Preparing the Deployment Destination for Verifying Application Operation](#)".
- Use the localhost server to check operation. Specify "localhost" as the host name of the server when adding a new server (with the New Server wizard).
- Specify the same URL of the server in the "localhost" as the URL specified in [WSDL file name] in the New Web Service (JAX-WS) wizard.
- Use ftp or file sharing functions to distribute to the operating environment. After sending the archive files to the server, deploy using the deployment functions of Interstage Application Server.

Application debugging

- Machines with applications to be debugged must be specified through "localhost" or IPv4 address.

Appendix A How to use Samples

This appendix describes the provided samples.

Sample Storage Destinations

The standard workbench and Java EE 6 workbench sample applications are stored in the following locations:

<Standard workbench installation folder>\sample

<Java EE 6 workbench installation folder>\sample

Samples List

Sample applications are stored in the following locations:

File Name	Project Name	Notes
WebSample.zip	WebSample	This is the application created as described in " 2.2 Introduction " in "Developing Web Applications".
EJBSample.zip	EJBSample	This is the application created as described in " 3.2 Introduction " in "Developing Enterprise JavaBeans (EJB)".
	EJBClientSample	
	EJBEARSample	
JPASample.zip	JPASample	This is the application created as described in " 4.2 Introduction " in "Developing Applications that use Java Persistence API (JPA)".
	JPAClientSample	
	JPAEARSample	
WebServiceSample.zip	WebServiceSample	This is the application created as described in " 5.2 Introduction " in the "Developing Web Service Applications".
	WebServiceClientSample	
	WebServiceEARSample	
AppletSample.zip	AppletSample	This is the application created as described in " 7.2 Introduction " in the "Developing Java Applications".

A "6" has been added to the end of the file and project names in the Java EE 6 workbench samples.

Procedure for using Samples

Use the following procedure to import samples to the Workspace:

1. Launch the workbench.
2. Select [File] > [Import] from the menu.
3. Select [General] > [Existing Projects into Workspace] as the import source selection, then click [Next].
4. On the next screen, select [Select archive file], then click [Browse] to specify the archive file to be imported.
5. Click [Finish].



Note

- When using the Java EE 6 workbench (exclude AppletSample6.zip), register the Interstage Application Server V11.1 (Java EE 6) runtime in "[9.6.1.1 Preparing to Operate Servers](#)" before importing the samples.

- If the Java EE 6 workbench Web service sample (WebServiceSample6.zip) is used in an environment in which this product has been installed (in an environment that is not "C:\Interstage"), refer to the notes in "[9.5.1.3 Developing Web Service Applications](#)" after importing the samples, before reconfiguring the WebServiceClientSample6 project build path.



Appendix B Migrating Resources from a Previous Version

This section explains how to migrate the resources developed with workbench of a previous version (Interstage Studio and Interstage Apworks).

Migration procedure

To use a workspace or project created in a previous version, follow the respective procedures below. For notes on workspace and project migration, refer to the sections in the document.

Using a workspace created in a previous version

Launch the workbench and specify the workspace of the previous version in the "Startup" dialog box - the workspace and the projects stored in it will then be ready for use.

Using projects created in a previous version

To migrate some projects in a workspace (instead of the entire workspace), follow the procedure below:

1. Copy the project to the workspace folder by using Explorer or another file management program.
2. From the menu bar, select [File] > [Import], and then select [General] > [Existing Projects into Workspace].
3. On the [Import Projects] page, in [Select root directory], specify the workspace folder to which the project was copied in step 1.
4. The list of the projects that can be imported is displayed in the [Projects]. Make sure that the project to be imported is selected, then click [Finish].

Using projects created by ComponentDesigner

If you want to migrate ComponentDesigner projects, they can be imported using the following procedure:

1. From the menu bar, select [File] > [Import], and then select [Other] > [Existing ComponentDesigner's Project].
2. At [Project to be Imported], specify the path of the project file to be imported.
3. Check that the project name is displayed at [Project name], and then click [Finish].

Point

Some workspace and project settings can be automatically updated using the [Update Workspace/Projects of Previous Version] command. For details, refer to "[B.6 Automatic Update of the Workspace and Projects](#)".

Note

The message "Project of an old version cannot be built. Update the project." may be displayed in the Problems view when the build is performed if "[B.6 Automatic Update of the Workspace and Projects](#)" is not performed. Building will be possible after performing "[B.6 Automatic Update of the Workspace and Projects](#)".

B.1 Notes on Migrating Resources of Up to V10

This section describes some issues with migrating V10 resources.

Note

Resources that can be migrated to the Java EE 6 workbench of this version are the resources created in the Java EE 6 workbench of a previous version and the resources in the Java Project and Java Application Project.

B.1.1 Notes on Workspace Migration

Refer to the following cautions when using a V10 workspace in this version's workbench.

Migrating Templates

The Template view that was provided with the V10 Compatible Workbench is not provided in the Java EE workbench. If a V10 Compatible Workbench workspace is opened in the workbench of this version, the template of the Template view that was used in the V10 Compatible Workbench can be used in the workbench. Templates for files with the extension java, xml, html, htm, jsp, js, css, dtd, and sql can be used. These templates can be used as the templates of each corresponding editor.

To use a template that was used in a V10 workbench Template view as the template of each editor, the template must be migrated. The procedure for migrating templates is shown below:

1. From the menu bar, select [Project] > [Update Workspace/Projects of Previous Version].
2. The [Update Workspace/Projects of Previous Version] dialog box is displayed. If update of the workspace has not been implemented, "Workspace" will display on the left side of the dialog box. Select "Workspace", and then click [OK]. The files required for template migration are generated.
3. Copy the following files to any projects for work use.
<workspace folder>\.metadata\templateBackups\studioTemplates.xml
4. Open the copied XML files in an XML editor. The opened XML file will be in the following tree structure:

```
<syntaxtemplate>
+ <root>
+ <category>
+ <template>
```

The name attribute of the <template> element is the name of the template. The description attribute is the description of the template. The context attribute is the context. The child text of the <template> element is the template.

5. Delete templates that do not require migration. To delete, in the XML editor design page, delete the <template> element.
6. The context of non-Java templates is not set. The attribute value of the context attribute is a null string. Enter an appropriate context in the context attribute. If the file type is not modified, the context can be modified after the import. Therefore, set the following values for each file type you want to import.

File Type	Attribute Value of Context Attribute
Java	Java
XML	xml_all
CSS	css_all
DTD	dtd_new
HTML	html_all
JavaScript	javaScript
JSP	jsp_all
SQL	org.eclipse.datatools.sqltools.editor.template.sql.generic

7. Open the source page of the XML editor, and modify the variable name. In the V10 Compatible Workbench, the variable name was not limited to the \${xxx} format, and Japanese characters etc could be used. In the workbench of this version, only alphanumeric characters and underscores ("_") can be used in the variable name. Migration cannot be performed if other characters are used.
8. Import templates from the template settings of each corresponding editor. With import, only the context templates associated with that editor are imported.

Migrating J2EE Associated Templates

A template definition file is prepared for the following J2EE templates.

Context	Name	Description
Java	EJB Home reference	Performs reference processing for EJB Home.
	EJB Local Home reference	Performs reference processing for EJB Local Home.
	MessageProducer	Sends a message to the EventChannel.
	Point-To-Point	Sends a message to the queue.
	Publish/Subscribe	Sends a message to the topic.

Import and apply the definition file as required. The procedure for applying the definition file is shown below.

1. From the menu bar, select [Window] > [Preferences].
2. The [Preferences] dialog box is displayed. Select [Java] > [Editor] > [Templates].
3. The [Templates] page is displayed. Click [Import].
4. The [Importing Templates] dialog box is displayed. Specify the following file to import the template definition.

```
<workbench installation folder>\etc\templates\templates_ejb_jms.xml
```

Snippets View Tags

If a V10 Compatible Workbench workspace is opened in the workbench for this version, there are some tags and attributes defined in the Snippets view that cannot be used in the workbench. To prevent these tags and attributes from displaying in the Snippets view, the following definition file must be deleted:

```
<user document folder>\Interstage Studio\workspace\.metadata\.plugins\org.eclipse.wst.common.snippets\user.xml
```



If individual tags were registered in the Snippets view of the V10 workbench, and the above-mentioned definition file is deleted, the individual tag information is also deleted. Therefore, use the customization function of the Snippets view again to register the tags.

B.1.2 Notes on Project Migration

Notes on WAR/EJB-JAR/EAR File Creation

In the workbench, WAR/EJB-JAR/EAR files are not created at build time. Use the Export wizard to create WAR/EJB-JAR/EAR files. If "B.6 Automatic Update of the Workspace and Projects" is performed on a V10 Compatible Workbench project, the J2EE package builder for creating WAR/EJB-JAR/EAR files is deleted. Instead, the ANT script file for creating these archive files is created in the project.

Archive File Type to be Generated	ANT Script File
WAR file	buildWAR.xml
EJB-JAR file	buildEJB-JAR.xml
EAR file	buildEAR.xml



When buildEAR.xml, the ANT script file for creating EAR files, is executed, it must be executed using the same JRE as the workspace. To do that, follow the procedure below:

1. Select the buildEAR.xml file, then from the context menu, select [Run As] > [Ant Build...]. The [Edit Configuration] dialog box is opened.
2. Click the [JRE] tab, and open the JRE page.

3. Select [Run in the same JRE as the workspace].
4. Click [Apply].

.....

To create WAR/EJB-JAR/EAR files at build time, add an ANT script file for creating these archive files to the builder. This procedure is shown below.

1. Display the Project Properties.
2. In the left pane of the Project Properties window, select [Builders]. The builder page is displayed.
3. In the right pane of the builder page, click the [New] button. The [Choose configuration type] dialog box is displayed.
4. In the [Choose Configuration type] dialog box, select [Ant Builder] from the list, then click [OK]. The [Edit Configuration] dialog box is opened.
5. In [Name], enter the name of this process.
6. In the [Main] tab, click the [Browse Workspace] button of [Buildfile]. The [Choose Location] dialog box is displayed.
7. In the [Choose Location] dialog box, select the ANT script file for creating archive files, then click [OK].
8. If creating EAR files, make settings to execute the ANT script file using the same JRE as the workspace. In the [JRE] tab, select [Run in the same JRE as the workspace].
9. Click [OK]. Return to the Project Properties window.
10. Make sure that the added item is the last one in the builder. If it is not, then use the [Up] and [Down] buttons to move the item to the last position in the builder.

Note

.....

Even if an ANT script file is executed to create or update an archive file, resources are not updated on the workbench. To reflect resource modifications in the workbench, from the context menu, select [Refresh], and so on.

.....

Information

.....

EAR files are not created directly using the buildEAR.xml ANT script file for creating EAR files. The EAR file is actually created by the buildEAR2.xml ANT script file. The buildEAR.xml ANT script file updates buildEAR2.xml to match the project structure, and invokes buildEAR2.xml. If there is no modification to the project structure, a buildEAR.xml need not be invoked. EAR files can be created by simply executing buildEAR2.xml.

.....

IJServer Launch Configuration

The IJServer launch configuration provided in the V10 Compatible Workbench is no longer provided.

The workbench in this version provides the "Interstage Application Server" launch configuration. However, if executing a Web application as a client application, in the Servers view, by selecting the deploying Web application project, then from the context menu, selecting [Web browser] the Web browser is started, and the Web application can be executed.

B.2 Notes on Migrating Resources of Up to V9

Most of the issues migrating resources up to V9 are the same as those for V10.

Refer to the following notes, being aware that points provided for the V10 Compatible Workbench are for V9 workbenches and points provided for the V10 standard workbench being are for the V9.2 Java EE workbench:

- [B.1 Notes on Migrating Resources of Up to V10](#)

B.3 Notes on Migrating Resources of Up to V8

This section provides notes on migrating resources of up to V8. For other notes on migrating resources of up to V8, also refer to the following:

- [B.2 Notes on Migrating Resources of Up to V9](#)

B.3.1 Notes on Project Migration

The sections below include notes on using a workspace of up to V8 in workbench of current version, and notes on importing and using a project of up to V8.

JAR package build tool settings

JAR package build tools for up to V8 are no longer provided. To create EJB JAR files, use the EJB JAR Export wizard.

Using JAR package build tools, resources to be contained in JAR could be specified one at a time. However, in the EJB JAR Export wizard, this specification is not inherited. If resources have been specified to be contained in a JAR file using a JAR package build tool, modify the deployment of resources as shown below.

- Migrate resources that you do not want to include in the JAR file to a location other than the source folder. Additionally, migrate resources that you want to include in the JAR file to the source folder. The EJB JAR Export wizard archives resources in the source folder to a JAR file.
- If you want to use your own manifest file, save it as (source-folder)/META-INF/MANIFEST.MF.

IDL compiler build tool

The IDL compiler build tool of up to V8 is not provided in current version for Java projects. Even if a project of up to V8 uses the IDL compiler build tool, IDL compilation is not performed during build in current version. If you want to perform IDL compilation, then add an IDL compiler (idlc.exe) as an external build tool to the builders of the project.

B.4 Notes on Migrating Resources of Up to V7

This section provides specific notes on migrating resources of up to V7. For other notes on migrating resources of up to V7, also refer to the following:

- [B.3 Notes on Migrating Resources of Up to V8](#)

B.4.1 Notes on Workspace Migration

For notes on using a workspace of up to V7 in workbench of current version, refer to this section and "[B.4.2 Notes on Project Migration](#)".

Updating a workspace

If a workspace of up to V7 is specified in the Startup dialog box of workbench, then a message box asking whether to update the workspace appears before the workspace is opened. Select [Yes] to update the internal setting information for the workspace, and the workspace will then be ready for use in workbench of current version. If you do not want to update the workspace, select [No] - Workbench will then exit without updating the workspace.

Perspective settings

When you first open a workspace of up to V7 with workbench of current version, the perspective is not the one last used in the workspace, but the default one. Open your favorite perspective as necessary. Additionally, if the perspective settings have been customized, then all of them are initialized. Create custom settings as necessary.

Enabling automatic build

The default setting of option 'Build automatically' has been changed to ON. Therefore, incremental build is performed when resources are saved. However, the setting is maintained without change when you open a workspace of up to V7 (which the default setting is OFF),

with the workbench of current version. To set automatic build to ON, select [Window] > [Preferences] from the menu bar, select [General] > [Workspace] in the [Preferences] dialog box, then check [Build automatically].

Editor settings

Editor settings, such as fonts, colors, margins, line number visibility, and highlighting, in a version of up to V7 are not maintained by workbench of current version. Make new settings as necessary.

Java Applet launch configuration

The Java Applet launch configuration created in a version up to V7 is not maintained by workbench of current version. Open the workspace, and re-create the Java Applet launch configuration. In the new Java Applet launch configuration, specify the applet display size in [Width] and [Height] on the [Parameters] tab instead of specifying it in an HTML file.

Histories used for JDBC driver names and connection destination URLs

When a database is accessed with a JDBC driver in the Enterprise Bean wizards, the JDBC driver name and connection destination URL are retained in histories, which enables the same values to be re-used the next time the wizards are used. However, the history information retained in a version up to V7 is not maintained by workbench of current version. Enter the JDBC driver name and connection destination URL as new information when using these wizards.

B.4.2 Notes on Project Migration

Refer to the following for notes on using a workspace of up to V7 in workbench of current version, and for notes on importing and using a project of up to V7.

JUnit classpath setting

The method for referring the JUnit JAR file has been changed. If "ECLIPSE_HOME/plugins/org.junit_3.7.0/junit.jar" has been set as a classpath in a project of up to V7, it should be changed to a new classpath. Delete the traditional junit.jar classpath settings, and instead set the JUnit library in the classpath. The JUnit library is a library installed for referencing JUnit JAR files.

By using "[B.6 Automatic Update of the Workspace and Projects](#)", the JUnit classpath can be automatically modified.

Creating an EAR file (earbuild.xml)

If a project of up to V7 uses the EAR file creation function using earbuild.xml, then an EAR file is not created in workbench of current version.

To automatically update the EAR file creation function, follow the procedure described in "[B.6 Automatic Update of the Workspace and Projects](#)".

Java compiler build tool settings

If you imported the project of up to V7, then Java compiler build tool settings are not maintained. This is also true if you are using a workspace of up to V7. If the settings have been changed from their default values, then make new settings as necessary.

SOAP application of up to V7

In the workbench, Web service functions that comply with JAX-WS are standard Web service execution environments. As for applications that use the RPC system of the SOAP service of up to V7, their implementation can be used to the new Java EE Web services, but some of the available types are different. To use the implementation, follow the procedure below (from the viewpoint of interconnection capability, however, it is not recommended to use a WSDL file of up to V7 in current version):

1. Migrate the implementation.
Migrate the existing implementation while considering changes in data type, differences in the execution environment, and so on.
2. Expose an implementation class as a Web service.
Add an @WebService annotation to an implementation class to expose it as a Web service.
For details on developing a Web service, refer to "[Chapter 5 Developing Web Service Applications](#)".

Using a Web application project

Note the following about using a Web application project:

- If a JAR file in the ContextRoot/WEB-INF/lib folder has been added to the project build path, then remove it from the build path. Even if the JAR file is deleted, build can be performed successfully. However, note the following about using the Java editor:
 - A class defined in the JAR file is highlighted as a problem.
 - Classes defined in the JAR file are not displayed in candidates of the content assist function.
- Using Interstage Application Server V6.0 or earlier servlet containers, servlets could be invoked even if servlet mapping was not defined in web.xml. If using servlets, add the servlet mapping definitions to web.xml. For details, refer to "Interstage Application Server J2EE User's Guide".

Tomcat launch configuration

The Tomcat launch configuration cannot be used in default setting. If debugging a Web application, debugging using the Servers view is recommended instead.

B.5 Version-Independent Notes

This section provides notes independent from any particular version.

B.5.1 Notes on Java Migration

This section provides notes on Java migration.

Java versions

Project Java versions are specified below.

- JRE system library in the Java build path
- Project Facets
- Compiler conformance level of the Java compiler

If a project was updated automatically with the standard workbench, basically the settings are as shown below. If a different version of Java is being used, it must be modified individually.

Java version before migration	Java version after migration
6	6
5 or earlier	6



With the standard workbench, the above conversion may not be performed, depending on the migration source workspace or project settings. After migrating, check the Java version so that there are no conflicts with the project facet settings.

For example: When the [Java Build Path] > [JRE System Library] setting is [Workspace default JRE], [JRE System Library] conforms to the migration destination workspace setting.

Predefined Libraries

The predefined libraries that can be used are shown below.

Predefined library name	Remarks
EAR Library	
JBK Library	Former compatible libraries are not supported.

Predefined library name	Remarks
JRE System Library	
JSF Library	
JUnit	
Web Application Library	
Server Runtime	J2EE libraries are converted to server runtimes when projects are automatically updated. The old versions of server runtime and target runtime are changed to the latest versions.
Plug-in Dependencies	
User Library	
Connectivity Driver Definition	
Fujitsu XML Library	Not newly added because this is not a recommended function.

Predefined libraries other than the above provided by older versions are deleted at the time of automatic update of projects because these functions are not provided by the latest version. If predefined libraries for which functions are not provided by the latest version are used, a compile error occurs. Therefore, the implementation must be changed to use an alternative method.

Archive files

In principle, the workbench can perform debugging even if there are no archive files. Therefore, the default is that archive files are not automatically created during a build.

If archive files need to be created for distribution to an operating environment, refer to "[6.2.7 Distributing to the Operating Environment](#)".



Point

.....

An Ant script for creating archive files is created by performing an automatic update of a project.

By registering this Ant script as an Ant build from [Builders] in the project properties, archive files can be created automatically during the build.

.....

B.5.2 Notes on J2EE Application Migration

This section provides notes on J2EE application migration.

Developing J2EE applications that operate under older versions of Interstage Application Server

If a J2EE application project has been migrated, the server runtime becomes "Interstage Application Server V11.1 IJServer (J2EE)" by default. For J2EE application projects that operate under older versions of Interstage Application Server, the server runtime must be changed at [Targeted Runtimes] under the project properties.



Note

.....

When developing a J2EE application that operates under an older version of Interstage Application Server, do not install the application server during installation. Instead, the client package for the version of Interstage Application Server to be used for the development must be installed, and a server in a remote environment must be used to perform debugging.

.....

Changing the execution environment from IJServer(J2EE) to IJServer Cluster

If a J2EE application that was operating under an older version of Interstage Application Server is run in an IJServer Cluster, the server runtime must be changed at [Targeted Runtimes] under the project properties.

Note

The following notes apply to J2EE applications:

- Enterprise Bean (Container-managed Persistence)

The CMP extension information files are for operating under IJServer(J2EE). To operate in an IJServer Cluster, consult the Interstage Application Server manual and change the files to those for IJServer (cluster).

- Web service

Web services (JAX-RPC) operating under IJServer(J2EE) cannot be executed in an IJServer Cluster. Refer to the Interstage Application Server manual for details.

Migration from a J2EE application to a Java EE application

To migrate a J2EE application to a Java EE application, refer to the modified points of each application and update the application.

- Web application: "[2.1.1.1 Modifications since J2EE1.4](#)"
- Enterprise JavaBeans (EJB): "[3.1.1.1 Modifications since J2EE1.4](#)"
- Java Persistence API: "[4.1.1.1 Modifications since EJB2.1](#)"
- Web service: "[5.1.1.1 Modifications since J2EE1.4](#)"

Migration from EJB2.0 to EJB2.1

To migrate to EJB2.1 resources that obey EJB2.0 specification, follow the procedures below.

Note

If migrating to EJB2.1, EJB Applications on the Interstage Application Server cannot be accessed from the remote environment.

1) Create a new project

Create a new EJB project using the EJB project wizard - select [2.1] in [EJB Module Version] in the wizard.

2) Import old resources

Use the Import wizard to import the old resources.

Select [File] > [Import] from the menu bar to start the Import wizard. Select [General] > [File System] in [Select] to import the necessary resources (source file, deployment descriptors and so on). Store the deployment descriptors to the META-INF folder in the source folder. A deployment descriptor is generated when an EJB project is created. Overwrite the deployment descriptor of the existing resources at import time.

Point

Remove the following folders from the import target:

- .externalToolBuilders
- .settings
- bin
- src
- distribute

3) Update the deployment descriptor

Modify the deployment descriptor from EJB2.0 to EJB2.1 format. The deployment descriptor must be opened in the XML Editor and modified directly.

- Modify the deployment descriptor Schema definition from DTD to XML Schema.

Define the XML Schema as shown below:

- EJB2.0

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE ejb-jar PUBLIC "-//Sun Microsystems, Inc.//DTD Enterprise JavaBeans 2.0//EN" "http://
java.sun.com/dtd/ejb-jar_2_0.dtd">
<ejb-jar>
```

- EJB2.1

```
<?xml version="1.0" encoding="UTF-8"?>
<ejb-jar version="2.1" xmlns="http://java.sun.com/xml/ns/j2ee"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://java.sun.com/xml/ns/j2ee
http://java.sun.com/xml/ns/j2ee/ejb-jar_2_1.xsd">
</ejb-jar>
```

- Modify tag configuration.

Correct tag configurations and tags for values that have changed from EJB2.0 to EJB2.1.

- <small-icon>, <large-icon> tags

Modify icon specification tags <small-icon> and <large-icon> so that they are enclosed by <icon> tag:

EJB2.0

```
<small-icon>icon/small.gif</small-icon>
<large-icon>icon/large.gif</large-icon>
```

EJB2.1

```
<icon>
  <small-icon>icon/small.gif</small-icon>
  <large-icon>icon/large.gif</large-icon>
</icon>
```

- <reentrant> tags

Modify the values of the <reentrant> tag to lowercase:

EJB2.0

```
<prim-key-class>sample.EventPrimaryKey</prim-key-class>
<reentrant>False</reentrant>
<cmp-version>2.x</cmp-version>
```

EJB2.1

```
<prim-key-class>sample.EventPrimaryKey</prim-key-class>
<reentrant>false</reentrant>
<cmp-version>2.x</cmp-version>
```

- <security-role-ref>, <security-identity> tags

Modify the position of the <security-role-ref> and <security-identity> tags from between the EJB reference tags (<ejb-ref>, <ejb-local-ref>) and resource reference tags (<resource-ref>, <resource-env-ref>) to after the resource reference tags:

EJB2.0

```
<ejb-local-ref>
  Omitted
```

```

</ejb-local-ref>
<security-role-ref>
  <role-name>Security1</role-name>
  <role-link>Security1</role-link>
</security-role-ref>
<security-identity>
  <run-as>
    <role-name>Security1</role-name>
  </run-as>
</security-identity>
<resource-ref>
  Omitted
</resource-ref>

```

EJB2.1

```

<ejb-local-ref>
  Omitted
</ejb-local-ref>
<resource-ref>
  Omitted
</resource-ref>
<security-role-ref>
  <role-name>Security1</role-name>
  <role-link>Security1</role-link>
</security-role-ref>
<security-identity>
  <run-as>
    <role-name>Security1</role-name>
  </run-as>
</security-identity>

```

- Modify Message-driven Bean-related tags

The following Message-driven Bean tags have been deleted. Redefine the property name and value in the <activation-config> tag.

- <message-selector>
- <acknowledge-mode>
- <message-driven-destination>
- <destination-type>
- <subscription-durability>

The method for specifying the property is described below. Note that the <message-driven-destination> tag does not have a value.

EJB2.0 Tag	EJB2.1 Tag	
	Property Name	Value
message-selector	messageSelector	Any character string
acknowledge-mode	acknowledgeMode	Specify one of the following. - AUTO_ACKNOWLEDGE - DUPS_OK_ACKNOWLEDGE
destination-type	destinationType	Specify one of the following. - javax.jms.Queue - javax.jms.Topic
subscription-durability	subscriptionDurability	Specify one of the following.

EJB2.0 Tag	EJB2.1 Tag	
	Property Name	Value
		- Durable
		- NonDurable

The following is an example of settings.

```

<message-driven>
  Omitted
  <activation-config>
    <activation-config-property>
      <activation-config-property-name> messageSelector </activation-config-property-name>
      <activation-config-property-value> JMSType = 'car' AND color = 'blue'</activation-
config-property-value>
    </activation-config-property>
    <activation-config-property>
      <activation-config-property-name> destinationType </activation-config-property-name>
      <activation-config-property-value> javax.jms.Topic </activation-config-property-value>
    </activation-config-property>
    <activation-config-property>
      <activation-config-property-name> subscriptionDurability </activation-config-property-
name>
      <activation-config-property-value> NonDurable </activation-config-property-value>
    </activation-config-property>
  </activation-config>
</message-driven>

```

4) Java Source Correction

The source file does not need to be modified when migrating from EJB2.0 to EJB2.1.

Migration from EJB1.0 or EJB1.1 to EJB2.0

To migrate to EJB2.0 resources that obey EJB1.0 or EJB1.1 specification, follow the procedure below:

1) Create a project, then import the resources

If the resources were not created using workbench, then follow the procedure below:

- a. Create an EJB project.
Specify [2.0] in [EJB Module Version].
- b. Import the resources using the Import wizard.

Select [File] > [Import] from the menu bar to select the import wizard, and import the necessary resources. If migrating from EJB1.0 or EJB1.1 to EJB2.0, the deployment descriptor is not imported.

2) Update deployment descriptors

The deployment descriptor generated when the project was created must be opened in an XML editor, and the XML must be directly updated. Set the required information, and update the deployment descriptor.

3) Modify the Java sources.

The source files do not need to be modified when migrating from EJB1.1 to EJB2.0. For migration from EJB1.0, modify them properly for the following processing:

- Lookup processing
- Business method invocation processing
- UserTransaction processing
- Processing of using Enterprise Bean Environment

Migration from J2EE Application Client1.3 to J2EE Application Client1.4

To migrate resources that obey J2EE Application Client1.3 to J2EE Application Client1.4, follow the procedure below:

1) Create a New Project

Create a new Application Client Project using the Application Client project wizard. In [Application Client module version], specify [1.4].

2) Import old resources using the Import wizard

Select [File] > [Import] from the menu bar to start the Import wizard. In [Select], select [General] > [File System] to import the necessary resources (source file, deployment descriptor and so on). Store the deployment descriptors to the META-INF folder in the source folder. A deployment descriptor is generated when an application client project is created. Overwrite the deployment descriptor of the existing resources at import time.

3) Update deployment descriptors

Modify the deployment descriptor from J2EE Application Client1.3 to J2EE Application Client1.4 format. The deployment descriptor must be opened in the XML Editor and modified directly.

- Modify the deployment descriptor Schema definition from DTD to XML Schema.

Define the XML Schema as shown below.

- J2EE Application Client1.3

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE application-client PUBLIC "-//Sun Microsystems, Inc.//DTD J2EE Application Client
1.3//EN" "http://java.sun.com/dtd/application-client_1_3.dtd">
<application-client>
```

- J2EE Application Client1.4

```
<?xml version="1.0" encoding="UTF-8"?>
<application-client id="Application-client_ID" version="1.4"
xmlns=http://java.sun.com/xml/ns/j2ee
xmlns:xsi=http://www.w3.org/2001/XMLSchema-instance
xsi:schemaLocation="http://java.sun.com/xml/ns/j2ee
http://java.sun.com/xml/ns/j2ee/application-client_1_4.xsd">
</application-client>
```

- Modify tag configuration

Correct tag configurations and tags for values that have changed from J2EE Application Client1.3 to J2EE Application Client1.4.

- <small-icon>, <large-icon> tags

Modify icon specification tags <small-icon> and <large-icon> so that they are enclosed by <icon>tag:

J2EE Application Client1.3

```
<small-icon>icon/small.gif</small-icon>
<large-icon>icon/large.gif</large-icon>
```

J2EE Application Client1.4

```
<icon>
  <small-icon>icon/small.gif</small-icon>
  <large-icon>icon/large.gif</large-icon>
</icon>
```

When JSF is used in Web applications

Add libraries using the following procedure after migrating resources to use JSF in Web applications:

1. Select the project. From the context menu, click [Properties] > [Java Build Path] > [Libraries] tab > [Add Library].
2. Select [JSF Libraries] and click [Next].

3. Click [Add].
4. Enter "JSF" for the library name, click [Add], and select the following jar file.
 <installation folder>\APS\F3FMisjee\lib\jsf-impl.jar
5. Select [Is JSF Implementation] and click [Finish].
6. Select the added JSF library and click [Finish].

B.5.3 Cautions when Migrating to JDK 6/7

Refer to the following for information about incompatibility and issues when migrating to JDK 6 and JDK 7:

- Migrating from JDK 1.3 to JDK 5.0
 <http://www.oracle.com/technetwork/java/javase/community/jm-white-paper-r6a-149981.pdf>
- Migrating from JDK 5.0 to JDK 6
 <http://www.oracle.com/technetwork/java/javase/adoptionguide-137484.html>
- Migrating from JDK 6 to JDK 7
 <http://docs.oracle.com/javase/7/docs/webnotes/adoptionGuide/index.html>

Note: The above URLs are managed and maintained by Oracle. They are subject to change without notice.

B.5.4 Notes on Migrating ComponentDesigner Resources

This section provides notes specific to the migration of ComponentDesigner resources.

Projects that can be migrated

You can use the import functions provided by workbench to migrate ComponentDesigner projects created with Interstage Apworks V5.0 or an earlier version to workbench.

The ComponentDesigner projects that can be migrated are listed below.

Projects that can be migrated
Pure Java Application (*1)
JavaBeans
Applet
Enterprise JavaBeans *2)
Web Application *3)
Web Application(Apcoordinator)

*1) Any applications that use the SOAP Server Applications(Apcoordinator) cannot be migrated.

*2) Any Enterprise JavaBeans compliant with EJB 1.0 specification cannot be migrated.

*3) Extension Tag Library applications cannot be migrated.



Note

- The following types of projects and applications are not supported by workbench and cannot be imported:
 - MIDP Application
 - Enterprise JavaBeans project compliant with EJB 1.0 specification
 - SOAP Server Application(Apcoordinator)
 - Extension Tag Library applications
 - COBOL Application

- CORBA Server
- CORBA Client
- All the classpaths set in ComponentDesigner are added to the build path of the imported project. If any library in classpath is missing, an error occurs during build. Modify the build path appropriately after importing a project.

Batched import of multiple projects

You can import two or more projects in one operation by using ComponentDesigner subprojects. To import them in one operation, follow the procedure below.

1. Create a work project for this import session in ComponentDesigner; and the project will be used as the parent project of other projects.
2. In Explorer, open the project folder of one of the projects to be imported. Then, drag the project file project-name.prj and drop it on the project file (work.prj where work is the import work project name) of the import work project displayed in the Project display area in ComponentDesigner.
Click [OK] in the [Add File] dialog box that appears.
The imported project is added as a subproject of the import work project.
3. Repeat step 2 for each of the ComponentDesigner projects to be imported.
4. Exit ComponentDesigner.
5. Start workbench, and select [File] > [Import] from the menu bar.
Select [Existing ComponentDesigner's Project] in [Select an import source].
6. In the [Project to be Imported], specify the project file of the import work project created in step 1.
7. Check [Import subprojects], and click [Finish]. The import work project and all its subprojects are imported.
8. After import is completed, delete the imported import work project from workbench.

Confirming the build path

For an imported project, the predefined libraries generally used for its project type are added to the build path. To use a predefined library that is not added as a standard library, add the predefined library to the build path.

All the classpaths set in ComponentDesigner are added to the build path of the imported project. If a JAR file that is resolved in a predefined library is directly specified in the build path, it need not be specified in the build path. Remove any such JAR files from the build path to prevent related problems. Another possible problem is that a JAR file cannot be found because its installation folder is different. Remove from the build path those JAR files that are not referenced during Java compilation or that can be resolved in a predefined library. Confirm that each library that you add yourself is at the proper location.

Confirming file locations

The location of each imported file is determined based on the project type, file type, and file settings at the time of import. In the standard layout of imported files, some files may be placed at locations not appropriate for development with Interstage Studio. After import is completed, confirm that each file is at the proper location; and if it is not, move it to the proper location.

The import function determines file locations based on the following rules:

- Java source files (*.java) are placed in the source folder.
- The files that are checked as Link to resources on the [Build] tab in the project properties dialog box are placed in the source folder with the original folder structure maintained as is.
- The files that are not checked as Link to resources on the [Build] tab in the project properties dialog box are placed in the project folder with the original folder structure maintained as is.
- SPT files (*.spt) are placed in the project folder.
- If the project to be imported is a Web Application project or Web Application(Apcoordinator) project, its HTML files and JSP files are placed in the ContextRoot folder with the original folder structure maintained as is. However, some JSP files may be placed in the

ContextRoot/pages folder.

The HTML files for other project types are placed in the project folder with the original folder structure maintained as is.

- If the project to be imported is a Web Application project or Web Application(Apcoordinator) project, the files contained in the WEB-INF folder in the import source project are placed in the ContextRoot/WEB-INF folder.
- If the project to be imported is an Enterprise JavaBeans project, the ejb-jar.xml file is placed in the META-INF folder in the source folder.
- The application-client.xml file is placed in the META-INF folder in the source folder.
- If the ComponentDesigner build option [Treat the Bean serialize files (*.ser) as resources] is checked, all the .ser files at the location immediately under the project folder in the import source project are placed in the source folder.

For a Web Application project or Web Application(Apcoordinator) project, only the HTML files, the JSP files, and the files contained in the WEB-INF folder are placed in the ContextRoot folder. Other files may be placed in the source folder or project folder, depending on whether the file is "Link to resources" or not, with their original folder structures maintained as is.

Confirm the locations of the resource files referenced from within HTML or JSP code, and if necessary, move the files so that their location is appropriate.

Generally, the files defined as "Link to resources" are placed in the source folder. This means that the source folder may be the location of text files and other files that are not to be built. Move files that are not to be built from the source folder to another appropriate location.

B.5.5 Notes on Applet or Java Form Migration

This section provides notes on Applet or Java Form migration.

Migration of Applet or Java Form that uses the Form Extension Function

An Applet or Java Form that uses the Form Extension Function cannot be migrated as is. This is because the Form Extension Function does not support JDK/JRE 6 or JDK/JRE 7.

To migrate an Applet or Java Form that uses the Form Extension Function, please change it to an Applet or Java Form that uses AWT or Swing. An Applet or Java Form that uses the Form Extension Function will be judged by whether the following classes are base classes. When migrating an Applet or Java Form, you will need to change the base classes to the following classes.

Base class before migration	Base class after migration
com.fujitsu.apworks.compod.ui.CDApplet	java.applet.Applet
com.fujitsu.apworks.compod.ui.CDJAplet	javax.swing.JApplet
com.fujitsu.apworks.compod.ui.CDFrame	java.awt.Frame
com.fujitsu.apworks.compod.ui.CDJFrame	javax.swing.JFrame
com.fujitsu.apworks.compod.ui.CDDialog	java.awt.Dialog
com.fujitsu.apworks.compod.ui.CDJDialog	javax.swing.JDialog
com.fujitsu.apworks.compod.ui.CDPanel	java.awt.Panel
com.fujitsu.apworks.compod.ui.CDJPanel	javax.swing.JPanel



Note

The following functions provided by the Form Extension Function cannot be migrated. For processing that relies on such functions, replace it with other kinds of processing or delete it.

- Focus traversal order definition
- Status bar

Note

An Applet or Java Form that is created using the Derived Form Function cannot be migrated. Please terminate the inheritance relationship by the merge with the source file of the base class.

You can use the following procedure to check whether the migration object file uses the Derived Form Function. For uncertainties, please consult our technical personnel.

1. Check the following comment part in the source file.

```
//@@Form Design Information start  
//@@Inherits From class name 8-digit values
```

2. The aforesaid class name is not included in the table "Base class before migration".
3. Turn the 8-digit values to corresponding hexadecimal values, and check the third byte.

If the third byte is 01, then the class is Derived Form.

Example 1) //@@Inherits From class name 16843267
16843267 (decimal) -> 0x01 01 02 03 (hexadecimal)
-> The class is Derived Form

Example 2) //@@Inherits From class name 16843267
16973315 (decimal) -> 0x01 02 FE 03 (hexadecimal)
-> The class is not Derived Form

Migration example: Change the Java Form with com.fujitsu.apworks.compod.ui.CDFrame class as the base class to the Java Form that uses AWT and has java.awt.Frame class as the base class. Follow the procedure below:

1. Creating a new Java Form

Create a new Java Form with java.awt.Frame class as the base class.

Create the Java Form after migration in the same project as the Java Form before migration is involved in. From the menu bar of the workbench, select [File] > [New] > [Other]. In the [New] wizard, select [Java] > [GUI] > [Form]. Enter information in the [Java Form Information] page. In the [New Java Form] dialog box, select [Frame] in the [Java] tab. In the [Frame] dialog box, enter a different name from the Java Form before migration in the [Java Form name] field and enter "java.awt.Frame" in the [Base class] combo box. Click the [Create] button, and a new Java Form will be created.

2. Bean copying

Open the Java Form before migration and the one after migration using the Graphical Editor. Copy all the Beans in the Java Form before migration to the Java Form after migration.

3. Various definitions

As the following definitions cannot be copied, related items in the Java Form after migration must be defined the same as those in the Java Form before migration. However, the attention processing method will be copied afterwards, so there is no need to define the attention processing definition.

- Menu definition
- Exclusive Selection Group definition
- Attention definition

4. Constructor copying

In the Java Form before migration, there exists a constructor with the parameter of com.fujitsu.apworks.compod.ui.CDApplicationContext class. This class is part of the Form Extension Function and cannot be copied to the Java Form after migration. Therefore, only the constructor processing is copied from the Java Form before migration to the Java Form after migration.

The constructor processing is copied using the Java Editor.

5. Source copying

Copy sources other than the constructor in the Java Form before migration to the Java Form after migration. However, remember not to copy the sources that prohibit changes. Sources that prohibit changes refer to the sources that begin with "// Graphical Editor

Form" and "//@@Form Design Information start" and end with "//@@Form Design Information end".
The sources are copied using the Java Editor.

6. Method name change

As the Java Form after migration is a class inherited form, you need to change the method names of event processing and attention processing.

Check whether there are methods in the following formats in the copied source. These are the methods of event processing and attention processing.

- public void "Bean name"_"event listener class name"_"event listener method name"("event class name")
- public boolean processAttention_"attention name"()

Add \$ to the ends of these methods according to the following formats.

- public void "Bean name"_"event listener class name"_"event listener method name"\$("event class name")
- public boolean processAttention_"attention name"\$()

After the Java Form is saved, the codes for association between bean and event processing method and those for association between attention and attention processing method will be created.

7. Java Form name change

Make the name of the Java Form after migration the same as that of the Java Form before migration.

Delete the Java Form before migration, or copy the Java Form after migration to other projects. Open the Java Form after migration using the Graphical Editor. From the menu bar of the workbench, select [File] > [Save As], and save the Java Form to a different name. Delete the original Java Form prior to the saving.

B.6 Automatic Update of the Workspace and Projects

Among the notes on migration of workspace and projects, the following can be automatically updated using workbench commands:

- [JUnit classpath setting](#)
- [Creating an EAR file \(earbuild.xml\)](#)

To convert these settings automatically, follow the procedure below.

1. To use workspace or project of previous version in the workbench of current version, follow the procedure described in "[Migration procedure](#)".
2. Select [Project] > [Update Workspace/Projects of Previous Version] from the menu bar.
3. The [Update Workspace/Projects of Previous Version] dialog box is displayed. On the left side of the dialog box, workspaces and projects requiring settings updates are displayed. Select the items to be updated, and then click [OK].
4. Workspace and project settings are updated. In the Console view, the status of the update is displayed. Make sure that the update has been performed successfully.

Note

- This command does not update closed projects, so open the projects to be updated before using it.
- If an automatic build is performed after the command, then the project may be displayed with an error mark in the Package Explorer view, even though the classpath is successfully set. In this case, clear the error mark by cleaning the project first and performing build again.
- Projects that meet all the following requirements may also be displayed with an error showing "Missing builder(com.fujitsu.apworks.apdesinger.java.packagerbuilder)" in the builders under the projects' properties, even though the projects have been automatically updated. In this case, please perform an automatic update once again.
 - Resources are developed in the V9 workbench or V10-compatible workbench.
 - Builders of the project are set as [JAR Packager Builder].

- Build path of the project is set as [Interstage J2EE] library.

Besides, problems such as builder mistakes will not occur, even though there is an error showing "Missing builder(com.fujitsu.apworks.apdesinger.java.packagerbuilder)".

Appendix C Procedure for Development Using JDK 6

This section describes the procedure for using JDK 6 in development on a Java EE 6 workbench.

- [Starting a Java EE 6 workbench that uses JDK 6](#)
- [Modifying existing workspace settings](#)
- [Initializing a Java EE 6 runtime environment that uses JDK 6](#)

Starting a Java EE 6 workbench that uses JDK 6

To start the Java EE 6 workbench using JDK 6, follow the procedures below:

1. Open the Command Prompt, and move to the installation folder of the Java EE 6 development function.

```
[installation folder of the Java EE 6 development function]
```

```
<product installation folder>\IDE\1101_WB36\eclipse
```

2. Execute isstudio.exe with the -vm option added, as shown below:

```
> isstudio.exe -vm <JDK 6 installation folder>\jre\bin
```

3. In the [Select a workspace] window, select a new workspace folder.

If an existing workspace is selected, follow the procedures in "Modifying existing workspace settings" below:

Modifying existing workspace settings

If an existing workspace is selected when the Java EE 6 workbench is started, follow the procedures below to modify settings to use JDK 6:

1. Add JDK 6 to [Installed JREs].
 - a. From the workbench menu, select [Window] > [Preferences].
 - b. In the [Preferences] dialog box on the left pane, select [Java] > [Installed JREs].
 - c. In the right [Installed JREs] window, click [Add].
 - d. In the [JRE Type] window, select "Standard VM", then click [Next].
 - e. In [JRE home] of the [JRE Definition] window, specify the JDK 6 installation folder.
"JDK 6" is displayed in [JRE name], and a list of JAR files for the JDK 6 library specified in [JRE system libraries] is displayed.
Click [Finish] to close the window.
 - f. "JDK6" is added to the list in [Installed JREs].
Select the "JDK 6" checkbox to make JDK 6 the default JDK. Click [OK] to close the dialog box.
2. Modify JDK used by an existing project to JDK 6.
 - a. From Project Explorer, select the project to be modified, and from the popup menu select [Properties].
 - b. In the left pane of the [Properties] dialog box, select [Java Build Path].
The [Java Build Path] page is displayed on the right. Select the [Libraries] tab.
 - c. From the list, select "JRE System Library[JavaSE-1.7]", then click [Edit].
 - d. In the [JRE System Library] window change the "Execution environment" to "JavaSE-1.6(JDK6)" and click [Finish].
 - e. Click [OK] to close the [Properties] dialog box.

Initializing a Java EE 6 runtime environment that uses JDK 6

The Java EE 6 runtime environment must be initialized for use with JDK 6 so the applications developed using JDK 6 can be deployed in the Java EE 6 runtime environment and tested. Any Interstage Java EE 6 DAS services created, or deployed applications, are deleted after initialization. If necessary, recreate Interstage Java EE 6 DAS services and redeploy applications after initialization.

Follow the steps below to initialize:

Table C.1 Initialization

No.	Steps	References
1	Backup Application Server resources. Note: Perform the tasks as required if J2EE and Java EE 5 functions are used with the Interstage Application Server bundled with this product.	"Interstage Application Server Operator's Guide" > "Resource Backup/Export"
2	Uninstall the Application Server.	"Interstage Studio Installation Guide" > "Installation" > "Overwrite installation"
3	After backing up any necessary files in the APS folder, delete the APS folder. (For example: If the installation folder was "C:\Interstage", delete the C:\Interstage\APS folder)	-
4	Install the Application Server. Select JDK 6 in the [Select the JDK to be used in the Java EE 6] window.	"Interstage Studio Installation Guide" > "Installation" > "Overwrite installation"
5	Restore the resources that were backed up. Note: Perform the tasks as required if J2EE and Java EE 5 functions are used with the Interstage Application Server bundled with this product.	"Interstage Application Server Operator's Guide" > "Resource Restore/Import"

 **Point**

If developing using JDK 7 after developing using JDK 6, follow the procedure below to revert the settings:

- The Java EE 6 workbench is usually started on the application page (in Windows 8 and Windows Server 2012) or from the start menu (in other Windows systems).
- If the settings of an existing workspace have been modified for JDK 6 use, revert the settings to those for JDK 7.
 - In the [Installed JREs] list, select the "JDK 6" checkbox to make JDK 7 the default JDK.
 - In the existing project, in the [JRE System Library] window, modify "Alternate JRE" to "JDK7".
 - In the [Java Compiler] window for the existing project, change the [Compiler compliance level] to [1.7].
- Initialize the Java EE 6 runtime environment for use with JDK 7. The procedure is the same as described in "Initialization". When installing the Application Server, select JDK7 in the [Select the JDK to be used in the Java EE 6] window.

Appendix D Migrating J2EE 1.4 Applications

This appendix explains how to develop J2EE 1.4 applications.



It is recommended to develop applications using JavaEE5 in Interstage Studio V11 rather than migrating applications developed with J2EE 1.4.

D.1 Developing Web Applications

This section explains J2EE 1.4 Web application development, focusing on the ways this differs from the Java EE Web application development method.

D.1.1 Preparing the Environment for Creating Web Applications

Refer to "[2.3.1 Preparing the Environment for Creating the Web Application](#)" for information on preparing to create a Web application.

Creating a project

Select [Interstage Application Server V11.1 IJServer (J2EE)] as the target runtime in the Dynamic Web Project wizard when developing J2EE1.4 Web applications.



The creation of new projects with dynamic Web module versions 2.2 and 2.3 is not supported.

D.1.2 Creating a Servlet

For details on creating and editing Servlets, refer to "[2.3.2 Creating a Servlet](#)".

D.1.3 Creating an HTML File

For details on creating and editing HTML files, refer to "[2.3.3 Creating an HTML File](#)".

D.1.4 Creating a JSP File

For details on creating and editing JSP files, refer to "[2.3.4 Creating a JSP File](#)".

D.1.5 Graphically Editing the HTML/JSP Files

Refer to "[2.3.5 Graphically Editing the HTML/JSP Files](#)" for information on graphically editing HTML/JSP files.

D.1.6 Creating the JavaScript

For details on creating and editing JavaScript files, refer to "[2.3.6 Creating the JavaScript](#)".

D.1.7 Creating the CSS

For details on creating and editing CSS files, refer to "[2.3.7 Creating the CSS](#)".

D.1.8 Editing the web.xml

For details on editing web.xml, refer to "[2.3.8 Editing the web.xml](#)".

D.1.9 Verifying the Web Application Behavior

Refer to "[D.4.6 Checking Application Behavior](#)" for information on verifying Web application behavior.

D.1.10 Distributing the Web Application to the Operating Environment

An archive must be created in order to distribute a Web application to an operating environment. Refer to "[6.2.7 Distributing to the Operating Environment](#)" for details.

D.2 Developing Enterprise JavaBeans

This section explains J2EE1.4 EJB development, focusing on the ways this differs from Java EE EJB development.

D.2.1 Preparing the Environment to Create EJBs

Refer to "[3.3.1 Preparing the Environment to create EJBs](#)" for information on preparing to create an EJB.

Creating a project

Select [Interstage Application Server V11.1 IJServer (J2EE)] as the target runtime in the EJB Project wizard when developing J2EE EJB.



Note

.....
The creation of new projects with EJB module version 1.1 is not supported.
.....

D.2.2 Creating an Enterprise Bean

Generating Enterprise Beans

To add or generate EJB2.1-compliant enterprise beans, use the Enterprise Bean wizard.

The Enterprise Bean wizard can create the following enterprise beans.

- Stateless Session Bean
- Stateful Session Bean
- EJB2.x Container-managed persistence Entity Bean
- Bean-managed persistence Entity Bean
- Point-To-Point model Message-driven Bean
- Publish/Subscribe model Message-driven Bean



Note

.....
Create an Enterprise Bean in a single source folder.

Since deployment descriptor files are stored in the source folder, if an Enterprise Bean is created in multiple source folders in one project, multiple deployment descriptors are created. When archiving in JAR format under these circumstances, the existence of files with the same name causes an error.
.....

From the [New] wizard, select [EJB] > [J2EE] > [Enterprise Bean]. After the wizard creates the Enterprise Bean, edit the generated Java source, deployment descriptor, and so on.

In the Enterprise Bean wizard, specify basic information on Enterprise Bean generation and the Enterprise Bean type.

Creating Stateless Session Bean

A session bean is the Enterprise Bean to perform a dialogue with client. Write the process (Business logic), which becomes necessary mainly in business process.

Stateless Session Bean does not retain transaction state and the value of the variable defined in Enterprise Bean over multiple methods. Therefore, use when function in one method is provided. Moreover two or more clients can share the instance of same session bean, this reduces the load of the server.

Specify information in the Enterprise Bean wizard as follows:

1. Specify Basic Information of Enterprise Bean.
Enter package name, class name.
2. Specify Session Bean Option.
Specify the transaction type.
3. Specify ejbCreate Method Definition.
Define the method for initializing Enterprise Bean.
The ejbCreate method for Stateless Session Bean is defined as an initial value. It is necessary to edit an initial value only to perform processing which adds the exception thrown by the ejbCreate method.
4. Specify Business Method Definition.
Add a business method. In Stateless Session Bean, it is necessary to create a method in the unit, which completes processing.

Creating Stateful Session Bean

A session Bean is the Enterprise Bean to perform a dialogue with client. Write the process (Business logic), which becomes necessary mainly in business process.

A stateful session bean corresponds to a client in a one-to-one relationship and can retain a transaction state and a variable value defined in an Enterprise Bean over multiple methods called by the client. Therefore, this type of session bean is used when providing a function that requires multiple methods.

Specify information in the Enterprise Bean wizard as follows:

1. Specify Basic Information of Enterprise Bean.
Enter package name, class name.
2. Specify Session Bean Option.
Specify transaction type and if SessionSynchronization is to be implemented or not.
3. Specify ejbCreate Method Definition.
Define the method for initializing Enterprise Bean.
ejbCreate method without argument is defined as initial value. Add, edit as per the requirement.
4. Specify Business Method Definition.
Add Business method.

Creating Bean-managed persistence Entity Bean

Entity Bean maps persistence data of the database in the Java object, and multiple clients can also access it.

Incase of Bean-managed persistence, the access processing to persistence data such as SQL sentence is described in the source of Entity Bean. Since data access processing is described directly, access control according to database classification can be performed.

In the Enterprise Bean wizard, you can generate a template of access process to database using JDBC.

Specify information in the Enterprise Bean wizard as follows:

1. Specify Basic Information of Enterprise Bean.
Enter package name, class name.
2. Specify Entity Bean Option.
Specify the information related to database access.
 - Reentrant
Specifies whether to enable Entity Bean to be recursively called.
 - Generate database access process
Incase of Bean-managed persistence, specify if the database access process template will be generated or not in the Enterprise Bean source.

- Make the table name in the SQL sentence modifiable
In case of Bean-managed persistence, when database access process template generation is performed, generate a source while using the [TableName] value environment property. Specify in case there will be a change in the schema and the table environment.
 - Optimize the retrieval logic
Entity search process searches two times in the template of database access process of Bean managed persistence based on normal EJB agreement because the timing of acquisition of primary key and acquisition of Persistence field differs at the time of search. Generate source by specifying this option, to acquire the primary key and persistence field information simultaneously to save in the memory.
 - Use record class for getting and setting data
To get/set the persistence field data once, generate the method that has used the record class.
 - Data source name
Specify the data source name used in persistence.
3. Specify Persistence Field Definition.
Defines the persistence field. You can add to the persistence field by referring to the database.
 4. Specify ejbCreate Method Definition.
Define the method for inserting persistence data in the database.
ejbCreate method which has persistence data field or record class in argument is defined as initial value. Add, edit as per the requirement.
 5. Specify Finder Method Definition.
Define the method for searching persistence data.
The ejbFindByPrimaryKey method for performing reference by the Primary Key class is defined as initial value. Add, edit as per the requirement.
 6. Specify Home Method Definition.
This function can be used with EJB2.1. The method is defined for performing the processing independent of the instance of Entity bean. (For example, records count).
 7. Specify Business Method Definition.
Add Business method.

Creating Container-managed persistence Entity Bean

Entity Bean maps the persistence data of the database in Java object, multiple clients can also access it.

In container managed persistence, this defines mapping process and search process with database as customize information, without the access process to persistence data in entity Bean source. Access process to persistence data is performed by container on the basis of customize information. Therefore it exceeds in priority to different database (data type).

CMP extension information file which is the customize information of the Interstage EJB can be generated in the Enterprise Bean wizard.

Specify information in the Enterprise Bean wizard as follows:

1. Specify Basic Information of Enterprise Bean.
Enter package name, class name.
2. Specify Entity Bean Option.
Specify the information related to database access.
 - Reentrant
Specifies whether to enable Entity Bean to be recursively called.
 - Use record class for getting and setting data
To get/set the persistence field data once, generate the method that has used the record class.
 - Data source name
Specify the data source name used in persistence.

Point

The file to which the data source name is written varies depending on whether the application target runtime is a J2EE container or a Java EE container.

- For J2EE: <fujitsu-cmp2x-mapping-definition>/ <datasource-name> in the FJCMP_XXX.xml
- For Java EE: <sun-ejb-jar>/ <enterprise-beans>/ <cmp-resource>/ <jndi-name> in the sun-ejb-jar.xml

3. Specify Persistence Field Definition.
Define the persistence field. You can add to the persistence field referring to the database.
4. Specify ejbCreate Method Definition.
Define the method for inserting persistence data in the database.
5. As an initial value, the ejbCreate method that has the persistence field or a record class in an argument is defined. Add, edit if required.
6. Specify Finder Method Definition.
Define the method for searching the persistence data.
7. As an initial value, according to the primary key class, findByPrimaryKey is defined in the method. Add if required.
8. Specify Home Method Definition.
This function can be used with EJB2.1. Define the method for performing processing independent of the Entity Bean instance (e.g., record count). Add if required.
9. Specify ejbSelect Method Definition.
This function can be used with EJB2.1. Define the method for search processing performed within an Entity Bean method. Add if required.
10. Specify Business Method Definition.
Add Business method.

Point

The final deliverables from the CMP Entity Bean vary depending on whether they are for a J2EE container or a Java EE container, as shown below.

Option	JavaEE Container	J2EE Container
CMP extension information file	sun-cmp-mappings.xml	FJCMP_XXXX.xml
Database definition information	sun-ejb-jar.xml	FJCMP_XXXX.xml
Database table information	XXX.dbschema (If "Use DB Schema" is selected when connecting to the database)	None

Creating Point-To-Point model Message-driven Bean

Message-driven Bean is an Enterprise Bean for carrying out unsynchronized communication process.

In Point-To-Point model Message-driven Bean, just a single receiver processes a message sent from a sender. This type of message-driven bean is used when a single receiver is assigned to a message.

Specify information in the Enterprise Bean wizard as follows:

1. Specify Basic Information of Enterprise Bean.
Enter package name, class name.

- Specify Message-driven Bean Option.
Specify transaction type and message selector as per the requirement.

Creating Publish/Subscribe model Message-driven Bean

Message-driven Bean is an Enterprise Bean for carrying out unsynchronized communication process.

In Publish/Subscribe model Message-driven Bean, Multiple receivers process a message sent from a sender. This type of message-driven bean is used when a single message must be delivered to multiple receivers.

Specify information in the Enterprise Bean wizard as follows:

- Specify Basic Information of Enterprise Bean.
Enter package name, class name.
- Specify Message-driven Bean Option.
Specify transaction type, subscription durability and message selector as per the requirement.

D.2.3 Developing a CMP Entity Bean

Defining CMP2.0 relationships

Use the Define CMP2.0 Relationships wizard to define the relationships of a container-managed persistence Entity Bean having EJB2.1 specifications. Use this wizard to set the relationships of an Entity Bean with other Entity Beans. Therefore, an EJB2.1 Container-managed persistence Entity Bean used for definitions of the relationships under the same project must be created before Define CMP2.0 Relationships wizard can be used.

From [New] wizard, select [EJB] > [J2EE] > [CMP2.0 Relationship] to select the Define CMP2.0 Relationships wizard. Check and enter the settings shown below.

Specifying the Deployment Descriptor

Specify the deployment descriptor file of Entity Beans to define their relationships.

Option	Description
ejb-jar.xml file	Specify the path of ejb-jar.xml.

Specifying CMP2.0 Relationship Information

Specify the relationships between Entity Beans.

Option	Description
Add	Add new relation. Click on [Add], "Add Relation" screen appears.
Delete	Delete the relation selected on the list.
Edit	Edit the relation selected on the list. Click on [Edit], "Edit Relation" screen appears.
Reflect relation information in a Home/Component interface	When this option is checked, the CMR field access information is reflected in the Local/Component interface. By calling the home/component interface generation function, the access method of the CMR field defined with the wizard is reflected in the Local interface.

Adding or Editing a CMP Relationship

Specify the relation between Entity Beans.

For ease of use, the two Enterprise Beans whose relationship is defined are assumed to be Enterprise Bean A and Enterprise Bean B.

Option	Description
EJB relation name	Specify a name for the relation between Entity Beans. The EJB relation name can be omitted.
Multiplicity	Specify the multiplicity of relation of Entity Beans. Multiplicity of relation indicates the correspondence relation between Enterprise Beans making up the relationship. The multiplicity of relation is a one-to-one, one-to-many, or many-to-many relation. For the one-to-many relation, either A or B can be defined as side 1. One:One, One:Many, Many:One, and Many:Many can be specified.
EJB relationship role name	Specify a relationship role name for each Entity Bean making up the relation. The EJB relationship role name can be omitted.
Enterprise Bean name	Specify an Enterprise Bean name for each Entity Bean making up the relationship.
CMR field for accessing Bean A/B	Specify a CMR field name used for accessing the other Enterprise Bean. Either A or B must be specified.
CMR field type	Specify the type of CMR field for accessing the other Enterprise Bean in the relation. If the multiplicity of relation is One:Many, Many:One, or Many:Many and the other Enterprise Bean is Many, java.util.Collection or java.util.Set can be specified.
Delete when Bean A/B is deleted	Specify whether an Enterprise Bean is deleted if the other Enterprise Bean making up the relation is deleted.
Description	Provide a description explaining this relation definition. The description can be omitted.
Mapping A tab	Specify the database information for an Enterprise Bean specified to be associated with Enterprise Bean A. If the multiplicity of relation is One:One, be sure to specify the column of the database used to access the primary key of the other Enterprise Bean. Select the [Mapping A] tab to display the "Mapping A" screen.
Mapping B tab	Specify the database information for an Enterprise Bean specified to be associated with Enterprise Bean B. If the multiplicity of relation is One:One, be sure to specify the column of the database used to access the primary key of the other Enterprise Bean. Select the [Mapping B] tab to display the "Mapping B" screen.
Join Table tab	Specify the information in the join table required when the multiplicity of relation is One:Many, Many:One, or Many:Many. Select the [Join Table] tab to display the "Join Table" screen.

Specifying Mapping A and Mapping B

Specify the information used for mapping a database table or database columns to the Enterprise Bean fields.

The information specified on this screen is saved in the CMP extension information file.

Option	Description
Data source name	Specify a data source name for the database used to make the associated Enterprise Bean fields persistent.
Schema name	Specify a schema name for the database used to make the associated Enterprise Bean fields persistent.
Table name	Specify a table name for the database used to make the associated Enterprise Bean fields persistent.
PK	Select this item for the Primary key.

Option	Description
DB Column Name	Specify a column name for the database to be associated with the persistence field
Related Enterprise Bean Name, Related Field Name	Specify an Enterprise Bean name and field name of the persistence field to be associated with the DB column name. If the multiplicity of relation is One:One, be sure to specify the DB column to be associated with the persistence field of the primary key for the destination Enterprise Bean.
Add	Add a new line.
Delete	Delete the line selected on the list.
Browse DB	Add a line by referring to a database table or column. Clicking the [Browse DB] button displays the login screen.

Specifying the Join Table

Specify information for the join table created for the multiplicity of relation of One:Many, Many:One, or Many:Many. The information specified on this screen is saved in the CMP extension information file of Enterprise Bean.

Option	Description
Schema name	Specify a schema name for the Join table used to associate tables with one another.
Join table name	Specify a name for the Join table used to associate tables with one another.
PK	Select the primary key item of the join table. The primary key item of the Enterprise Bean on the many side of multiplicity of relation is also used for the join table. Therefore, if the multiplicity of relation is Many:Many, all items become primary keys.
DB Column Name	Specify a name for a column in the database.
Related Enterprise Bean Name, Related Field Name	Specify an Enterprise Bean name and field name of the persistence field to be associated with a DB column name. The DB column is required for an association using the primary key items of two Enterprise Beans whose relationship is defined.
Add	Add a new line.
Delete	Delete the line selected on the list.
Browse DB	Add a line by referring to a database table or column. Clicking the [Browse DB] button displays the login screen.

Refer to the "Interstage Application Server J2EE User's Guide" for relationship definition details.

Editing Deployment Descriptors (ejb-jar.xml)

To edit the deployment descriptor(ejb-jar.xml), use an XML editor. For details, refer to "[Editing XML Files](#)".

Creating a CMP extension information file

When the source of an Enterprise Bean class is created separately and if the CMP extension information corresponding to the Enterprise Bean does not exist because of certain reason, then the required CMP extension information file can be generated.

From [New] wizard, select [EJB] > [J2EE] > [CMP Extension Information], and use the wizard to create a CMP extension information file. See below for the wizard settings.

Option	Description
Ejb-jar.xml file	Specify the path of ejb-jar.xml.

Option	Description
Enterprise Bean name	If the path of ejb-jar.xml is specified, since the list of Container-managed persistence Entity Bean defined will be displayed. Select a CMP extension information file of Enterprise Bean to be generated.

Note

The CMP Extension Information wizard does not operate with applications for Java EE containers.

Point

CMP extension information file is generated in the root of source folder where the specified ejb-jar.xml exists.

Editing CMP Extension Information Files

To edit CMP extension information files, use the CMP Extension Information File Editor. It is used to edit the mapping between CMF(Container-managed Fields) and databases, and to edit finder method search criteria.

To edit a CMP extension information file, select FJCMF_(Enterprise Bean class name).xml from the [Project Explorer] or similar view, and then select [Open With] > [CMP Extension Information File Editor] from the context menu. Because an editor is associated once you edit a file, you can also start the CMP extension information editor by double-clicking the file or selecting [Open] from the popup menu.

Point

The CMP Extension Information File Editor does not provide a function to display XML as is. To edit XML source directly, use an XML editor.

D.2.4 Creating an EJB Test Client

Create a test client (client for operation check) from the created Enterprise Beans and check the operation of each business method. This section explains how to create a test client and how to conduct the test. Here, It explains how to create an EJB test client as a Web application.

Point

If you are creating an EJB test client that is connected remotely to an EJB, create a Java application project, and in [Client type] in the EJB test client wizard, specify a client type other than "Web".

"Web" cannot be selected in [Client type] in an EJB test client if the EJB does not contain a LocalHome/Local interface.

Creating a project

From [New] wizard, select [Web] > [Dynamic Web Project] to create a Web application project for use as a test client.

Select the created Web application project, and then, in the context menu, select [Properties] > [Java Build Path] > [Projects] tab. Add the Enterprise Bean project targeted for testing.

Creating the EJB test client source

From the [New] wizard, select [EJB] > [J2EE] > [EJB Test Client], and use the wizard to create the EJB test client source.

See below for the wizard settings. Since the EJB test client is being created as a Web application in this case, select "Web" at [Client type].

- Source folder
Specify the folder of source generation.
- Package
Specify package name. Used in all the classes to be generated.

- Name
Specify the main class name of test client.
- EJB JAR file/EJB project
Specify the path or the project of EJB JAR file including the Enterprise Bean carried out for operation confirmation. The Enterprise Bean carried out for operation confirmation must have either a Home/Remote interface or a LocalHome/Local interface. The Message-driven Bean does not require an interface.
- EJB Test Client Detail Information
See below.

Option		Description
Enterprise Bean test clients		<p>Display the list of Enterprise Bean included in EJB JAR file. According to the specification in [Client type], the Enterprise Beans displayed in the list will change.</p> <p>Specify with a check on the left side of the list, if the test client is generated or not for the Enterprise Bean,</p>
	Enterprise Bean Name	<p>The Enterprise Bean name described in the deployment descriptor that is included in the EJB JAR file will be displayed.</p> <p>The Enterprise Bean list, which is displayed according to the conditions specified in [Client type], will change as shown below.</p> <ul style="list-style-type: none"> - If [EJB] or [J2EE1.3] was specified. <ul style="list-style-type: none"> - Enterprise Bean that has a Home/Remote interface (in the case of a Session/Entity Bean) - Message-driven Bean - If [Web] was specified. <ul style="list-style-type: none"> - Enterprise Bean that has a LocalHome/Local interface (in the case of a Session/Entity Bean) - Message-driven Bean
	Class Name	<p>Specify the class name of test client corresponding to the Enterprise Bean</p> <p>In Message-driven Bean, it is not necessary to specify, since source is not generated for each Enterprise Bean.</p>
	lookup ID	<p>Incase of Session Bean and Entity Bean, for acquiring the Home interface specify the lookup identifier.</p> <p>Specify lookup identifier for getting the destination incases of Message-driven Bean.</p>
Client type		Client classification is specified.
	EJB	Deployment descriptor is not created. Transaction management from a client etc. cannot be performed for those therefore simply call only Enterprise Bean.
	J2EE1.3	Deployment descriptor is not created. Transaction management from a client etc. cannot be performed for those therefore simply call only Enterprise Bean.
	Web	<p>Generates a Servlet class which calls the Enterprise Bean and displays the results.</p> <p>Selection becomes possible if the project specified in [Source folder] is a Web application.</p>



Developing a test client requires the following classes included in the Enterprise Bean to be tested:

- If [Client type] is "EJB" or "J2EE1.3".
 - Home interface class and remote interface class (mandatory)
- If [Client type] id "Web".
 - Local interface class and localhome interface class (mandatory)
- Common conditions
 - User-defined class to be passed as a parameter or return value in Enterprise Bean invocation (valid when this class is being used)



Note

EJB2.1 applications cannot be accessed from remote environment. Therefore specify [Web] in [Client type] in the EJB Test Client wizard to check the operation of the EJB2.1 application from the created Web application, via the local interface.

A test client cannot be created for an Enterprise Bean that exposed a Stateless Session Bean as a Web service.

If J2EE1.3 is selected as the client type, a J2EE1.3 format deployment descriptor is generated in the EJB test client project, but this project cannot be added to an enterprise application project.

Editing generated files

If required, edit the files generated by wizards.

- The business methods expanded into the doGet method are invoked in the same order as in the analysis. For this reason, rearrange the business methods in correct order.
- Check the values to be set in the method parameter.
- To obtain the values of the array elements, add output processing.

Run

Deploy the EJB application to be tested and the created Web application to a J2EE execution environment where the IJServer type is [Web and EJB Applications run in the same Java VM], and launch the applications. Check the operation of the Web application from the Web browser.

D.2.5 Creating EJB Clients

This section describes preparation for creating an EJB client application and the support function for generating an EJB client application.

Preparation for Creating a Session or Entity Bean Client Application

Developing an application that invokes a session bean or an entity bean requires the following classes included in the Enterprise Beans to be invoked:

- Home interface class and remote interface class or localhome interface class and local interface class.
- User-defined class to be passed as a parameter or return value in Enterprise Bean invocation (valid if this class is being used)

To create an application that invokes the Enterprise Bean, in addition to the requirements above, set the client distribution data to the classpath. Refer to "[Client distribution data](#)" for information on client distribution data.



Point

If performing an EJB application test, the procedure below must be used to set Interstage Application Server V11.1 IJServer (J2EE) for the server runtime:

- Select the project. From the context menu, click [Properties] > [Java Build Path] > [Libraries]Tab > [Add Library...]. At [Server Runtime], specify [Interstage Application Server V11.1 IJServer (J2EE)].

If J2EE1.3 is selected as the client type, a J2EE1.3 format deployment descriptor is generated in the EJB test client project, but this project cannot be added to an enterprise application project.

D.2.6 Checking EJB Operation

Refer to "[D.4.6 Checking Application Behavior](#)" for information on checking EJB operation.



Deploy EJB 2.1 applications to the J2EE execution environment shown below. Deployment fails if attempted for a different J2EE execution environment.

- Run Web applications and EJB applications on the same Java VM.

D.2.7 Distributing EJBs to the Operating Environment

An archive must be created in order to distribute EJBs to an operating environment. For details, refer to "[6.2.7 Distributing to the Operating Environment](#)".

D.2.8 Exposing a Stateless Session Bean as a Web Service

This section explains the procedure for exposing a stateless session bean as a Web service.

(1) Creating Stateless Session Bean

Creating a Project

Refer to "[3.3.1 Preparing the Environment to create EJBs](#)" for information on creating projects.

In this case, select [2.1] for [EJB Module version].

Creating an Enterprise Bean

Create an Enterprise Bean with the [New Enterprise Bean] wizard.

1. Enterprise Bean Type
In [Enterprise Bean type], select [Stateless Session Bean].
2. Interface Creation Options
If creating a stateless session bean, select [Generate LocalHome/Local interface]. If a stateless session bean will be invoked from a Web service only, the localhome/local interface is not required. However, in this case, select [Generate LocalHome/Local interface].



Remote access via the home/remote interface is not supported for EJB2.1 Applications that run on Interstage Application Server.

3. Business Method Definition
Define the method that the Web service will publish. After the wizard finished, implement the method that was defined in the created Enterprise Bean class.

Creating a Service Endpoint Interface

Create a service endpoint interface from the Enterprise Bean class that was created.

Ensure that the following conditions are met in the service endpoint interface.

- Inherit the java.rmi.Remote interface
- The method throws a java.rmi.RemoteException
- Restrictions apply to the Java data types that can be specified in method arguments and return values. For details, refer to "Interstage Application Server J2EE User's Guide".

- Do not overload the method
- EJBObject or EJBLocalObject are not included as the argument and return value of the method
- Do not define the following in elements of an array, structure, bean type member to be used in the argument and return value of the method
 - Local interface
 - Remote interface
 - LocalHome interface
 - Home interface
 - Timers interface
 - Timers handle interface
 - The return value of the Collection CMP finder method

Point

Interfaces can be created using the [Extract Interface] functionality.

1. Select the Enterprise Bean class that was created with Project Explorer.
2. From the context menu select [Refactor] > [Extract Interface].
3. In the [Extract Interface] dialog box, in [Members to declare in the interface] select the method to be published.
4. Correct the created interface so that the conditions of the service endpoint interface are met.

(2) Exposing a stateless session bean as a Web service

The following measures are required when exposing a stateless session bean as a Web service.

- Define the service endpoint interface information in the deployment descriptor(ejb-jar.xml)
- The following Web service-related definition files must be created to publish the Web service.
 - WSDL file
 - <WSDL filename>_mapping.xml
 - webservices.xml

By using the Web service wizard, updating and creating the above files can be performed automatically from the service endpoint interface.

Exposing a Stateless Session Bean as a Web service with the Web service wizard

Expose a stateless session bean as a Web service with the Web service wizard. For details on the wizard, refer to "[Generating the files required for a Web service](#)".

1. Source Folder

In the [Source folder], specify the source folder of the Enterprise JavaBeans project that you need to expose as Web service. A project that is not Enterprise JavaBeans version 2.1 cannot be specified.
2. Service Endpoint Interface Name

Specify the name of the service endpoint interface that was created.

Point

When a stateless session bean is exposed as a Web service, the Enterprise Bean class becomes an implementation class, therefore setting the implementation class name is not required.



Note

If you try to expose the following Enterprise JavaBeans as a Web service, an error will occur in the wizard.

- If implementing identical service endpoint interfaces in multiple Enterprise Bean classes
- If registering the same Enterprise Bean class with different EJB names

(3) Packaging to the EAR File

If deploying a stateless session bean that has been exposed as a Web service, perform packaging to the EAR file.

(4) Deployment

Deploy the created EAR file to the Interstage Application Server J2EE execution environment.



Note

If an EAR file contains a Stateless Session Bean that has been exposed as a Web service, deploy the EAR file to the J2EE execution environment shown below. Deployment fails if attempted for a different J2EE execution environment.

- Run the Web application and EJB application on the same Java VM.



Point

If a stateless session bean that was exposed as a Web service is deployed to the IJServer, a Web application named "EJB SOAP router" will be integrated automatically.

If running a stateless session bean that was exposed as a Web service, a servlet for the purpose of transmitting a request from SOAP (HTTP) is required. This servlet is called an "EJB SOAP router".

For every EJB JAR file which includes a stateless session bean that was exposed as a Web service, one EJB SOAP router is created.

D.3 Developing Web Service Applications

This section explains J2EE 1.4 Web service development, focusing on the ways this differs from the Java EE EJB development method.

D.3.1 Preparing the Environment for Creating Web Service Applications

This section explains the provision of a Web service by a Web application.

Refer to "[D.2.8 Exposing a Stateless Session Bean as a Web Service](#)" if the Web service is provided by an EJB application.

Creating a project

Refer to "[2.3.1 Preparing the Environment for Creating the Web Application](#)" for information on creating a project.

Select [Interstage Application Server V11.1 IJServer (J2EE)] as the target runtime specification in the Dynamic Web Project wizard.



Point

Add the required libraries using the following procedure to create Web service applications:

1. Select the project. From the context menu, click [Properties] > [Java Build Path] > [Libraries] tab > [Add External JARs] and add the JAR file from the following location:
`<product installation folder>\APS\J2EE\lib\isws-saaj-api.jar`
2. In the [Order and Export] tab, move the JAR files that were added in 1 above [JRE System Library] then click [OK].

D.3.2 Creating the Web Service

Create the Service Endpoint Interface, and then use the Web service wizard to create the files required for the Web service application.

Creating a Service Endpoint Interface

Create an interface in Java for a service to be made publicly available as a Web service.

From [New] wizard, select [Interface] and create the Service Endpoint Interface.

The service endpoint interface must satisfy the following conditions:

- It extends `java.rmi.Remote`.
- Any defined method for it declares `java.rmi.RemoteException` in the throws clause.



When defining a method, comply with the range of data types that can be used. For details on the data types that can be used, refer to the "Interstage Application Server J2EE User's Guide".

Generating the files required for a Web service

You can use the Service Endpoint Interface to create files required for creating a Web service. From [New] wizard, select [Web Service] > [JServer] > [Web Service (JAX-RPC)] and use the wizard to create the files. See below for the wizard settings.

[Service endpoint related information] page

Option	Description						
Source folder	Specify the destination folder for the implementation template class to be created. The WSDL file, deployment descriptor, and other elements will be created under ContextRoot in the project to which the specified source folder belongs. A source folder included in a project cannot be specified as anything other than an Enterprise JavaBeans project with version 2.1 Web Application Project or Enterprise JavaBeans.						
Service end point interface name	Specify the fully qualified name of the service endpoint interface to be made open for the Web service. The specified service endpoint interface must satisfy the following conditions: <ul style="list-style-type: none">- <code>java.rmi.Remote</code> must be extended.- <code>java.rmi.RemoteException</code> must be specified in the throws clause of each method.- Methods must not be overloaded.						
Implemented template class	Specify information on the implementation template class. This cannot be specified if the project specified in [Source folder] is an Enterprise JavaBeans Project.						
	<table border="1"><tbody><tr><td data-bbox="239 1671 603 1809">Package</td><td data-bbox="603 1671 1406 1809">Specify the package for the implementation templates to be created. The name of the package of the service endpoint interface is displayed by default. If the package for the template differs from the package of the service endpoint interface, change the displayed name.</td></tr><tr><td data-bbox="239 1809 603 1917">Implemented template class name</td><td data-bbox="603 1809 1406 1917">Specify a name for the implementation template class to be created. The service endpoint interface name followed by the character string "SOAPBindingImpl" is displayed by default.</td></tr><tr><td data-bbox="239 1917 603 1993">Not overwrite</td><td data-bbox="603 1917 1406 1993">To prevent overwriting of an existing implementation template class, select this option.</td></tr></tbody></table>	Package	Specify the package for the implementation templates to be created. The name of the package of the service endpoint interface is displayed by default. If the package for the template differs from the package of the service endpoint interface, change the displayed name.	Implemented template class name	Specify a name for the implementation template class to be created. The service endpoint interface name followed by the character string "SOAPBindingImpl" is displayed by default.	Not overwrite	To prevent overwriting of an existing implementation template class, select this option.
Package	Specify the package for the implementation templates to be created. The name of the package of the service endpoint interface is displayed by default. If the package for the template differs from the package of the service endpoint interface, change the displayed name.						
Implemented template class name	Specify a name for the implementation template class to be created. The service endpoint interface name followed by the character string "SOAPBindingImpl" is displayed by default.						
Not overwrite	To prevent overwriting of an existing implementation template class, select this option.						

[WSDL file related information] page

Option	Description
WSDL file name	Indicates a name for the created WSDL file, using a path relative to ContextRoot.
style attribute/use attribute	<p>Specifies a combination of the style attribute and use attribute for writing to the WSDL file. One of the following combinations can be specified:</p> <ul style="list-style-type: none"> - DOCUMENT/LITERAL - RPC/ENCODED - RPC/LITERAL <p>To satisfy the requirements in WS-I Basic Profile 1.0, specify DOCUMENT/LITERAL or RPC/LITERAL.</p> <p>DOCUMENT/LITERAL is more suitable for program communication. Use DOCUMENT/LITERAL, which is our recommendation and is more widely used, unless you have a reason not to use it.</p> <p>RPC/ENCODED is a combination that was widely used before WS-I Basic Profile 1.0 was established. If an RPC/ENCODED-based connection is required, such as when a connection must be established to a system that uses an old SOAP service, use RPC/ENCODED.</p>
Location information	<p>Specifies the endpoint URL used to access the Web service.</p> <p>This value depends on the operating environment. The default value is set to check the operation in combination with a client in the local environment.</p>
Attached file	<p>Specifies how to create a WSDL to develop an application that handles attached files in a Web service. One of the following two methods can be selected:</p> <ul style="list-style-type: none"> - WSDL file is created conforming to WS-I Attachments Profile 1.0 - WSDL file is created using a specific data type <p>To create a WSDL file conforming to WS-I Attachments Profile 1.0 by using the attached files, specify [WSDL file is created conforming to WS-I Attachments Profile 1.0] by using the javax.activation.DataHandler class with a service end-point interface. If the following classes used in the attached files are used when [WSDL file is created conforming to WS-I Attachments Profile 1.0] is specified, a creation error occurs:</p> <ul style="list-style-type: none"> - java.awt.Image - javax.mail.internet.MimeMultipart - javax.xml.transform.Source <p>If a class other than the javax.activation.DataHandler class is used, conversion from binary data to a Java object prevents the original file data from being completely retained. Therefore, the recommended class for use is the javax.activation.DataHandler class.</p>

 Note

Before you can use the Web Service wizard, the Interstage Application Server function or Interstage Application Server client package must be installed.

Since the Web Service(JAX-RPC) wizard examines a class file (.class) to acquire required information, the service endpoint interface must be compiled before activation of the wizard.

 Point

- In the initial workbench state, options are configured such that a build is automatically performed when a file is saved. If no compile error occurs in the process of saving the service endpoint interface, there is no problem.

- If you choose the service endpoint interface in Project Explorer when starting the Web service wizard, the wizard service endpoint interface name and other input items are set as initial values.

The table below lists files created by the Web Service wizard.

Created file	File name	Description
Web service endpoint	<Service-endpoint-interface-name>SOAPBindingImpl.java	Defines the service endpoint interface implementation class. A Web service implementation is defined in this class. The file is created in the source folder and identified from the package and name specified with the wizard.
WSDL file	<Service-endpoint-interface-name>.wsdl	Contains a Web service interface definition. The file is created at the level immediately below ContextRoot/WEB-INF/wsdl.
deployment descriptor	webservices.xml	Contains Web service deployment information. The file is created at a level below ContextRoot/WEB-INF. (If an existing file has the same name, this information is added to the existing file.)
	web.xml	Contains WAR file deployment information. The file is created at a level below ContextRoot/WEB-INF. (If an existing file has the same name, this information is added to the existing file.)
Other file	<WSDL-file-name>_mapping.xml	Automatically generated file required for running a Web service. The file is created at a level below ContextRoot/WEB-INF.



Point

For the rules used during file creation regarding conversion from the Java type to an XML type for WSDL, refer to the "Interstage Application Server J2EE User's Guide".

Editing WSDL files

Use a WSDL editor to edit WSDL files. For details on WSDL editors, refer to "[WSDL Editor](#)".

Validating the propriety of a WSDL file

The following validator is provided as a WSDL propriety validation tool:

- WSDL validator

To execute this validator during build, select the [Validation] builder, and then enable execution of the validator in the [Validation] properties. In the general settings, execution of this validator is enabled.

You can use the above validator for validation even when no build is in progress, by selecting [Validate] from the context menu with the target resource selected.

Editing Deployment Descriptors

To edit the deployment descriptors below, use an XML editor (for details on XML editors, refer to "[Editing XML Files](#)"):

- webservices.xml
- web.xml (if providing a Web service as a Web application)
- ejb.xml (if providing a Web service as an EJB application)

Implementing a Web service application

Implement your Web service application. Write the Web service implementation code in the Web service endpoint file created by the wizard.

Normally, you need not edit the other created files. If you need to edit any of these files, make sure you have a good understanding of the file. For details on each file, refer to the "Interstage Application Server J2EE User's Guide".

Point

.....

If the service endpoint interface uses a Holder class, parameters are treated as INOUT parameters. To treat parameters as OUT parameters, edit the WSDL file.

.....

D.3.3 Exposing a Stateless Session Bean as a Web Service

For details, refer to "[D.2.8 Exposing a Stateless Session Bean as a Web Service](#)".

D.3.4 Creating a Service Endpoint Interface from WSDL

If a WSDL file already exists when you have decided an interface, you can use a command (iswsgen) provided by Interstage Application Server to create a Web service application.

However, this command does not automatically create a deployment descriptor. Therefore, you would have to create a deployment descriptor.

For details on this command and on development using the command, refer to the "Interstage Application Server J2EE User's Guide" or the "Interstage Application Server Reference Manual (Command Edition)".

D.3.5 Creating a Web Service Client

This section explains the procedure for creating a Web service client application.

Point

.....

Add the required libraries using the following procedure to create Web service clients:

1. Select the project. From the context menu, click [Properties] > [Java Build Path] > [Libraries] tab > [Add Library...] and in [Server Runtime] specify [Interstage Application Server V11.1 IJServer (J2EE)] and click [Finish].
 2. Next, add the following JAR files in [Add External JARs]:
<product installation folder>\APS\J2EE\lib\isws-saaj-api.jar
 3. In the [Order and Export] tab, move the JAR files that were added in 2 above [JRE System Library] then click [OK].
-

Acquiring Web service interface information (WSDL files)

WSDL files are required for creating a Web service client.

When creating an Interstage Application Server Web service, from the tree on the left side of the Interstage Management Console, click [Web Service] > [Web Service Class Name]. On the [Web Service] page, select the [General] tab. The WSDL file URL can be obtained from [WSDL] on the [Web Service] page.

If Web service development resources already exist, you can also use the WSDL files of the development resources.

Point

.....

Location information (URL of the Web service) as defined in a WSDL file will be output to files that are created from the WSDL file. To change the runtime environment, you can change the connection destination by customizing the environment setting without rebuilding the application. However, it is easier to use the WSDL file that contains location information for the Web service to be used.

Publicly available WSDL files that can be acquired through Interstage Management Console contain location information that is updated to reflect the environment at the deployment destination. By using these files, you need not consider the connection destination.

If you are also developing a Web service application at the same time, however, you can use the WSDL files of development resources as is, without having to perform deployment or other such work, by specifying debugging environment location information in the Web Service wizard in advance. Web service applications and Web service clients can thereby be developed in parallel.

If the Web Service Client is a J2EE application and JNDI lookup is used to acquire the service interface, you do not need to know the location information that is described in the WSDL file.

.....

Generating a stub

You can create a Java class required for access to a Web service, from the WSDL file that defines a Web service interface. From [New] wizard, select [Web Services] > [JServer] > [Web Service Client (JAX-RPC)] and use the wizard to generate a stub. See below for the wizard settings.

- Source folder
J Specify the destination folder for the created Java source files.
- WSDL file name
Select the WSDL file that contains the interface information defining the accessed Web service.
- Generate user-defined classes
Specify whether to create user-defined classes for use by the Service Endpoint Interface.



Note

Before you can use the Web Service Client(JAX-RPC) wizard, the Interstage Application Server function or Interstage Application Server client package must be installed.

.....

The table below lists files created by the Web Service Client(JAX-RPC) wizard.

Created file	File name	Description
Service endpoint interface	<WSDL portType-name>.java	Contains a description of the Java interface that represents the Web service endpoint interface to be used.
Service interface	<WSDL service-name>.java	Contains a description of the interface used to acquire the stub defining the Web service endpoint in a Web service.
User-defined-type class	xxxxx.java	Created only for a Web service that uses user-specific types.
Holder class	xxxxxHolder.java	Created only if standard Holder classes do not exist, as in a case with a user-defined type or array.
Other class	_isws_+XXXXXX+.java	Class required for running a Web service client. Your application need not consider whether this class is used.



Point

For the rules regarding conversion from an XML type for WSDL to the Java type used during file creation, refer to the "Interstage Application Server J2EE User's Guide".

.....

Note

If you are also creating a Web service application, the service endpoint interface created for the Web service application may be different from the service endpoint interface created from the WSDL file for a Web service client. Therefore, when developing a Web service client, use the service endpoint interface generated from the WSDL file.

Developing a Web service client

Create a client application that uses the generated stub to access the Web service.

The client application gets the stub and invokes the stub method to access the Web service.

The following methods describe how to acquire stubs.

- The method using JNDI to lookup the service object.
- The method using ServiceFactory

Acquiring stubs when JNDI was used to lookup the Service object

If invoking a Web service from a Web application, an EJB Application or a J2EE application client, it is possible to perform lookup of the service object using JNDI.

Editing the deployment descriptor

If using JNDI to lookup the service object from the Web Service client application, depending on the mode of the client application, the service reference description must be defined in the deployment descriptor shown below.

Client Application Mode	deployment descriptor
Web Application	web.xml
EJB Application	ejb-jar.xml
J2EE Application client	application-client.xml

Accessing Web Service from stubs by acquiring a service object

1. Create an InitialContext object
Create a new InitialContext object.
2. Acquire a Service object
Use the lookup method of the acquired InitialContext object to acquire a service object.
In the argument of the lookup, specify the following character string.
java:comp/env/[value specified in <service-ref-name> of the deployment descriptor]
3. Acquire the stub object
Use the method of the acquired service object to acquire the stub object (an instance of the class that implements the service endpoint interface).
4. Invoke the stub object method
Invoke the method of the acquired stub object to access the Web service.

Note

If JNDI is used to perform lookup of a service object, it is not possible to use a single InitialContext object with multiple threads.

Point

If using JNDI from a J2EE application client, the environment of the JNDI service provider must be configured. For details on the JNDI service provider environment settings, refer to the "Interstage Application Server J2EE User's Guide".

Acquiring stubs using ServiceFactory

1. Acquiring the ServiceFactory object
Invoke the newInstance method of the javax.xml.rpc.ServiceFactory class provided by JAX-RPC to acquire the ServiceFactory object.
2. Acquiring the Service object
Use the loadService (java.lang.Class) method of the acquired ServiceFactory object to acquire a Service object (an instance of the class that implements the service interface).
3. Acquiring a stub object
Use the method of the acquired Service object to acquire a stub object (an instance of the class that implements the service endpoint interface).
4. Invoking the stub object method
Call the method of the acquired stub object to access the Web service.

For details on the methods used, refer to the "Interstage Application Server J2EE User's Guide" and javadoc documents about Web services.

D.3.6 Checking Web Service Operation

Refer to "[D.4.6 Checking Application Behavior](#)" for information on checking Web service operation.

D.3.7 Distributing Web Services to the Operating Environment

An archive must be created in order to distribute a Web service to an operating environment. Refer to "[6.2.7 Distributing to the Operating Environment](#)" for details.

D.4 J2EE Application Common Items

This section explains common items related to creating applications that run in a J2EE 1.4 application execution environment, focusing on the ways the development method differs from that of Java EE applications.

D.4.1 Preparing the Deployment Destination that is used for Application Operation Verification

Use the method below to create an IJServer for use as the deployment destination that performs application operation verification.

Creating an IJServer(MyDebug/My1VMDebug)

Normally, the Interstage management console is used to create an IJServer. However, in the method described here, the definition file for the IJServer intended for local debugging is used, and commands are used to create the IJServer.

For details on how to create the IJServer from the Interstage management console, refer to "Interstage Application Server J2EE User's Guide".

1. Select [Interstage Studio Vxx] > [Interstage Management Service Operation Tool] from on the application page (in Windows 8 and Windows Server 2012) or the start menu (in other Windows systems).
2. In the tool window, check that [Use J2EE execution environment] is selected for [Debug environment used at localhost]. If not already selected, select it. Check [Status] under [Service information]. If the service is "Stopped" or "Stopped(some services)", click [Start only mandatory services] to launch the service.
3. On completion of the operation, click [Close] and then close the tool window.
4. Issue the J2EE environment setting/definition operation command (isj2eeadmin).

Issue the command below with the IJServer definition file set as the argument (for details on this command, refer to "Interstage Application Server Reference Manual (Command Edition)"):

```
isstart  
isj2eeadmin ijserver -a -f <Java IDE installation folder>\etc\ijserver\ijserver_studio_debug.xml
```



Reference module information is set in the <Location> tag of the IJServer definition file. Normally, the value is set from the installation folder information during installation. However, if there is an error in the path structure, then change the settings to match the appropriate installation environment.

D.4.2 Preparing the Environment to Create an Application

In order to create the application, projects must be created to suit the modules being developed.

If an application is to comprise multiple modules, these can be grouped together as an enterprise application.

Refer to "[6.2.2 Preparing to Create Applications](#)" for information on preparing to create applications.

Creating a project

Select [Java EE] > [Enterprise ApplicationProject], and select [Interstage Application Server V11.1 IJServer (J2EE)] as the target runtime in the Ear Application Project wizard when developing J2EE1.4 enterprise applications.



The creation of new projects with an EAR version of 1.3 or earlier is not supported.

D.4.3 Creating the Java Class and the Interface

Refer to "[6.2.3 Creating the Java Class and the Interface](#)" for information on creating Java classes and interfaces.

D.4.4 Creating XML Files

Refer to "[6.2.4 Creating XML Files](#)" for information on creating XML files.

D.4.5 Detecting and Correcting Problems

Refer to "[6.2.5 Detecting and Correcting Problems](#)" for information on problem detection and correction.

D.4.6 Checking Application Behavior

This section describes how to check the behavior of each J2EE1.4 application in the Interstage Application Server J2EE container (IJServer).

To check the behavior of a J2EE application, deploy the J2EE application to the server, then start the server. These operations are performed in the Servers view. Check behavior by invoking the J2EE application from the client. Additionally, perform debugging to interrupt program execution and execute code one line at a time, or to check the value of variables.

Preparations for Operating Servers

To perform operations such as starting and stopping servers in the workbench, add the servers to be operated in the Servers view.

Adding Servers

Behavior check is performed when the Servers view that is the deployment destination of the application is added to the server. To add a server, use the New Server wizard. In the Servers view, from the context menu, select [New] > [Server]. The table below lists the wizard setup items:

Setup Items	Setup Content
Server's host name	Name of the server host. If using a local machine, specify "localhost".

Setup Items	Setup Content
Server type	Target version server type. For an Interstage V11 J2EE container, select [FUJITSU LIMITED] > [Interstage Application Server V11.1 IJServer (J2EE)].
Server name	The server name can be customized, but modifications made here are not reflected in Interstage Studio, and are in the following format. "IJServer name [server type name] (host name)"
Server runtime environment	Runtime configuration corresponding to the server type. The server type and server runtime are associated one-to-one, therefore modification is not required. For an Interstage V11 J2EE container, select [Interstage Application Server V11.1 IJServer (J2EE)]. If the server runtime environment is not registered, then the [Server runtime environment] item is not displayed, and the [Interstage Application Server] page is displayed. The server runtime environment is automatically registered. Click the [Next] button.
HTTPS is used for the connection to the management console	When the Interstage Management Console is displayed, select this if using SSL encoding communication. It must match the server settings. For the SSL encoding communication settings of the Interstage Management Console, "Select Operation Mode" can be selected during Interstage installation.
Interstage Management Console Port Number	Interstage Management Console port number. Normally, modification is not required. The default port number is 12000. Modify to match the server environment.
Interstage JMX ServicePort Number	Port number used by the Interstage JMX service to receive requests from the Interstage Management Console. The default port number is 12200. Modify to match the server environment. This item can be set when the host is remote.
Login	Connects to the server. The states are already connected, or, when the button is clicked and login is successful, the button cannot be clicked. If the connection destination server is not 'localhost', then an authentication dialog box is displayed. Specify the user name and password. However, even in the case of the localhost, if the user does not have administrator permissions, an authentication dialog box will be displayed. Set a username and password with administrator permissions.
User name	Username used in server connection. Application Server authentication is performed using the username specified here. Operations are restricted depending on the username used in authentication. For details on authentication and restricted operations, refer to "Interstage Application Server Operator's Guide".
Password	Password used in authentication.
IJServer Cluster	List of IJServer cluster defined in the connected server. Select the target IJServer cluster.
HTTP Port Number	HTTP port number of the server. Modify to match the server environment.
Add and Remove Projects	Project to be deployed to the server. A list of projects is displayed in [Available projects]. Add the project to be deployed to [Configured projects]. Projects can also be added from the context menu after the server is added.

Note

- The only type of IJServer on which Web service applications and EJB 2.1 applications can run is one on which the same Java VM runs EJB applications and Web applications.
- If an old version server type is selected in [Server type], an Interstage Application Server client function with the version selected in [Server type] must be installed. Additionally, in the server to be connected, specify the server in the remote environment.

Associating Projects and Servers

The specifications of the project to be deployed to the server are performed by adding the project to the server of the Servers view. To add the project to be deployed to the server, use the Add and Remove Projects wizard. To start the Add and Remove Projects wizard, in the Servers view, select the server, then from the context menu, select [Add and Remove Projects].

The table below lists the dialog box setup items:

Setup Items	Setup Content
Add and Remove Projects	Project to be deployed to the server. A list of projects is displayed in [Available projects]. Add the project to be deployed to [Configured projects].



Note

If applications that have already been deployed in a separate environment exist on the server, then the Servers view will display the module name, instead of the project name. If applications that have already been deployed exist, and those projects are imported and deployed, in the Servers view, both "Project name" and "Module name" will be displayed. However, modules deployed to APS are recognized as one application.

Starting Servers

In the Servers view, select the server, then from the context menu, select [Start] or [Debug]. The application can be set to deploy automatically when the server starts.



Point

If a connection to the IJServer cluster has not been made yet, in the Servers view, nothing will display in the IJServer cluster [State] field, and operations such as start and stop cannot be performed. In such cases, in the Servers view, select that IJServer cluster, then from the context menu, select [Connect/Login] to connect to the IJServer cluster.

To stop a server, in the Servers view, select the server, then from the toolbar, select [Stop].



Note

If the Interstage Management Service of the server is stopped while connected to the IJServer cluster, connection to the IJServer cluster will be interrupted, causing failure of operations of the IJServer cluster in the Servers view. If operations fail, in the Servers view, select IJServer cluster, and from the context menu, select [Refresh]. After updating the status of the IJServer cluster, from the context menu, select [Connect/Login] to reconnect to the IJServer cluster.

Executing Applications

To check the behavior of the J2EE application deployed to the server, execute a client application that invokes the J2EE application.

In the case of a Web application, the client application starts a Web browser, and checks the behavior by accessing the URL of the application.



Point

In the case of a Web project, in the Servers view, select the deploying Web project, then from the context menu, select [Web browser] to start the Web browser with the URL internally structured.

Debugging Applications

To debug applications, in the Servers view, select the server, then from the context menu, select [Debug]. To perform debugging, set a breakpoint, interrupt the started program, and then execute code one line at a time, and check the content of variables.

For details on debugging, refer to "6.2.6.1 Debugging".

D.5 J2EE1.4 Application Notes

D.5.1 EJB Notes

Client distribution data

The Enterprise Bean Home interface/Remote interface compiled class must be set in the classpath to create an application that invokes the Enterprise Bean. This already compiled class is called "client distribution data".

Refer to the following manual when acquiring the client distribution data.

- "Interstage Application Server J2EE User's Guide" > "Environment Setup for Referencing EJB" > "Setting Client Distribution Data"

java.naming.factory.initial property settings

If an Enterprise Bean is invoked from the client application, the system property must be set using the -D option of the java command in the following combinations:

Client type	Key	Value	Remarks
EJB client	java.naming.factory.initial	com.fujitsu.interstage.ejb.jndi.FJCNContextFactoryForClient	
J2EE application client		com.fujitsu.interstage.j2ee.jndi.InitialContextFactoryForClient	The EJB information to be referenced must be specified in the deployment descriptor of the J2EE application client.

Using EAR if exposed as a Web service

If a Stateless Session Bean is exposed as a Web service, it must be deployed by an EAR file. Execute the [New EAR Application Project] wizard and create an enterprise application project. Select the Enterprise JavaBeans project as a J2EE module to be added to the enterprise application.

Naming service environment variables if using a remote server

To specify implementation of a class that accesses a naming service, specify the name of the implemented class in an environment property (java.naming.factory.initial), such as those shown below.

An EJB client can call only Enterprise Beans. (EJB clients are not capable of transaction control or accessing resources.)

Client type	Value
EJB client	com.fujitsu.interstage.ejb.jndi.FJCNContextFactoryForClient
J2EE application client	com.fujitsu.interstage.j2ee.jndi.InitialContextFactoryForClient

The setting can be specified in the following:

- jndi.properties file
- FJjndi.properties file
- javax.naming.InitialContext(Hashtable environment) argument
- Argument (-D) on the java command line that starts an application

If an error occurs when a client application is actually run, the environment may not be correctly set up. Referring to the troubleshooting section in the "Interstage Studio User's Guide" and appropriate Interstage Application Server documentation, review the environment.

Specifying a naming service connection destination server for client products

Information about the destination of a naming service must be specified because the naming service is accessed, for example, to call Enterprise Beans in remote environment with Interstage Application Server client functions. Specify the applicable server in the following file:

Interstage Application Server installation folder\odwin\etc\inithost

Enterprise Bean cannot be invoked from a remote environment if the IJServer type at the deployment destination is [Web and EJB Applications run in the same Java VM]

If the IJServer type at the deployment destination is [Web and EJB Applications run in the same Java VM], an Enterprise Bean cannot be invoked from a remote environment. In order to invoke it from a client, the Enterprise Bean must be deployed to either the [Web and EJB Applications run in separate Java VMs] or the [EJB Applications Only] server type.

EJB2.1 applications not accessible from a remote environment

EJB2.1 Applications that run on Interstage Application Server cannot be accessed from remote environments. Therefore access EJB2.1 Applications locally from the Web Application.

JMS connection factory and destination

A JMS connection factory and destination must be created, even on the client side, to enable Message-driven Beans to run on a client server. Use the JMS operation commands to create these on the client side.

Perform the following registration processing on the client-side using the JMS operation commands of Interstage Application Server.

- Registration of the JMS connection factory

The JMS connection factory need not be registered if the default JMS connection factory is used.

Use the jmsmkfact command if a JMS connection factory other than the default is to be used.

For details about the jmsmkfact command, refer to the "Interstage Application Server Reference Manual (Command Edition)".

- Registration of the Destination definition

The jmsmkdst command is used to register the destination definition.

For details about the jmsmkdst command, refer to the "Interstage Application Server Reference Manual (Command Edition)".

D.5.2 Web Service Application Notes

Mandatory application server

Before you can use Interstage Studio to develop Web services, the Interstage Application Server function or Interstage Application Server client package must be installed.

WSDL location information

Location information (URL of the Web service) as defined in a WSDL file will be output to files that are created from the WSDL file. To change the runtime environment, you can change the connection destination by customizing the environment setting without rebuilding the application. However, it is easier to use the WSDL file that contains location information for the Web service to be used.

Publicly available WSDL files that can be acquired through Interstage Management Console contain location information that is updated to reflect the environment at the deployment destination. By using these files, you need not consider the connection destination.

If you are also developing a Web service application at the same time, however, you can use the WSDL files of development resources as is, without having to perform deployment or other such work, by specifying debugging environment location information in the Web Service wizard in advance. Web service applications and Web service clients can thereby be developed in parallel.

If the Web Service Client is a J2EE application and JNDI lookup is used to acquire the service interface, you do not need to know the location information that is described in the WSDL file.

Interoperable destinations

The WS-I Basic Profile has been established to provide guidelines for Web services to improve interoperability. These guidelines mainly specify the formats of messages exchanged by Web services, and they include WSDL coding instructions.

Migration of old resources

Older versions supported the Messaging and RPC methods as message exchange methods.

The RPC method can still be applied to the application implementation code, though it involves an incompatibility, such as in the available types.

From the viewpoint of interconnectivity and for other reasons, Interstage does not recommend using an existing WSDL file. To reuse existing development resources, follow the procedure below.

1. Create a service endpoint interface.

If you have been using an interface that conforms to the applicable standards for service endpoint interfaces, you can continue using it as is. If the used data type of the interface does not conform to the standards, re-create the service endpoint interface with a valid data type.

If you have no existing interface, create a service endpoint interface with a valid data type appropriate for publicly available functions.

2. Create files with the Web Service wizard.

Create the files required for Web services from the service endpoint interface.

3. Port the implementation process.

Using existing implementation code, port the process with consideration of essential information such as the changes to the data type or the differences in the execution environment.

Web services operating only in J2EE execution environments with the same VM type

Web services operate only in the [Web and EJB Applications run in the same Java VM] type of J2EE execution environment.

D.5.3 Notes Common to All J2EE Applications

Undeploying J2EE modules when migrating from J2EE module development to J2EE application development

When development of J2EE modules is completed and a transition to the J2EE application development phase is taking place, deployed J2EE modules must be undeployed to prevent a possible deployment error caused by a difference in J2EE module management methods.

Before the EAR file is created for Interstage rapid deployment, J2EE modules must be undeployed, because the EAR file is deployed internally when the file is created.

D.6 Compatibility Information

D.6.1 Migration from J2EE Container to Java EE Container

This section explains how to change a J2EE1.4 application intended for a J2EE container to one intended for a Java EE container.



The following applications cannot be ported as applications intended for Java EE containers:

- Web services (JAX-RPC)
 - Web service clients (JAX-RPC)
-

Web applications

Use the procedure below to enable operation in Java EE containers:

1. Select properties from the project context menu.
2. On the [Targeted Runtimes] page, select [Show all runtimes].
3. Select Java EE runtime from the displayed runtimes, and then click [OK].

EJB applications

Migration processing for EJB applications that do not contain CMPs

Use the procedure below to enable EJB applications that do not contain CMPs to run in a Java EE container:

1. Select properties from the project context menu.
2. On the [Targeted Runtimes] page, select [Show all runtimes].
3. Select Java EE runtime from the displayed runtimes, and then click [OK].

CMP2.0 migration processing

The following differences related to CMP2.0 apply to Java EE containers and J2EE containers:

Option	JavaEE Container	J2EE Container
Target runtime	JavaEE container	J2EE container
CMP extension information file	sun-cmp-mappings.xml	FJCMP_XXXX.xml
Database definition information	sun-ejb-jar.xml	FJCMP_XXXX.xml
Database table information	XXX.dbschema	None

Accordingly, new sun-cmp-mappings.xml, sun-ejb-jar.xml, and XXX.dbschema files must be generated to use CMP2.0 in Java EE containers. Use the [Update to CMP 2.0 Java EE environment] wizard to generate these files and change the target runtime. From [New] wizard, select [EJB] > [J2EE] > [Update to CMP 2.0 Java EE environment].

Check and enter the settings below.

Option	Description
Project selection	Select the project to be migrated. The displayed projects are EJB projects that contain CMPs.
Use DB Schema	Specify whether or not to use database schemas.
Connection	A list of already created database connections is displayed. Select one of them.
Add connection	Opens the [New Connection Profile] wizard.
Connection	Connects to the database selected in the connection list box.
JNDI Name	Specify the JNDI name for the database connection used by the Java EE container.

After the above information is set, click [Next], set the items below and then click [Finish].

Option	Description
Project name	A selected project list is displayed.
EJB name	A list of CMPs included in the selected projects is displayed.
Table name	Information concerning the table that sets CMP relationships is displayed
Schema name	From the information concerning the database connected from the previous page, select the schema used by the table. If relationships exist between CMPs, the information concerning all CMP schemas that have relationships can be changed.

Note

The wizard for updating to a CMP2.0 Java EE environment does not support CMP1.1 migration. If CMP2.0 and CMP1.1 both exist in a project, migration processing is performed for CMP2.0 only, not for CMP1.1. Refer to "CMP1.1 migration processing" for information on migrating CMP1.1.

CMP1.1 migration processing

The following tasks are required for CMP1.1 migration:

1. Information migration from FJCMP_XXX.xml to sun-cmp-mapping.xml

Code the information in FJCMP_XXX.xml in CMP1.1 to sun-cmp-mapping.xml.

Item correspondences are shown below.

sun-cmp-mappings.xml			FJCMP_XXX.xml
sun-cmp-mappings			
sun-cmp-mapping+			
schema			fujitsu-cmp-definition/schema-name
entity-mapping+		<-	fujitsu-cmp-definition
ejb-name			EJB name(ejb-jar.xml/ejb-name)
table-name		<-	fujitsu-cmp-definition /table-name
cmp-field-mapping+		<-	fujitsu-cmp-definition /field-map
field-name		<-	fujitsu-cmp-definition /field-map/field-map-entry/field-name
column-name+		<-	fujitsu-cmp-definition /field-map/field-map-entry/dbcolumn-name

Refer to the Interstage Application Server manual for further details.

2. Coding JDOQL queries to sun-ejb-jar.xml

Since SQL query specifications are being changed, code the SQL query information to sun-ejb-jar.xml in JDOQL format.

sun-ejb-jar.xml : <ejb></ejb-name>, <ejb></cmp></one-one-finders></finder>

Code this item in the description format adopted for JDOQL, that is, <method-name>, <query-params>, <query-filter>, <query-ordering>.

An example of finder method coding is shown below.

```

<ejb>
  <ejb-name>TestCMP11</ejb-name>
  <jndi-name>jdbc/ORACLE</jndi-name>
  <cmp>
    <one-one-finders>
      <finder>
        <method-name>testMethod</method-name>
        <query-params>int param1</query-params>
        <query-filter> id &lt; param1</query-filter>
      </finder>
    </one-one-finders>
  </cmp>
</ejb>

```

Refer to the Interstage Application Server manual for details.

3. Generating database schema files

Use the following command to generate database schema files:

<Interstage Studio installation folder>\APS\F3FMisjee\bin\capture-schema.bat

The following information must be specified when executing this command:

Mandatory item	Value
database user name	Database connection user name
database user password	Database password
database connection destinations URL	JDBC connection destination URL
JDBC driver name	Driver name of the JDBC driver to be used
output file path	{Workspace directory}/{project}/{source directory}/{sun-cmp-mappings/sun-cmp-mapping/schema}.dbschema
name of schema to be used	fujitsu-cmp-definition/schema-name
name of table to be used	fujitsu-cmp-definition/table-name

Execute the command in the following format:

```
capture-schema -username {database user name} -password {database user password}
-dburl {database connection destination URL} -driver {JDBC driver name} -out {output file
path}
[-schemaname {name of schema to be used}] [-table {name of table to be used}]*
```

Ear applications

Use the procedure below to enable operation in Java EE containers:

1. Select properties from the project context menu.
2. On the [Targeted Runtime] page, select [Show all runtimes].
3. Select Java EE runtime from the displayed runtimes, and then click [OK].

Appendix E Troubleshooting

This appendix describes the methods for resolving problems that are issued by the workbench.

E.1 General Problems Related to Workbench

A file, folder, and project cannot be deleted.

[Symptom]

Workbench operations may generate a file incorrectly such that the file path exceeds the maximum value for a file system path length, and attempts from the workbench or Explorer to delete that file, or a folder or project that contains that file, may fail.

[Action]

Use the following command to delete the file and the folder.

```
java -classpath <Workbench installation folder>\bin\f5drprfc.jar RemoveFolderContents folder-name
```

For each file and folder in the specified folder, the command asks whether to delete it and deletes the files and folders individually. The specified folder itself is not deleted.

The [Windows Security Alert] dialog box is displayed.

[Symptom]

The [Windows Security Alert] dialog box shown below may be displayed in the following cases:

- An application is debugged.
- A web application or an Apcoordinator application is executed using the Tomcat launch configuration.
- Select [Help] > [Help Contents] from workbench menu.

The [Windows Security Alert] dialog box is displayed with a message as shown below.

```
To help protect your computer, Windows Firewall has blocked some features of this program.  
Do you want to keep blocking this program?
```

```
Name (N): Java(TM) 2 Platform Standard Edition binary  
Publisher (P): FUJITSU LIMITED
```

[Action]

In such cases, click any of [Keep Blocking], [Unblock], and [Ask Me Later] in the dialog box. Regardless of which button is clicked, the program can be used without any problems.



The messages in the [Windows Security Alert] dialog box depend on the operating system you use. The messages described in this document are based on Windows XP Service Pack 2.

A memory shortage error occurs and Workbench may terminate.

[Symptom]

If a process that uses a large amount of memory is performed on the workbench, the following error message may appear and the workbench may terminate.

'An out of memory error has occurred. Consult the "Running Eclipse" section of the read me file for information on preventing this kind of error in the future.'

[Action]

If this occurs, refer to the following article and extend the heap area:

<Workbench installation folder>\eclipse\isstudio.htm

The Interstage Management Console or Interstage Java EE Admin Console does not start.

[Symptom]

The stopped state is standard for locally installed application server services. Since services are stopped, the Interstage Management Console and the Interstage Java EE Admin Console are not started.

[Action]

If a local Interstage Management Console or Interstage Java EE Admin Console is used, use the Interstage Management Service Operation Tool to start the application server services.



Note

Notes related to problems that occur in the workbench

Fujitsu cannot provide support for problems that occur in the following:

- Features that are hidden by default in the standard workbench

Of the features displayed using [General] > [Capabilities] > [Advanced] on the settings page, this means the deprecated features that are in a non-selected state by default so they remain hidden, such as "Plug-in Development", "EMF Development", and "Non-recommended Function of Server Runtime".

- The following Java EE 6 workbench features:

- CVS
 - EMF (Eclipse Modeling Framework)
 - Plug-in Development features
 - EJB XDoclet related features
 - Static Web Project related features
 - Web service Axis related features
 - Server features of non-Interstage Application Server
 - Database features related to database products not described in "Related Software" of the Software Release Guide
-

E.2 Problems Related to Java

A warning error related to a generic type in a source generated by a wizard or similar is displayed.

[Symptom]

A warning error related to a generic type introduced under Java 5 may be displayed when a build uses a source generated by a wizard or similar or an imported sample.

[Action]

To avoid displays of generic type warnings, under the [Window] menu > [Preferences] > [Java] > [Compiler] > [Errors/Warnings], change each value of [Generic types] to [Ignore].

E.3 Problems Related to Databases

Even if a view filter is set, all views are displayed in the Data Source Explorer view

[Symptom]

In the Data Source Explorer view, in the "view" item properties, even if [Disable filter] is cleared to enable filters, all views are displayed in the Data Source Explorer view.

[Action]

In the Data Source Explorer view, in the "view" item properties, view filtering cannot be performed. If you want to filter a view, in the schema that view belongs to, under "table" item properties, specify a filter. The "table" item filter settings are applied to both the table and the view.

It might not be possible to delete Symfoware Server tables

[Symptom]

After a Symfoware Server table is edited using the Table Data editor, the "DROP TABLE" SQL statement fails. If table deletion fails, in the SQL Results view, "JYP3913E Table "table name" being used exclusively by another user." will be displayed.

[Action]

In the Data Source Explorer view, disconnecting the database, then connect to it again. After this, execute the "DROP TABLE" SQL statement.

Symfoware Server indexes are displayed using a different name (DSO name)

[Symptom]

In the Data Source Explorer view, the list of indexes defined in the database can be displayed. If connected to the Symfoware Server, not the index name but rather the DSO name will be displayed in the index name.

When the table [Delete] context menu is selected, the "DROP TABLE" SQL statement is generated. If an index is defined in a table, then the "DROP INDEX" SQL statement is also generated. If connected to the Symfoware Server, then the DSO name (instead of the index name) will be specified in the SQL statement that deletes indexes. As a result, when the SQL statement that deletes indexes is executed, this feature might fail.

[Action]

Either specify the index name in the SQL statement that deletes indexes or execute the "DROP DSO" and "DROP DIS" SQL statements. The DSO name of the index is the same as the DSI name.

E.4 Problems Related to JavaScript

JavaScript parsing errors might occur

[Symptom]

The literal format of reserved words and regular expressions might not be handled correctly in the JavaScript syntactic analysis and result in an error.

[Action]

Disable the JavaScript syntactic analysis is not executed, according to the procedure below:

- If disabling the JavaScript validator:
 1. In the Project Explorer, select the target project.
 2. From the context menu, select [Properties] to open the [Properties] dialog box.
 3. In the left trees view of the [Properties] dialog box, select [Builders].
 4. In the [Builders] page, unselect [JavaScript Validator].
- If disabling JavaScript syntax validation:
 1. In the Project Explorer, select the target project.
 2. From the context menu, select [Properties] to display the [Properties] dialog box. Or, from the workbench menu, select [Window] > [Preferences], to display the [Preferences] dialog box.
 3. In the [Properties] page, or, the left tree of the [Preferences] page, select [Validation].

4. In the [Validation] page, clear [Build] of [JavaScript Syntax Validation].

Point

To delete a problem from the Problem view, before disabling JavaScript syntactic analysis, in a state where build is not performed automatically, project clean must be executed.

Note

When the JSP editor enters a JSP tag in JavaScript code between `<script>` - `</script>` tags, a JavaScript syntax error may be displayed in the Problems View.

JSP tags are dynamically converted when JSP is executed. However, because JSP tags cannot be converted when the editor performs a JavaScript syntax check, they may appear to be syntax errors.

If this applies, perform the build or execution operation as is. Even though a syntax error is detected, the build or execution operation can be performed.

E.5 Problems Related to Editor

The Java editor or Ant editor cannot perform breakpoint operations.

[Symptom]

The Java editor and Ant editor cannot be used to delete or disable a breakpoint under the following condition:

- After a line is added or deleted at a location before the line containing a set breakpoint, an attempt is made to use the breakpoint without saving the file.

[Action]

If this symptom occurs, take one of the following actions to solve the problem:

- Perform the breakpoint operation from the breakpoint view.
- Save the file before performing the breakpoint operation.

Operation may become slow when the CSS editor handles large files.

[Symptom]

If one of the following operations is performed with the CSS editor handling a large file containing 1000 or more lines, the CSS editor may operate slowly:

- Deleting a large number of lines at the same time
- Restoring a large number of edited parts to their original (unedited) state at the same time
Example) An entire document is formatted once, and then the document is restored to its original state.

[Action]

Therefore, use a text editor for an operation that involves deleting a large number of lines. Also, save the file before performing formatting or another operation that may affect the entire document. In addition, to restore the file to its original state, cancel the changes made, close the file, and then reopen it.

E.6 Problems Related to IJServer Clusters and Interstage Java EE DAS Services

IJServer Cluster debugging fails to start.

[Symptom]

IJServer Cluster debugging fails to start and the following error dialog box is displayed:

Title: A problem occurred.

Content: A problem occurred in 'During server start - {0}'.

The server failed to start.

Details: The server failed to start.

OM2014: All server instances in cluster {1} were not started.OM1051: Server Instance unable to start: NAME={2}

[Variable information]

{0}: Server name displayed in the Server View

{1}: IJServer Cluster name

{2}: Instance name specified when the IJServer Cluster was created.

[Cause]

The cause is that the [Debug Port] port specified when the IJServer Cluster was registered in the Server View is already being used.

[Action]

Take one of the following actions:

- Terminate the other application that is using the port specified at [Debug Port], and then repeat the operation.
- Temporarily delete the IJServer Cluster from the Server View and use the [New Server] wizard again. Specify a [Debug Port] that does not duplicate that of other applications, and then repeat the operation.

An Interstage Java EE DAS service project operation failed.

[Symptom]

An addition, deletion, or other project operation for the Interstage Java EE DAS service was performed, but the operation failed and the following error dialog box is displayed:

Title: "A problem occurred." or "Problems occurred."

Content: Publication failed.

Details: Publication failed.

Module publication cancellation failed: {0}

OM2997: Unable to connect to admin-server at given host: [{1}] and port: [{2}]. Please check if this server is up and running and that the host and port provided are correct.

[Variable information]

{0}: Operation target project name

{1}: Host name

{2}: Interstage JMX service port number specified during server registration

[Cause]

The cause is one of the following:

- The project was used when the Interstage Java EE DAS service was in a stopped state.
- The [Interstage JMX Service Port Number] specified when the Interstage Java EE DAS was registered in the Server View is incorrect.

[Action]

Check that the Interstage JMX service port number is correct. From the Server View, repeat [Connect/Login] or [Connect(Debug Launch)/Login], and then perform the operation again.

The [New Server] wizard displays "Unexpected end of file from server" and login fails.

[Symptom]

When the [Login] button is clicked on the [New Interstage Application Server] page of the [New Server] wizard, the following error message is displayed and login fails:

Content: Login failed for the following reason: Unexpected end of file from server

[Cause]

HTTP connections are not permitted by the Interstage Application Server installed at the host specified at [Server's host name] on the [Define a New Server] page of the [New Server] wizard, but [Use HTTPS communication to connect to target] is not selected on the [New Interstage Application Server] page.

[Action]

Check the communication settings of the connection destination Interstage Application Server and, if necessary, change them. Alternatively, select [Use HTTPS communication to connect to target] on the [New Interstage Application Server] page and click the [Login] button.

E.7 Problems Related to Server Operation Verification

The "Bad version number in .class" error is displayed when performing operation verification for EJB, Web, or similar applications.

[Symptom]

An error message is displayed during EJB or Web application operation verification.

The following is an example of Web application operation verification, but errors are output to the server log in a similar way for EJB applications:

```
HTTP Status 500 -
type
    Exception report
message
description
    The server encountered an internal error () that prevented it from > fulfilling this request.
exception
    javax.servlet.ServletException: JSVLT52352: Error during servlet instance allocation
root cause
    java.lang.UnsupportedClassVersionError: Bad version number in .class > file
```

[Cause]

The cause is that the JDK/JRE version of the server performing operation verification is an older version than the [Compiler compliance level] of the [Java Compiler] used for the build. Example: The JDK conformance level is 1.6 and the server JDK/JRE version is 1.5.

[Action]

Make the [Compiler compliance level] of the [Java Compiler] and the JDK/JRE version on the server performing operation verification the same, or perform the build in the older environment and perform operation verification again.

Appendix F Tutorial

This tutorial uses examples to explain the procedure for developing applications.

F.1 Java Application

Java application projects can develop the applications shown below.

- [Client Application](#)
- [Applet](#)
- [JavaBeans](#)

F.2 Client Application

You will learn how to develop client applications, by creating a simple application.

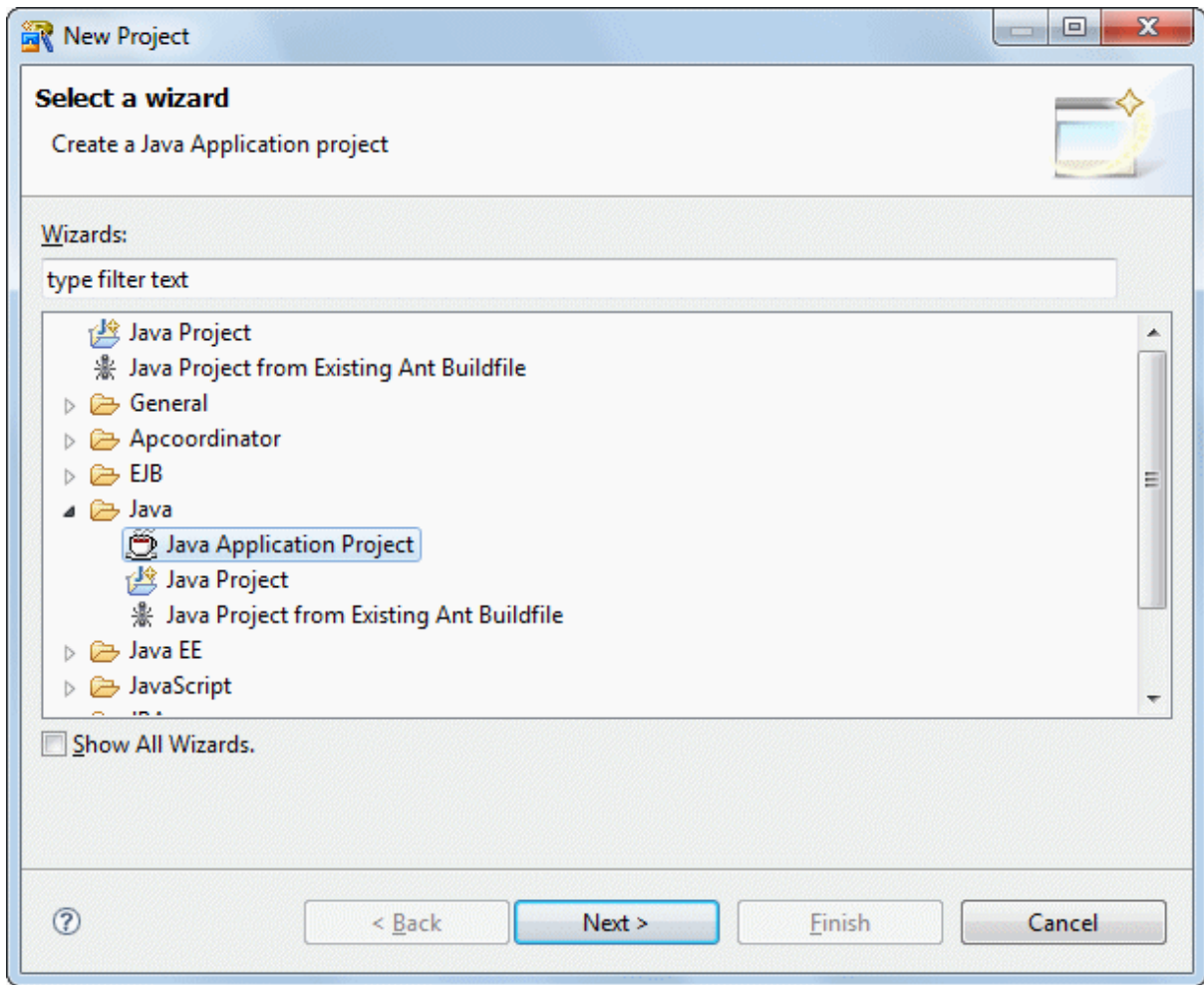
F.2.1 Lesson1 Calendar

This section explains by means of a simple example how to develop a simple application using the JBK (J Business Kit) calendar Bean, create a new project, and run the created application.

Creating a Java Application Project

1. Start the workbench.
2. Select [File] > [New] > [Project] from the menu bar.

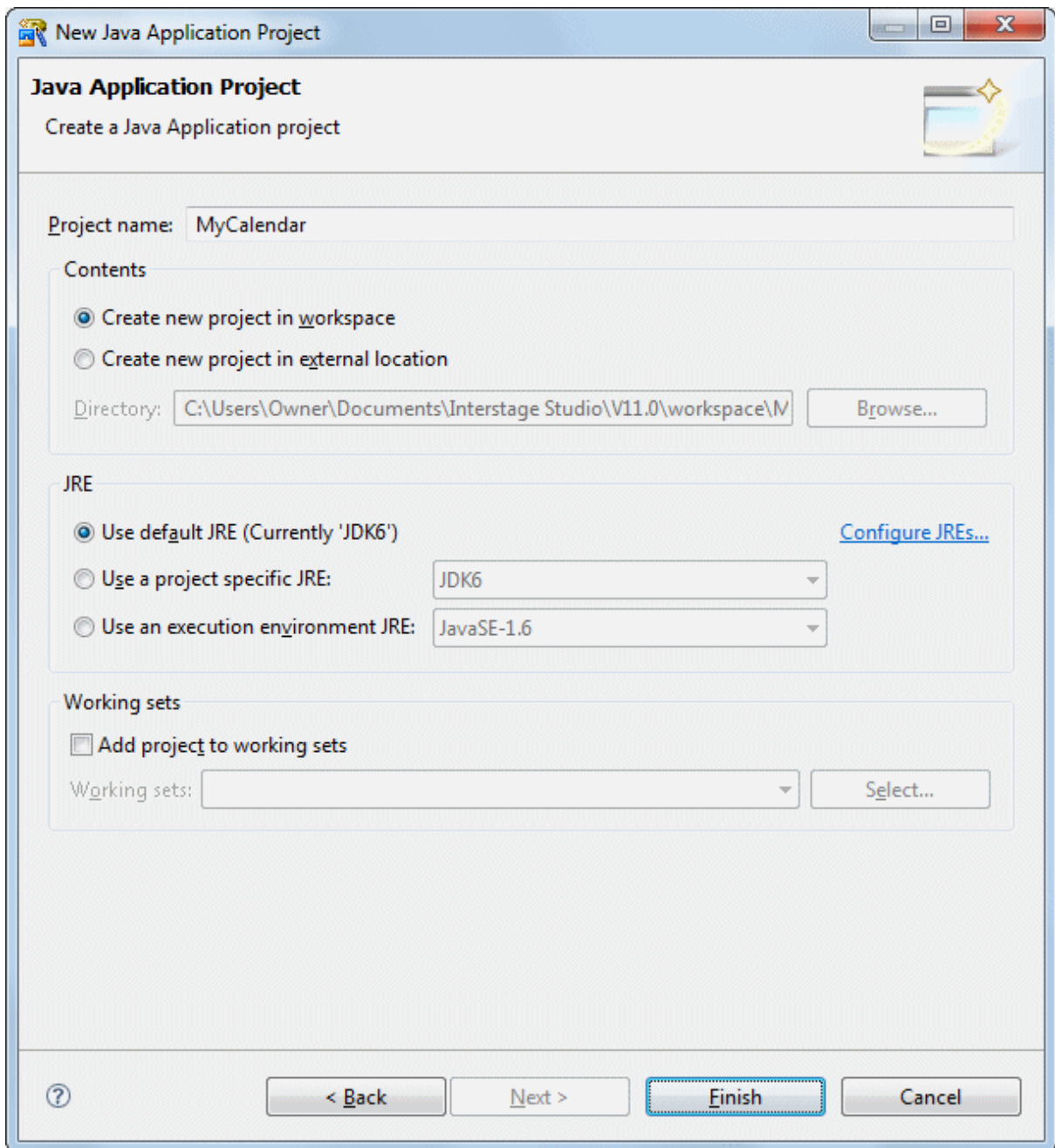
3. The [New Project] wizard is displayed. Select [Java Application Project] from the tree.



Click [Next].

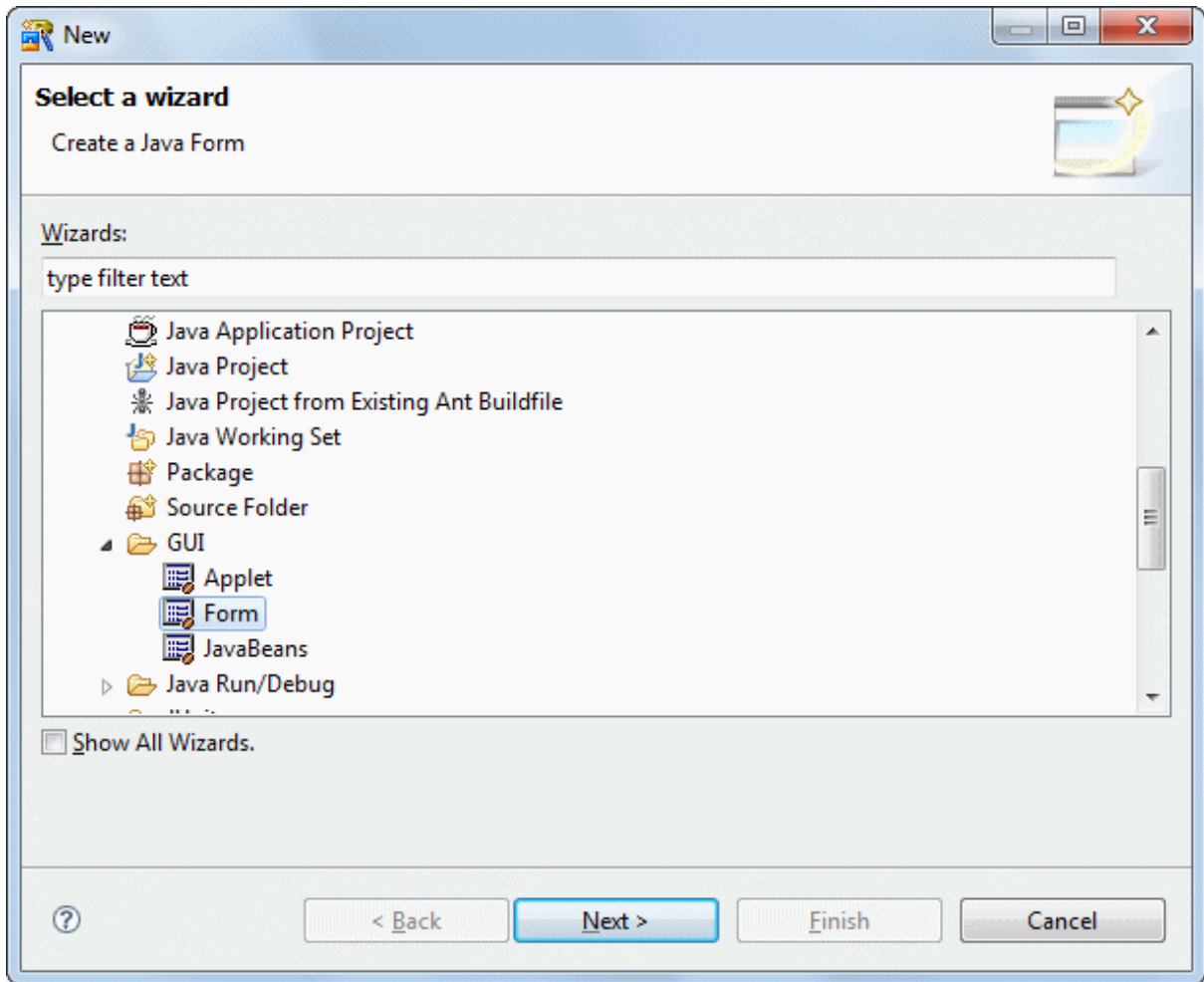
4. The [Java Application Project] page is displayed. Enter the project information as follows:

Setup Items	Setup Content
Project name	MyCalendar



Click [Finish]. This step creates the project.

- The next step is to create the frame.
Select [File] > [New] > [Other] from the menu bar. Then, select [Java] > [GUI] > [Form] from the tree in the [New] wizard that appears.



Click [Next].

- The [Java Form Information] page is displayed.
Enter information as follows.

Setup Items	Setup Content
Source folder	MyCalendar/src
Package	myapp.javaform

New Form

Java Form Information

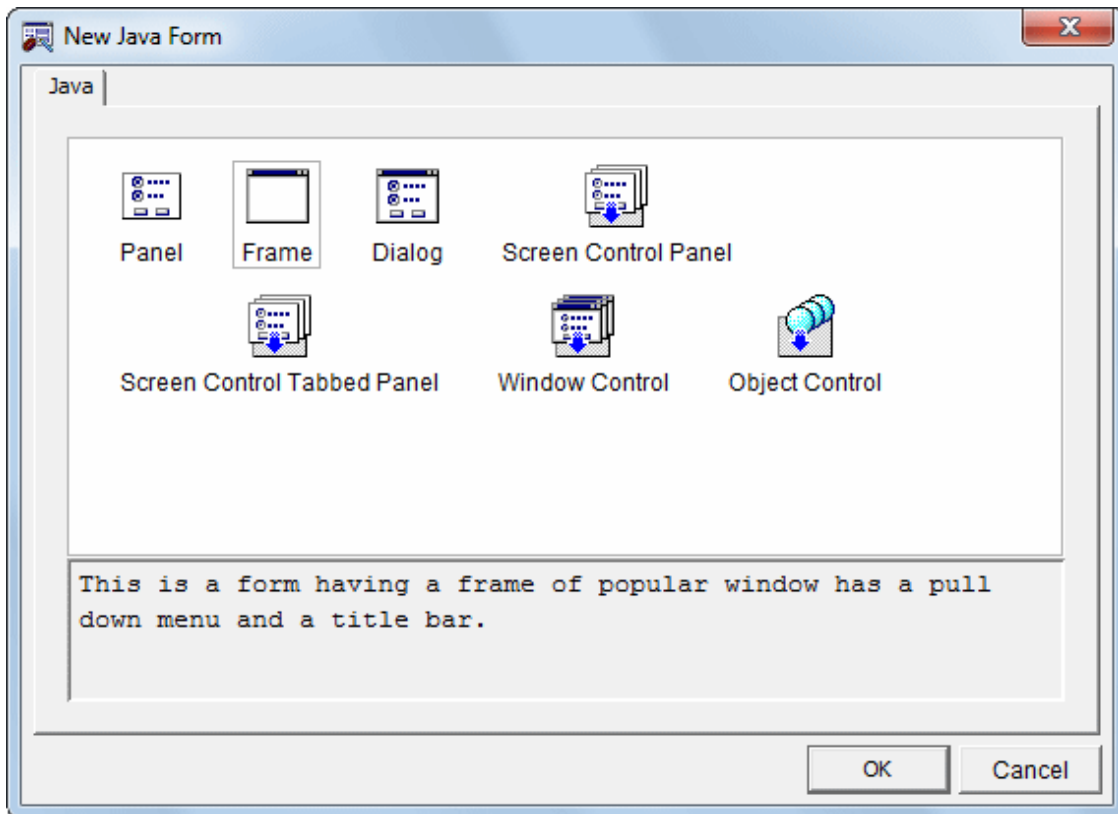
Please input the Java Form information.

Source Folder:

Package:

Click [Next].

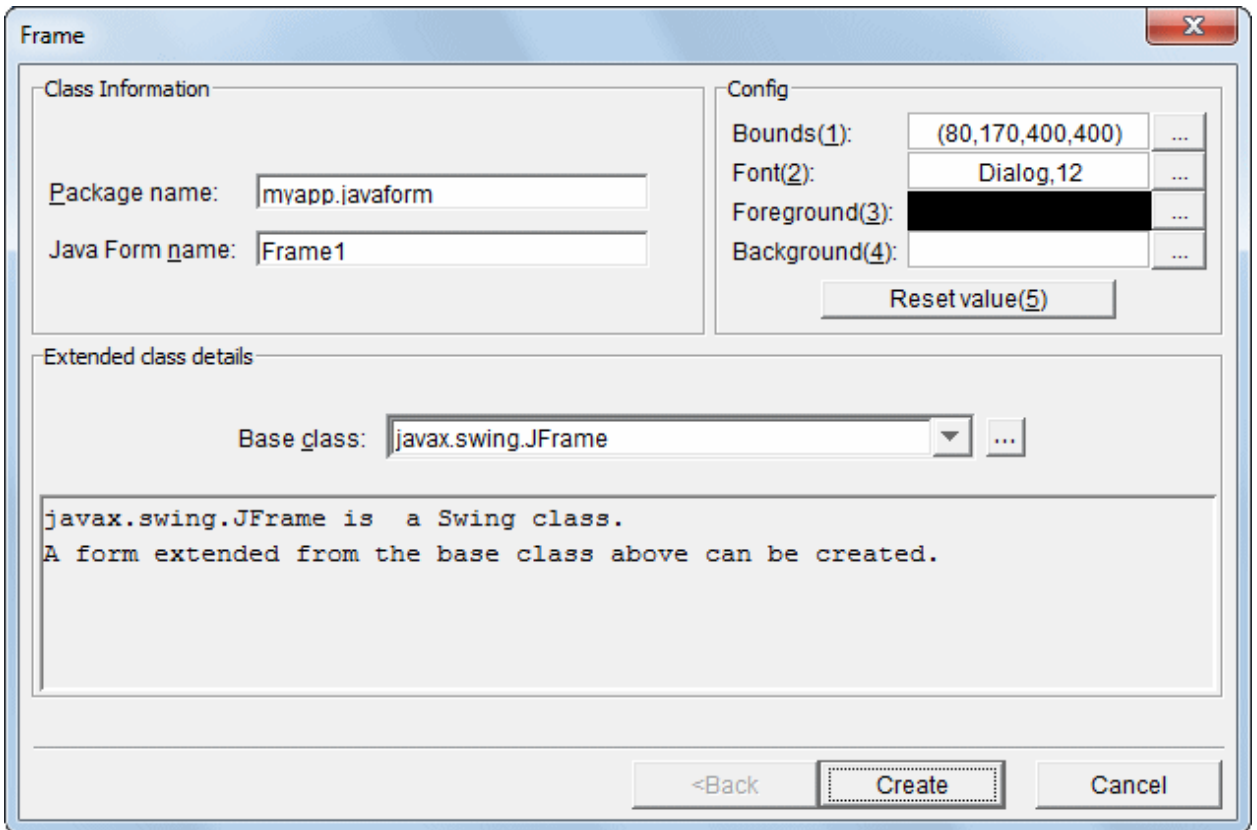
- The [New Java Form] dialog box is displayed.
Select [Frame].



Click [OK].

- The [Frame] dialog box is displayed.
On this page, specify the information required to create the frame.
Enter information as follows.

Setup Items	Setup Content
Package name	myapp.javaform
Java Form name	Frame1
Base class	javax.swing.JFrame



Click [Create].

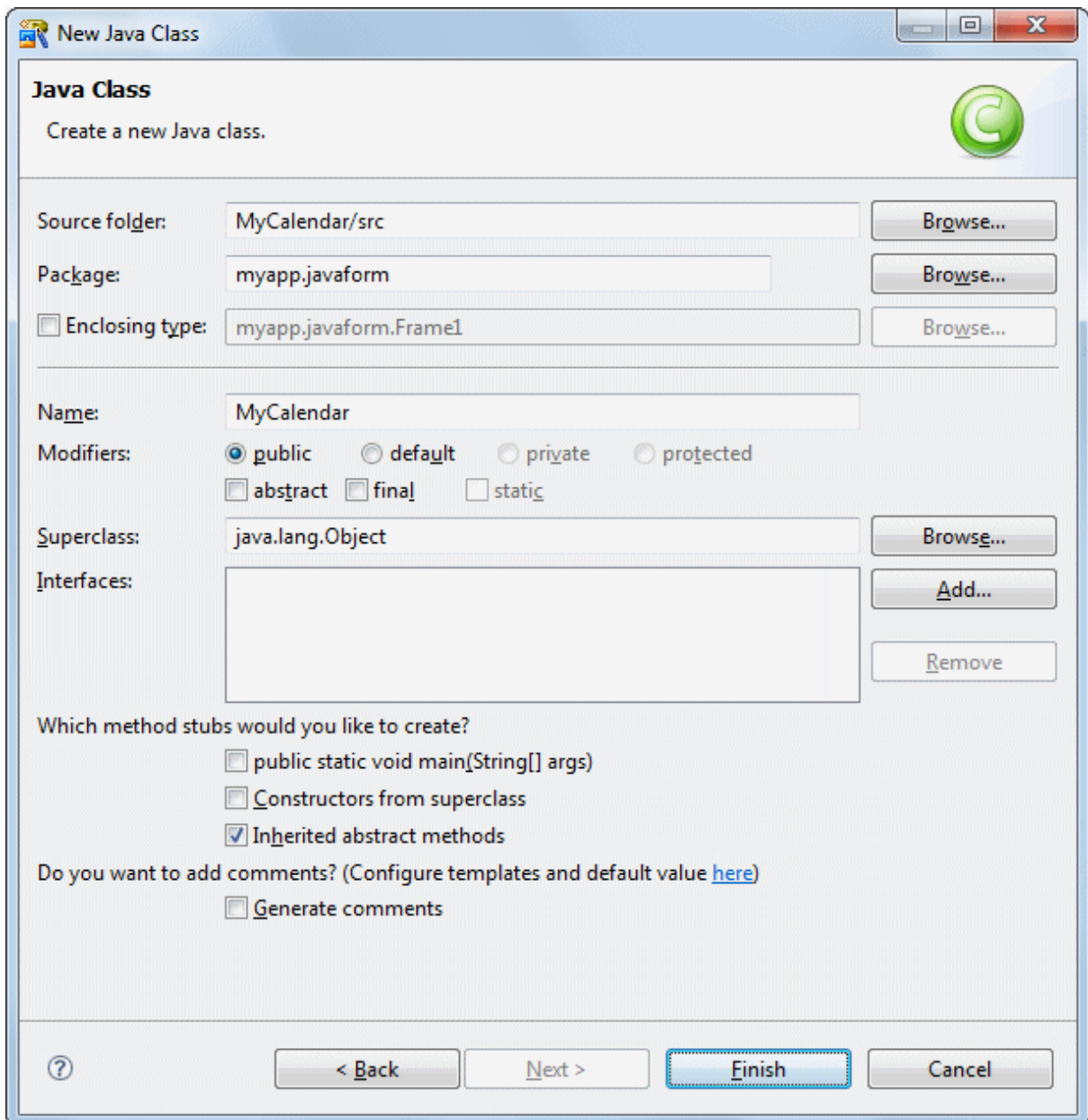
Point

Heavyweight components and lightweight components

There are two types of containers for Beans: heavyweight (AWT) and lightweight (Swing). You are strongly advised to be consistent with your use of heavyweight and lightweight components when creating Java Forms and applets. If heavyweight and lightweight components are mixed in the same application, the following problems may occur: overlap (the heavyweight component is always displayed on top of the lightweight one) and creeping of the menu into a Bean.

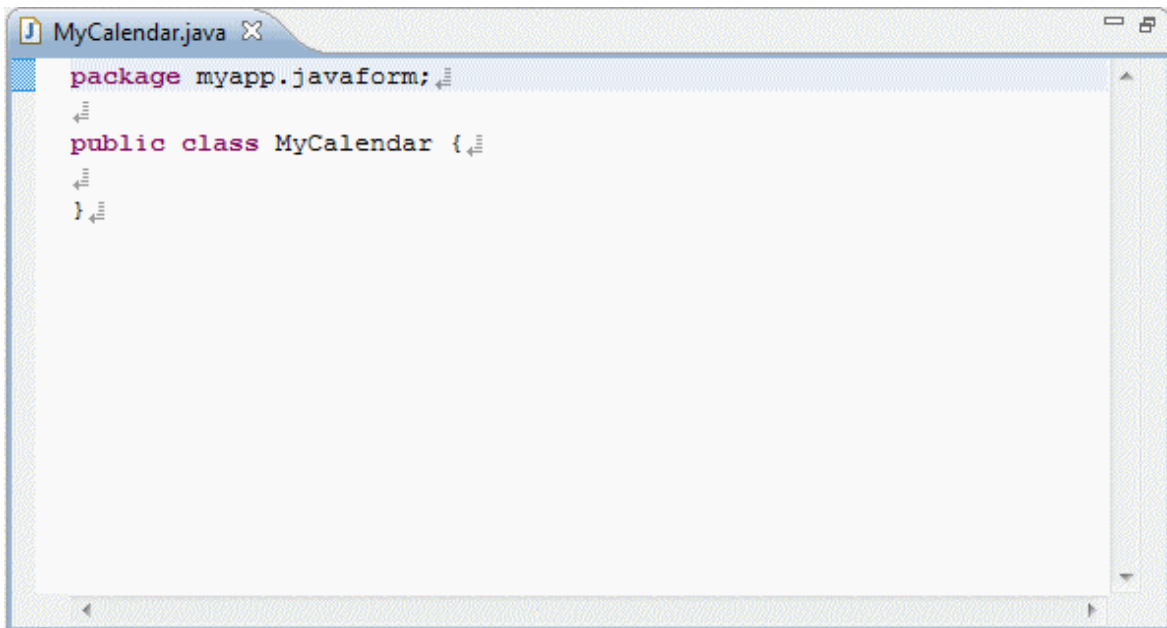
9. The next step is to create a class of the form and an executable class that will display the form. Select [File] > [New] > [Class] from the menu bar.
10. The [Java Class] page is displayed. On this page, specify basic information for the Java class to be created. Enter information as follows.

Setup Items	Setup Content
Source folder	MyCalendar/src
Package	myapp.javaform
Name	MyCalendar
Inherited abstract methods	Checked



Click [Finish].

11. The source created in the [Java Editor] is displayed.



```
MyCalendar.java X
package myapp.javaform;
public class MyCalendar {
}
```

Edit the source that was created. Add the parts in **red**.

```
//Constructor
public MyCalendar() {
}

public void run() {
    //Create new form project
    Frame1 form = new Frame1();
    //Show form
    form.setVisible(true);
}

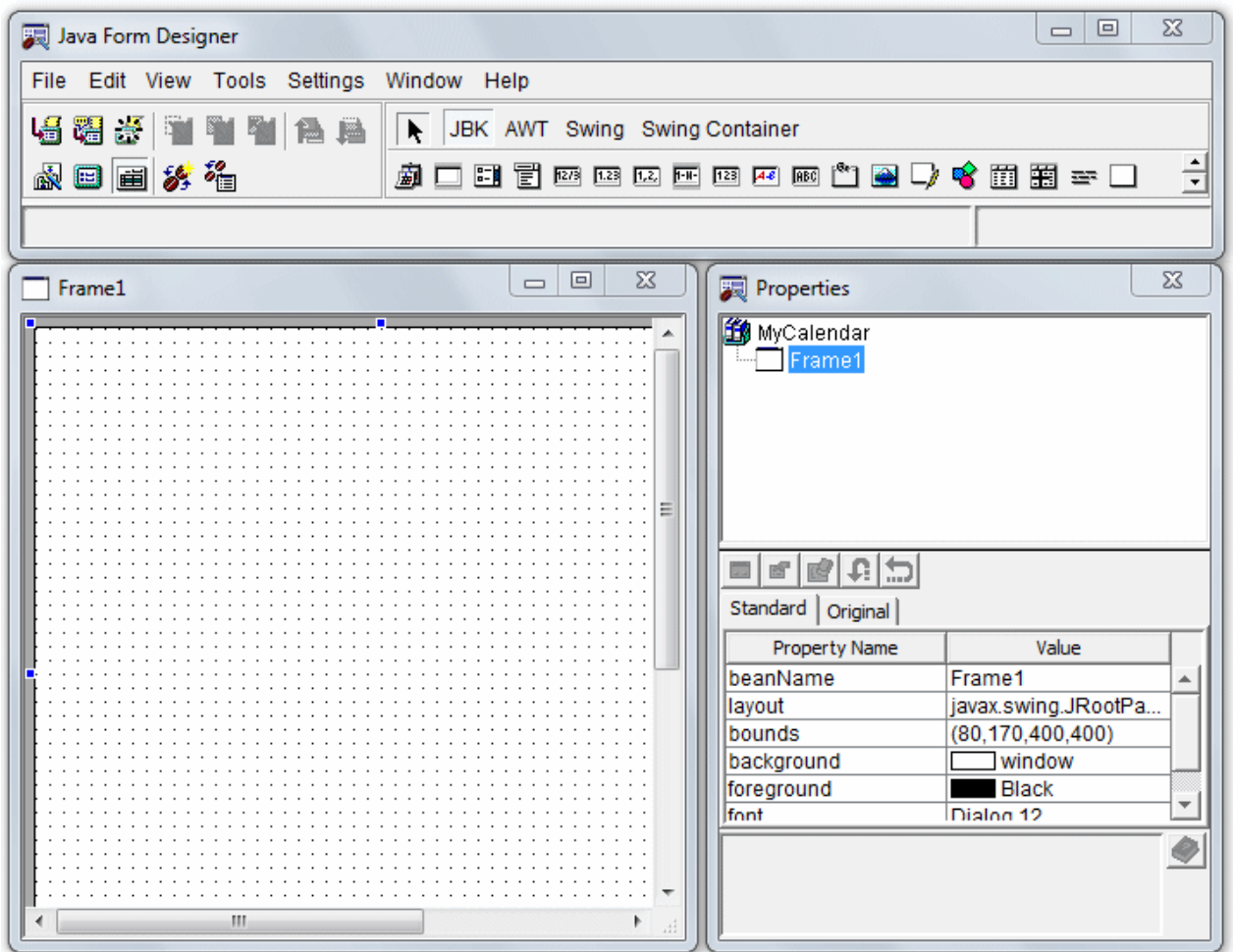
//Main process
public static void main(String[] args) {
    // Create MyCalendar class instance
    MyCalendar object = new MyCalendar();
    // Call run method of the MyCalendar class instance
    object.run();
}
```

12. The above steps have created a class of the form and an executable class that can display the form.

Defining Java Forms

1. A Java Form has a specific screen shape and layout defined in a Java class. This section explains how to edit this Java Form and make it complete.

2. In Java Form Designer, place Beans, define properties (attributes), and code processing procedures to complete the form.



3. Modify Frame1 properties.

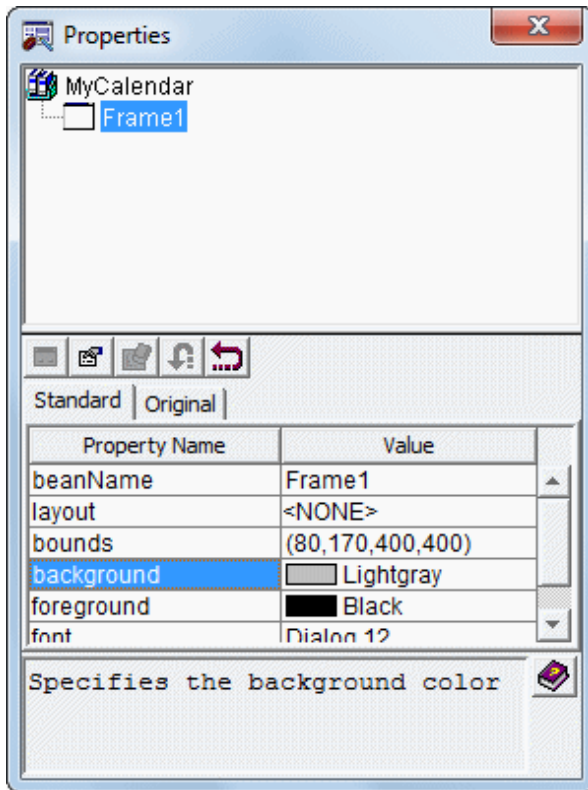
You can browse and define properties in "Property Sheet" of the Properties window.

If the Properties window is not displayed, select [View] > [Properties] from the Java Form Designer menu bar. The Properties window then is displayed.

Property Sheet lists properties of the selected Bean. If no Bean is selected, properties of the Java Form are listed.

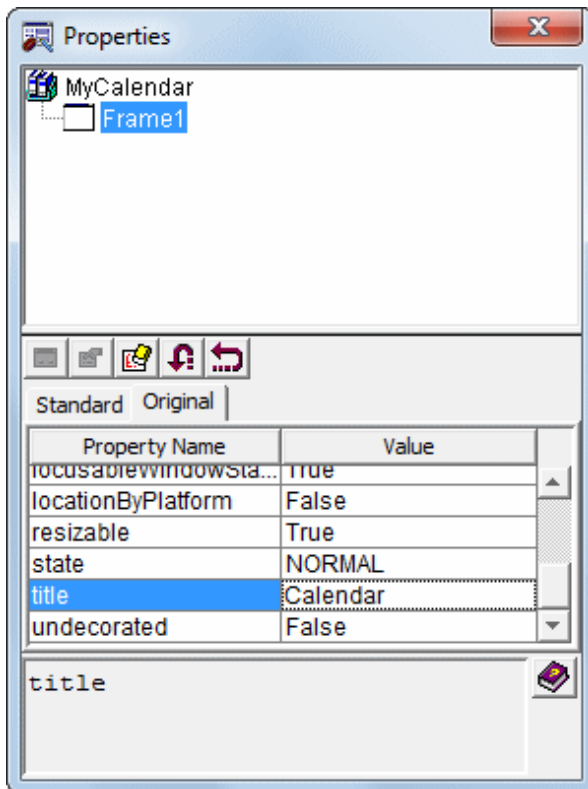
Define the properties of Frame1 as follows:

Property name	Value
layout	<NONE>
background	Lightgray



Bean information is displayed on two pages: Standard and Original. Click the [Original] tab, and specify the following:

Property name	Value
title	Calendar



Placing the Calendar Bean

1. Beans such those for a button and listbox are used to control input and output between a user and the screen displayed. This section explains how to place the JBK calendar Bean on a Java Form. Using the mouse, edit the Bean by placing, resizing, and moving it.
If [JBK] is not selected on the Object Palette, click [JBK].

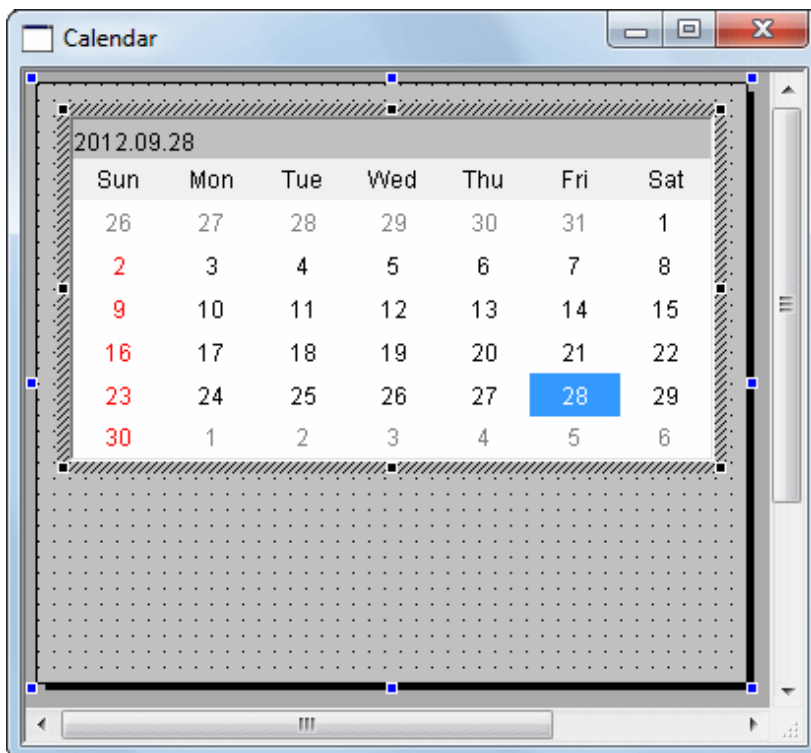
Point

Object Palette



Use the Object Palette to place components (Beans and controls) on the form. Click an icon on the palette to select a component.

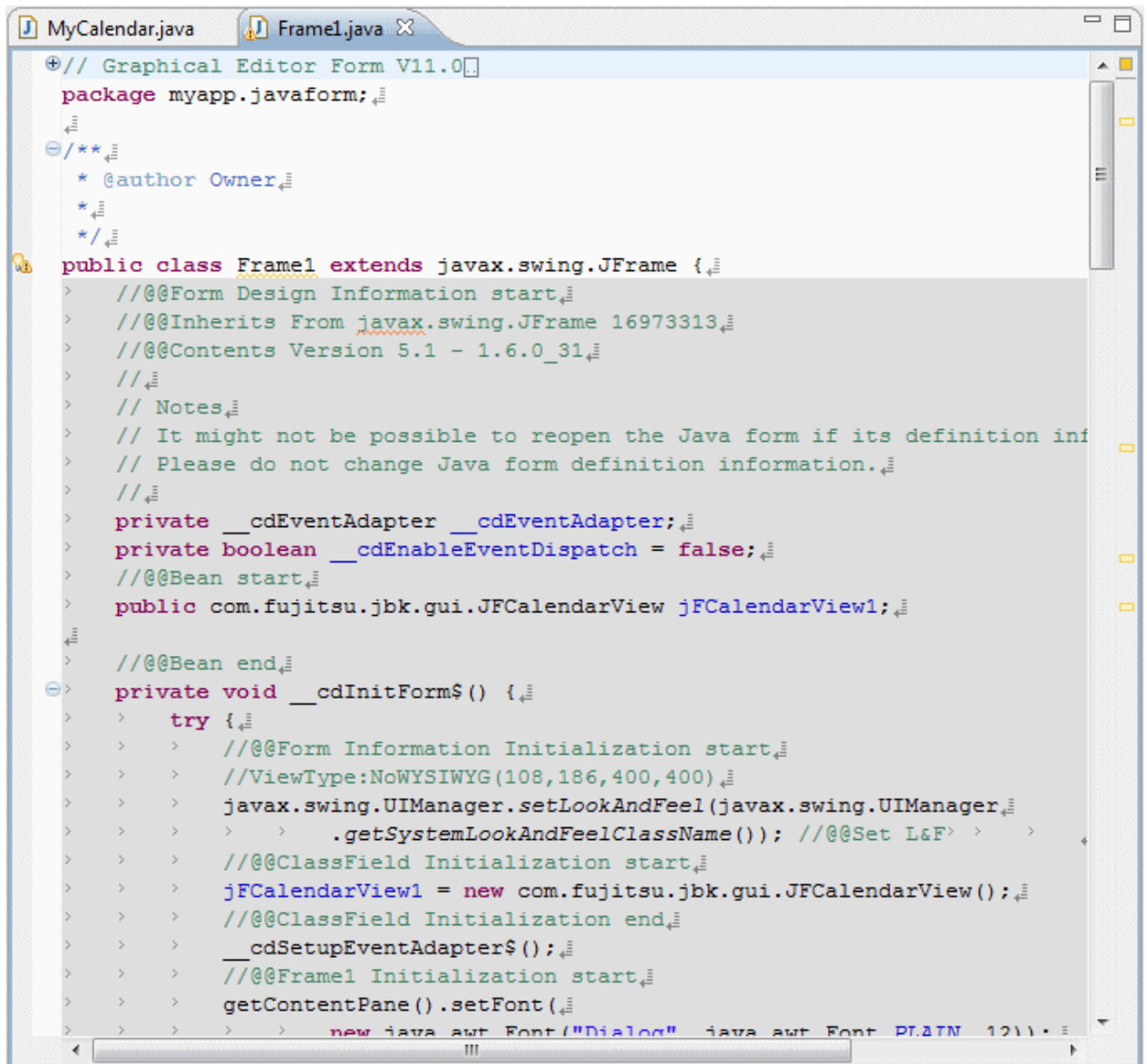
2. Click [Calendar] on the Object Palette.
3. Place the Bean on the Java Form.
Click and hold down the left mouse button at the point where you want to paste the Bean on the Java Form.
Drag the mouse and release the button of the mouse at an appropriate point. The Bean can thus be pasted on the form.



Coding an event process

1. An operation performed by clicking the mouse or using the keyboard is referred to as an event. A processing procedure coded for an event is referred to as an event process.
Code event processes in the Java editor.

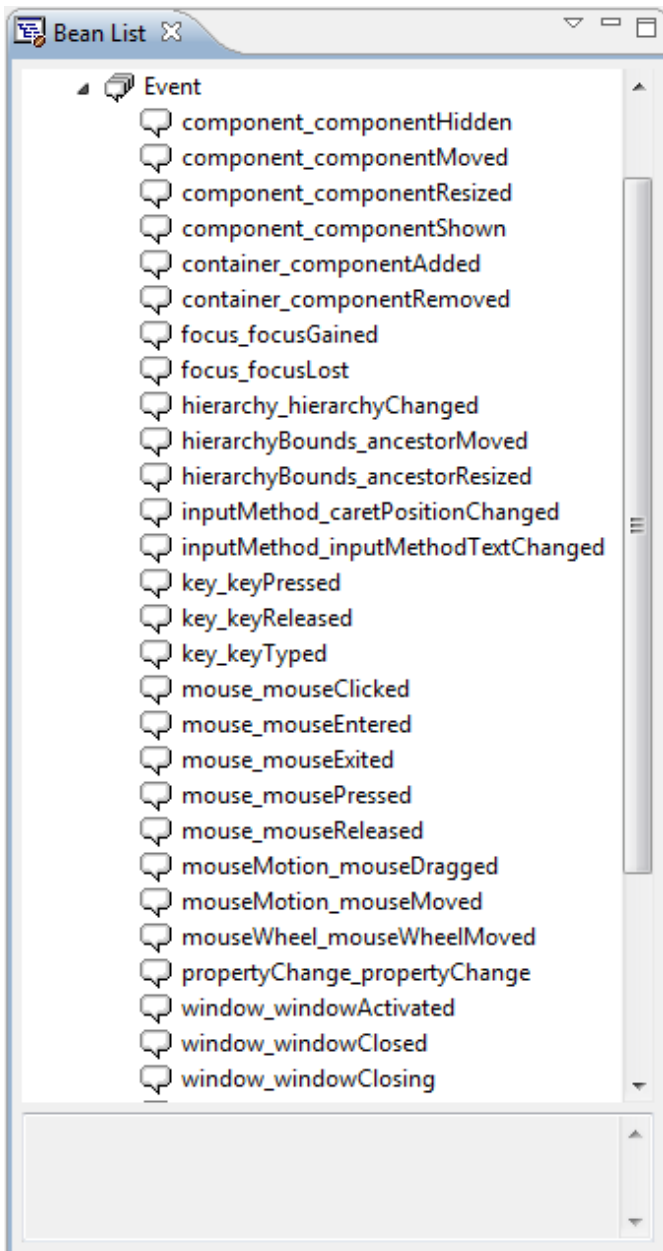
To start the Java editor, select [View] > [Java Editor] from the Java Form Designer menu bar. You can also start the Java editor by double-clicking on a Java Form or pasted Bean.



```
// Graphical Editor Form V11.0
package myapp.javaform;

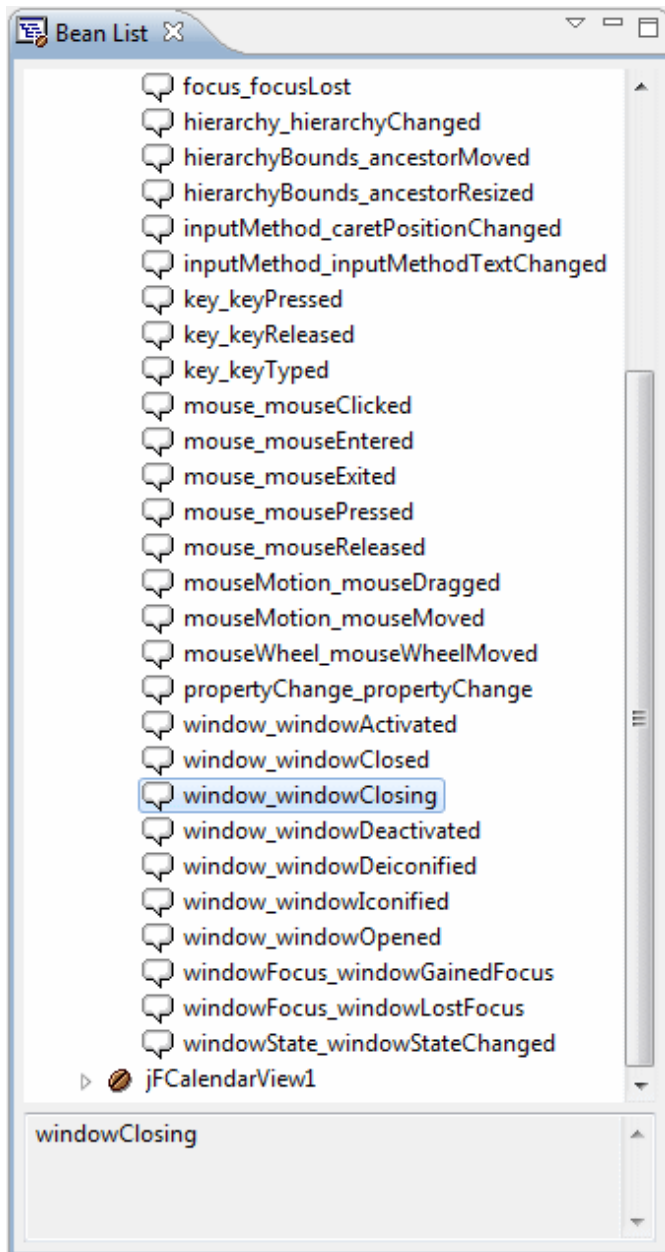
/**
 * @author Owner
 */
public class Frame1 extends javax.swing.JFrame {
    > //@@Form Design Information start
    > //@@Inherits From javax.swing.JFrame 16973313
    > //@@Contents Version 5.1 - 1.6.0_31
    > //
    > // Notes
    > // It might not be possible to reopen the Java form if its definition info
    > // Please do not change Java form definition information.
    > //
    > private __cdEventAdapter __cdEventAdapter;
    > private boolean __cdEnableEventDispatch = false;
    > //@@Bean start
    > public com.fujitsu.jbk.gui.JFCalendarView jFCalendarView1;
    >
    > //@@Bean end
    > private void __cdInitForm$() {
    >     > try {
    >         > //@@Form Information Initialization start
    >         > //ViewType:NoWYSIWYG(108,186,400,400)
    >         > javax.swing.UIManager.setLookAndFeel(javax.swing.UIManager
    >         >         > .getSystemLookAndFeelClassName()); //@@Set L&F
    >         > //@@ClassField Initialization start
    >         > jFCalendarView1 = new com.fujitsu.jbk.gui.JFCalendarView();
    >         > //@@ClassField Initialization end
    >         > __cdSetupEventAdapter$();
    >         > //@@Frame1 Initialization start
    >         > getContentPane().setFont(
    >         >         > new java.awt.Font("Dialog", java.awt.Font.PLAIN, 12));
    >     }
    > }
```

2. Display the [Bean List] view to code a process in which a specific event occurs.
Select [Window] > [Show View] > [Other] from the menu bar. The [Show View] dialog is displayed. Select [Java] > [Bean List], then click [OK].



3. Code the process such that the application ends when the frame closes.
When the frame closes, a window event occurs. In the "window_windowClosing" event, code a process corresponding to the window event.

Double-click [Frame1] > [Event] > [window_windowClosing] in the [Bean List] view.
An event process creation confirmation dialog is displayed. Click [Yes].



4. Code the text shown below in **red** in the "Frame1_window_windowClosing" event.

```
public void Frame1_window_windowClosing(java.awt.event.WindowEvent e) {  
    if(!defaultEventProc(e)) {  
        // The procedure when the event is generated is described below.  
        System.exit(0);  
    }  
}
```

Point

Change prohibition part which is managed by Java Form Designer

Java Forms include sources for calling screen (Bean) information and event processes. Because Java Form Designer is used to manage the sources, changing the sources is prohibited.

These sources that must not be changed are enclosed by comments in **blue**, as shown below.

```
public class Frame1 extends javax.swing.JFrame {
    //@Form Design Information start

    - Sources that are managed with Java Form Designer and that must not be changed -

    //@Form Design Information end
}
```

5. Save the Java Form.

Select [File] > [Save] from the workbench menu bar. Alternatively, select [File] > [Save] from the Java Form Designer menu bar. Workbench or Java Form Designer can be used to save the file.

Select [File] > [Close] from the Java Form Designer menu bar, to close the Java Form.

Building

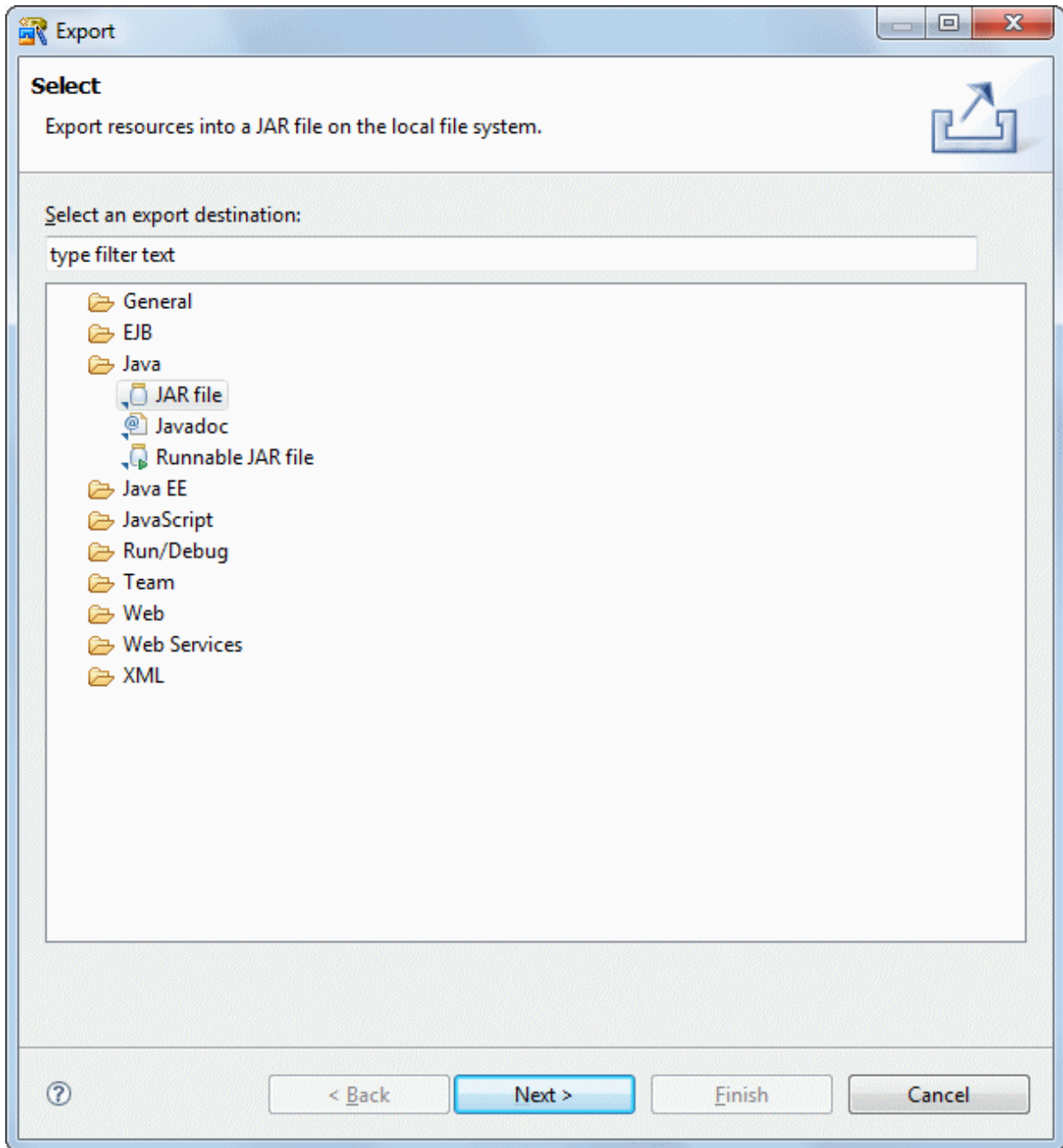
1. Building is executed automatically when resource files (such as the Java files) are saved if the [Build Automatically] item is enabled in the [Project] menu.

If [Build Automatically] is disabled, right-click on the project and select [Build Project] or select [Build Project] in the [Project] menu.

2. A JAR file is required so create it.

Use the export wizard to create the JAR file.

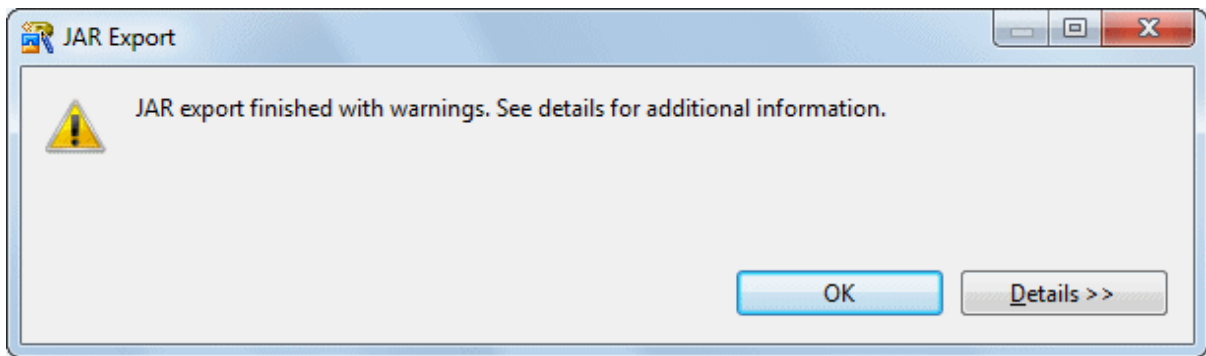
Select [File] > [Export] to start the export wizard.
 Select [Java] > [JAR file] in the export wizard.



- The JAR export wizard is displayed.
 Select [MyCalendar] > [src] folder in [Select the resources to export] and enter the following settings.
 After setting the information, click [Finish].

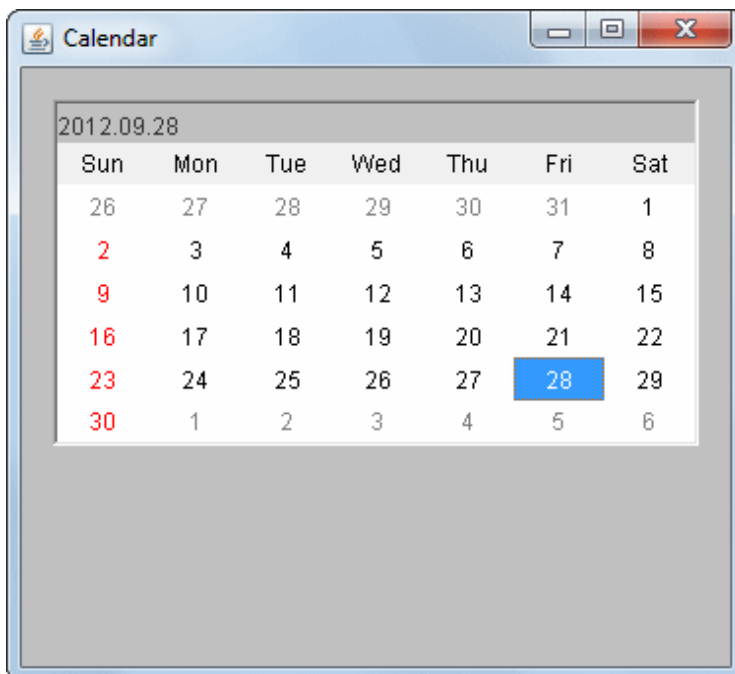
Setup Items	Setup Content
Select the resources to export	src/
Export generated class files and resources	Checked
JAR file	MyCalendar/MyCalendar.jar
Compress the contents of the JAR file	Checked

The following message is output when exporting JAR, but this does not indicate a problem.



Running

1. Select the file (class) to be executed. Click [MyCalendar] > [src] > [myapp.javaform] > [MyCalendar.java] in the [Package Explorer] view of workbench.
2. Select [Run] > [Run As] > [Java Application] from the menu bar. As a result, the application starts, and the Java Form is displayed.



3. Check the calendar Bean operations.
The calendar Bean includes functions that you can use to display a calendar and select a date.
Click a date to select it.
You can use the cursor keys or the [Page Up] and [Page Down] keys to change the selected display for each day, week, or month.
4. To close an application, click the Close [X] button on the title bar.

F.2.2 Lesson2 Input and Output Fields

This lesson describes how to add JBK input and output field Beans to the Java Form created in Lesson 1, and how to display the number of days from the current date to the day selected in the calendar. It also describes how to switch the displayed calendar according to the number of days specified in the input and output fields.

The types of input and output field Bean are as follows:

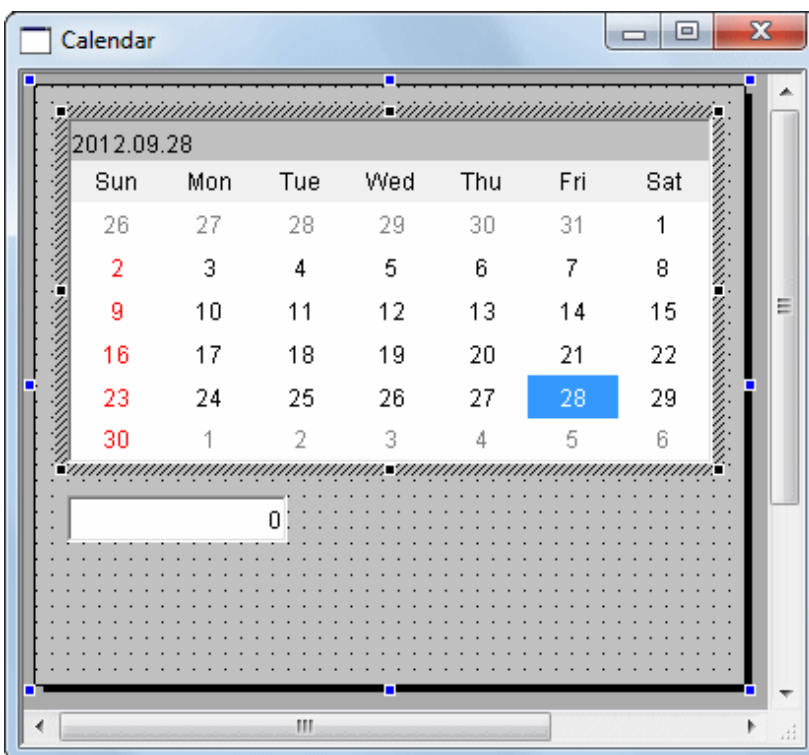
- Data/Time field (com.fujitsu.jbk.gui.JFFieldDate)
- Real number field (com.fujitsu.jbk.gui.JFFieldDouble)

- Embedded character string field (com.fujitsu.jbk.gui.JFFieldFilled)
- Integer field (com.fujitsu.jbk.gui.JFFieldLong)
- Character string field (com.fujitsu.jbk.gui.JFFieldString)

Because you want to input and output numerical values, use an integer field Bean.

Placing an integer field Bean

1. In workbench, open the "MyCalendar" project created in Lesson 1.
Open [Frame1.java] in Java Form Designer. To display the context menu, right-click [MyCalendar] > [src] > [myapp.javaform] > [Frame1.java] in the [Package Explorer] view of workbench. Then, select [Open With] > [Graphical Editor].
2. If [JBK] is not already selected on the Object Palette, click [JBK].
3. Click [Integer Field] on the Object Palette to select it.
4. Place the Bean on the Java Form by dragging it with the mouse.



Coding an event process

1. Code event processes in the Java editor.
To start the Java editor, select [View] > [Java Editor] from the Java Form Designer menu bar. You can also start the Java editor by double-clicking on a Java Form or pasted Bean.
2. Since the JDK Date class is used, add the import keyword.
The Date class is contained in the "java.util" package of JDK. Code it as follows:

```
import java.util.Date;
```

3. Calculate the number of days from the current date to the date selected in the calendar Bean, and code the process to be displayed in the integer field Bean.
Create this process as a Frame1 class method so that it can be called from other event processes.
Code the text shown below in **red**.

```
/*  
 * User definition initialization
```

```

    */
protected void initUser$ () {
    // The user definition initialization is described at this point.
}

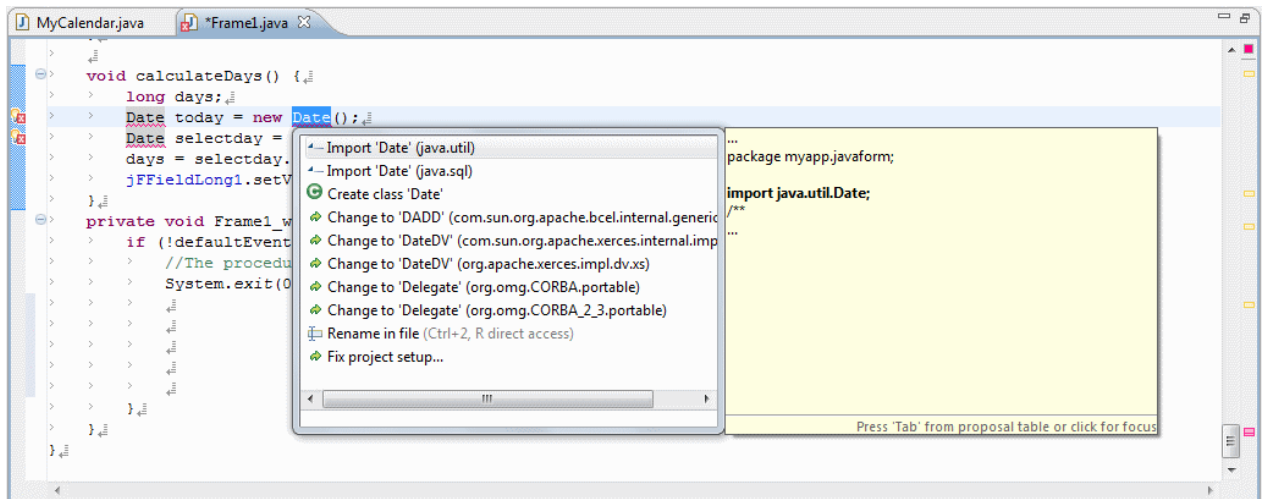
void calculateDays() {
    long days;
    Date today = new Date();
    Date selectday = jFCalendarView1.getDate().getTime();
    days = selectday.getTime() / 86400000 - today.getTime() / 86400000;
    jTextFieldLong1.setValue(days);
}

```

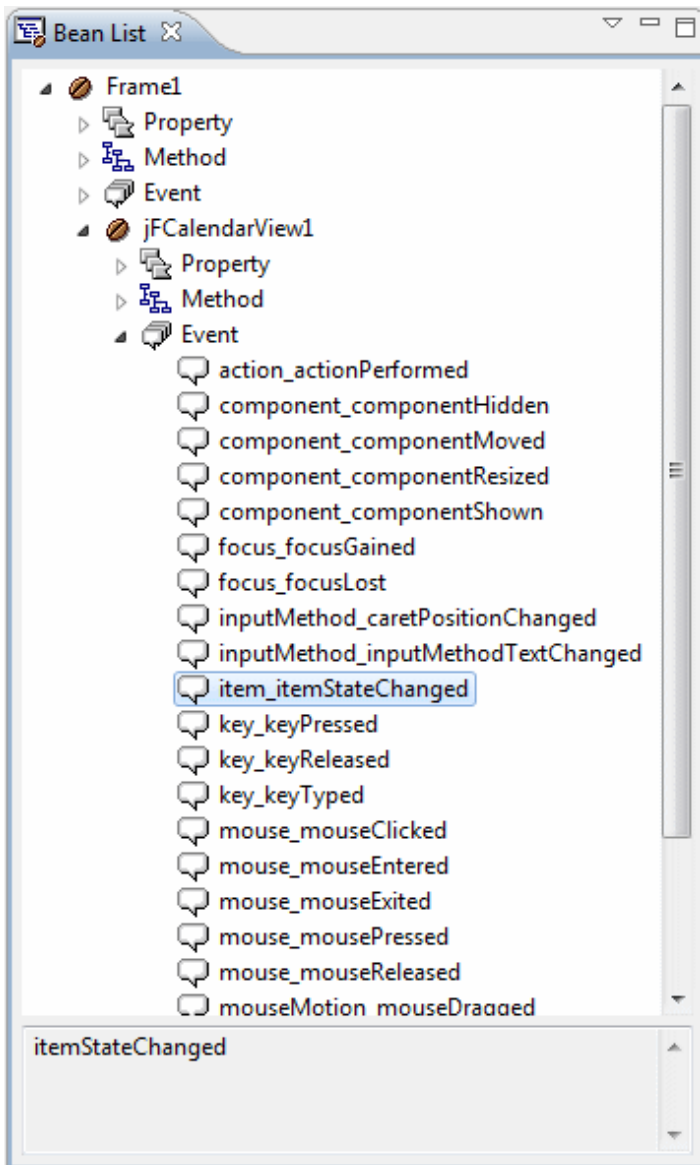
Point

Java editor code assist function

Java editor has a function that displays errors, such as invalid method descriptions and missing import keywords and classes, and proposes corrections of the errors. If an import keyword is missing in a class used, the class is indicated by a wavy line, and the electric bulb icon is displayed on the vertical ruler. Packages that can be imported and processed corrections can be displayed by clicking this electric bulb icon. Also, you can add an import keyword by selecting [Import java.util.Date] from the proposed corrections.



- Display the [Bean List] view to code a process in which a specific event occurs.
Select [Window] > [Show View] > [Other] from the menu bar. The [Show View] dialog is displayed. Select [Java] > [Bean List], then click [OK].



- Code a process that calls the calculateDays method when a calendar date is selected.
If another date has already been selected, an item event occurs. In the "item_itemStateChanged" event, code a process corresponding to the item event.
Double-click [Frame1] > [jFCalendarView1] > [Event] > [item_itemStateChanged] in the [Bean List] view.
An event process creation confirmation dialog is displayed. Click [Yes].
- Code the text shown below in **red** in the "jFCalendarView1_item_itemStateChanged" event.

```

public void jFCalendarView1_item_itemStateChanged(java.awt.event.ItemEvent e) {
    if(!defaultEventProc(e)) {
        // The procedure when the event is generated is described below.
        calculateDays();
    }
}

```

- Code the process so that when a value is entered in the integer field, calendar selection is modified according to the number of days elapsed from the current date.
An action event occurs when the [Enter] key is pressed for the active integer field. In the "action_actionPerformed\$" event, code a

process corresponding to the action event.

Double-click [Frame1] > [jFFiledLong1] > [Event] > [action_actionPerformed] in the [Bean List] view.

An event process creation confirmation dialog is displayed. Click [Yes].

8. Code the text shown below in **red** in the "jFFiledLong1_action_actionPerformed\$" event.

```
public void jFFiledLong1_action_actionPerformed$(java.awt.event.ActionEvent e) {
    if(!defaultEventProc(e)) {
        // The procedure when the event is generated is described below.
        jFCalendarView1.moveToday();
        jFCalendarView1.moveDate((int) jFFiledLong1.getValue());
    }
}
```

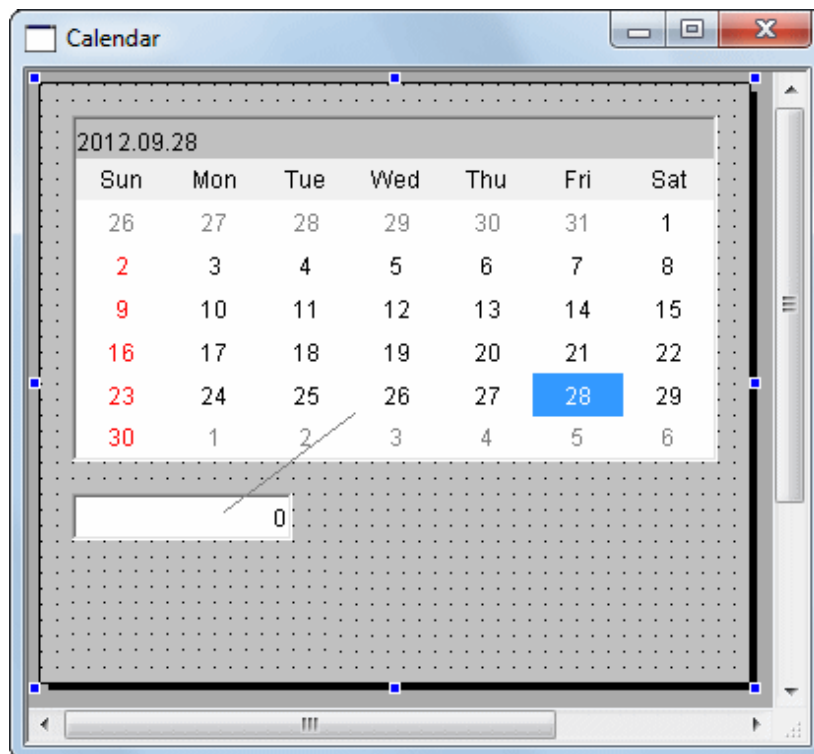
Point

You can create interactive source code using JavaBeans properties and methods.

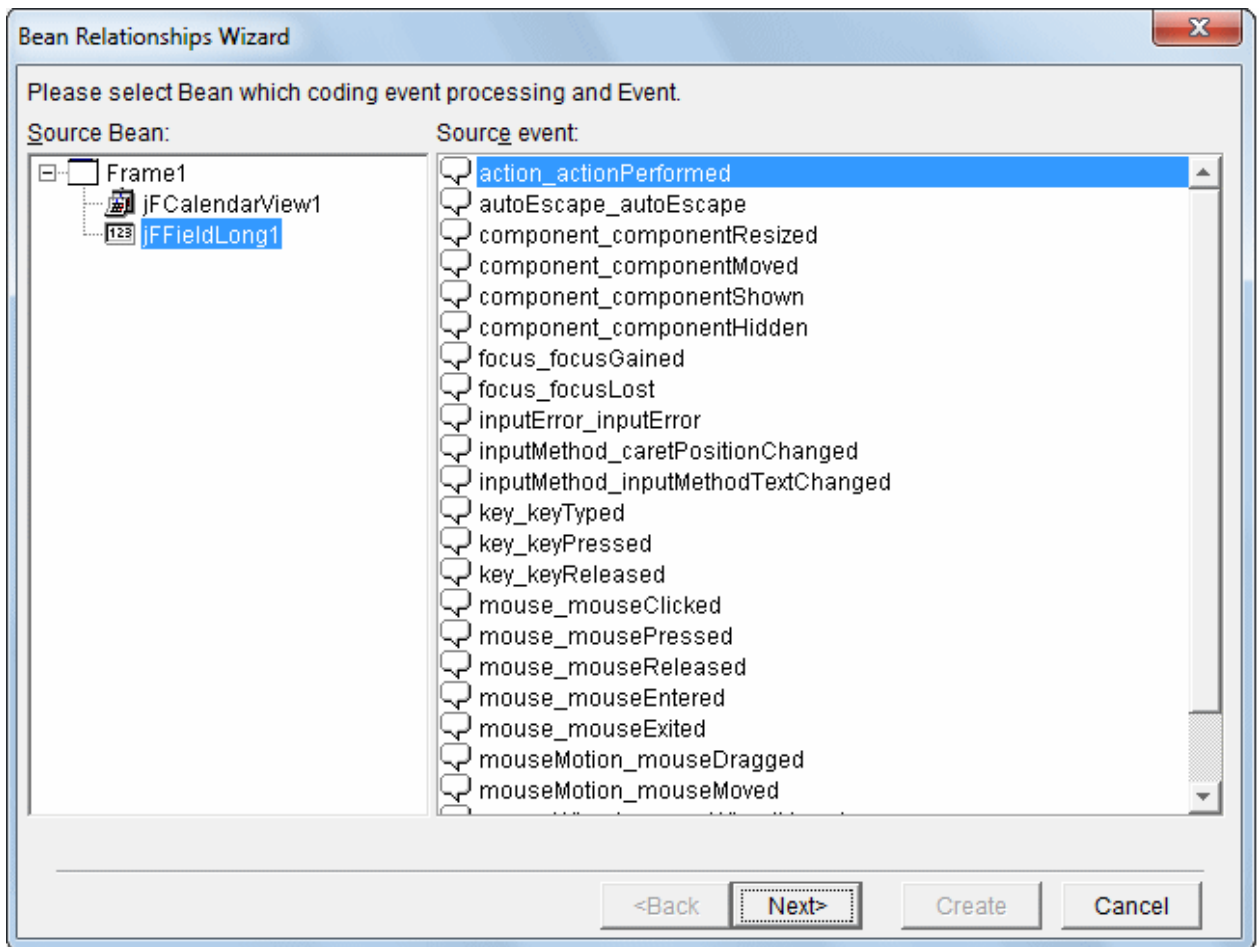
Interstage Studio provides a "Create Bean Relationships function" that supports creation of source code using JavaBeans properties and methods in event process methods. You can use this function to improve the efficiency of development of event processes.

In the above example, the text in **red** was coded in the Java editor. However, the Create Bean Relationships function can be used because "**jFCalendarView1.moveToday();**" is the JavaBeans method call source code in the event process method.

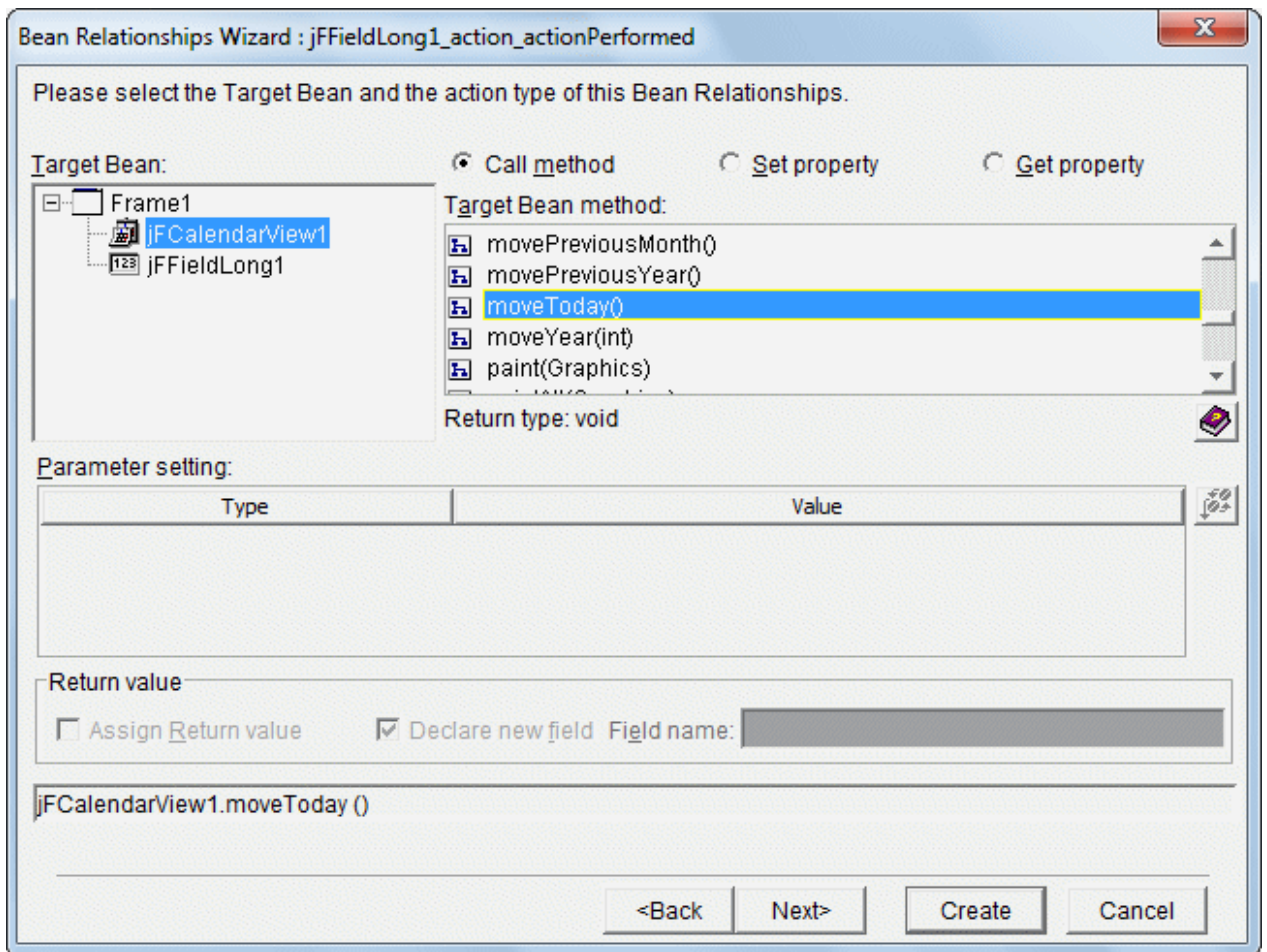
Click and hold down the right mouse button at jFFiledLong1, drag jFFiledLong1 to jFCalendarView1, and then release the button at an appropriate point. This operation is known as wiring.



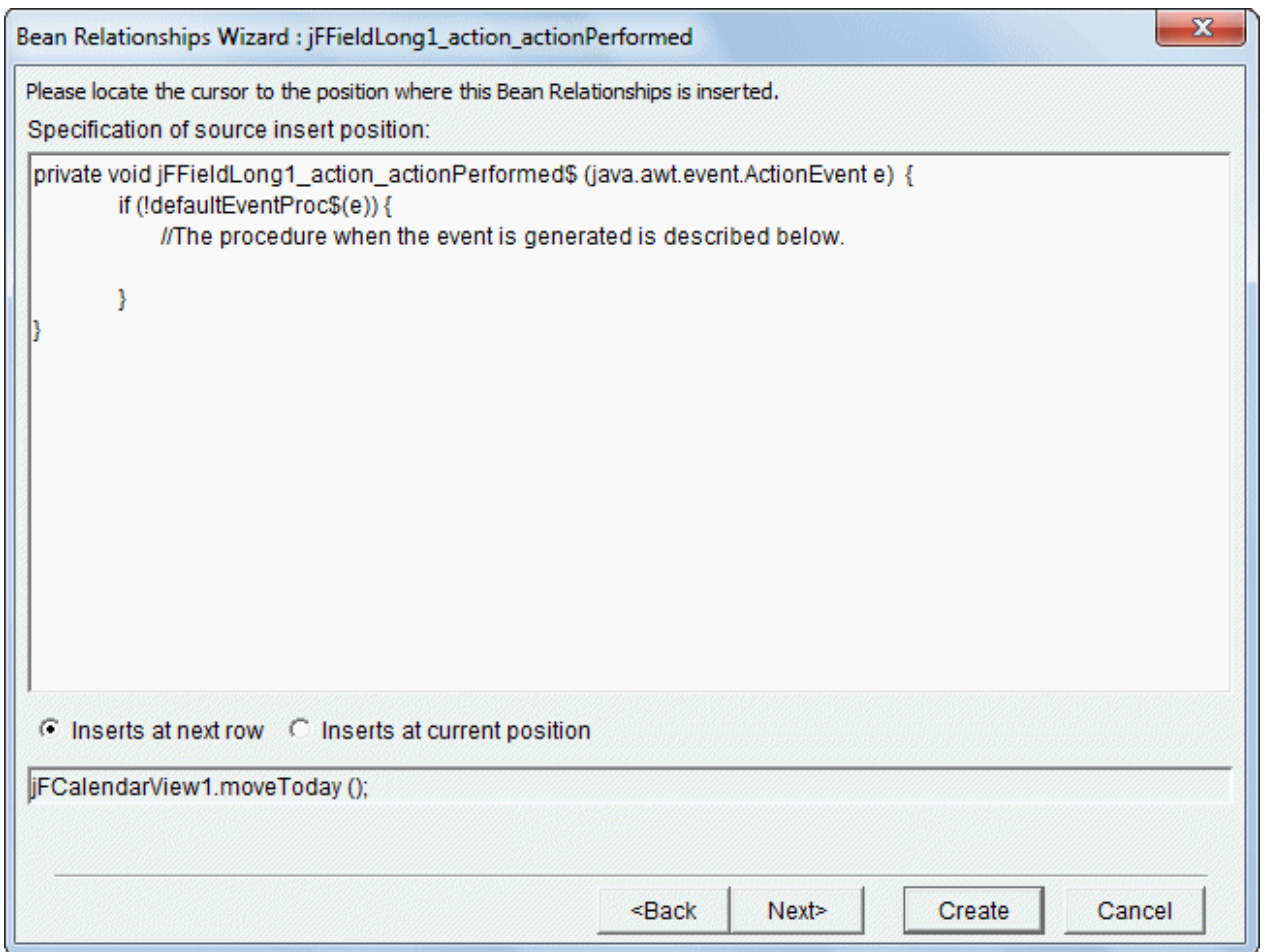
The Bean Relationships Wizard is displayed. On the Source Bean list, the Bean that was the starting point of wiring is already selected. From the Source event list, select [action_actionPerformed] corresponding to the action event, then click [Next].



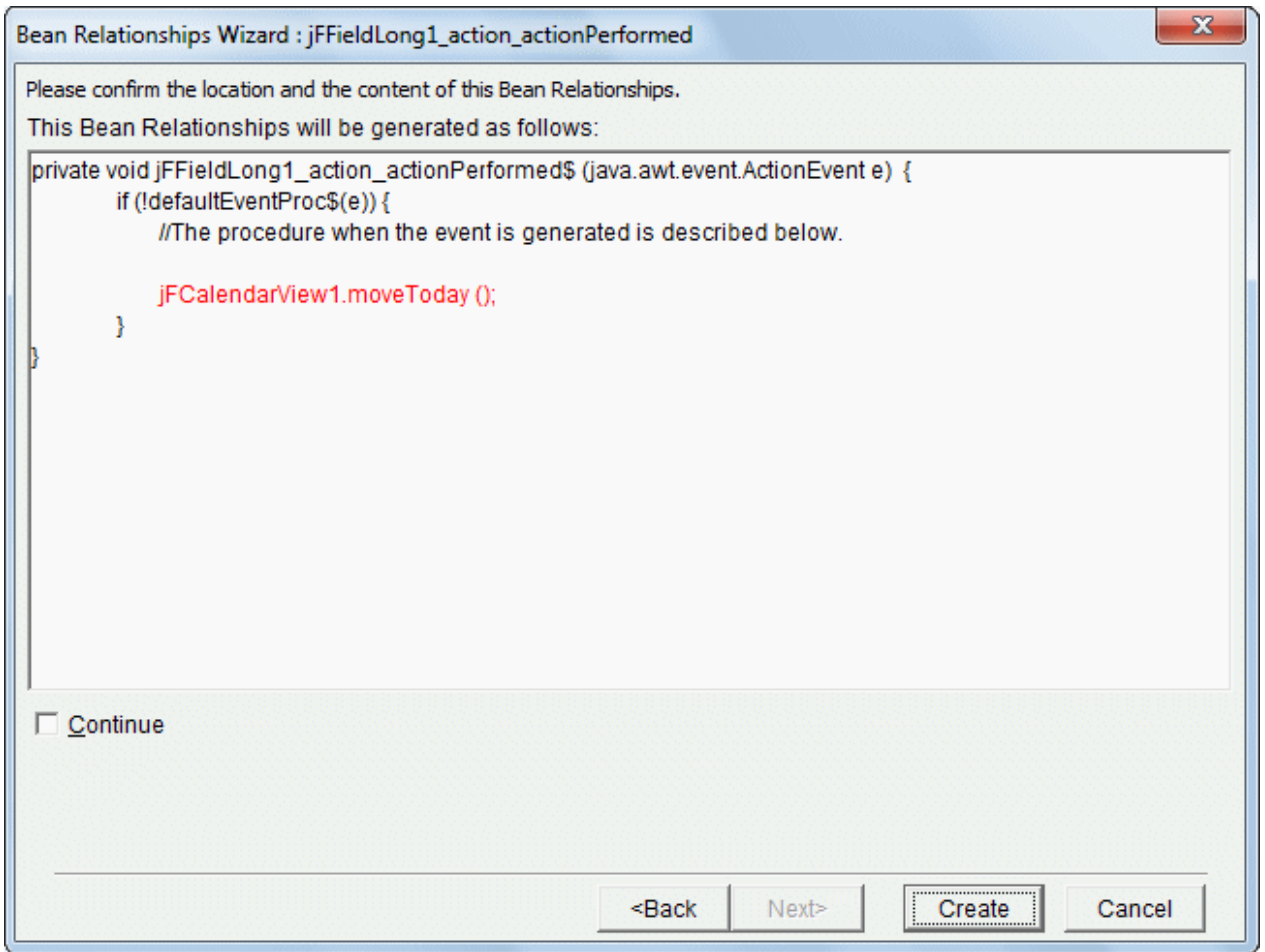
On the Target Bean list, the Bean that was the end point of wiring is already selected. Select [Call method] as the Bean Relationships type, select [moveToday()] from the Target Bean method list, then click [Next].



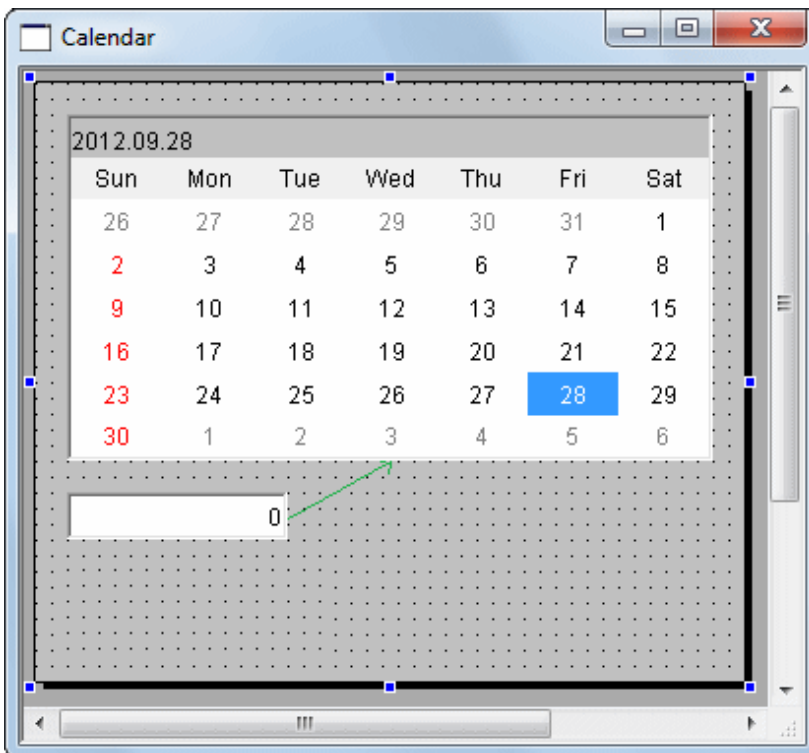
Select the location at which to insert source code, then click [Next].



The entire event process method with the inserted source code is displayed. Text in **red** is source code created by the wizard. Verify the contents of the source and the insertion point, then click [Create].



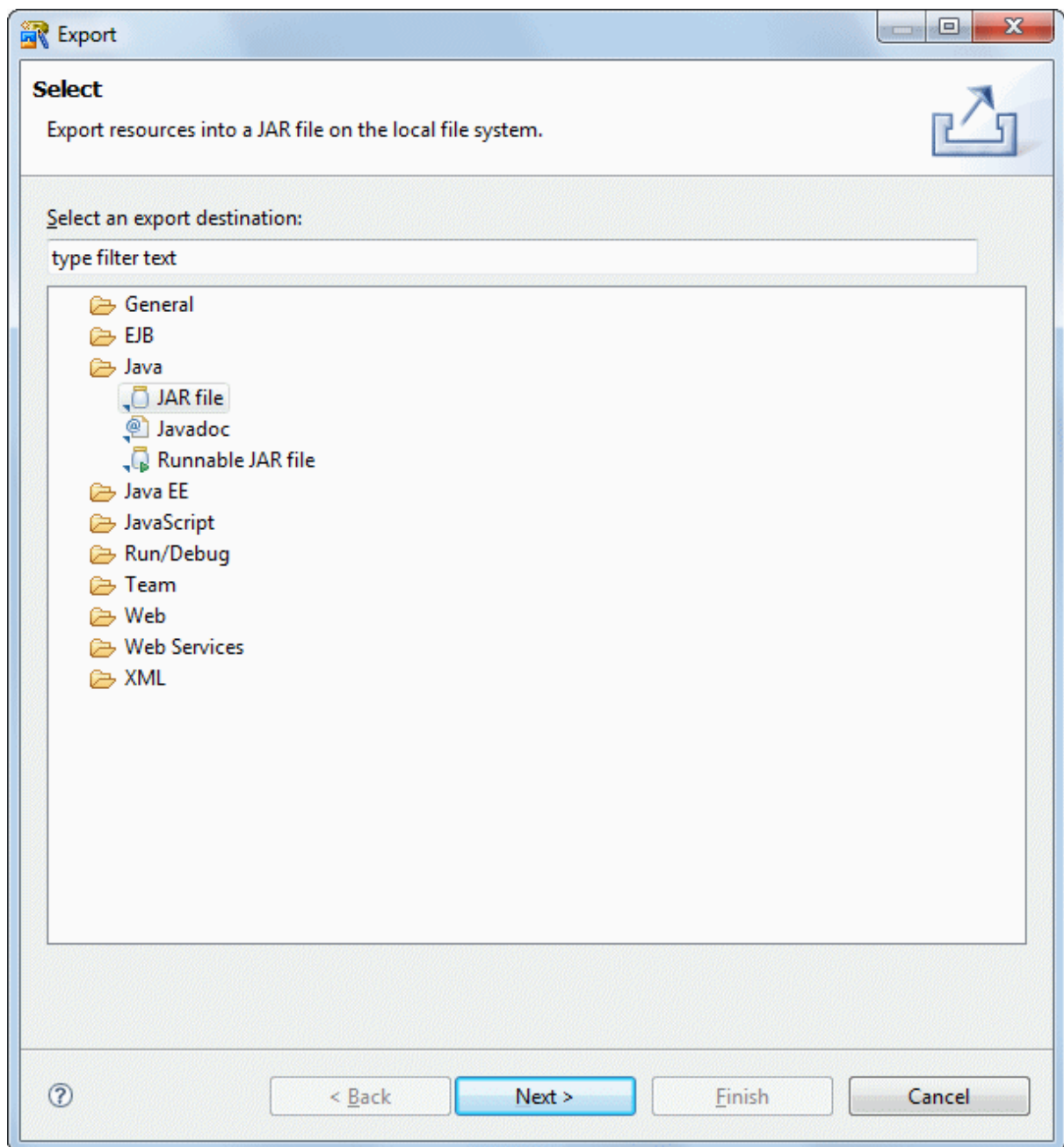
Source code using JavaBeans properties and methods in event process methods is represented on the screen by lines linking Beans.



9. Save the Java Form.
Select [File] > [Save] from the workbench menu bar.
To close the Java Form, select [File] > [Close] from the Java Form Designer menu bar.

Building

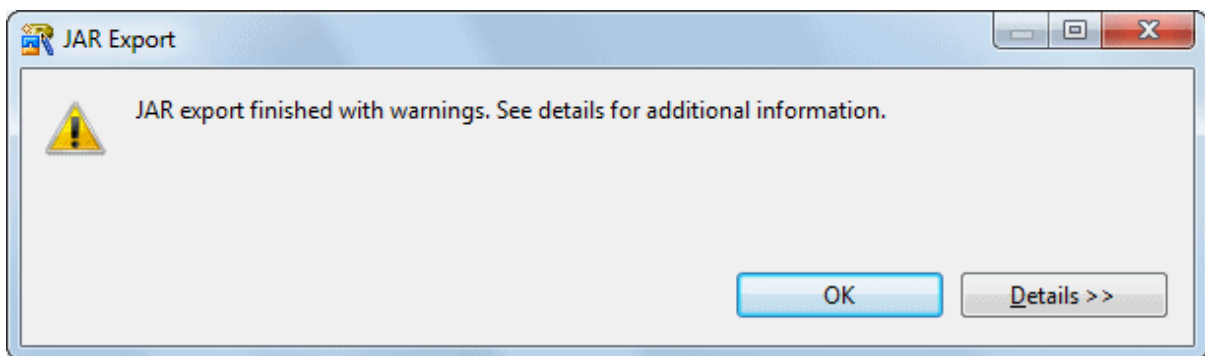
1. Building is executed automatically when resource files (such as the Java files) are saved if the [Build Automatically] item is enabled in the [Project] menu.
If [Build Automatically] is disabled, right-click on the project and select [Build Project] or select [Build Project] in the [Project] menu.
2. A JAR file is required so create it.
Use the export wizard to create the JAR file.
Select [File] > [Export] to start the export wizard.
Select [Java] > [JAR file] in the export wizard.



- The JAR export wizard is displayed.
Select [MyCalendar] > [src] folder in [Select the resources to export] and enter the following settings.
After setting the information, click [Finish].

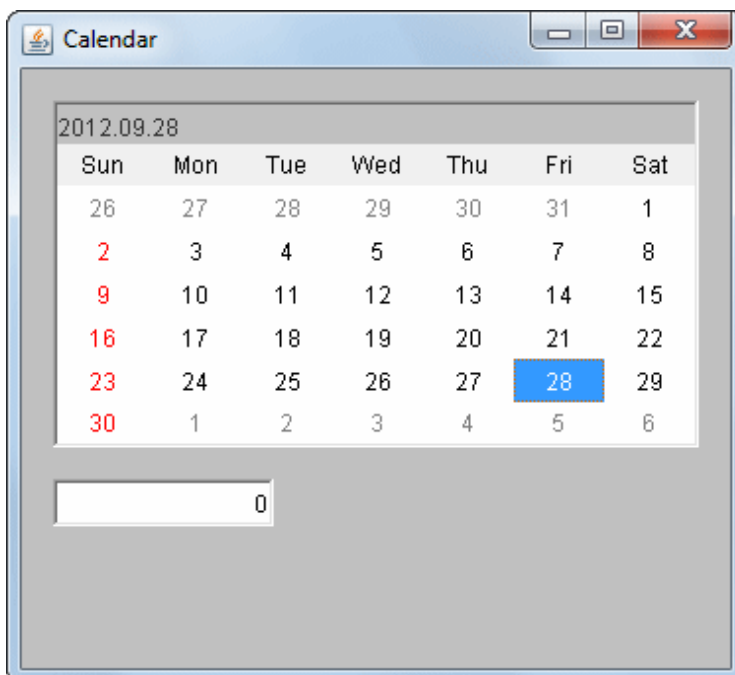
Setup Items	Setup Content
Select the resources to export	src/
Export generated class files and resources	Checked
JAR file	MyCalendar/MyCalendar.jar
Compress the contents of the JAR file	Checked

The following message is output when exporting JAR, but this does not indicate a problem.

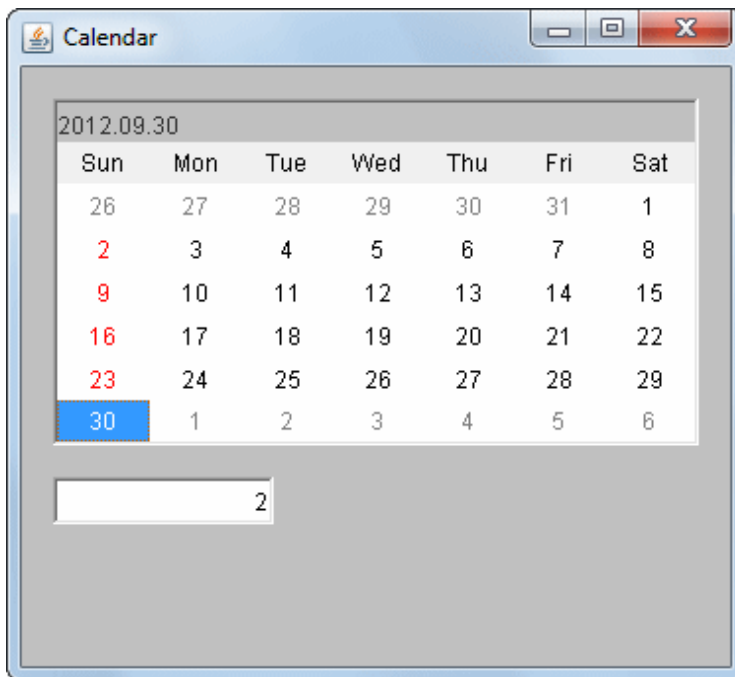


Running

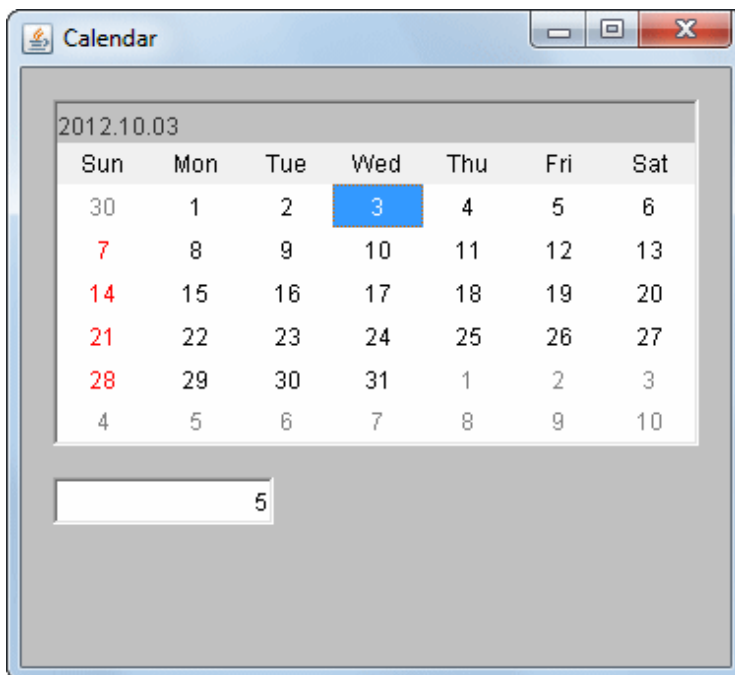
- Select the file (class) to be executed. Click [MyCalendar] > [src] > [myapp.javaform] > [MyCalendar.java] in the [Package Explorer] view of workbench.
- Select [Run] > [Run As] > [Java Application] from the menu bar. As a result, the application starts, and the Java Form is displayed.
- Click a date other than the selected date.
The number of elapsed days is displayed in the integer field.



Select two days later.



4. Enter a number in the integer field, and press the [Enter] key.
Confirm that the calendar display changes.
Enter "5" in the integer field.



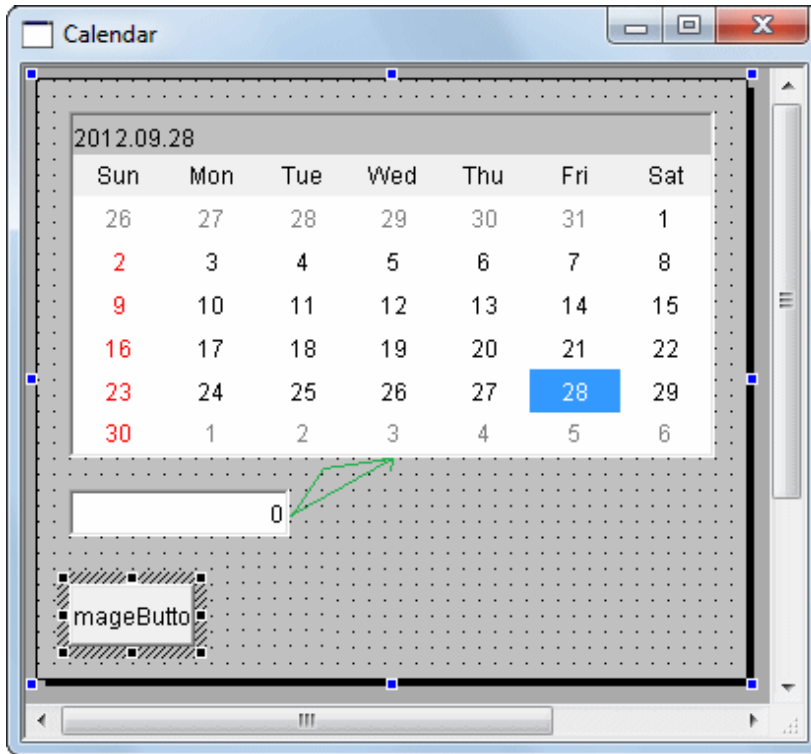
5. Close the application by clicking the Close [X] button on the title bar.

F.2.3 Lesson3 Buttons

This lesson describes how to add a JBK image button Bean to the Java Form created in Lesson 2, and how to use this button so that calendar display can be changed.

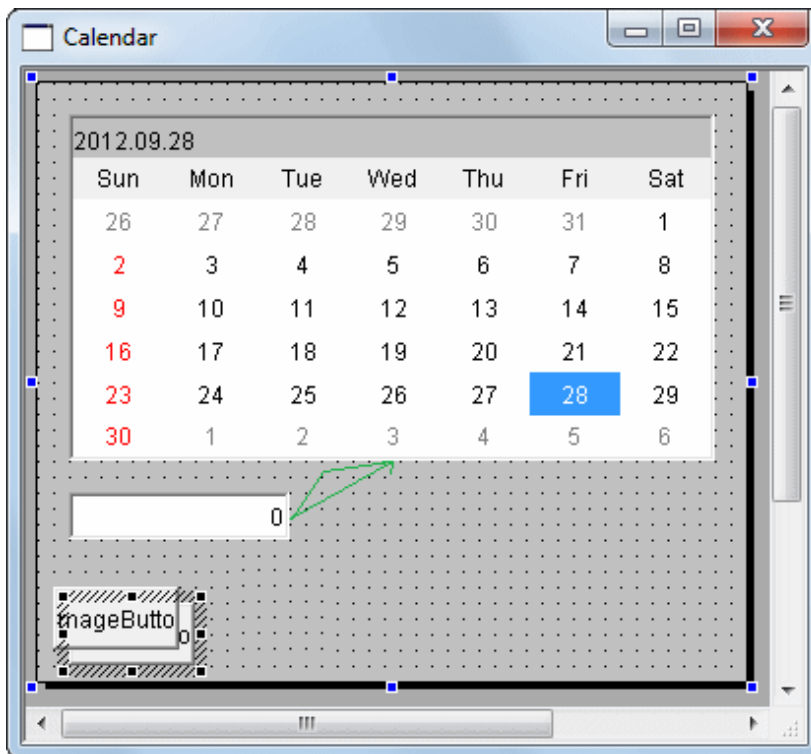
Placing a button Bean

1. In workbench, open the "MyCalendar" project created in Lesson 2.
Open [Frame1.java] in Java Form Designer. To display the context menu, right-click [MyCalendar] > [src] > [myapp.javaform] > [Frame1.java] in the [Package Explorer] view of workbench. Then, select [Open With] > [Graphical Editor].
2. If [JBK] is not already selected on the Object Palette, click [JBK] to select it.
3. Click [Image button] on the Object Palette to select it.
4. Place the Bean on the Java Form by dragging it with the mouse.

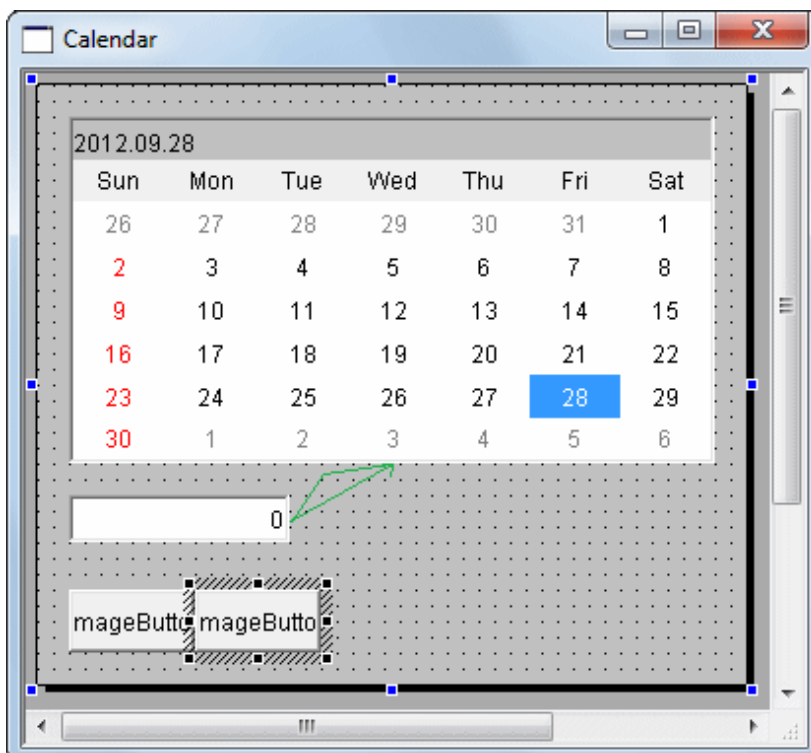


With the Bean selected, select [Edit] > [Copy] from the Java Form Designer menu bar.

Select [Edit] > [Paste] from the menu bar.



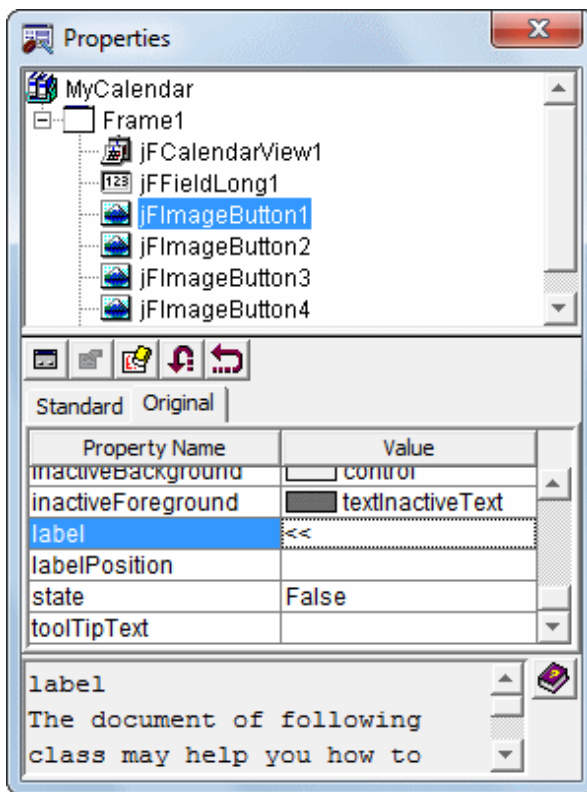
Drag the pasted Bean to change its location.



Repeat the procedure until you have added five button Beans. Modify the label properties and code event processes as follows:

Bean Name	Label Property Value	action Event (action_actionPerformed) Process
jFImageButton1	<<	jFCalendarView1.moveYear(-1); calculateDays();
jFImageButton2	<	jFCalendarView1.moveMonth(-1); calculateDays();
jFImageButton3	Today	jFCalendarView1.moveToday(); calculateDays();
jFImageButton4	>	jFCalendarView1.moveMonth(1); calculateDays();
jFImageButton5	>>	jFCalendarView1.moveYear(1); calculateDays();

Modify properties on the property sheet. For jFImageButton1, the procedure for modifying text properties is as follows:

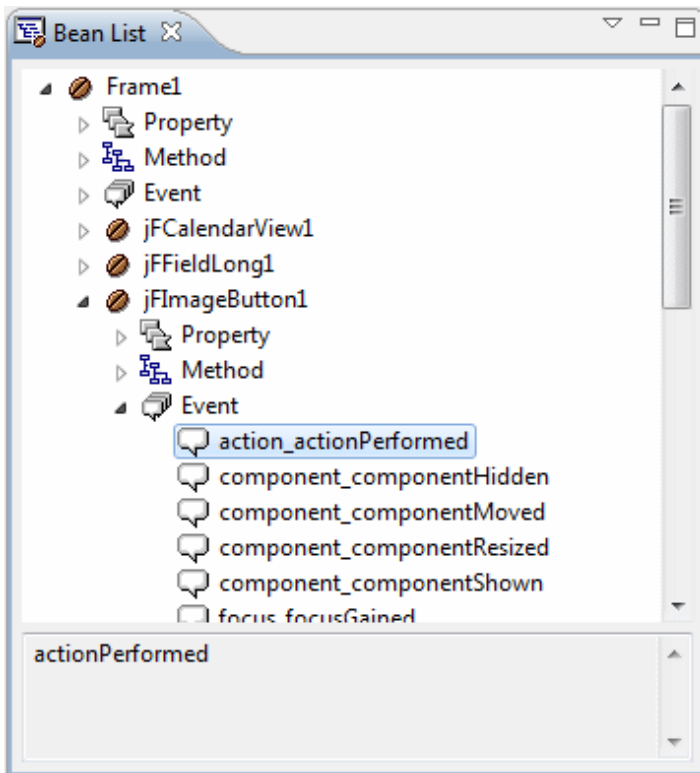


Code the event process in "action_actionPerformed" for each button in the Java editor.

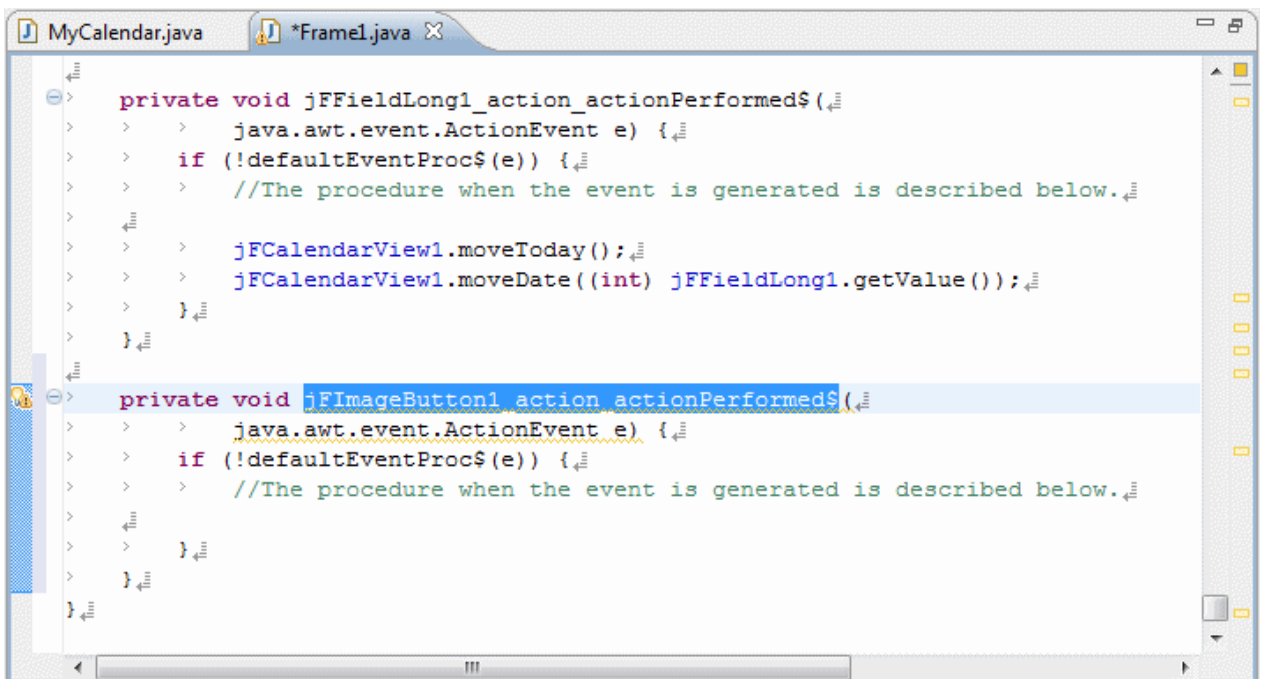
To code the process in "action_actionPerformed" for jFImageButton1, double-click [Frame1] > [jFImageButton1] > [Event] >

[action_actionPerformed] in the [Bean List] view.

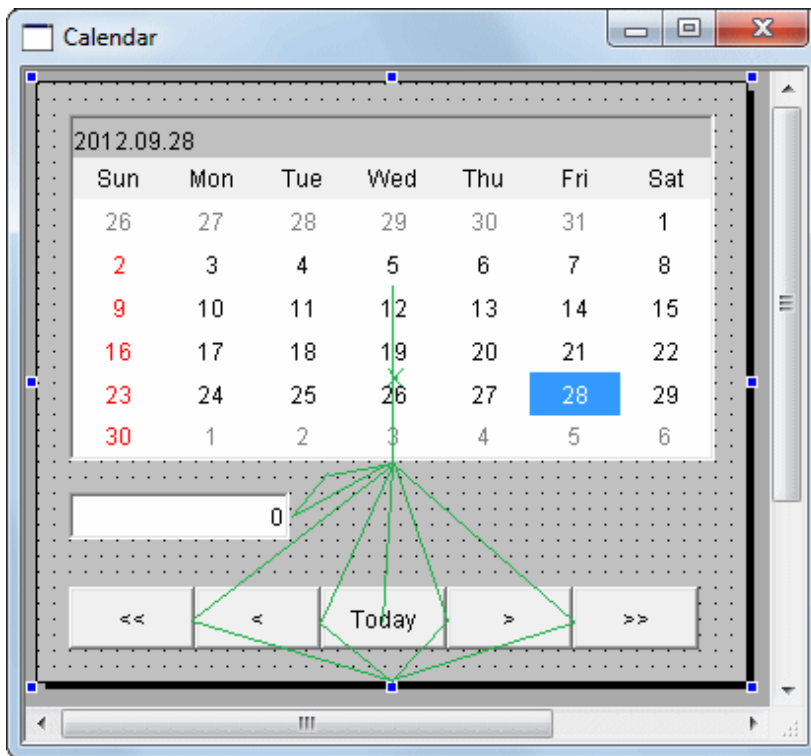
When an event process creation confirmation dialog is displayed, click [Yes].



The processing procedure of the "jFImageButton1_action_actionPerformed\$" event is displayed. Enter the procedure according to the above table.



When you have placed the button Beans and modified their label properties, the Java Form has the following appearance:



5. Save the Java Form.

Select [File] > [Save] from the Java Form Designer menu bar.

To close the Java Form, select [File] > [Close] from the Java Form Designer menu bar.

Building

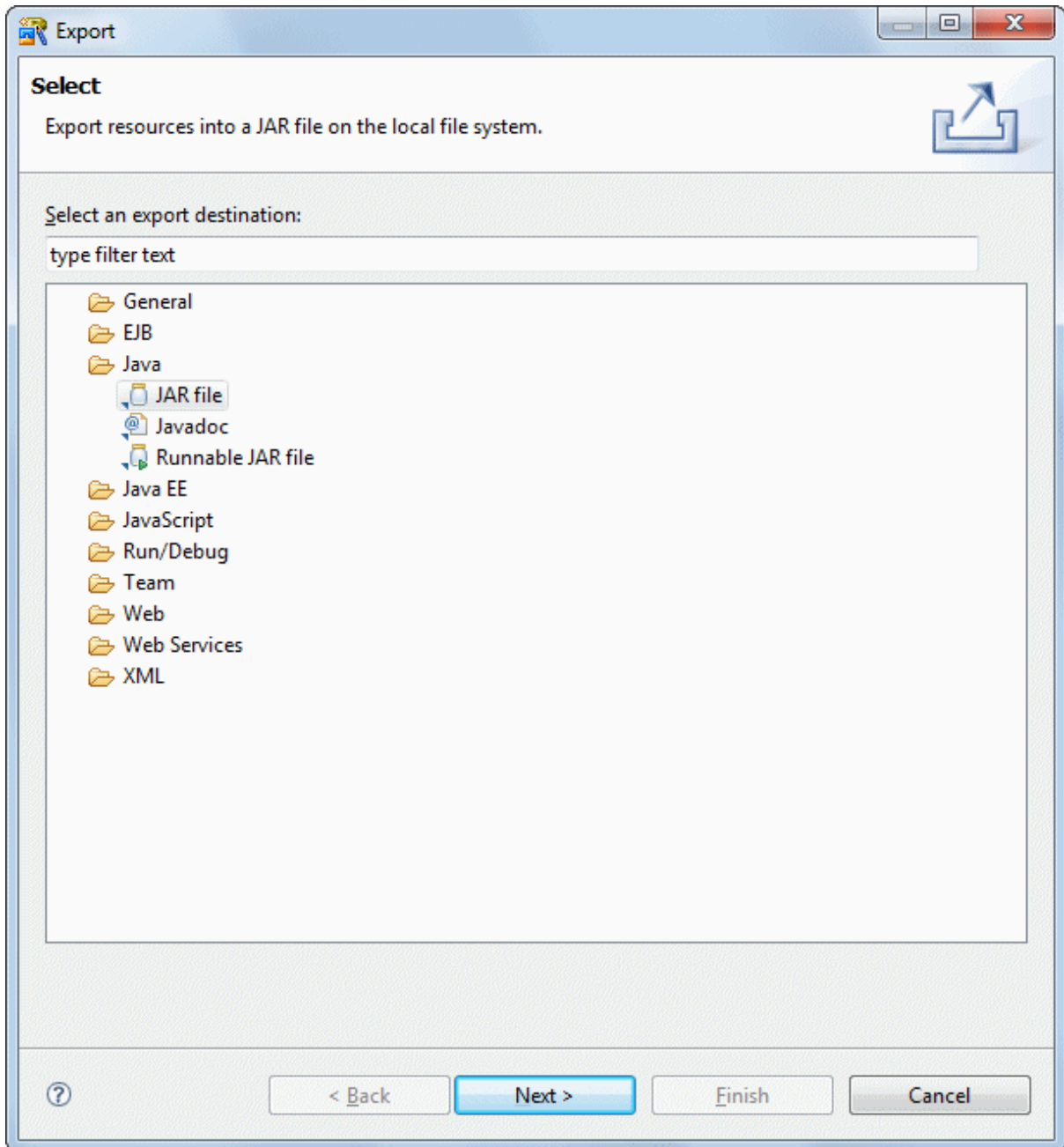
1. Building is executed automatically when resource files (such as the Java files) are saved if the [Build Automatically] item is enabled in the [Project] menu.

If [Build Automatically] is disabled, right-click on the project and select [Build Project] or select [Build Project] in the [Project] menu.

2. A JAR file is required so create it.

Use the export wizard to create the JAR file.

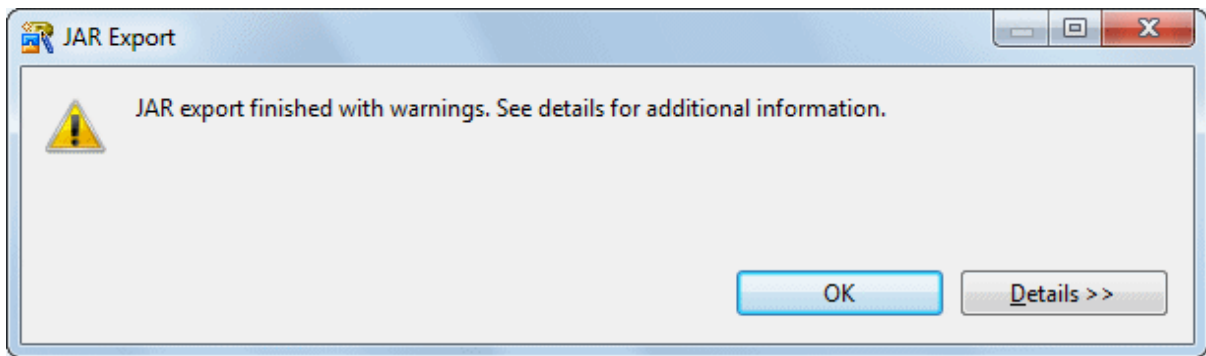
Select [File] > [Export] to start the export wizard.
 Select [Java] > [JAR file] in the export wizard.



- The JAR export wizard is displayed.
 Select [MyCalendar] > [src] folder in [Select the resources to export] and enter the following settings.
 After setting the information, click [Finish].

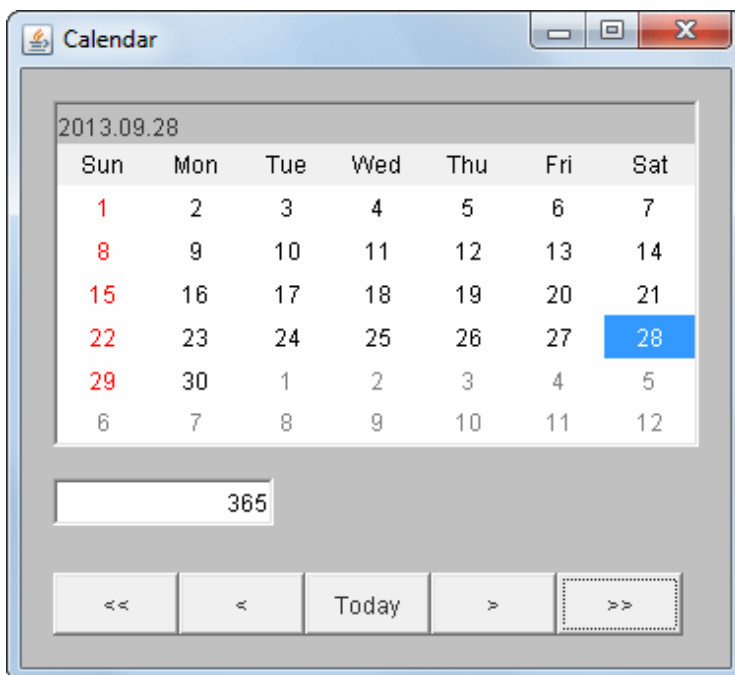
Setup Items	Setup Content
Select the resources to export	src/
Export generated class files and resources	Checked
JAR file	MyCalendar/MyCalendar.jar
Compress the contents of the JAR file	Checked

The following message is output when exporting JAR, but this does not indicate a problem.



Running

1. Select the file (class) to be executed. Click [MyCalendar] > [src] > [myapp.javaform] > [MyCalendar.java] in the [Package Explorer] view of workbench.
2. Select [Run] > [Run As] > [Java Application] from the menu bar. As a result, the application starts, and the Java Form is displayed.
3. Click the added [<<], [<], [Today], [>], and [>>].
The calendar changes according to the button you click.
The following shows the results of clicking [>>] when "Today" is displayed.



4. Close the application by clicking the Close [X] button on the title bar.

F.2.4 Lesson4 Dialog

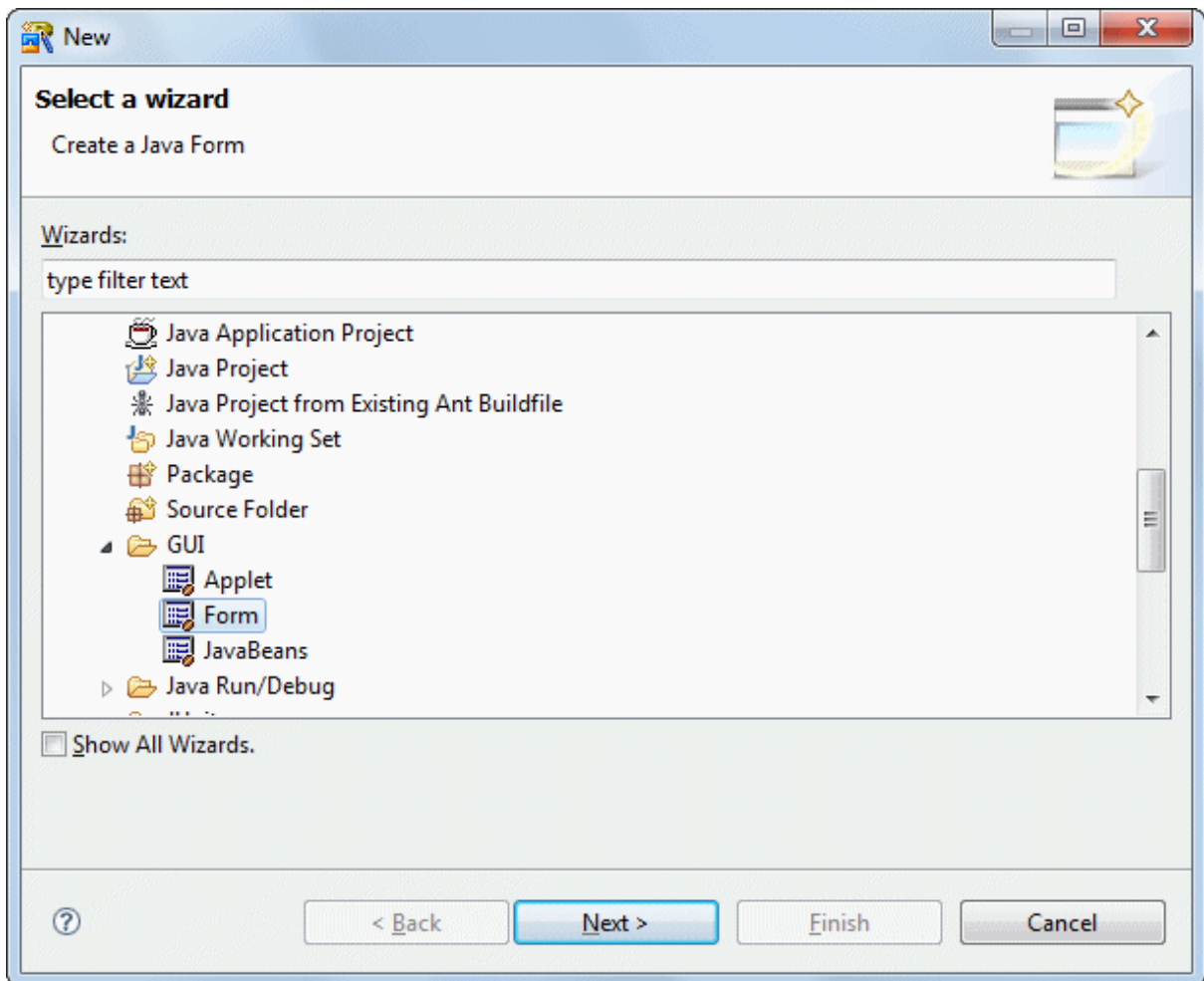
Dialog is the auxiliary screen using for input of information and display of message etc.

The calendar Bean has a function for setting a memo that is displayed as a tool tip for a single date. This lesson describes how to implement a screen to enter a memo for a date, using a dialog.

The section also describes how to specify settings so that a dialog for entry of a memo is displayed when a date on the calendar is double-clicked. The dialog is added to the project created in Lesson 3.

Creating a dialog

1. In workbench, open the "MyCalendar" project created in Lesson 3.
Select [File] > [New] > [Other] from the menu bar.
2. The [New] wizard is displayed. Select [Java] > [GUI] > [Form] from the tree.



Click [Next].

3. The [Java Form Information] page is displayed. Enter information as follows.

Setup Items	Setup Content
Package	myapp.javaform

New Form

Java Form Information

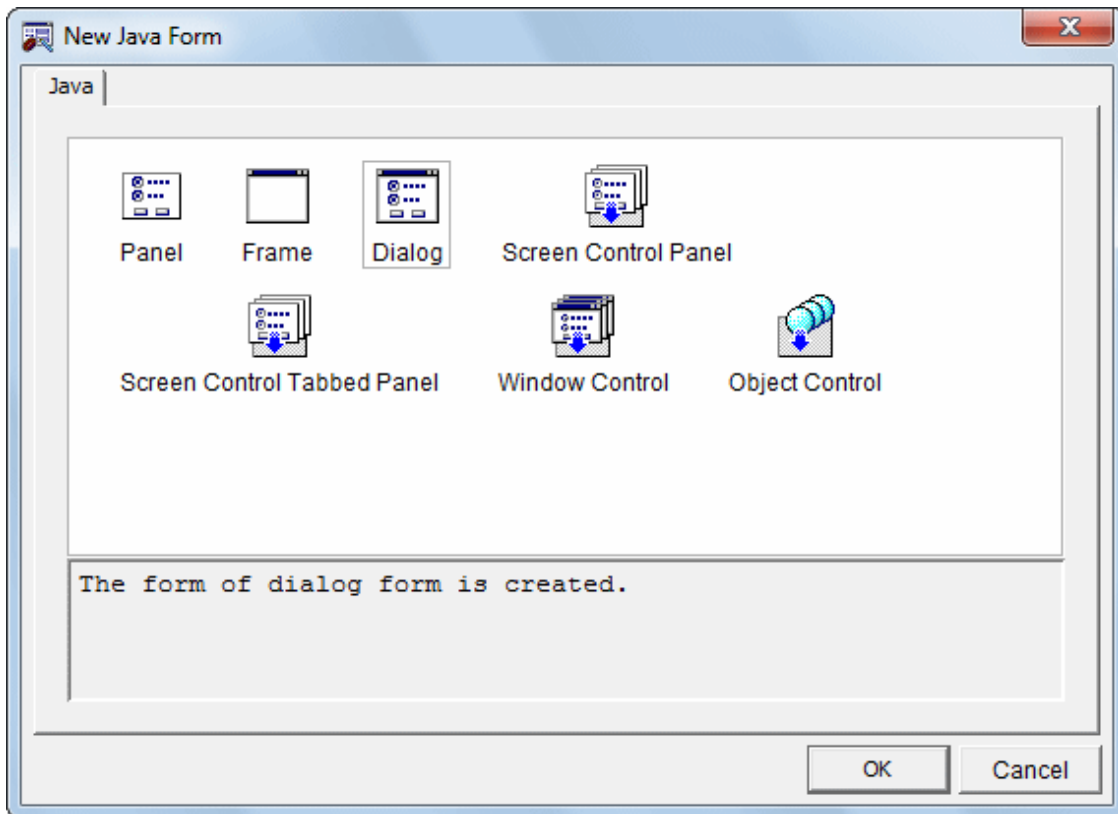
Please input the Java Form information.

Source Folder:

Package:

Click [Next].

- The [New Java Form] dialog box is displayed. Select [Dialog]. Click [OK].



Point

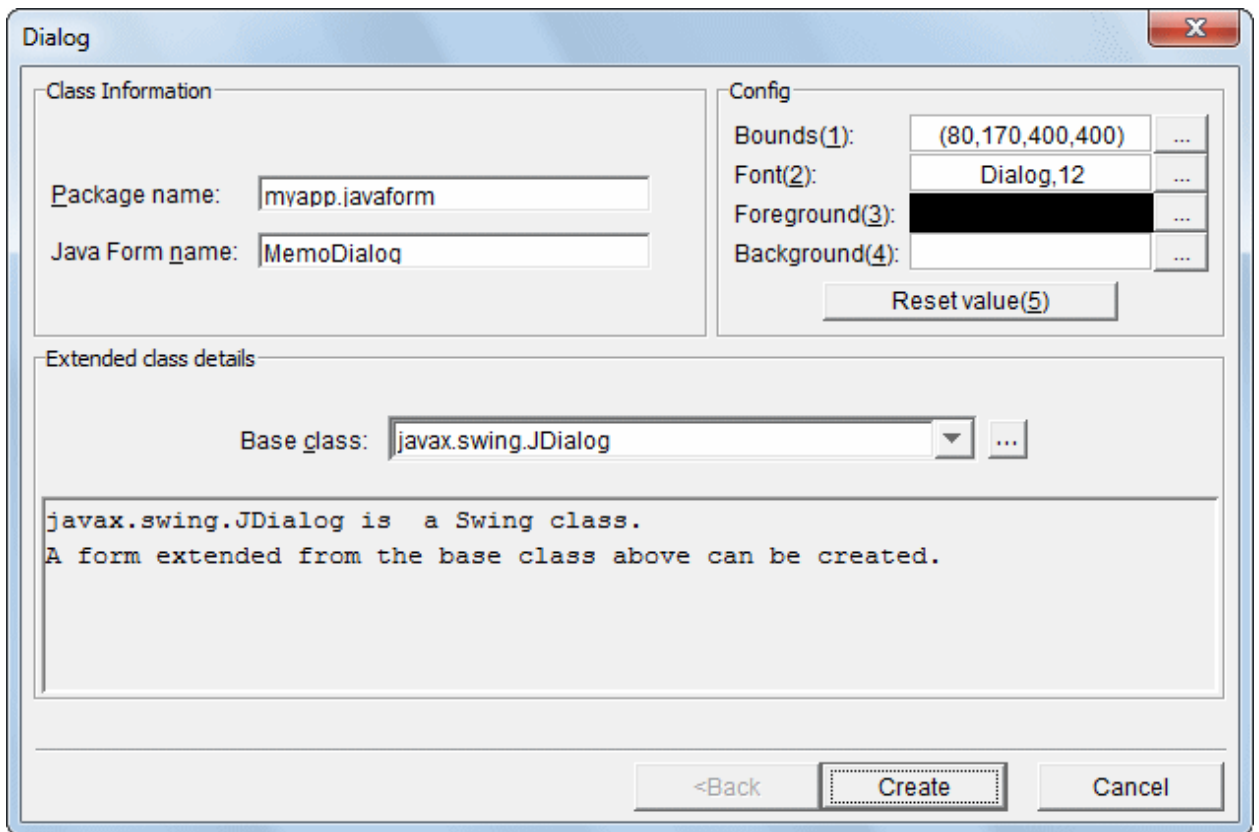
About Java Form types

The three types of Java Form are listed below. For each type, there is a class extended form. Note that form layouts that are often used can be registered in advance. To use such forms, click the [Wizard] tab, and select the type of form to be automatically generated.

- Frame
A form may contain a general screen frame consisting of pulldown menus, title bars, and other parts. Such a form is used as the basic initial screen of an application.
- Dialog
Although similar to a frame, a dialog has certain differences from a frame: for example, a dialog can be made modal, and it cannot contain a title bar icon. A dialog works as an auxiliary to a frame. It is displayed temporarily for user input and output of information.
- Panel
A panel is a form without a frame. One or more panels can be pasted into frames and dialogs. Two or more Beans are pasted into a panel to create a compound GUI part.

- Create a new System Dialog.
The [Dialog] page is displayed. Enter information as follows.

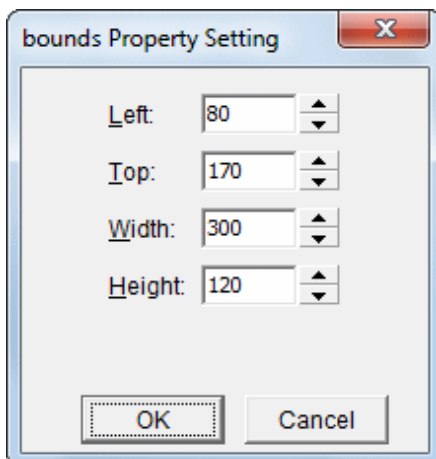
Setup Items	Setup Content
Package name	myapp.javaform
Java Form name	MemoDialog
Base class	javax.swing.JDialog



To set up Bounds, click [...].

The [bounds Property Setting] dialog is displayed. Specify the following properties, then click [OK].

Property name	Value
Width	300
Height	120



Click [Create].

Dialog

Class Information

Package name: myapp.javaform

Java Form name: MemoDialog

Config

Bounds(1): (80,170,300,120) ...

Font(2): Dialog,12 ...

Foreground(3): ...

Background(4): ...

Reset value(5)

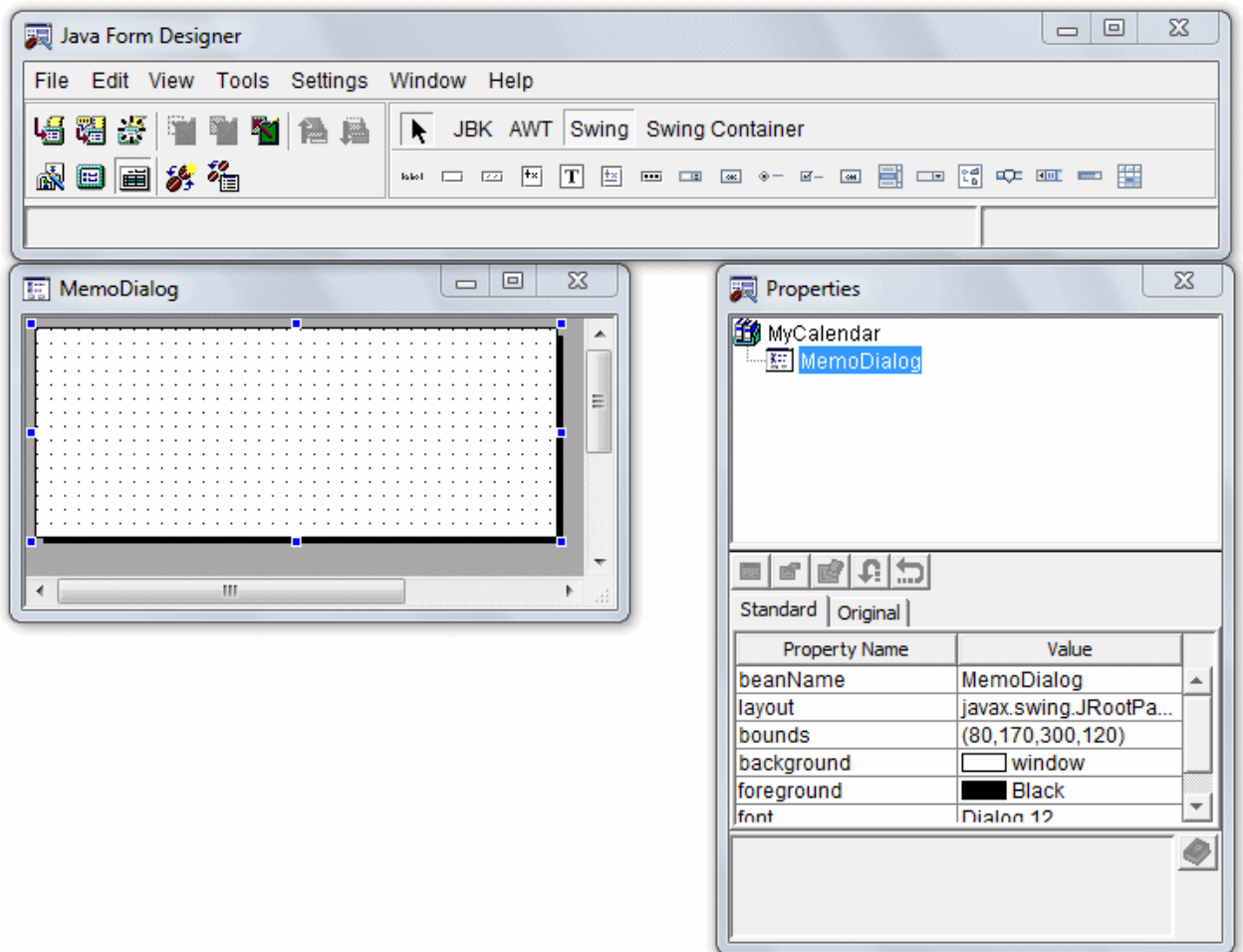
Extended class details

Base class: javax.swing.JDialog

javax.swing.JDialog is a Swing class.
A form extended from the base class above can be created.

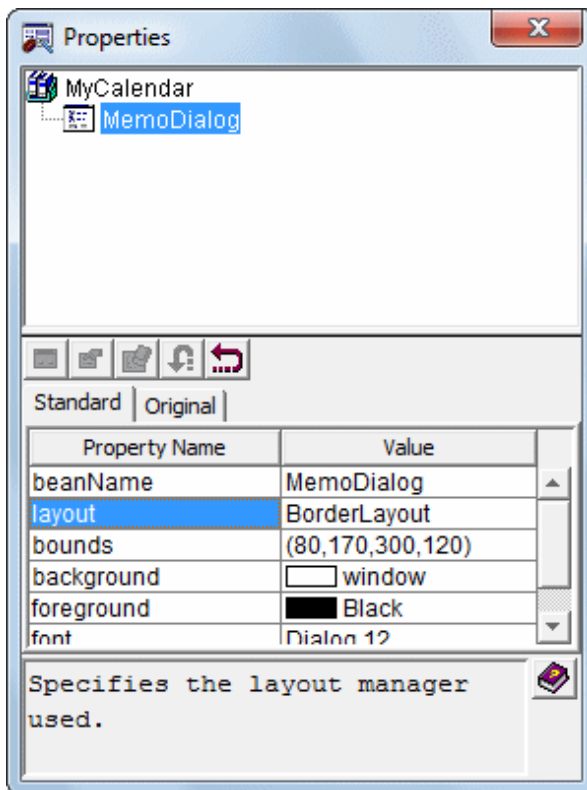
<Back Create Cancel

6. Java Form Designer starts.

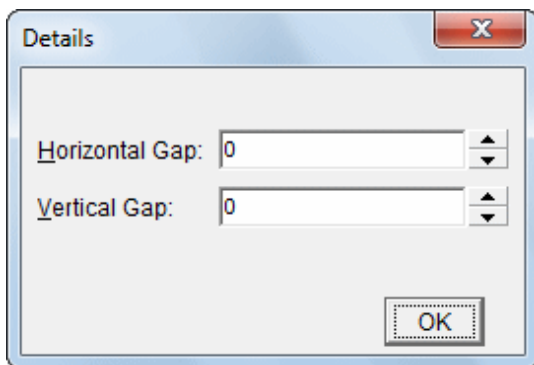


Placing Beans

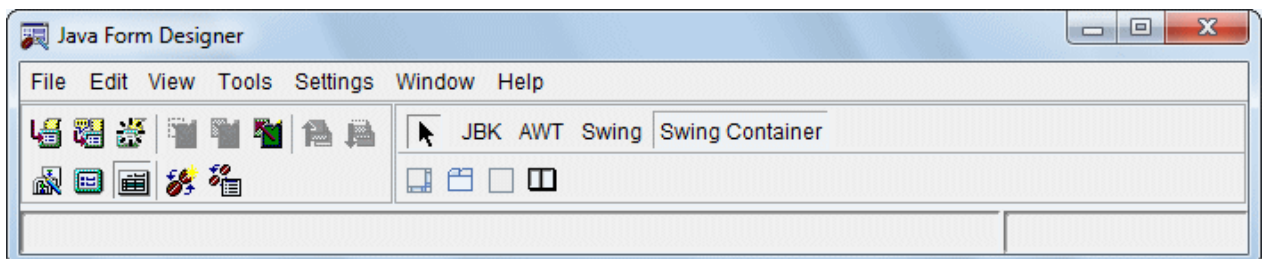
1. Change the [layout] property of the created MemoDialog to [BorderLayout].



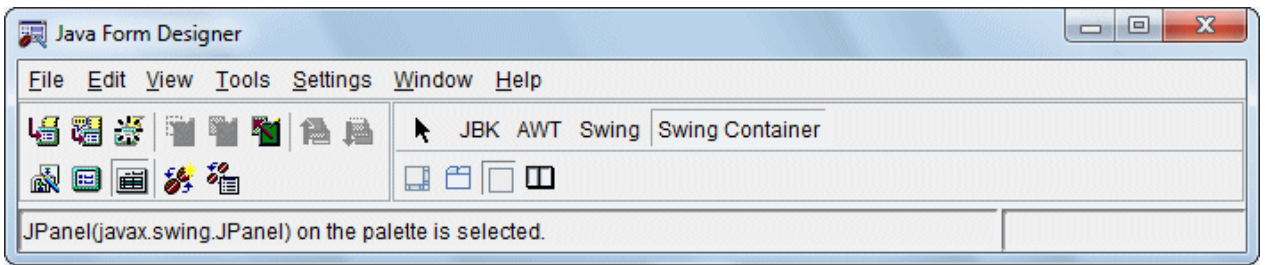
The [Details] dialog box is displayed when you change the [layout]. Click [OK].



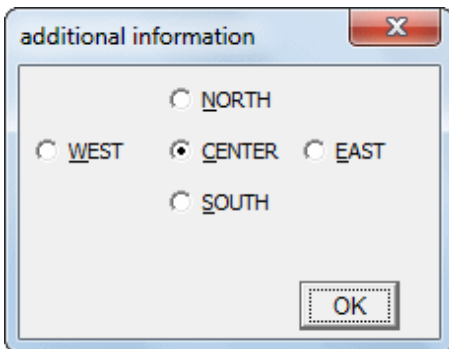
2. Select [Swing Container] in the object palette if it is not already selected.



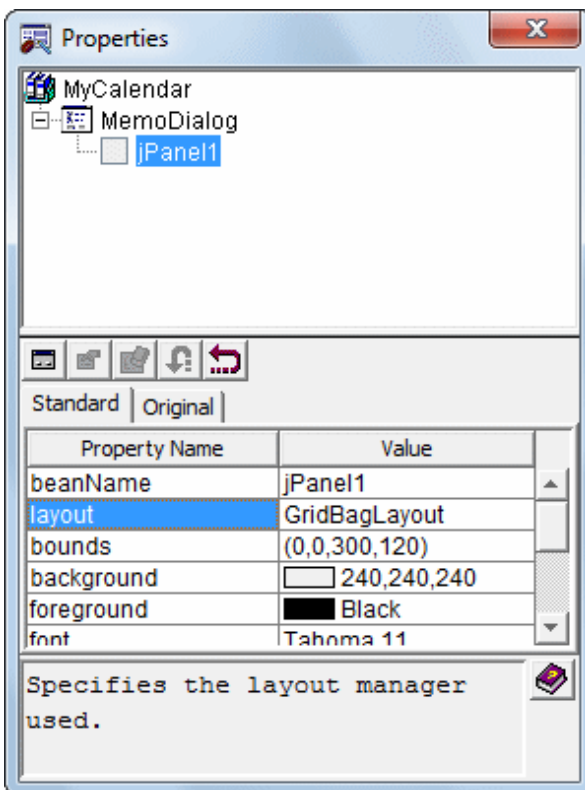
3. Click [JPanel] in the object palette.



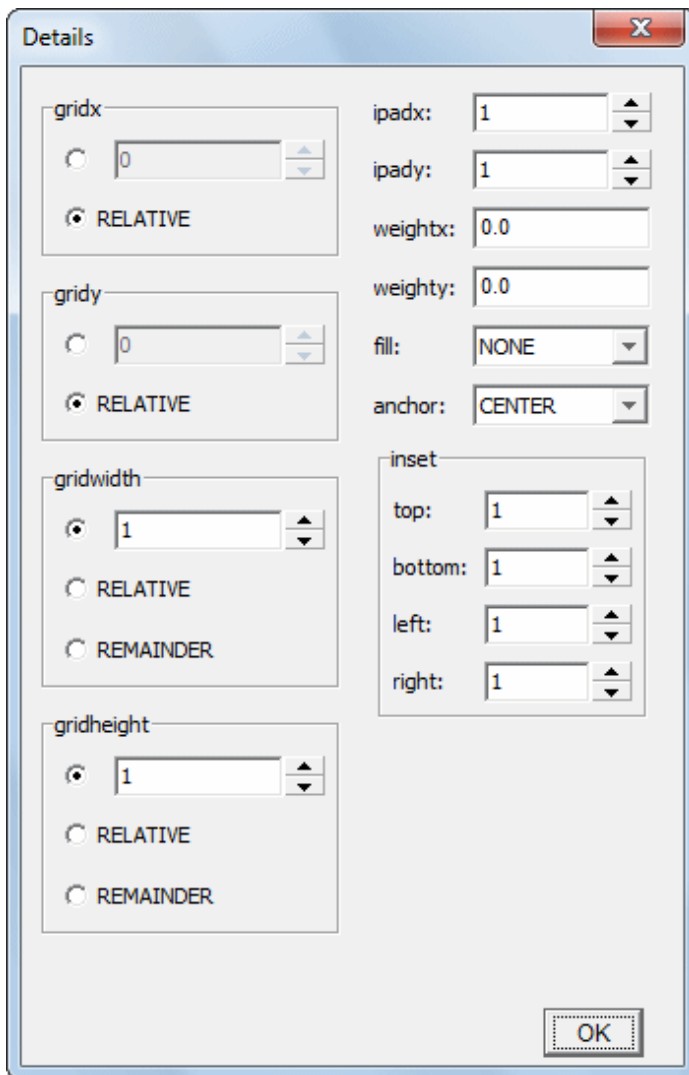
4. Place on the Java Form by dragging with the mouse.
5. Select [CENTER] in the [additional information] and click [OK].



6. Change the layout of the created panel.
Change the [layout] property of the panel to [GridBagLayout].



Click [OK] when the [Details] dialog box is displayed.



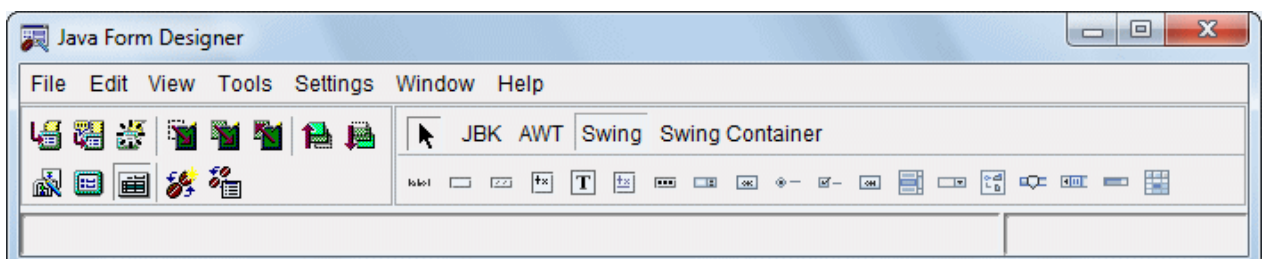
7. Place the second panel.

As this is to be placed directly under the dialog box, do not drag onto the panel that was just created, but rather drag to the grey area outside the dialog box.

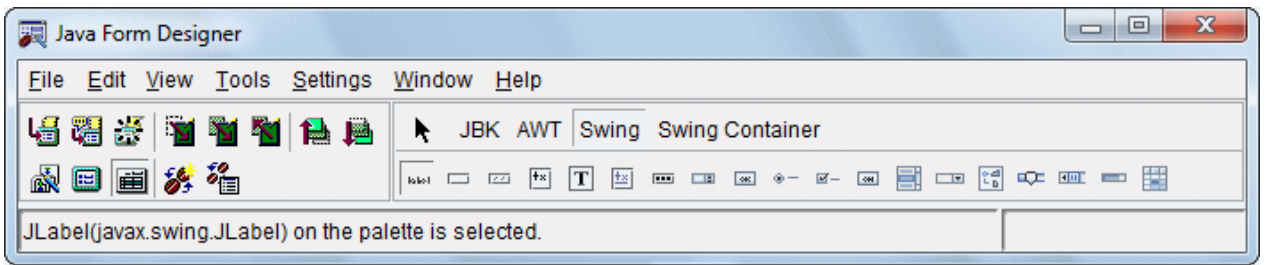
Select [SOUTH] in the [additional information] and click [OK].

8. Place the label.

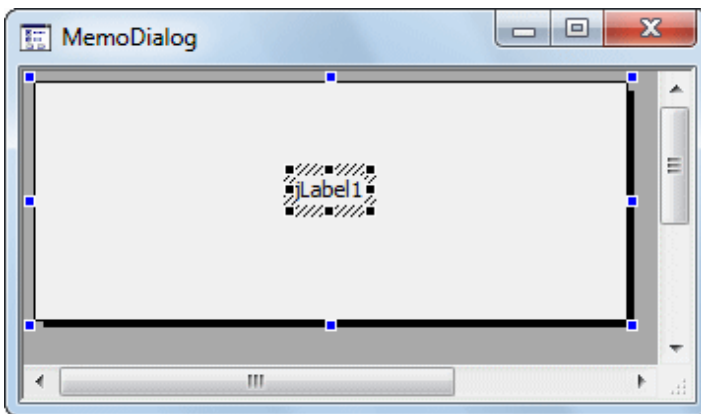
Select [Swing] in the object palette if it is not already selected.



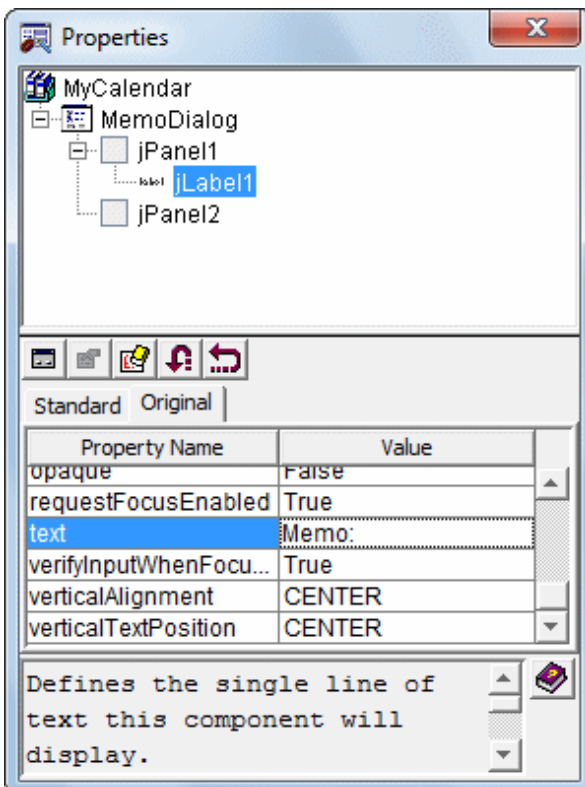
9. Click [JLabel] in the object palette.



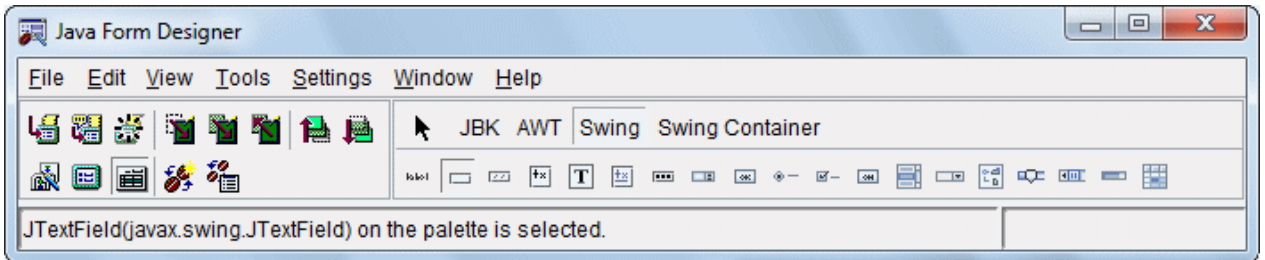
10. Drag to the top of the created dialog so that it is placed on the panel that was created first. Click [OK] when the [additional information] dialog box is displayed.



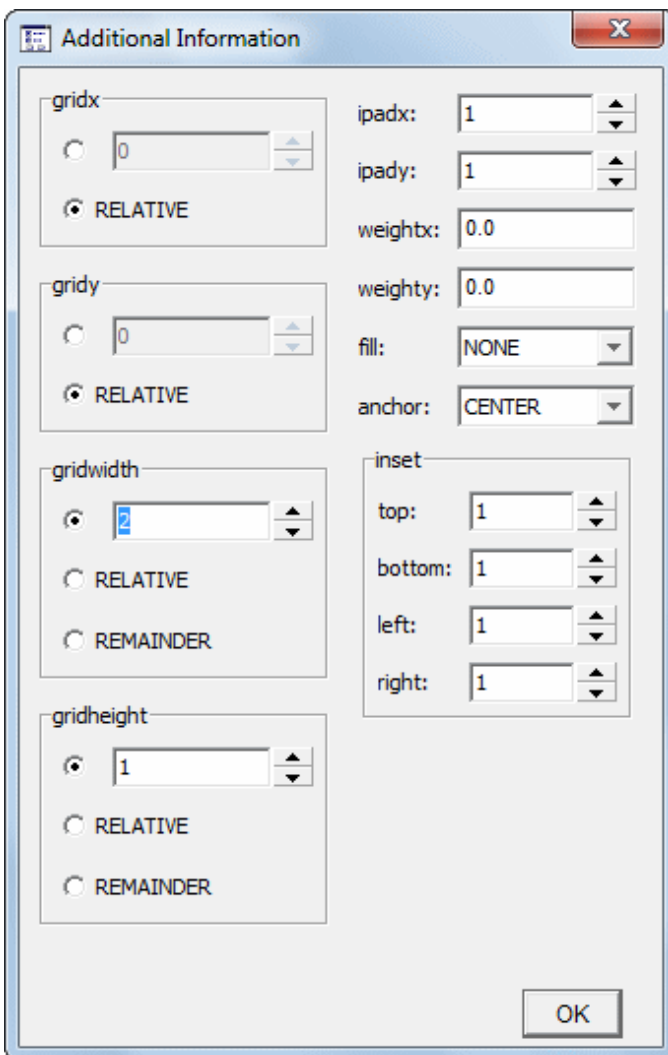
11. Change the text of the created label. Select the created label and select the [Original] tab of the Properties window. Change the [text] property to [Memo:].

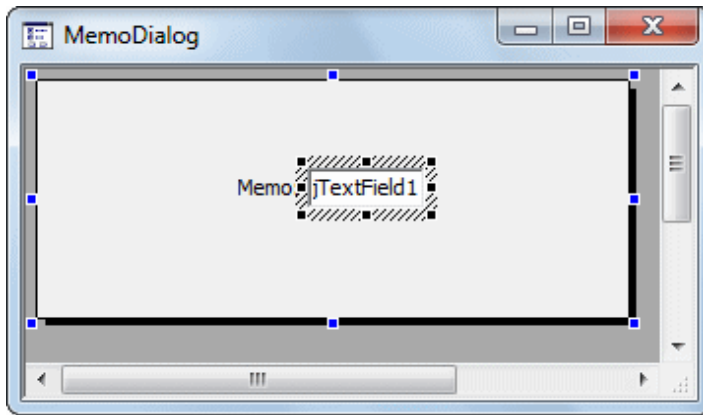


- Place the text field.
Click [JTextField] in the object palette.

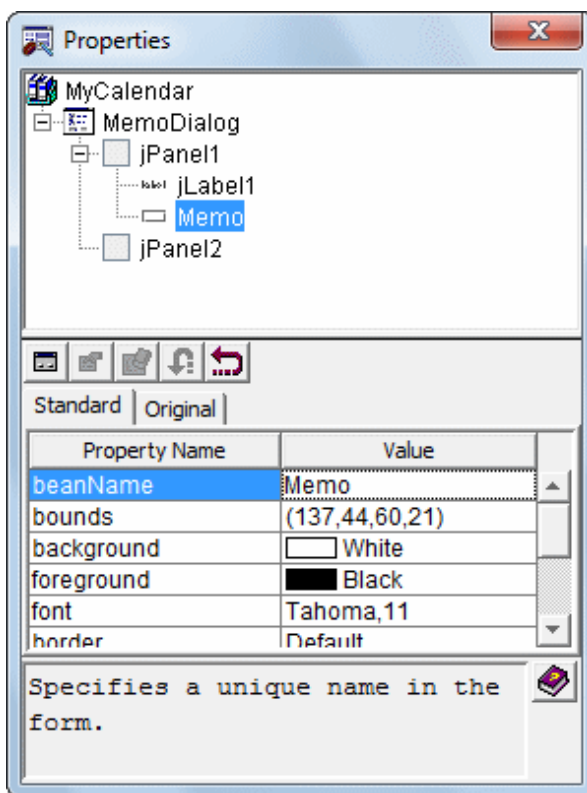


- Drag to the top of the created dialog so that it is placed on the panel that was created first.
Change [gridwidth] to [2] in the [additional information] and click [OK].

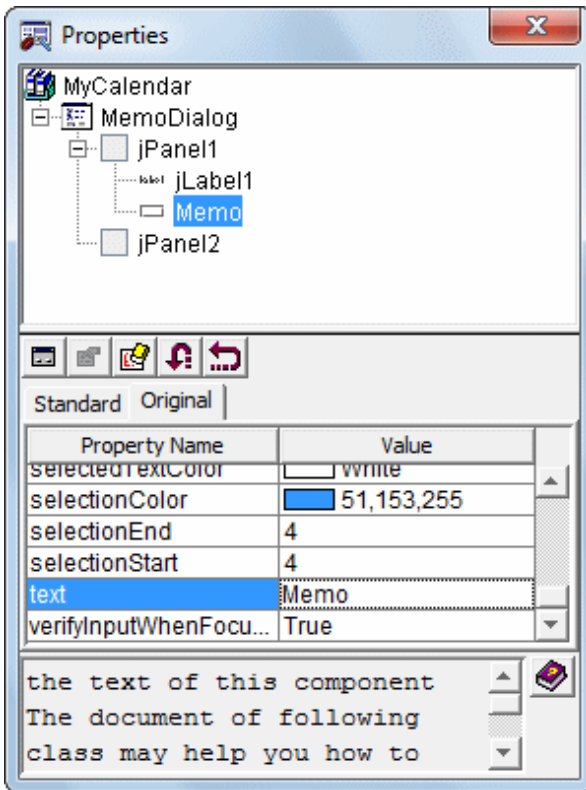




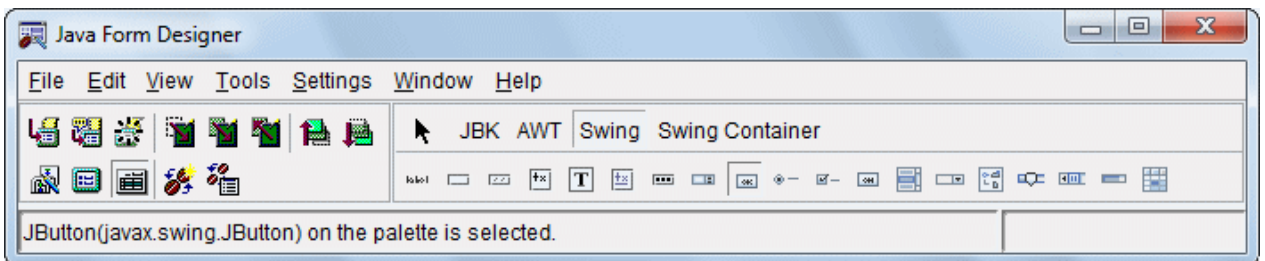
14. Change the [beanName] of the created text field.
 Select the created text field and select the [Standard] tab of the Properties window.
 Change the [beanName] property to [Memo].



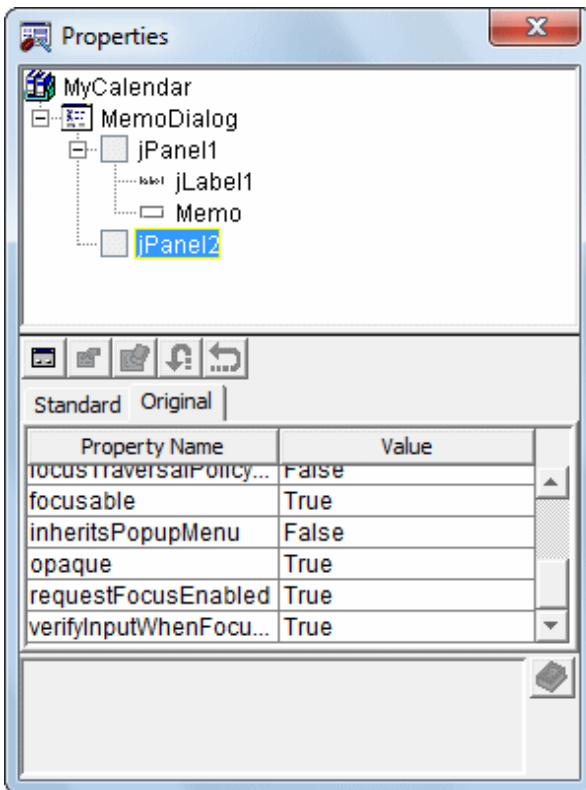
- Change the text of the created text field.
Select the created text field and select the [Original] tab of the Properties window.
Change the [text] property to [Memo].



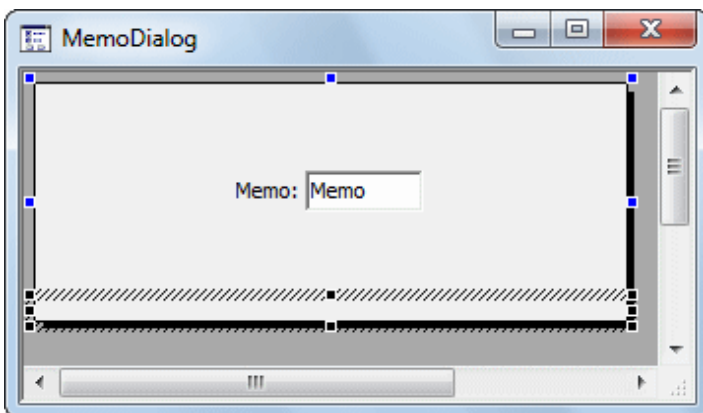
- Place the button.
Click [JButton] in the object palette.



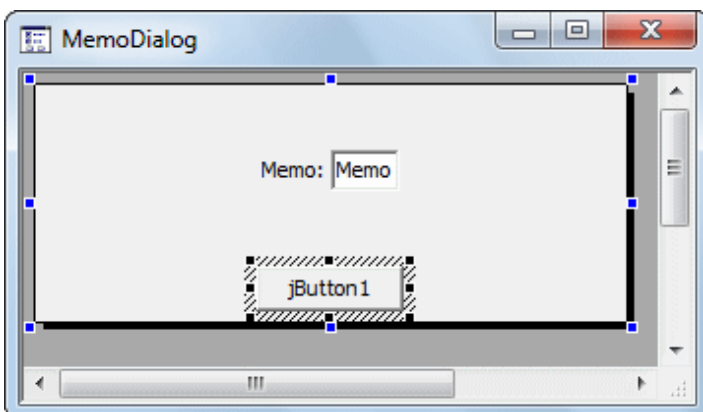
17. Select the [jPanel2] that is in the tree in the Properties window so that it is placed on the panel that was created second.



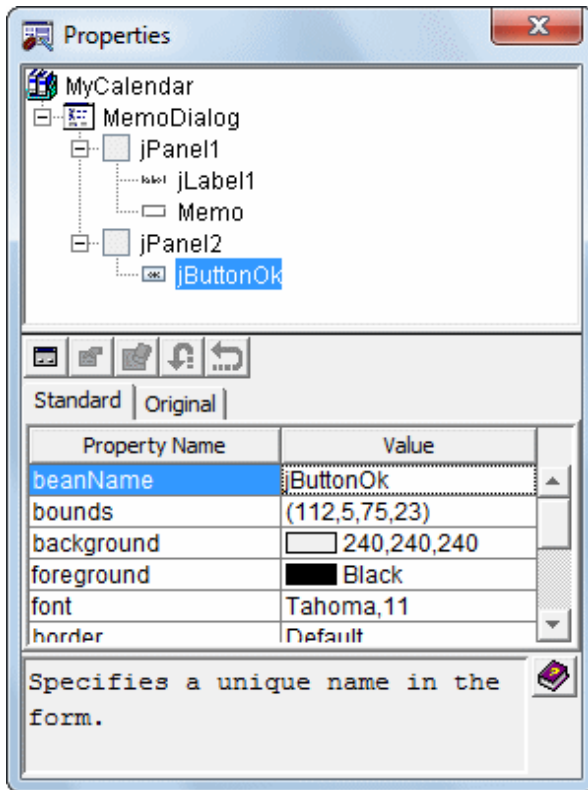
18. The border of the panel is displayed when it is selected.



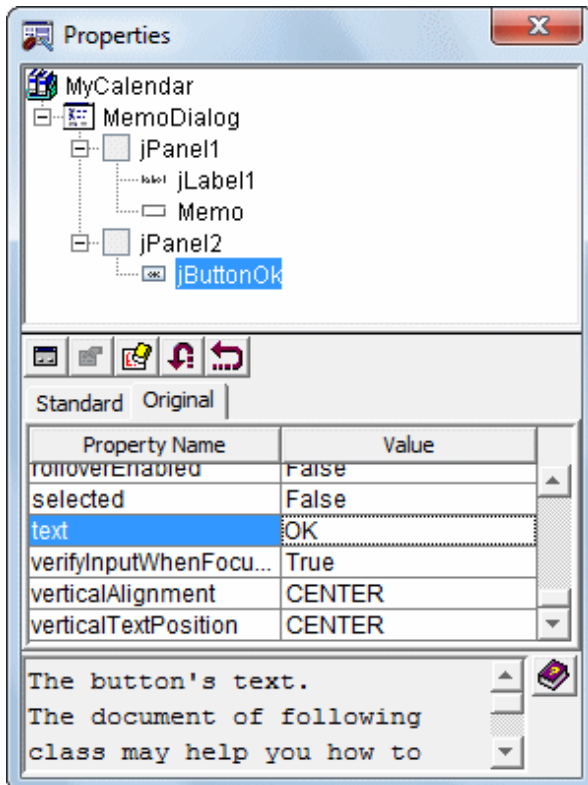
19. Drag into the border.



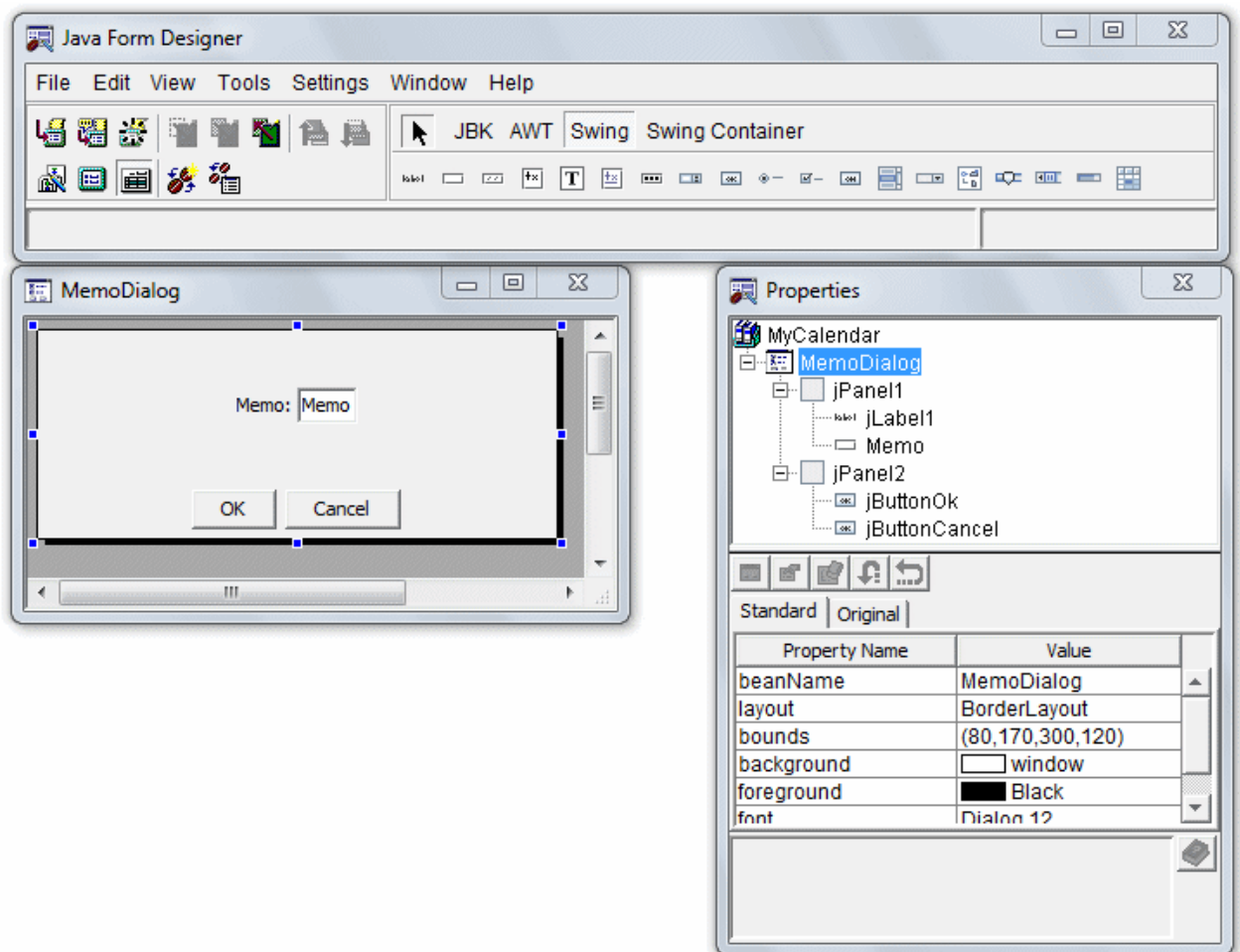
20. Change the [beanName] of the created button.
 Select the created button and select the [Standard] tab of the Properties window.
 Change the [beanName] property to [jButtonOk].



21. Change the text of the created button.
 Select the created button and select the [Original] tab of the Properties window.
 Change the [text] property to [OK].



22. Place the second button.
Click [JButton] in the object palette.
23. Select the [jPanel2] that is in the tree in the Properties window so that it is similarly placed on the panel that was created second.
The border of the panel is displayed when it is selected.
Drag into the border.
24. Change the [beanName] of the created button.
Select the created button and select the [Standard] tab of the Properties window.
Change the [beanName] property to [jButtonCancel].
25. Change the text of the created button.
Select the created button and select the [Original] tab of the Properties window.
Change the [text] property to [Cancel].



26. To start the Java editor, select [View] > [Java Editor] from the Java Form Designer menu bar.
27. Since the JBK class is used, add the import keyword.
Add the text shown below in **red**.

```
import com.fujitsu.jbk.gui.JFCalendarView;
```

28. Add a field to the MemoDialog class.
This field is used to save the calendar class of the calling frame. Add the text shown below in **red**.

```
/**
 * The form is constructed.
 */
JFCalendarView cv;
```



```

*/
public MemoDialog(java.awt.Frame owner, boolean modal) {
    super(owner, modal);
}

```

29. Add processes to the MemoDialog class constructor.

The constructor is a method that is automatically called when an object is generated. It usually performs the initial processing of an object. A method with the same name as a class name is regarded as a constructor. Add the text shown below in **red**.

```

public MemoDialog(java.awt.Frame param0, boolean param1) {
    // Invokes the constructor of the base class.
    super(param0, param1);
    cv = ((Frame1) param0).jFCalendarView1;
    // initialize form
    init$();
    // call user definition initialization
    initUser$();
}

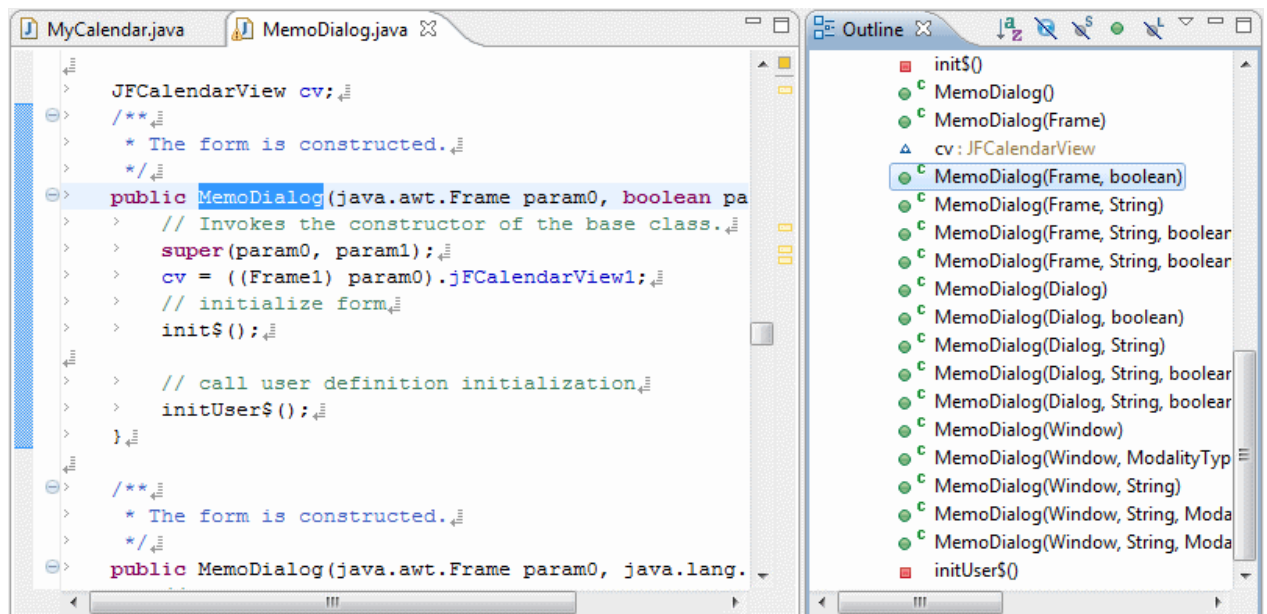
```

Point

Scrolling in Java editor

To easily scroll to a method or field you want to edit in Java editor, you can use the [Outline] view.

Clicking "MemoDialog(Frame, boolean)" in the [Outline] view causes the Java editor display to scroll to the constructor.



30. Add processes to the MemoDialog class initUser\$ method, and define the memo settings in the calendar so that it appears initially. Add the text shown below in **red** to the processing procedure of initUser\$().

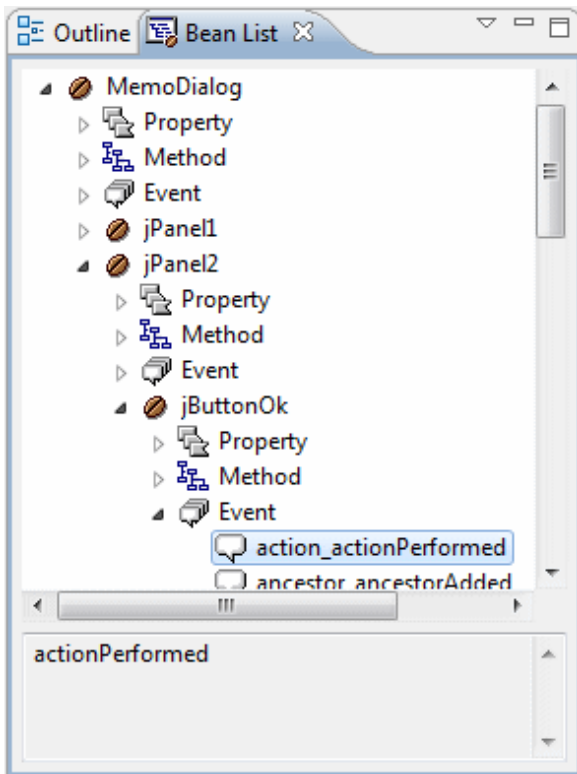
```

protected void initUser$() {
    // The user definition initialization is described at this position.
    Memo.setText(cv.getMemo(cv.getDate()));
    Memo.setColumns(10);
}

```

31. Code the process in which [OK] is clicked.

Double-click [MemoDialog] > [jPanel2] > [jButtonOk] > [Event] > [action_actionPerformed] in the [Bean List] view.
When an event process creation confirmation dialog is displayed, click [Yes].



When the processing procedure of "jButtonOk_action_actionPerformed\$" is displayed, code the text shown below in **red**.

```
public void jButtonOk_action_actionPerformed$(java.awt.event.ActionEvent e) {
    if(!defaultEventProc(e)) {
        // The procedure when the event is generated is described below.
        cv.setMemo(cv.getDate(), Memo.getText());
        dispose();
    }
}
```

Next, code the process in which [Cancel] is clicked.

Double-click [MemoDialog] > [jPanel2] > [jButtonCancel] > [Event] > [action_actionPerformed] in the [Bean List] view.
When an event process creation confirmation dialog is displayed, click [Yes].

When the processing procedure of "jButtonCancel_action_actionPerformed\$" is displayed, code the text shown below in **red**.

```
public void jButtonCancel_action_actionPerformed$(java.awt.event.ActionEvent e) {
    if(!defaultEventProc(e)) {
        // The procedure when the event is generated is described below.
        dispose();
    }
}
```

32. Save the Java Form.

Select [File] > [Save] from the workbench menu bar.

To close the Java Form, select [File] > [Close] from the Java Form Designer menu bar.

Adding processes to the calendar

1. Define settings so that a dialog for entry of a memo is displayed when a date on the calendar is double-clicked.

Open [Frame1.java] in Java Form Designer. To display the context menu, right-click [MyCalendar] > [src] > [myapp.javaform] > [Frame1.java] in [Package Explorer] view of workbench. Then, select [Open With] > [Graphical Editor].

- To start the Java editor, select [View] > [Java Editor] from the Java Form Designer menu bar.
- An action event occurs when a date on the calendar is double-clicked. Code processes for the action event. Double-click [Frame1] > [jFCalendarView1] > [Event] > [action_actionPerformed] in the [Bean List] view. When an event process creation confirmation dialog is displayed, click [Yes]. When the processing procedure of "jFCalendarView1_action_actionPerformed\$" is displayed, code the text shown below in **red**.

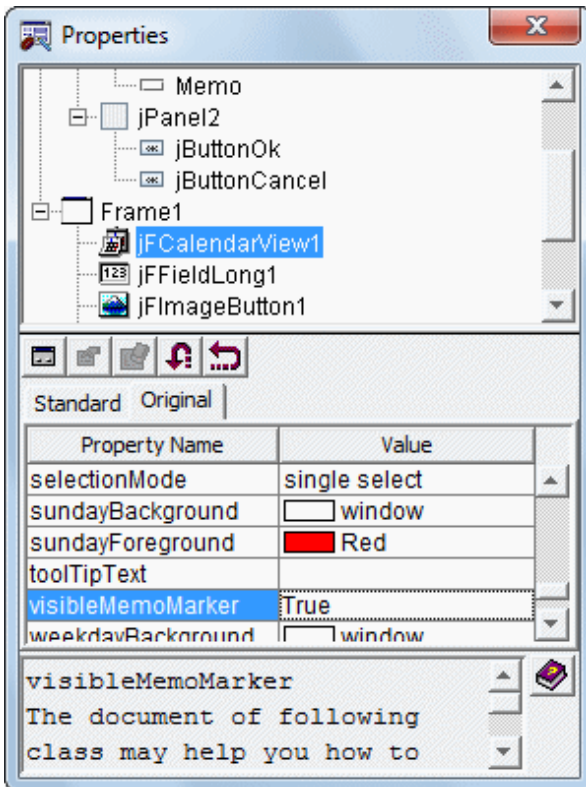
```

public void jFCalendarView1_action_actionPerformed$( java.awt.event.ActionEvent e) {
    if( !defaultEventProc(e) ) {
        // The procedure when the event is generated is described below.
        MemoDialog dlg = new MemoDialog(this, false);
        dlg.setVisible(true);
    }
}

```

- Modify the calendar Bean properties. Modify the following properties of the [Original] tab, using the property sheet:

Property Name	Value
displayMemo	True
visibleMemoMarker	True

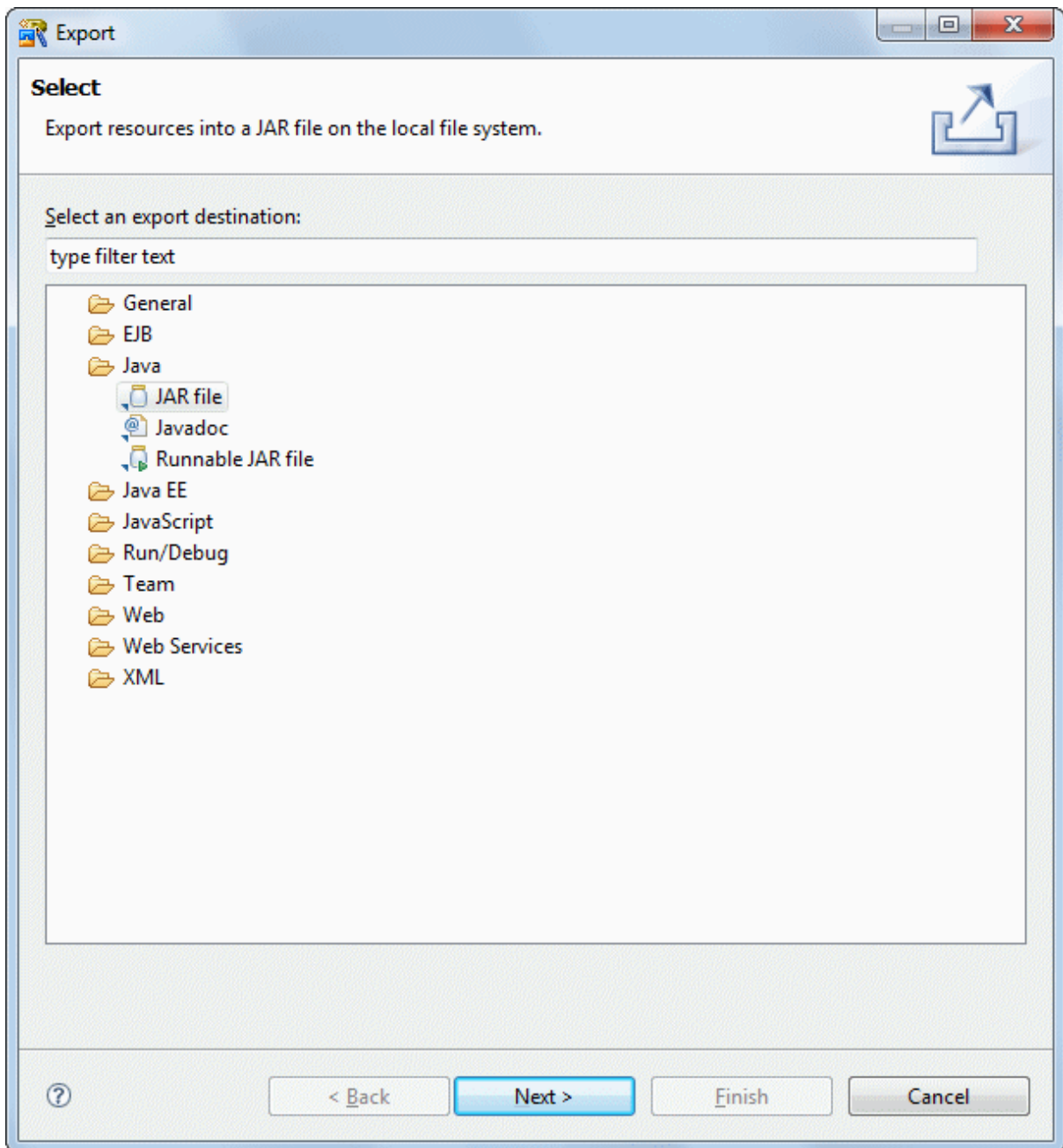


- Save the Java Form. Select [File] > [Save] from the Java Form Designer menu bar. To close the Java Form, select [File] > [Close] from the Java Form Designer menu bar.

Building

- Building is executed automatically when resource files (such as the Java files) are saved if the [Build Automatically] item is enabled in the [Project] menu. If [Build Automatically] is disabled, right-click on the project and select [Build Project] or select [Build Project] in the [Project] menu.

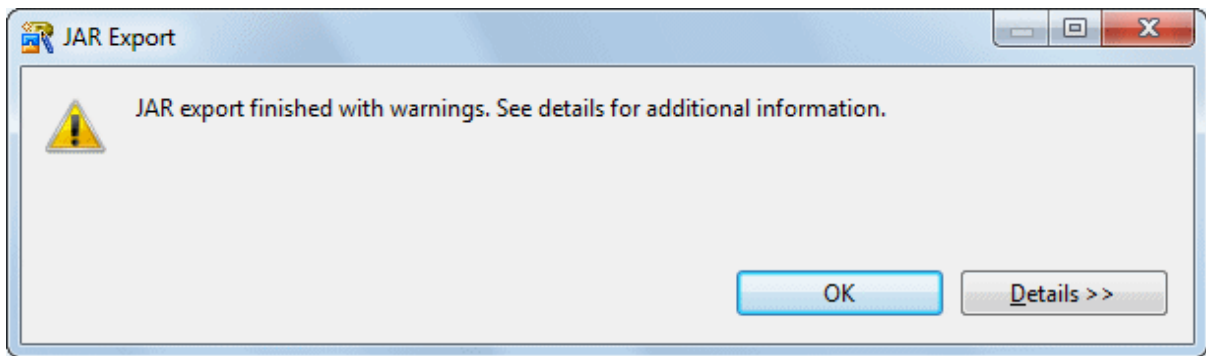
2. A JAR file is required so create it.
 Use the export wizard to create the JAR file.
 Select [File] > [Export] to start the export wizard.
 Select [Java] > [JAR file] in the export wizard.



3. The JAR export wizard is displayed.
 Select [MyCalendar] > [src] folder in [Select the resources to export] and enter the following settings.
 After setting the information, click [Finish].

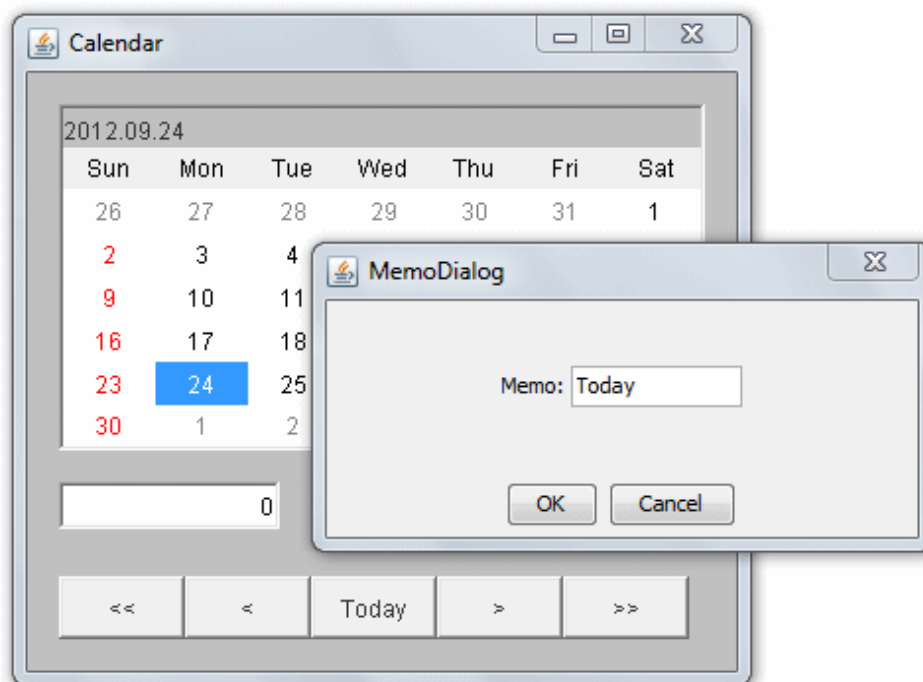
Setup Items	Setup Content
Select the resources to export	src/
Export generated class files and resources	Checked
JAR file	MyCalendar/MyCalendar.jar
Compress the contents of the JAR file	Checked

The following message is output when exporting JAR, but this does not indicate a problem.

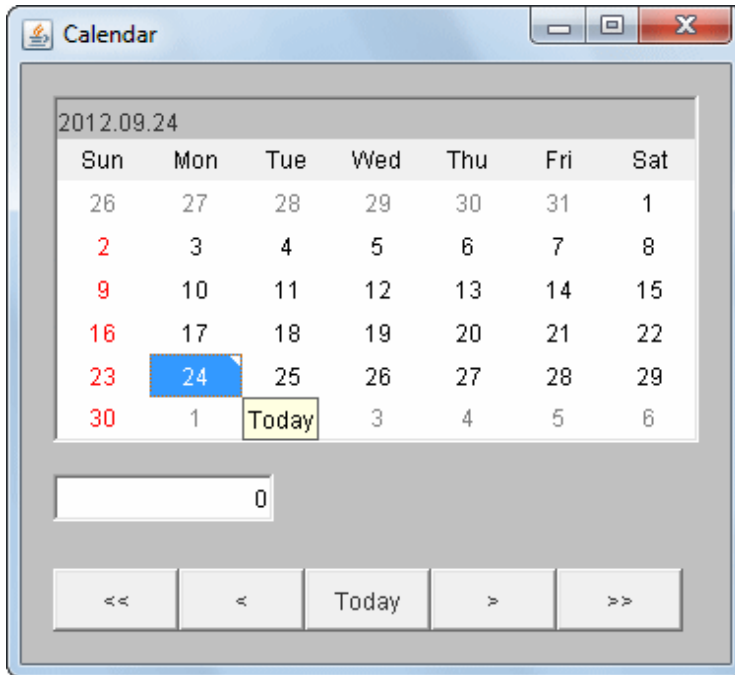


Running

1. Select the file (class) to be executed. Click [MyCalendar] > [src] > [myapp.javaform] > [MyCalendar.java] in the [Package Explorer] view of workbench.
2. Select [Run] > [Run As] > [Java Application] from the menu bar. As a result, the application starts, and the Java Form is displayed.
3. Double-click a date on the calendar. The dialog is displayed. Enter a memo, then click [OK].



4. A marker is displayed on the date that has the memo.
When the cursor passes over that date, the memo is displayed as a tool tip.



5. To close the application, click the Close [X] button on the title bar.

F.3 Applet

You will learn how to develop applets, by creating a simple applet and an applet that involves transitions between multiple pages.

F.3.1 Developing Applet

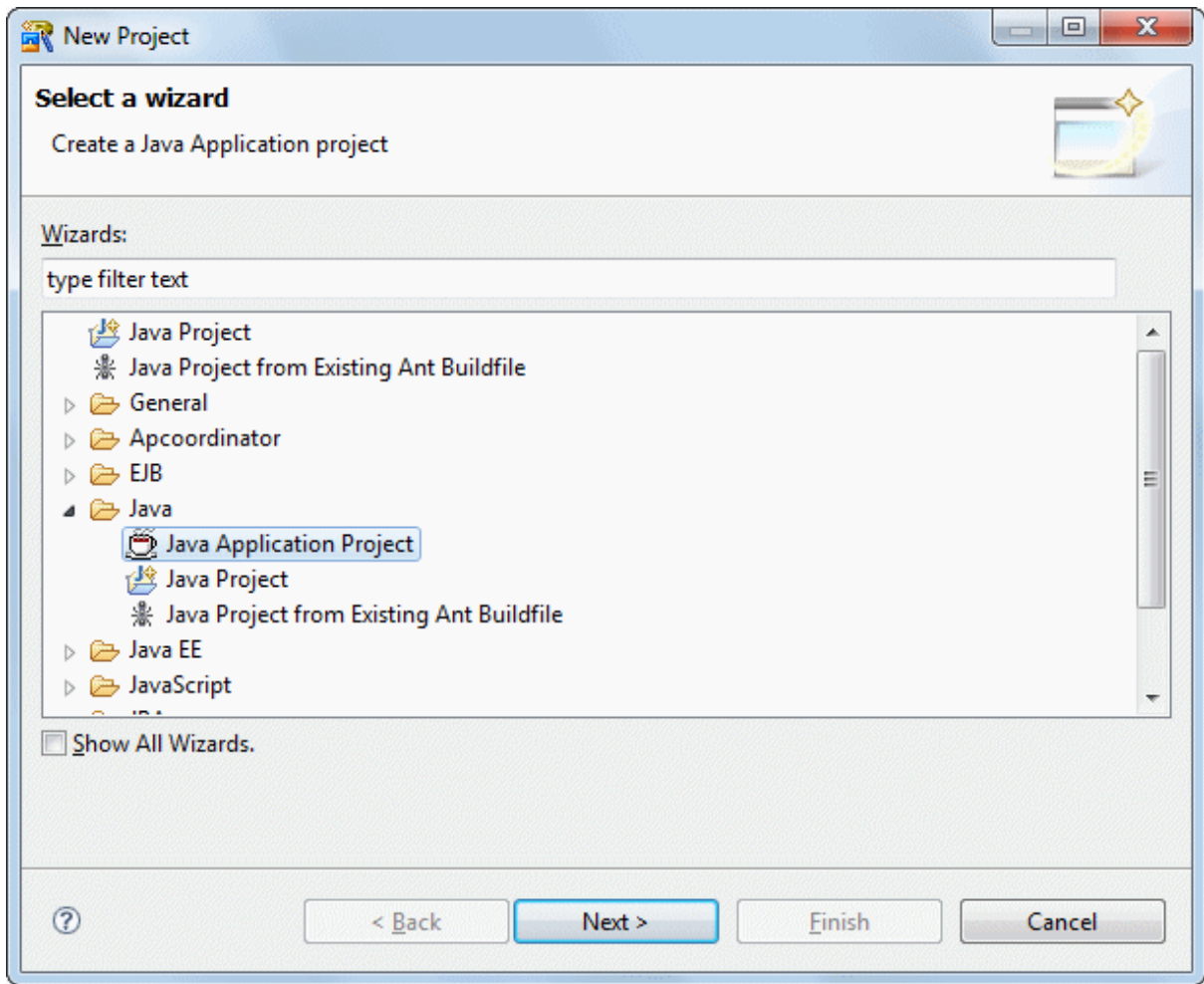
An applet is a program that is embedded in and run on HTML when that HTML is displayed in a Web browser. In Interstage Studio, you can develop applets, which are a type of Java application.

Follow the instructions below to develop an applet that works in the same way as the application you created in Lessons 1 to 3 of "[F.2 Client Application](#)".

Creating an Java Application Project

1. Start the workbench.
2. Select [File] > [New] > [Project] from the menu bar.

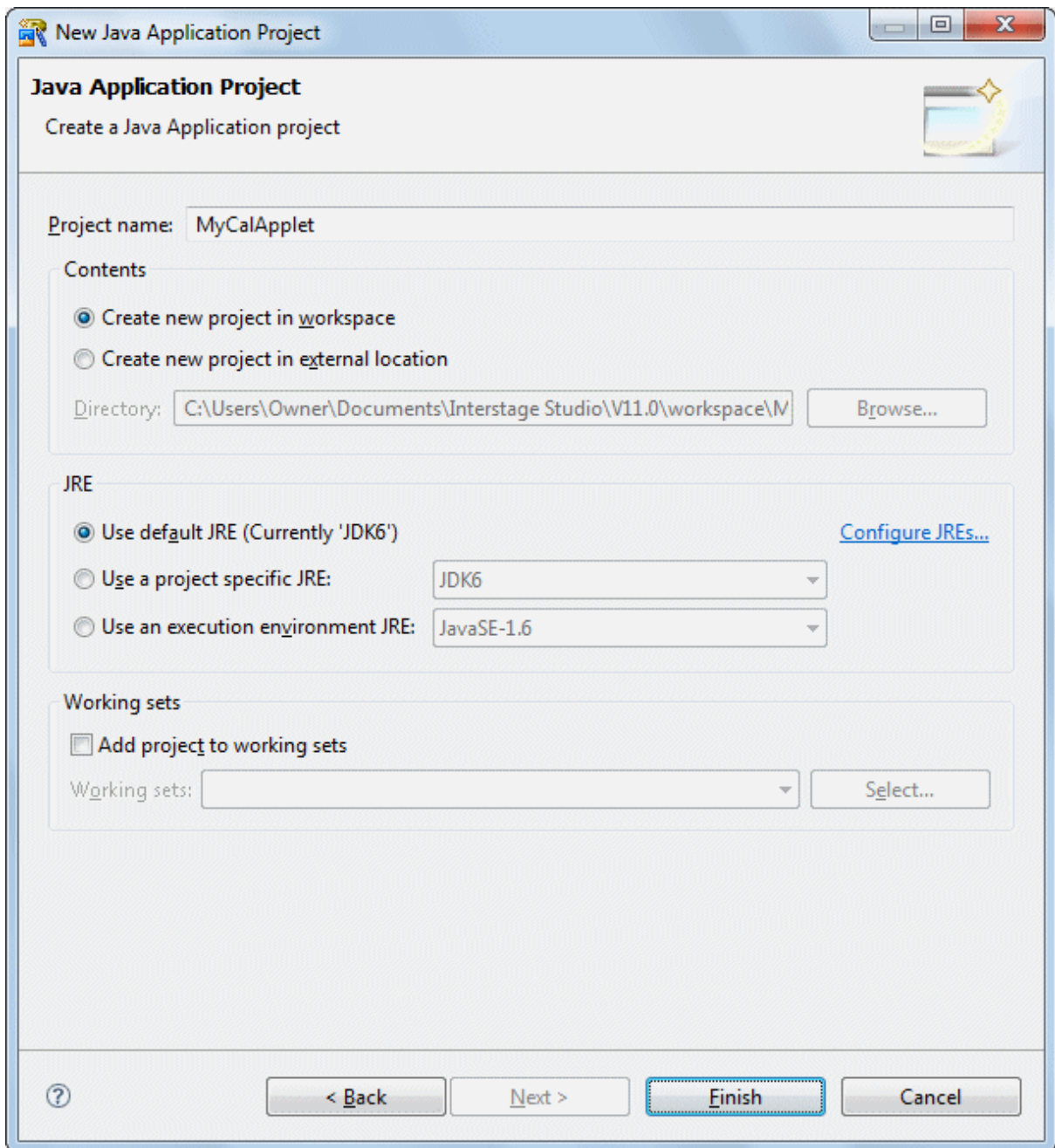
3. The [New Project] wizard is displayed. Select [Java Application Project] from the tree.



Click [Next].

4. The [Java Application Project] page is displayed. A project is created according to the information entered for items on this page. Enter the project information as follows.

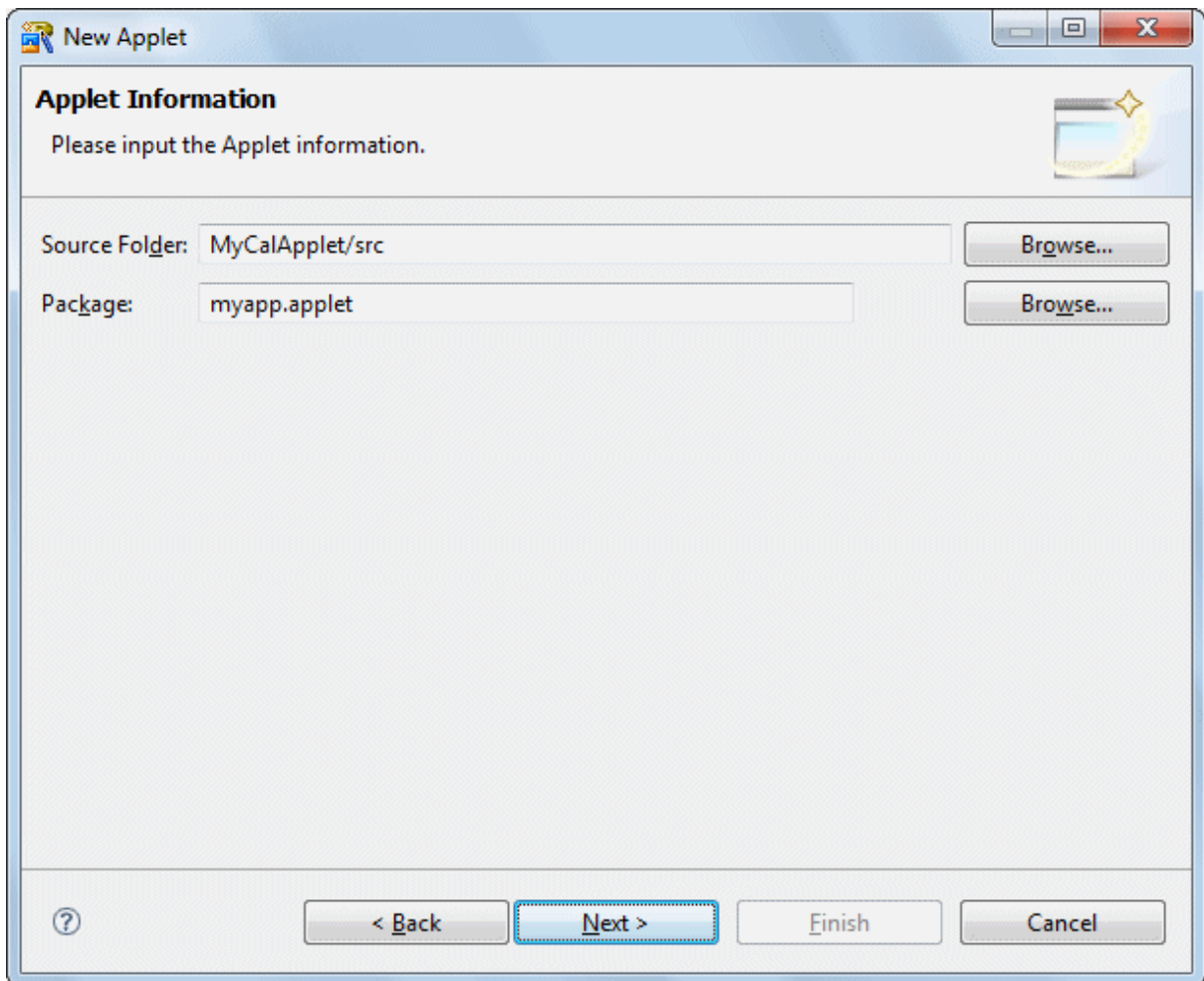
Setup Items	Setup Content
Project name	MyCalApplet



Click [Finish].

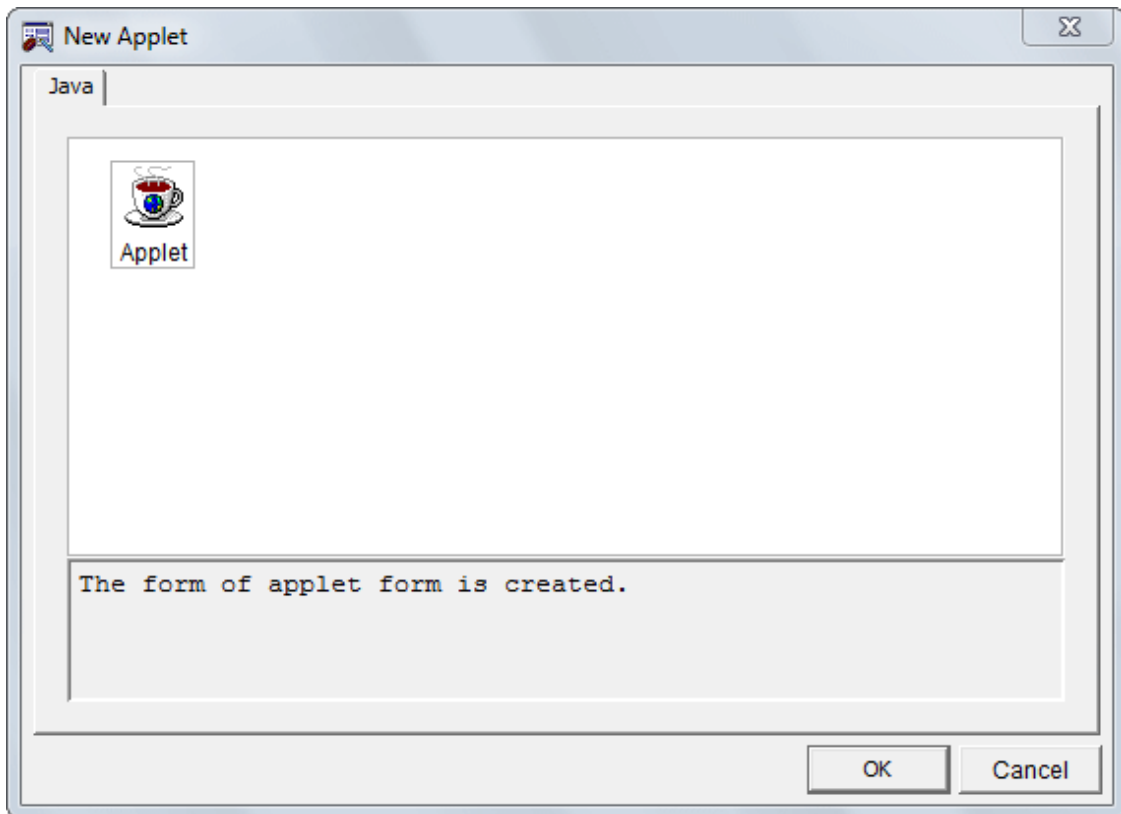
- The next step is to create an applet. Select [File] > [New] > [Other] from the menu bar in workbench. Then, select [Java] > [GUI] > [Applet] from the tree in the [New] wizard that appears. The [Applet Information] page is displayed. Enter information as follows.

Setup Items	Setup Content
Package	myapp.applet

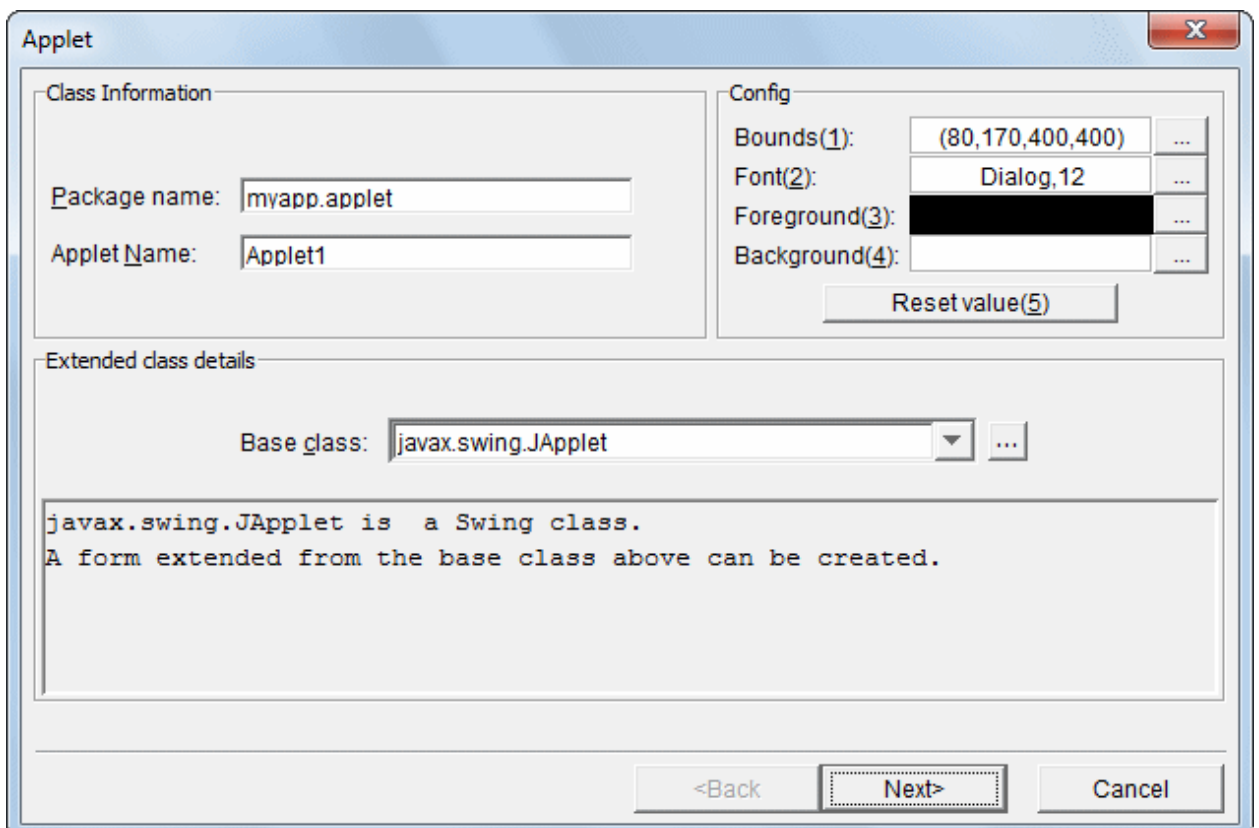


Click [Next].

- The [New Applet] dialog box is displayed.
Select [Applet] on the [Java] tab, and then click [OK].



- The page for entering applet class information is displayed. Enter "Applet1" in the [Applet Name] field.
Select [javax.swing.JApplet] as the base class.



Click [Next].

8. The [Create HTML] page is displayed. You can create an HTML template in which the applet is pasted. Specify the following settings.

Setup Items	Setup Content
Generate HTML	Checked
Generate HTML for JBK plug-in	Checked
Page title	Applet1
Width of applet	400
Height of applet	400

Applet

Create HTML

Generate HTML

Generate HTML for JBK plug-in

Page title: Applet1

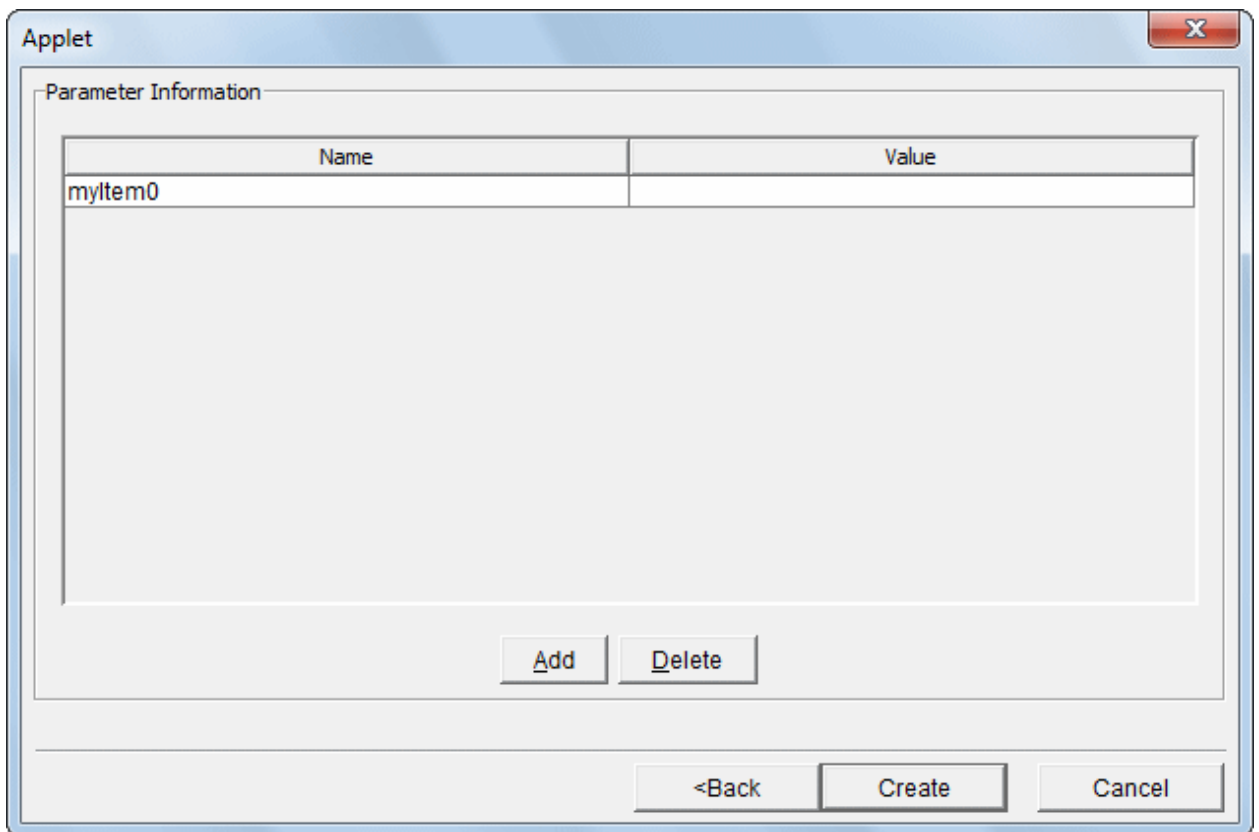
Width of applet: 400

Height of applet: 400

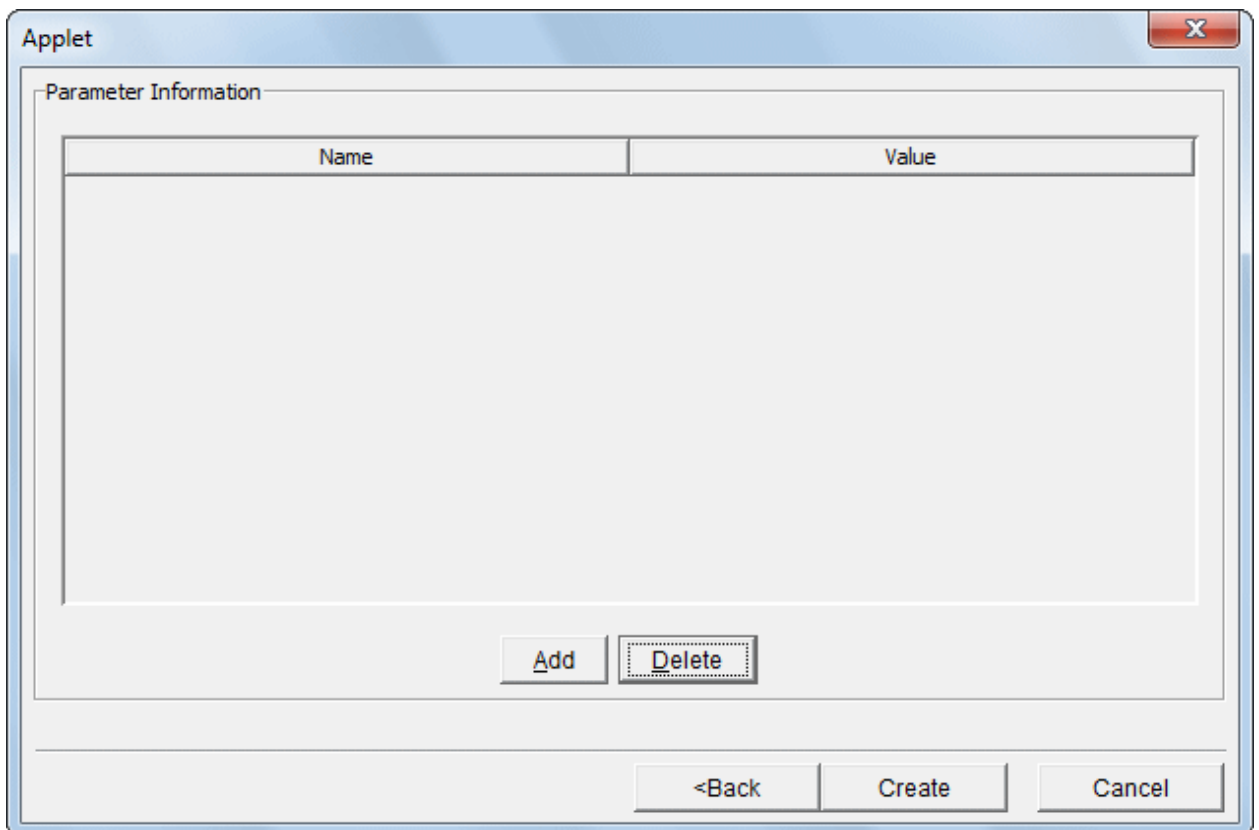
<Back Next> Cancel

Click [Next].

9. The [Parameter Information] page is displayed.



10. Parameters are not used, so you can delete "myItem0." Click [Delete].

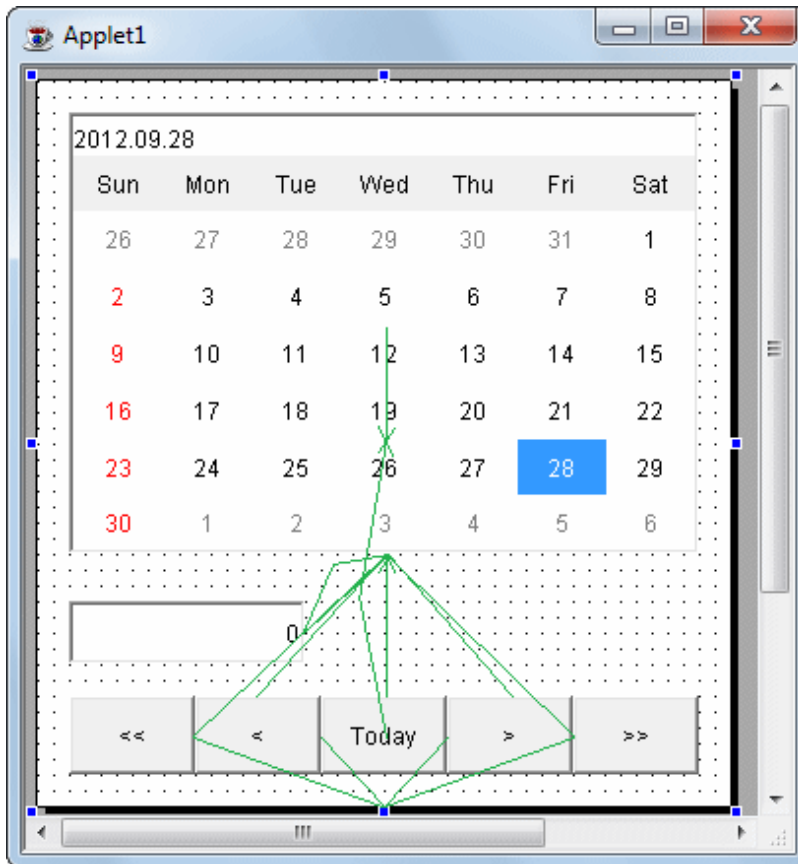


Click [Create].

11. Java Form Designer starts.

Editing an Applet

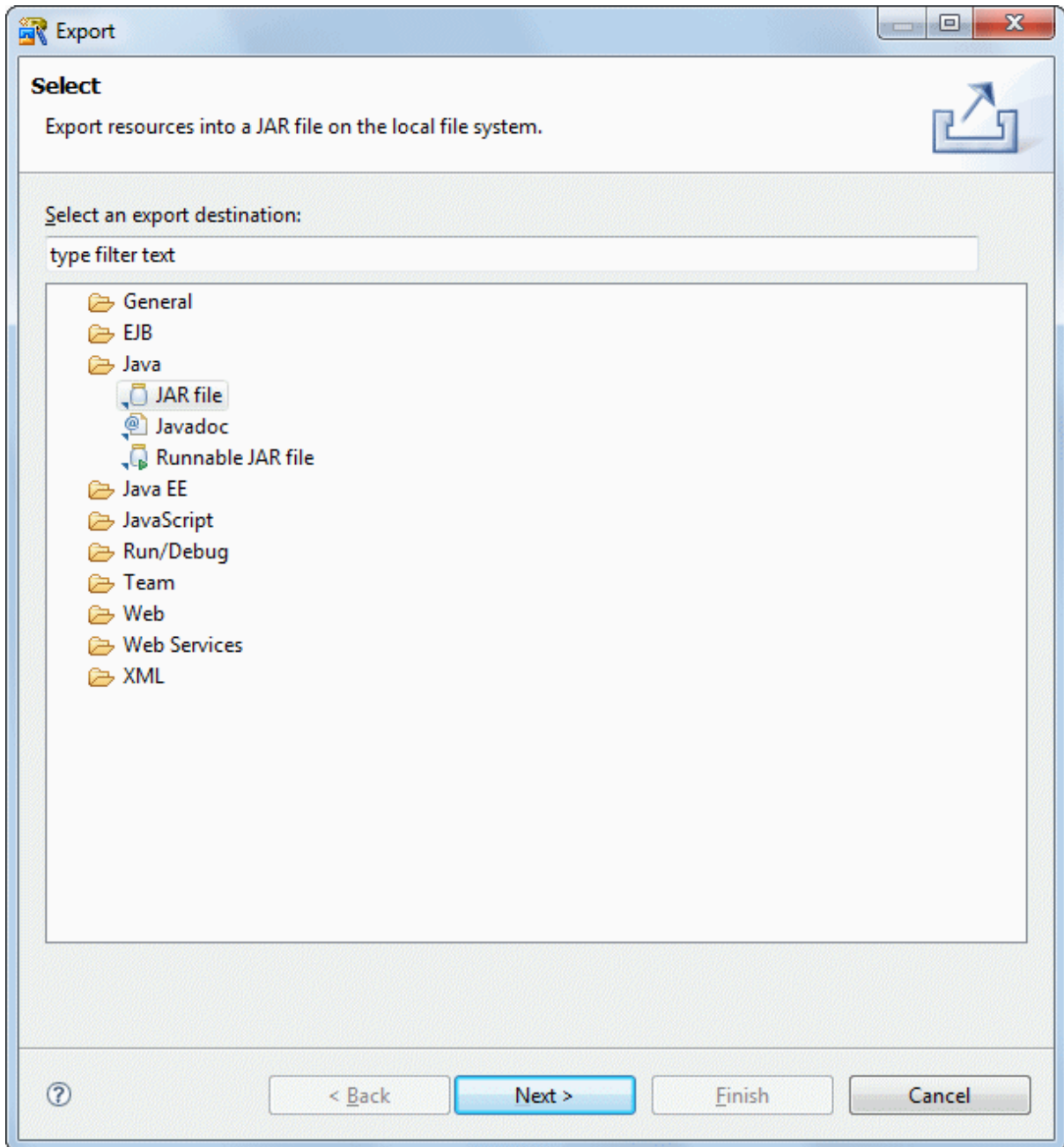
1. Refer to Lessons 1 to 3 of "F.2 Client Application" and, define properties, and describe event processes for the applet.



Building

1. Building is executed automatically when resource files (such as the Java files) are saved if the [Build Automatically] item is enabled in the [Project] menu.
If [Build Automatically] is disabled, right-click on the project and select [Build Project] or select [Build Project] in the [Project] menu.
2. A JAR file is required so create it.
Use the export wizard to create the JAR file.

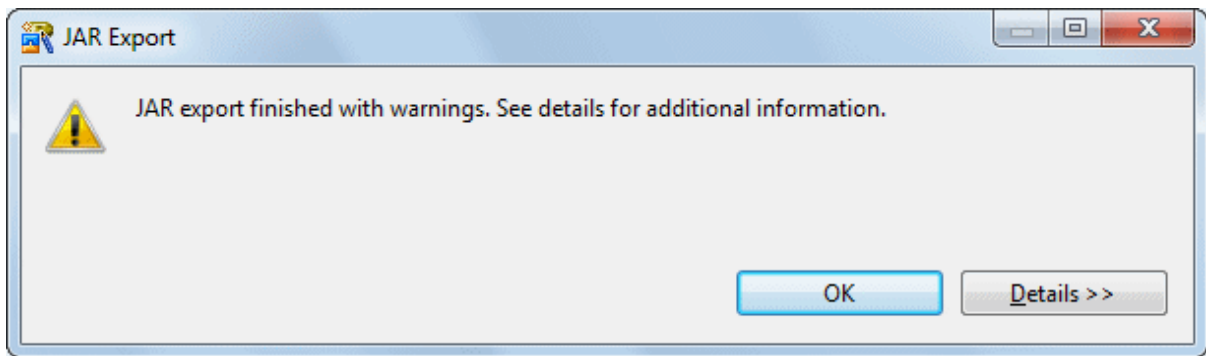
Select [File] > [Export] to start the export wizard.
 Select [Java] > [JAR file] in the export wizard.



- The JAR export wizard is displayed.
 Select [MyCalApplet] > [src] folder in [Select the resources to export] and enter the following settings.
 After setting the information, click [Finish].

Setup Items	Setup Content
Select the resources to export	src/
Export generated class files and resources	Checked
JAR file	MyCalApplet/MyCalApplet.jar
Compress the contents of the JAR file	Checked

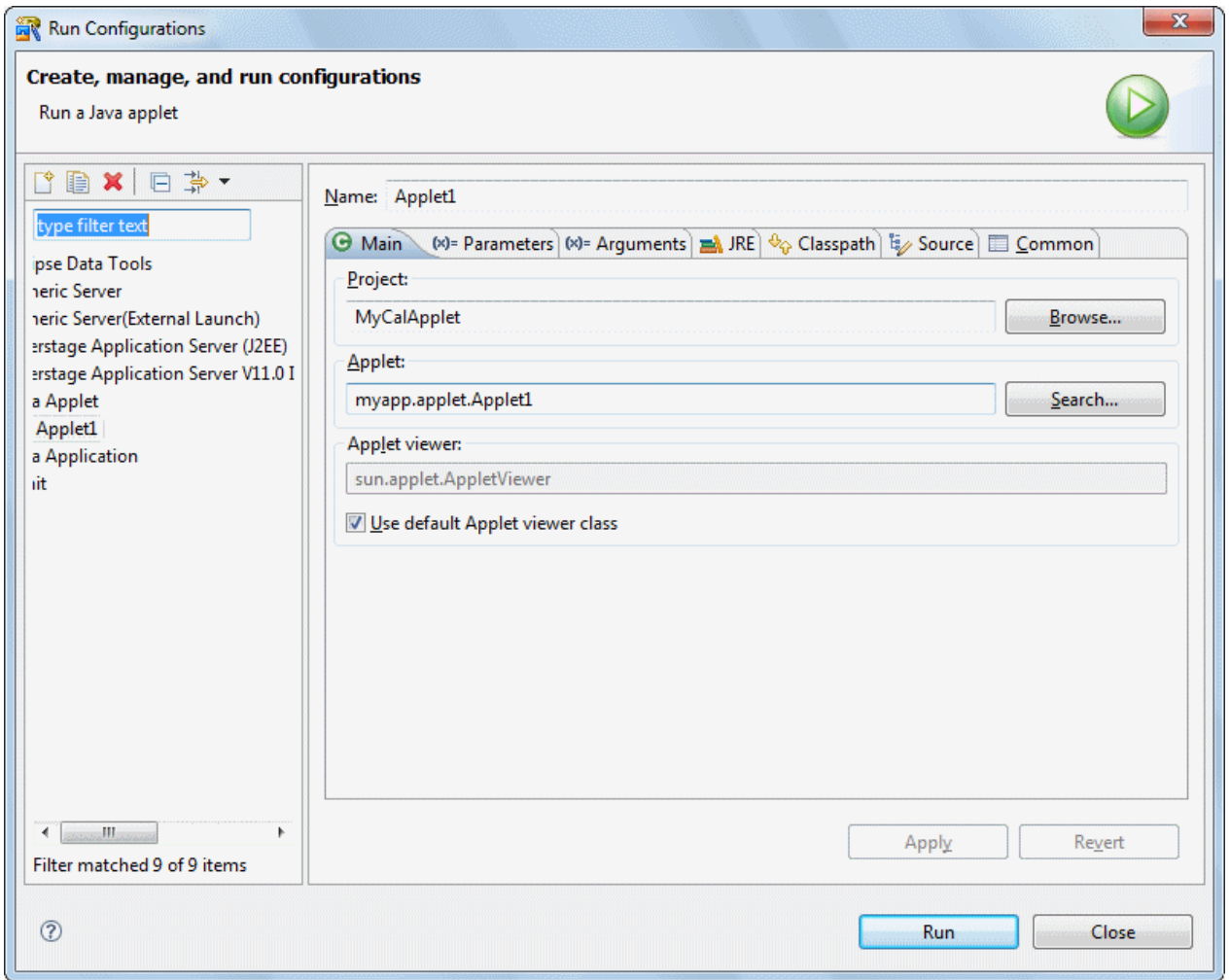
The following message is output when exporting JAR, but this does not indicate a problem.



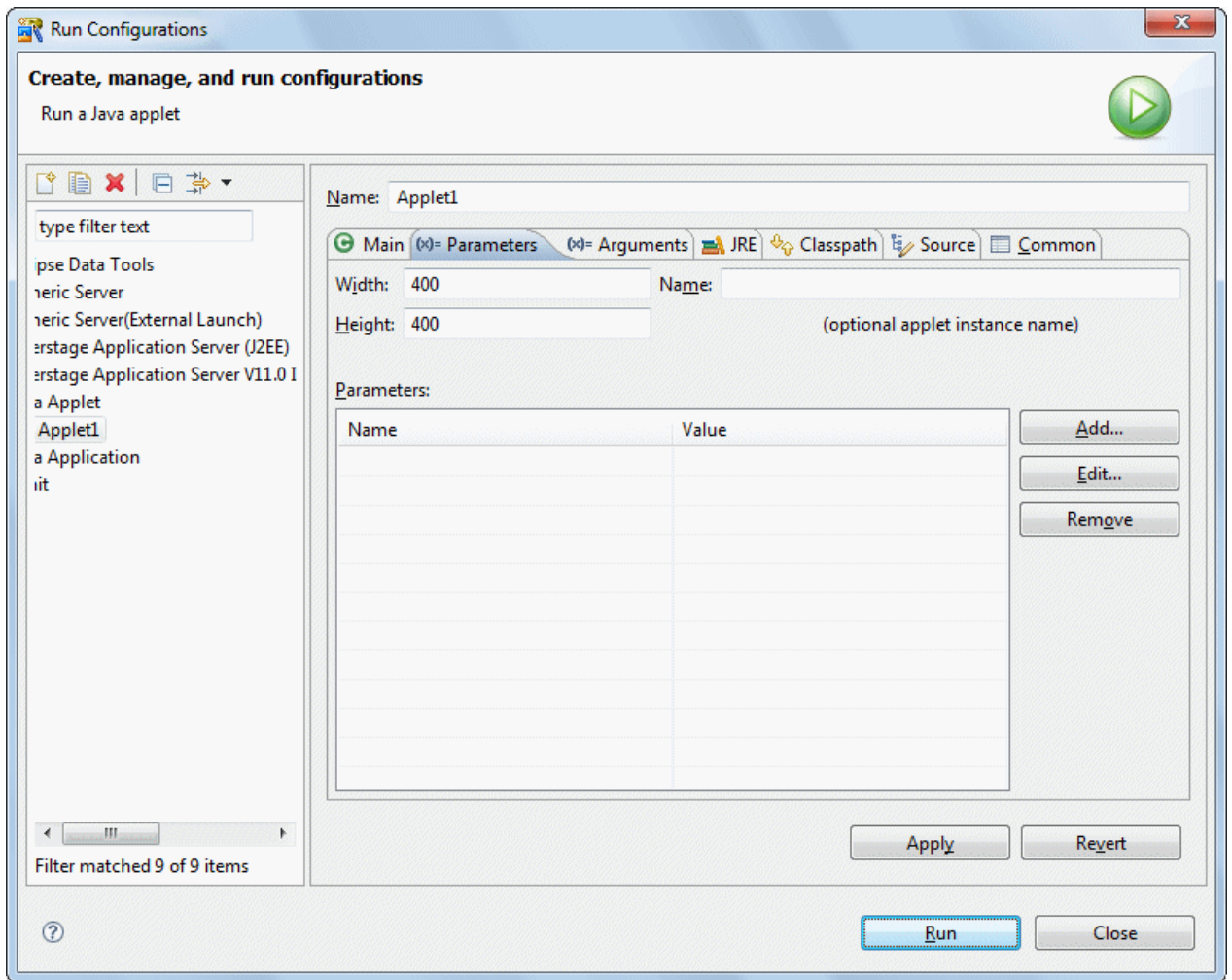
Running

1. Select the file (class) to be run. Then, click [MyCalApplet] > [src] > [myapp.applet] > [Applet1.java] in the [Package Explorer] view in workbench.
2. Select [Run] > [Run] on the workbench menu bar.
3. The [Run] dialog box opens.
Select [Java Applet] from the [Configurations] tree. Then, click [New] at the bottom. A new configuration is added.
Set the following in the [Main] tab:

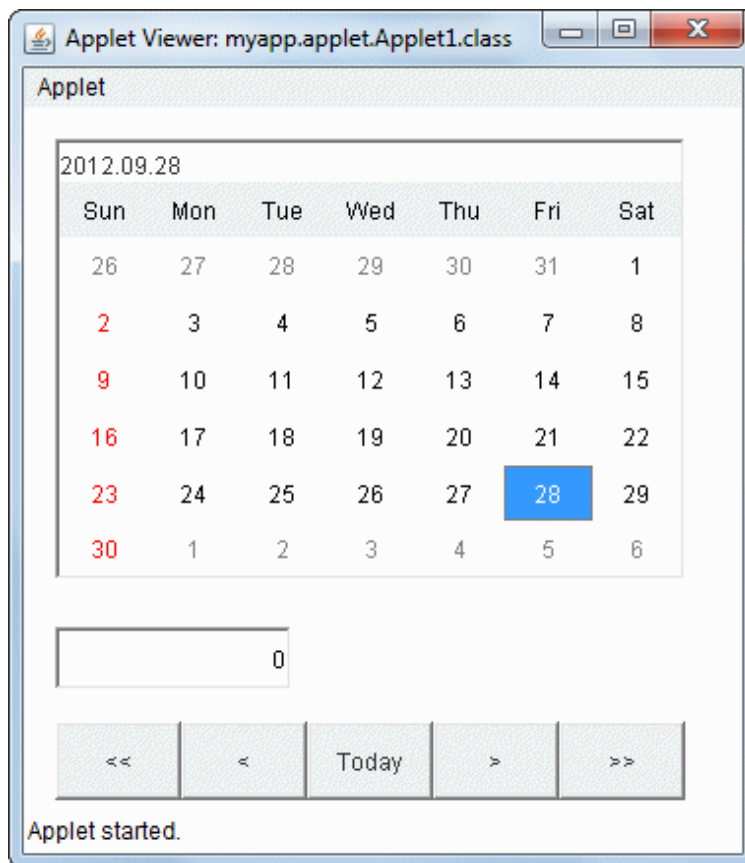
Setup Items	Setup Content
Name	Applet1
Project	MyCalApplet
Applet	myapp.applet.Applet1



- Specify an applet width and height. Click the [Parameters] tab, and specify 400 in [Width] and [Height].

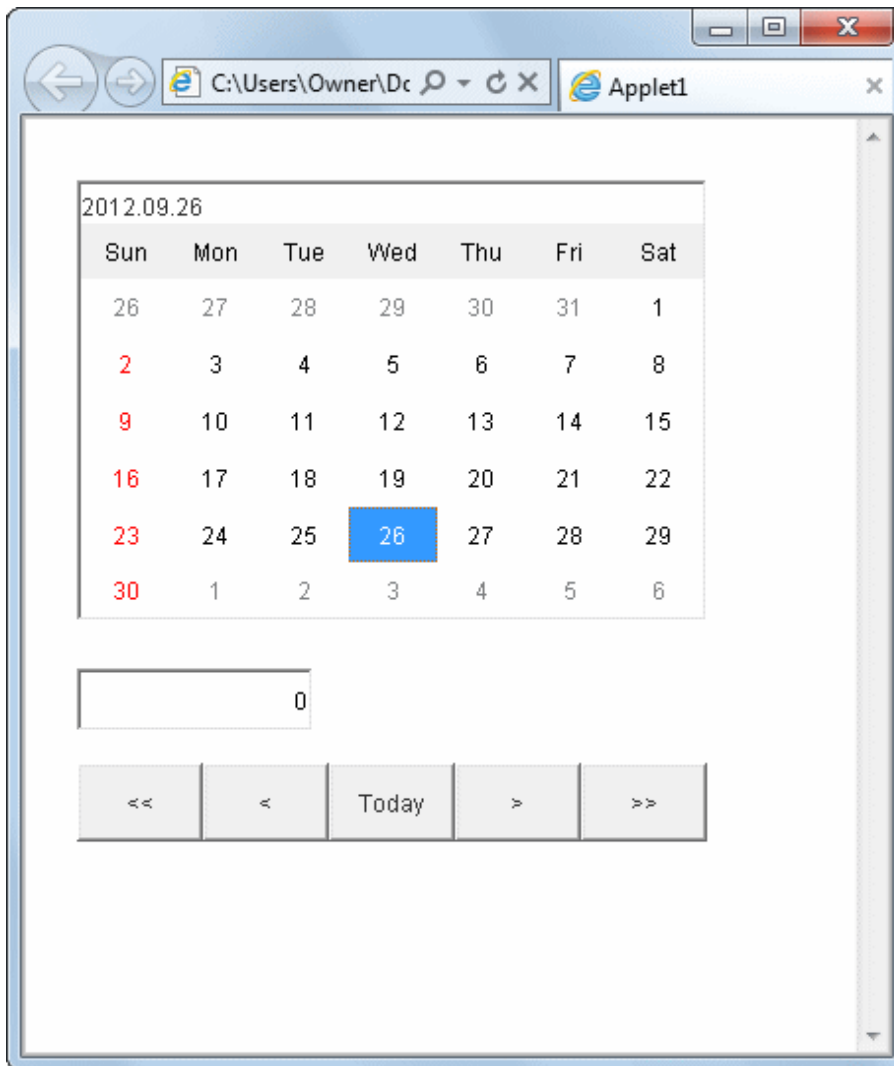


5. Click [Run]. Applet Viewer opens with the applet displayed.



6. Close Applet Viewer by clicking the [Close (X)] button on the title bar.

7. Next, confirm that the applet runs correctly in a Web browser.
To do this, open the "Applet1-JBKPlugin.html" file stored in the project folder in Internet Explorer.



F.3.2 Developing Applet That Use Multiple Screens

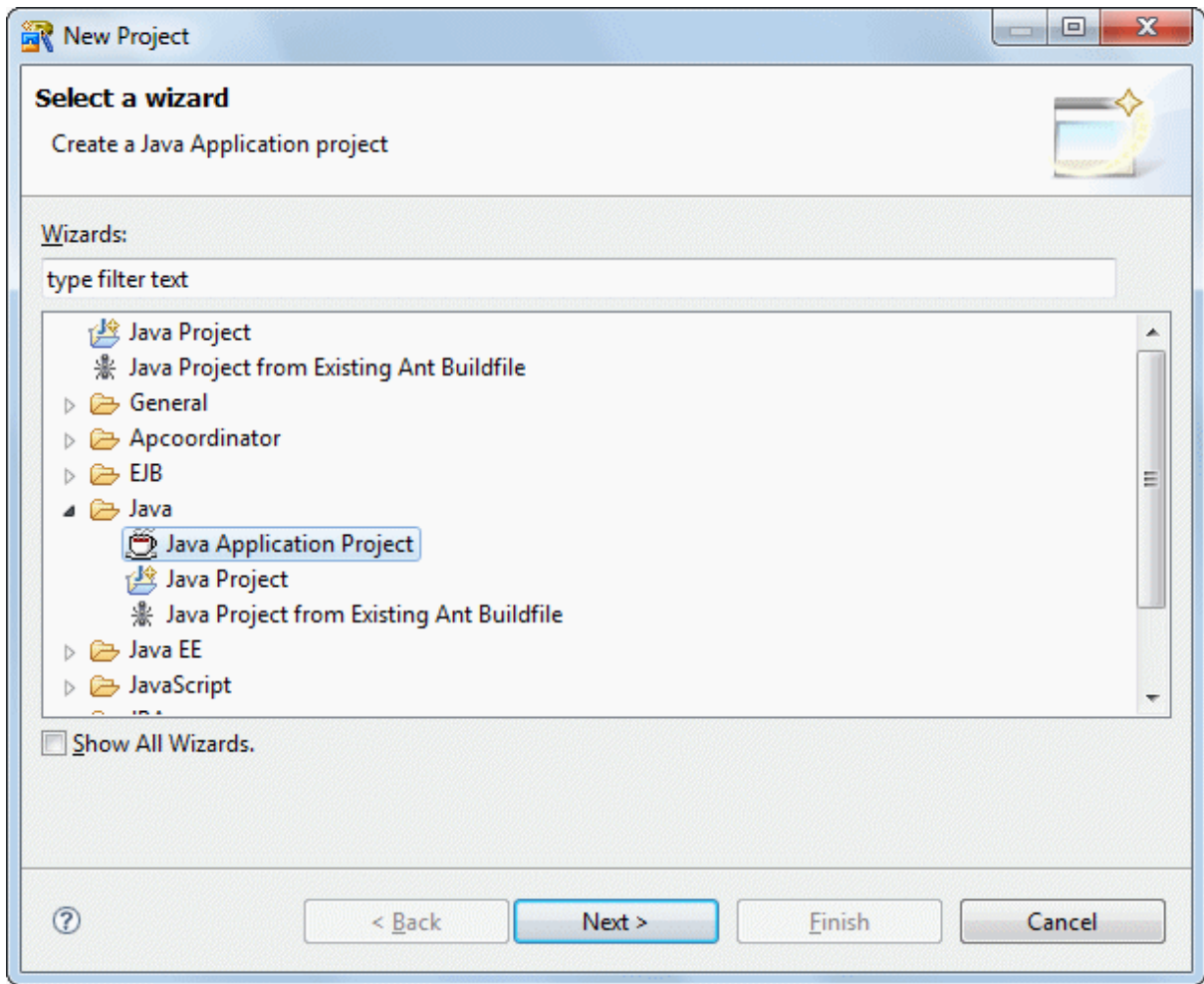
The Screen Control Library is a library for managing large numbers of screens in applications and applets concerted with window transitions. The library contains control functions for reading, displaying and deleting screens.

Follow the instructions below to develop an applet for switching between the displays of two different screens, using the Screen Control Library.

Creating a Java Application Project

1. Start the workbench.
2. Select [File] > [New] > [Project] from the menu bar.

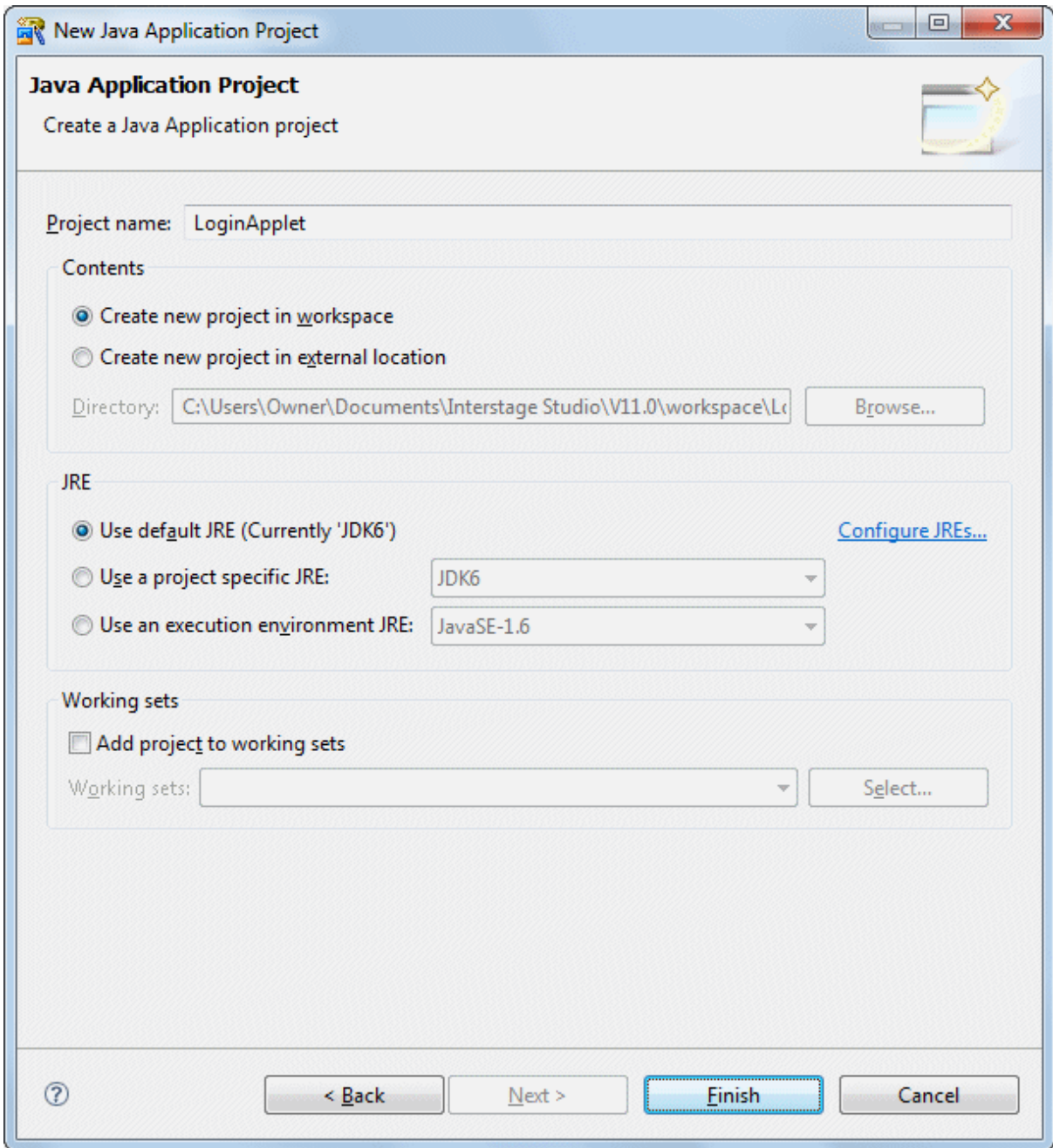
3. The [New Project] wizard is displayed. Select [Java Application Project] from the tree. Then, click [Next].



Click [Next].

4. The [Java Application Project] page is displayed. A project is created according to the information entered for items on this page. Enter information as follows.

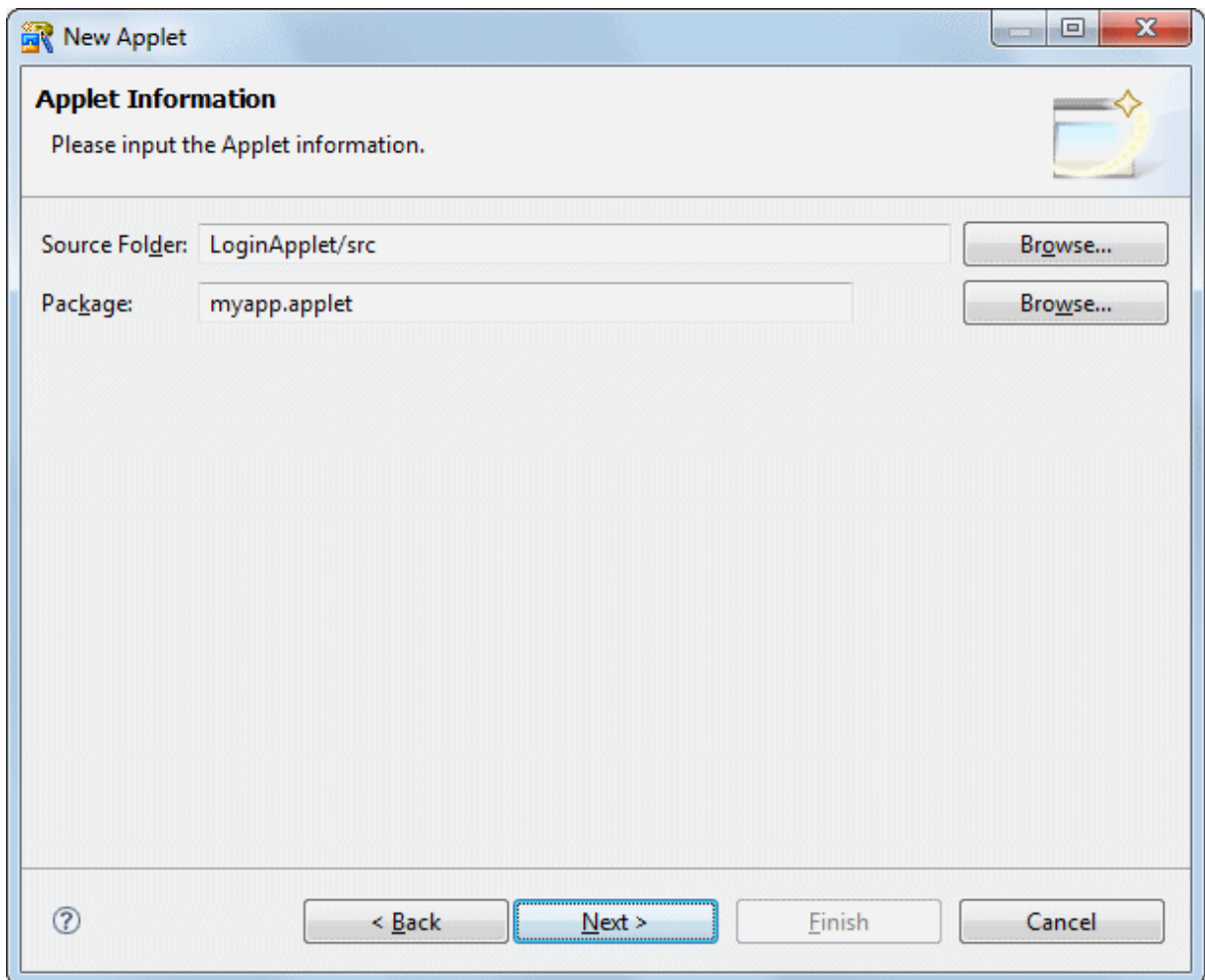
Setup Items	Setup Content
Project name	LoginApplet



Click [Finish].

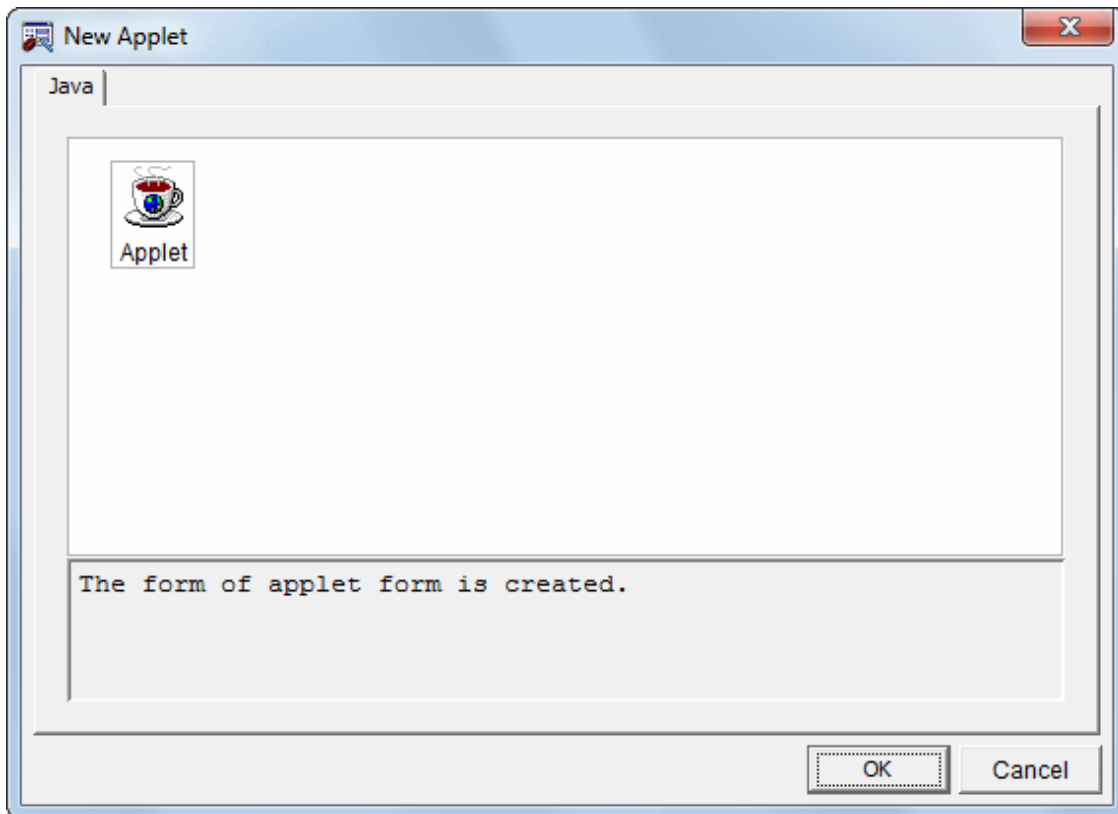
5. The next step is to create an applet. Select [File] > [New] > [Other] from the menu bar in workbench. Then, select [Java] > [GUI] > [Applet] from the tree in the [New] wizard that appears.
6. The [Applet Information] page is displayed. Enter information as follows.

Setup Items	Setup Content
Package	myapp.applet

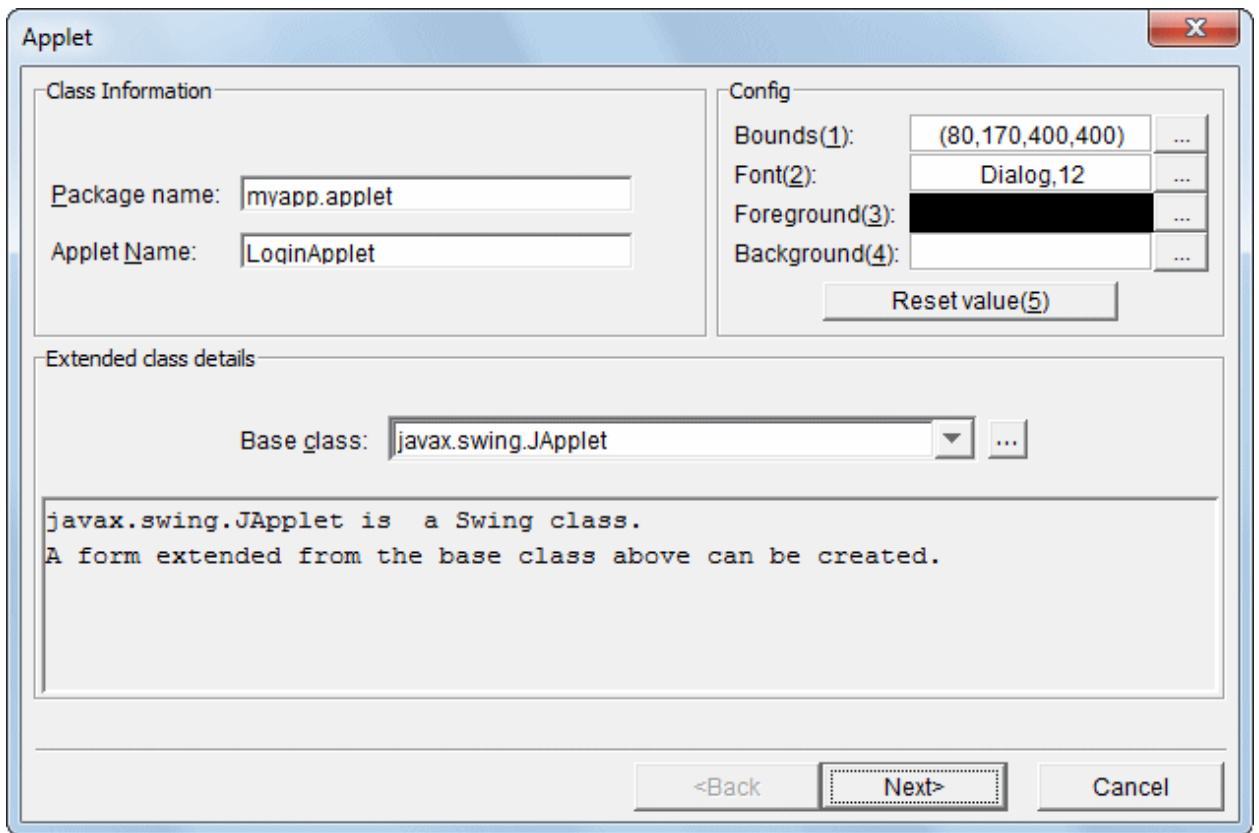


Click [Next].

7. The [New Applet] dialog box is displayed.
Select [Applet] on the [Java] tab, and then click [OK].



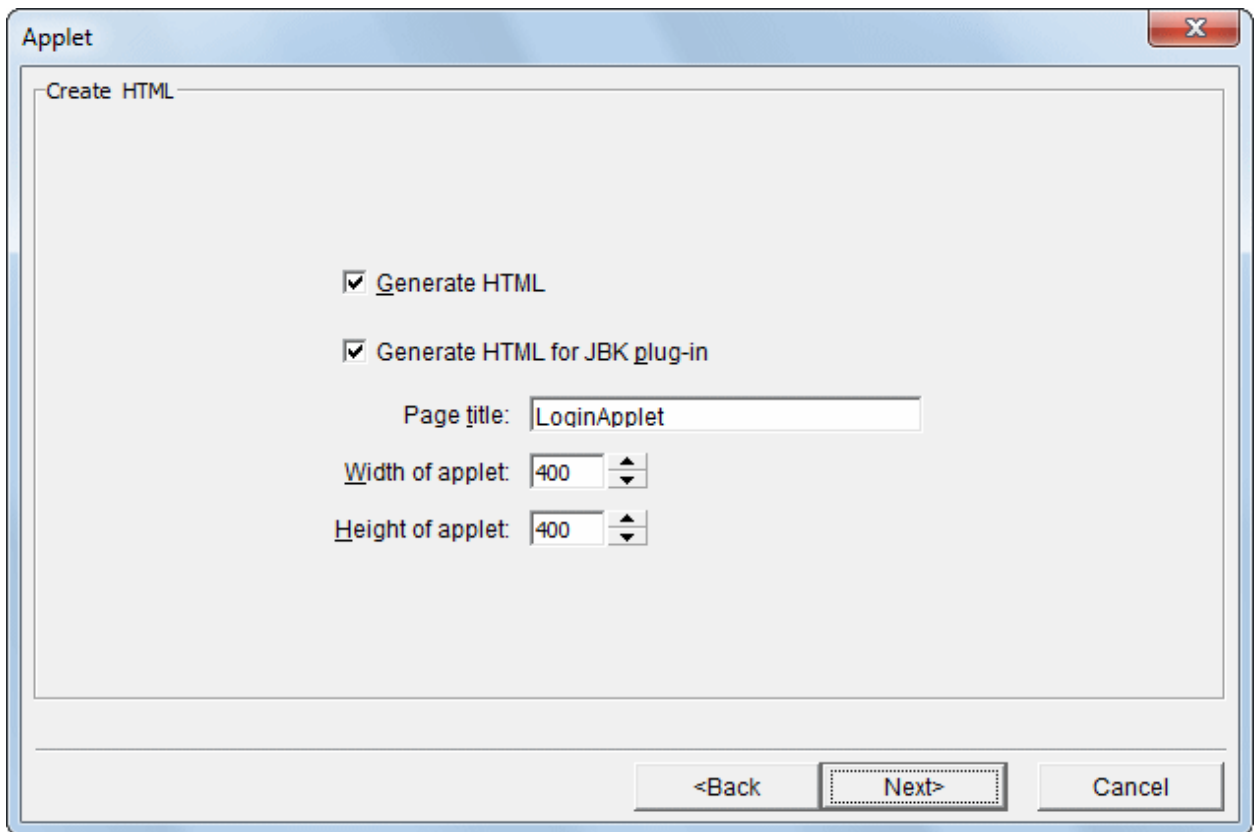
- The page for entering applet class information is displayed.
Enter "LoginApplet" in the [Applet Name] field.
Select [javax.swing.JApplet] as the base class.



Click [Next].

- The [Create HTML] page is displayed. You can create an HTML template in which the applet is pasted.
Specify the following settings:

Setup Items	Setup Content
Generate HTML	Checked
Generate HTMLfor JBK plug-in	Checked
Page title	LoginApplet
Width of applet	400
Height of applet	400



Click [Next].

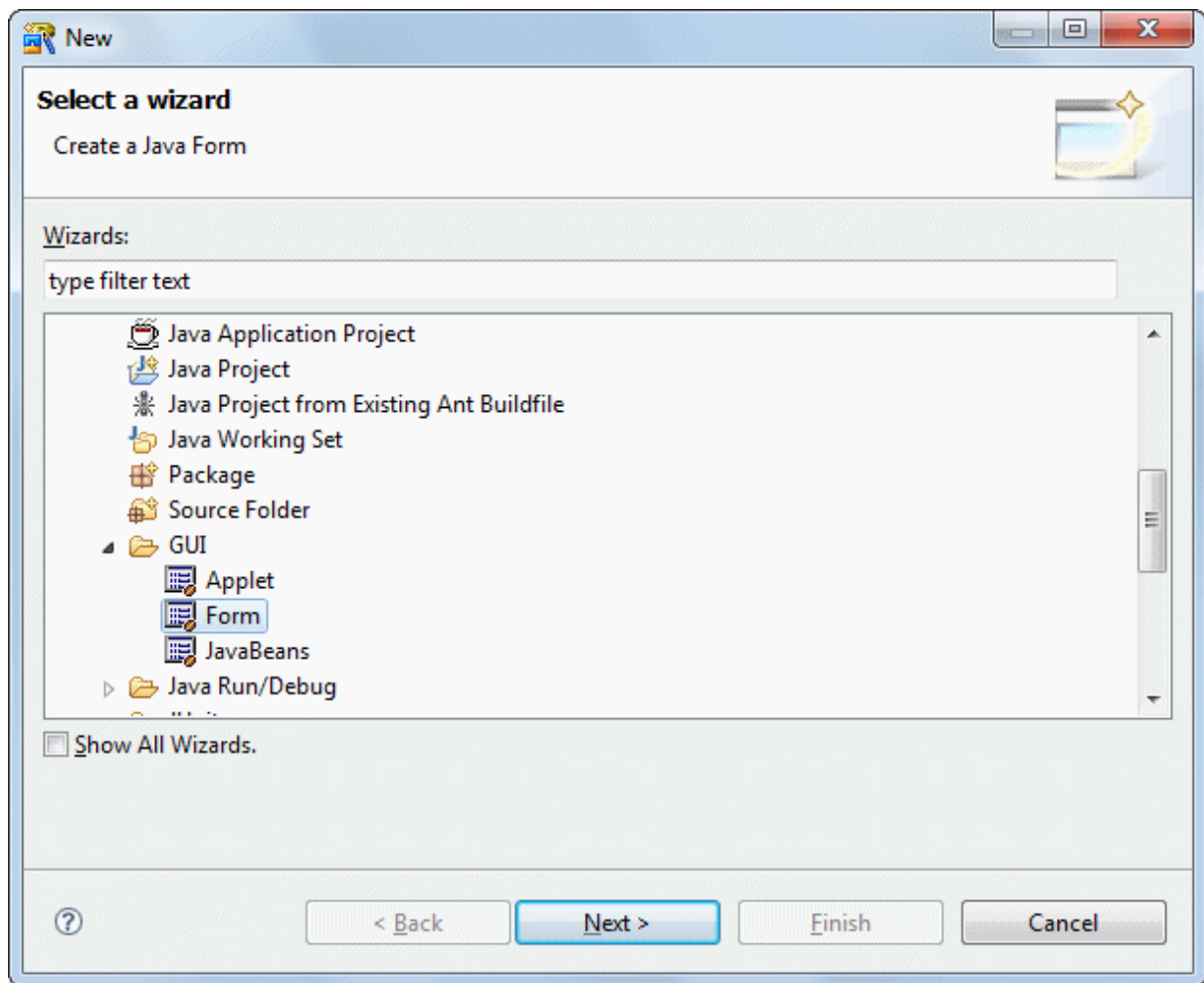
10. The [Parameter Information] page is displayed.
Parameters are not used, so you can delete "myItem0." Click [Delete]. Then, click [Create].
11. Java Form Designer starts.
To use the Screen Control Library, paste screen control class (extended JFLCardPanel class) into the applet, but this task is performed at a subsequent time. For this reason, close Java Form Designer.
From the [File] menu of Java Form Designer, select [Close] to close the applet form.

This is the end of the procedure for creating a project.

Creating a Screen Control Panel

1. Create the Screen Control Panel.
Select [File] > [New] > [Other] from the workbench menu bar.

2. The [New] wizard is displayed. Select [Java] > [GUI] > [Form] from the tree.



Click [Next].

3. The [Java Form Information] page is displayed. Enter information as follows.

Setup Items	Setup Content
Package	myapp.applet

New Form

Java Form Information

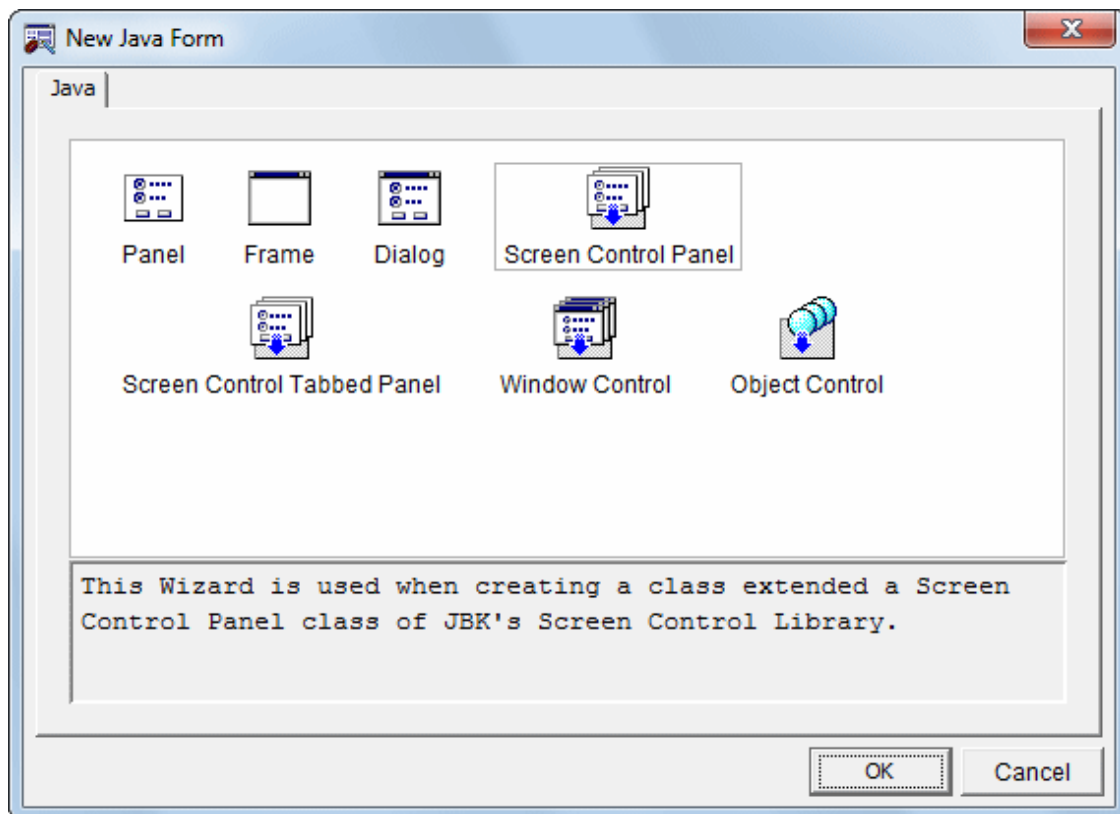
Please input the Java Form information.

Source Folder: LoginApplet/src

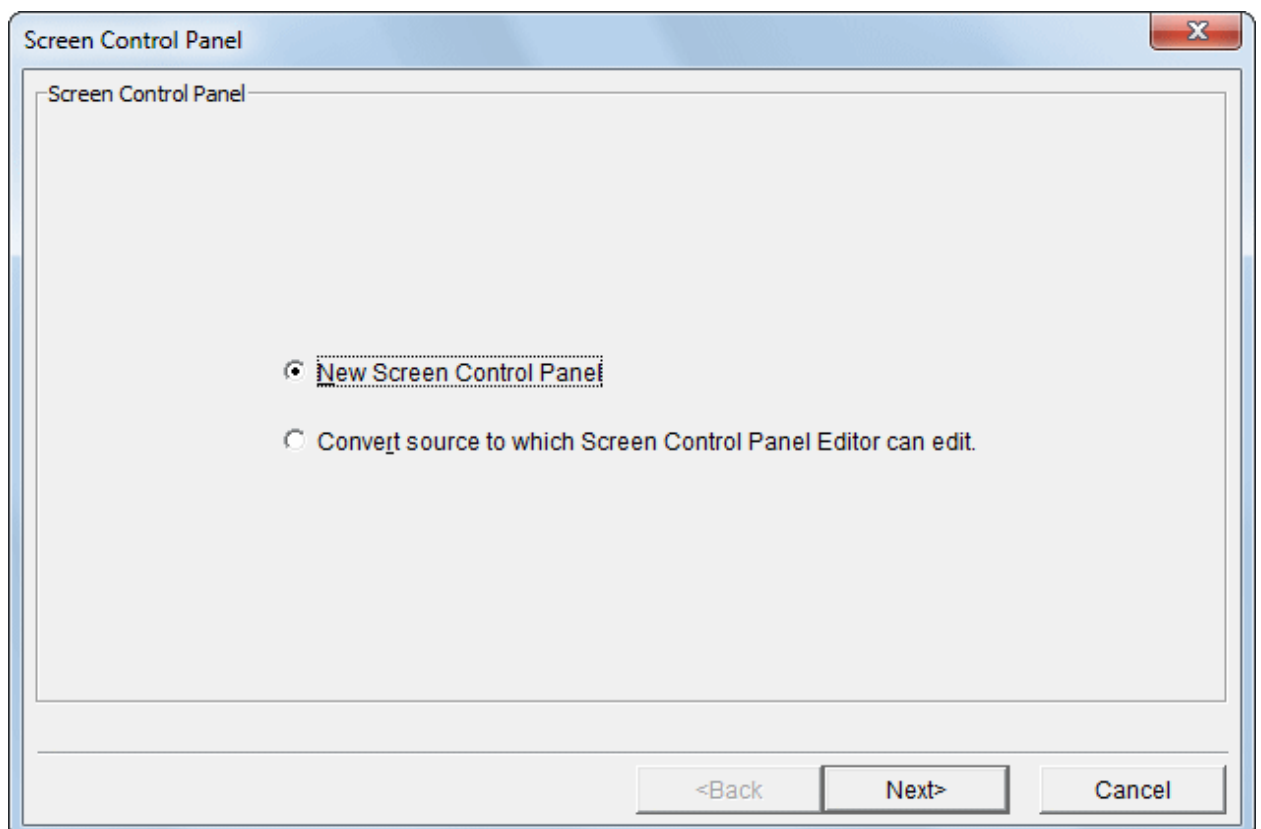
Package: myapp.applet

Click [Next].

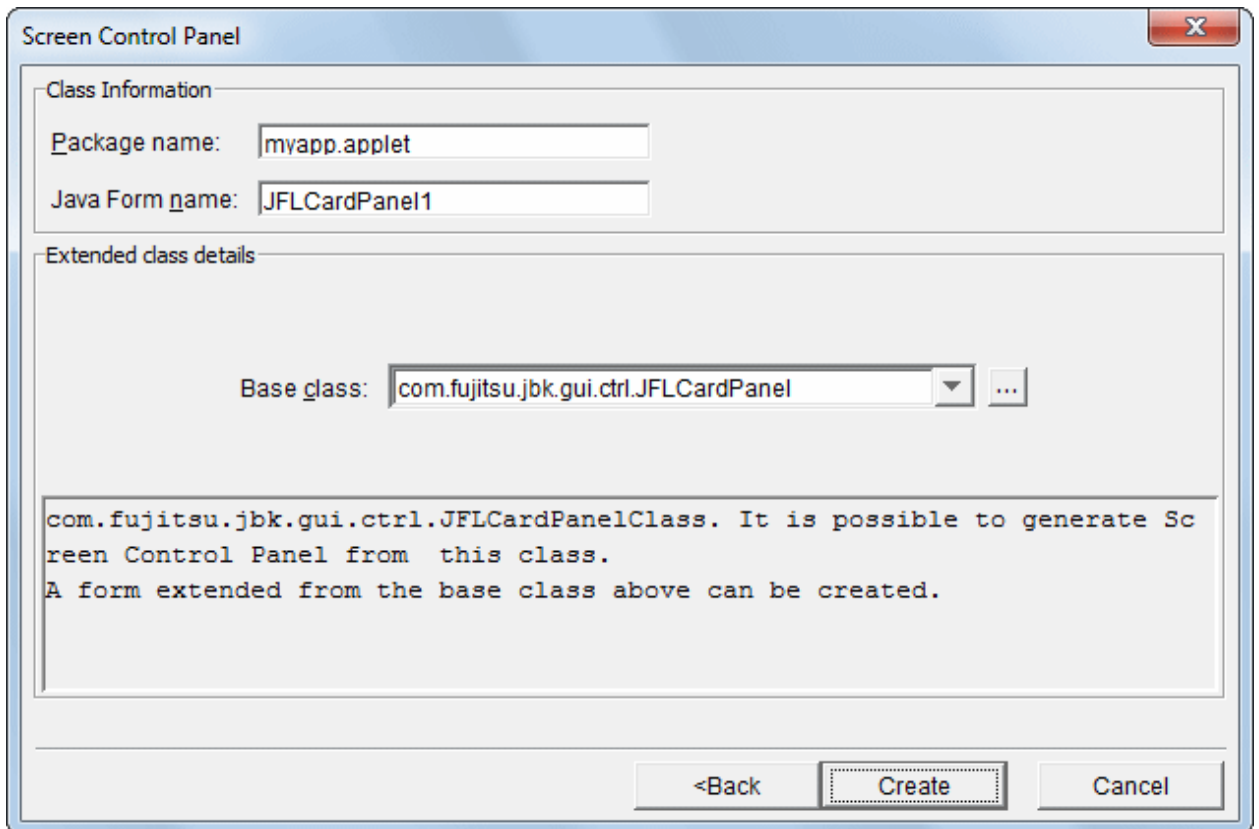
4. The [New Java Form] wizard is displayed. Select [Screen Control Panel] on the [Java] tab. Then, click [OK].



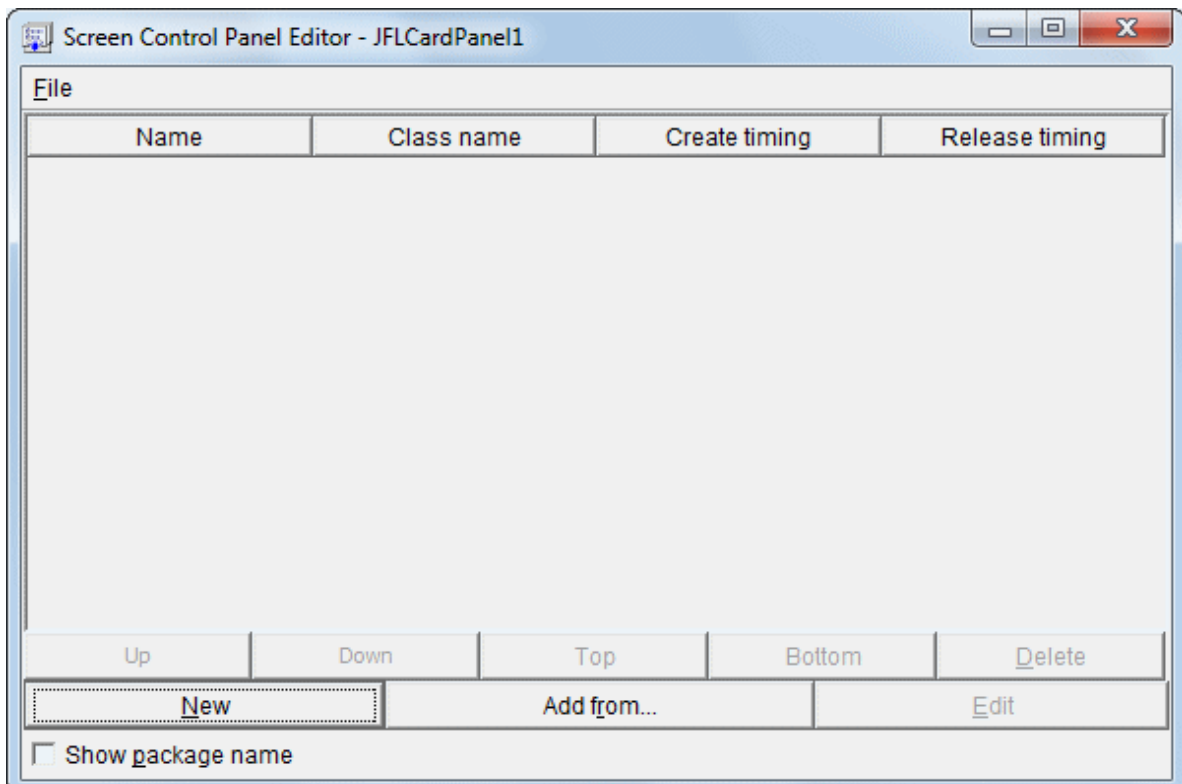
5. On the first page of the wizard, select whether to create a resource or use an existing resource. Because you are creating a panel in this example, select the [New Screen Control Panel] option. Then, click [Next].



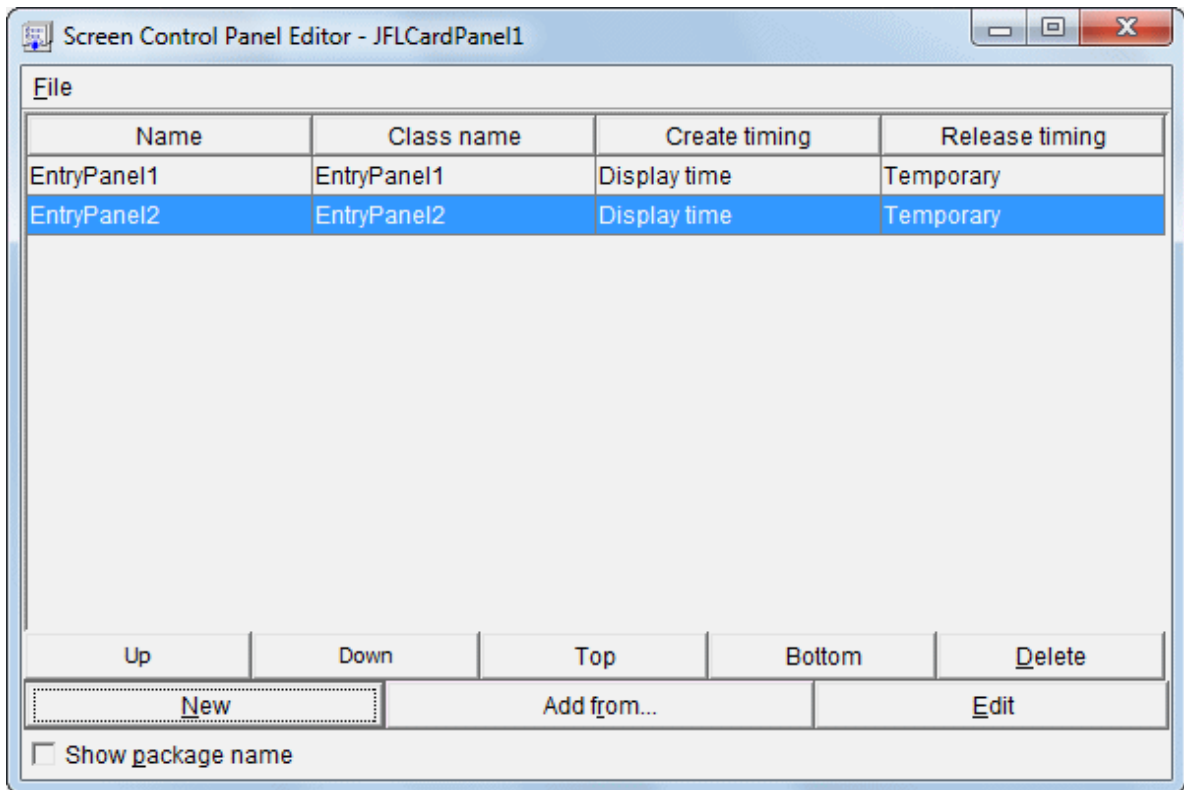
- The page for entering class information is displayed. Confirm that "JFLCardPanel1" is specified as the Java form name and "com.fujitsu.jbk.gui.ctrl.JFLCardPanel" is specified for the base class. Then, click [Create].



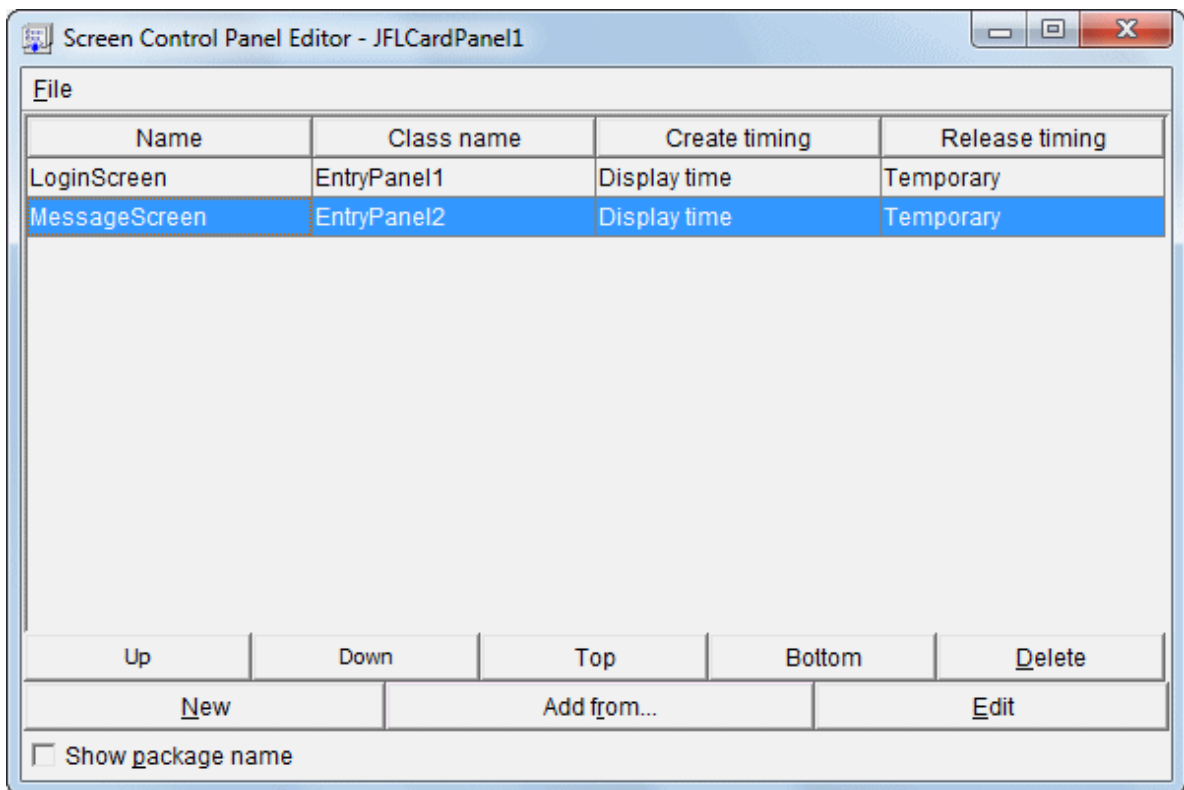
- Screen Control Panel Editor is displayed. In this window, you control such details as the setup sequence and names of screen parts, create timing, and release timing. For this project, you will use two task screens (task panels): Login screen and Message screen. Click [New] twice.



8. Two panels are added. Specify "Login screen" for "EntryPanel1" and "Message screen" for "EntryPanel2."

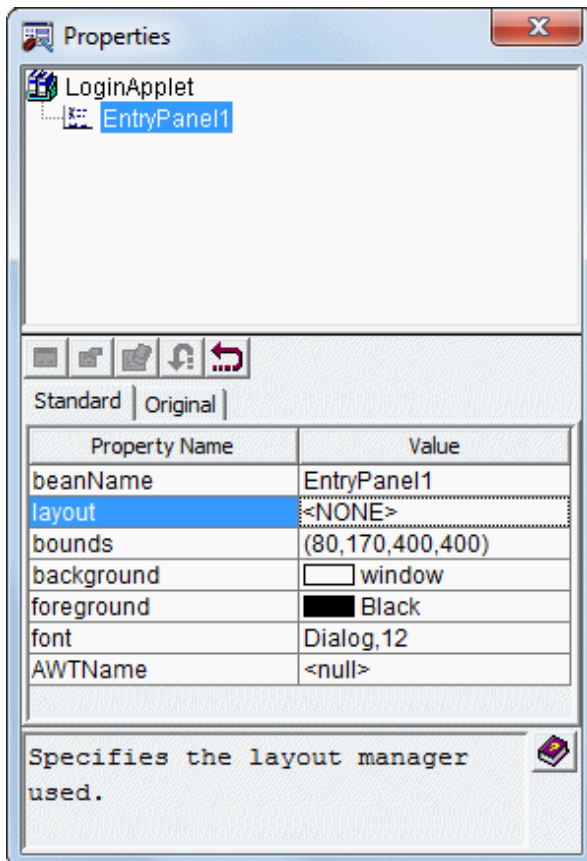


9. Click a screen name to edit it. Change "EntryPanel1" to "LoginScreen" and "EntryPanel2" to "MessageScreen".

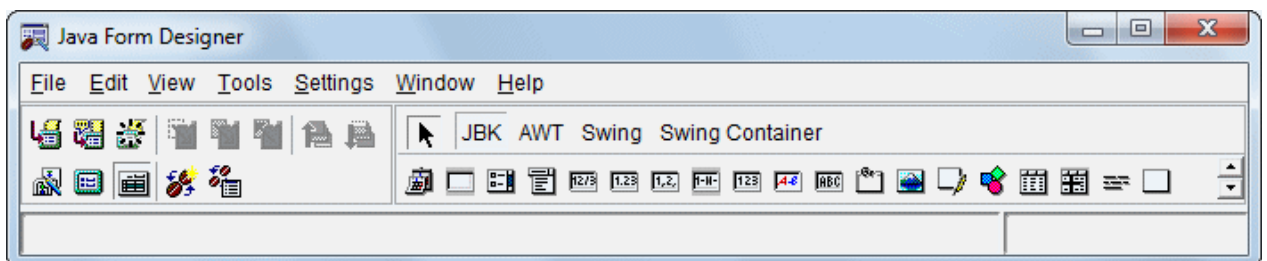


Creating a Login Screen

1. Design the Login screen. In Screen Control Panel Editor, select "LoginScreen" and click [Edit].
A confirmation message is displayed. Click [Yes].
2. Java Form Designer starts. EntryPanel1 is displayed.
In its initial state, Layout Manager for EntryPanel1 is set to "FlowLayout." FlowLayout places components in sequence from left to right. Because you want to place Beans freely, release the Layout Manager setting.
From the Java Form Designer [View] menu, select [Properties] if the Properties window is not already displayed.
Change the Layout Manager setting from "FlowLayout" to "<NONE>."



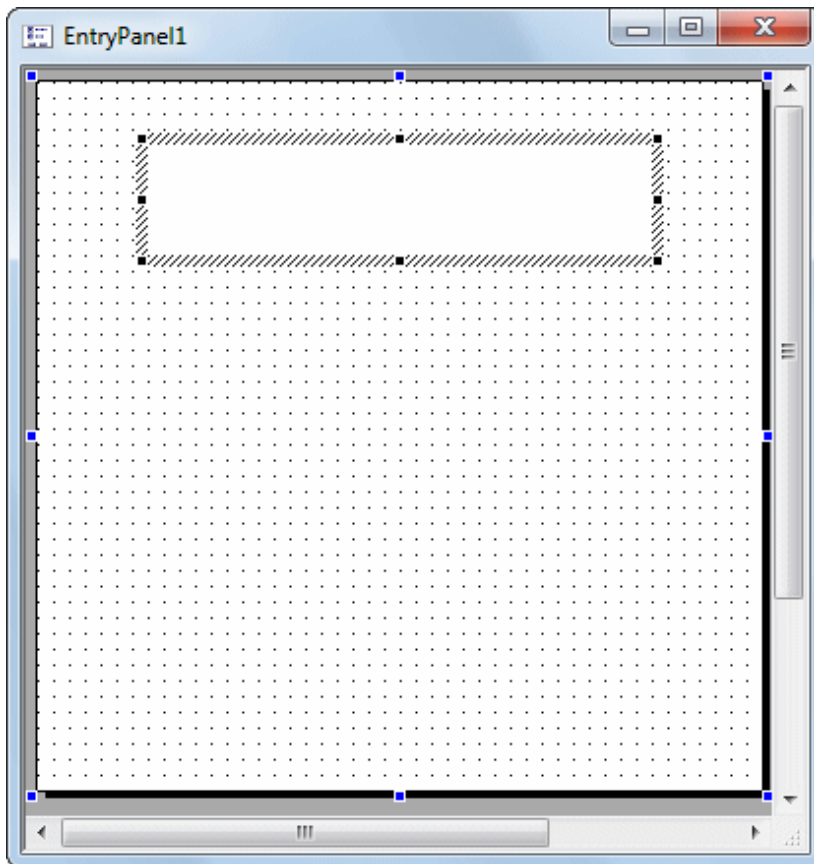
3. Place Beans.
First, place the title character string "Login screen".
If [JBK] is not already selected in the Object Palette, click [JBK].



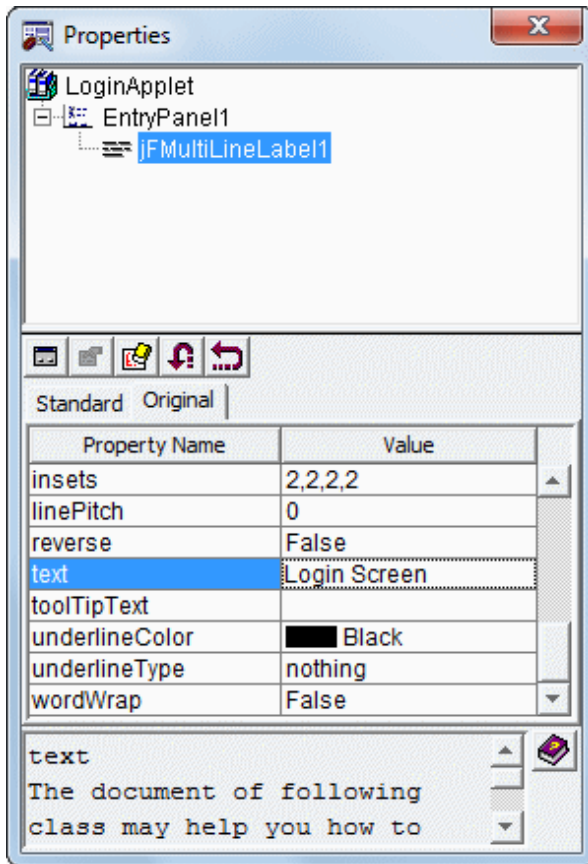
Click [Multiple-Line Label] in the Object Palette.

Place it in the Java form. In the Java form, click on the location where you want to paste the Bean.

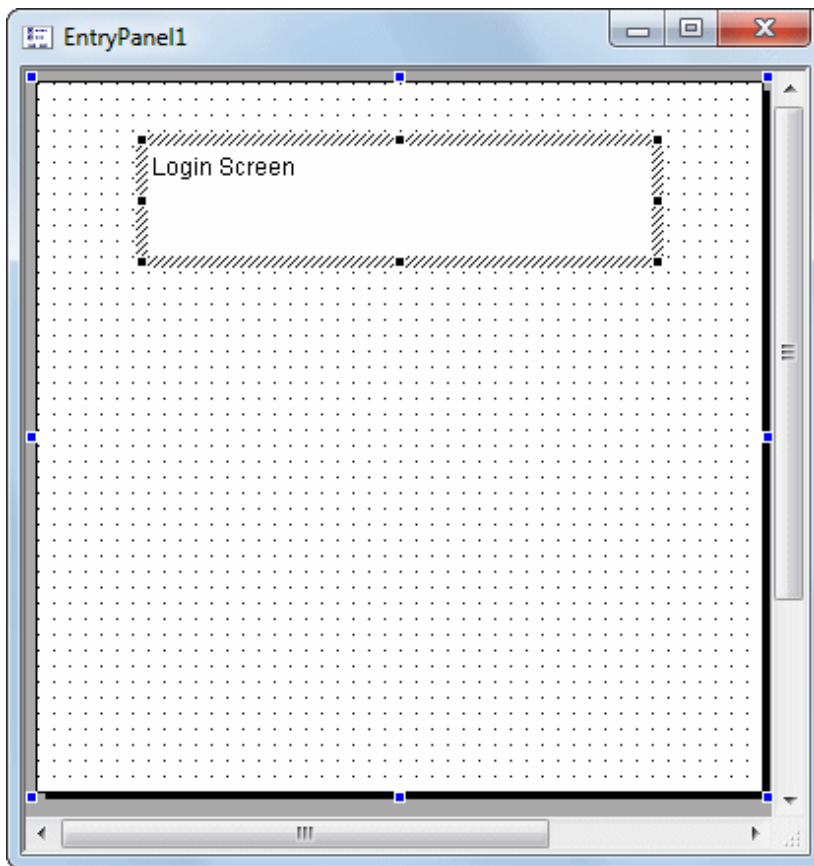
Drag the mouse to select the area to be filled with the Bean, and release the left mouse button at the appropriate point.



Specify the character string to be displayed.
With the Bean selected, click the [Original] tab in the Properties window.

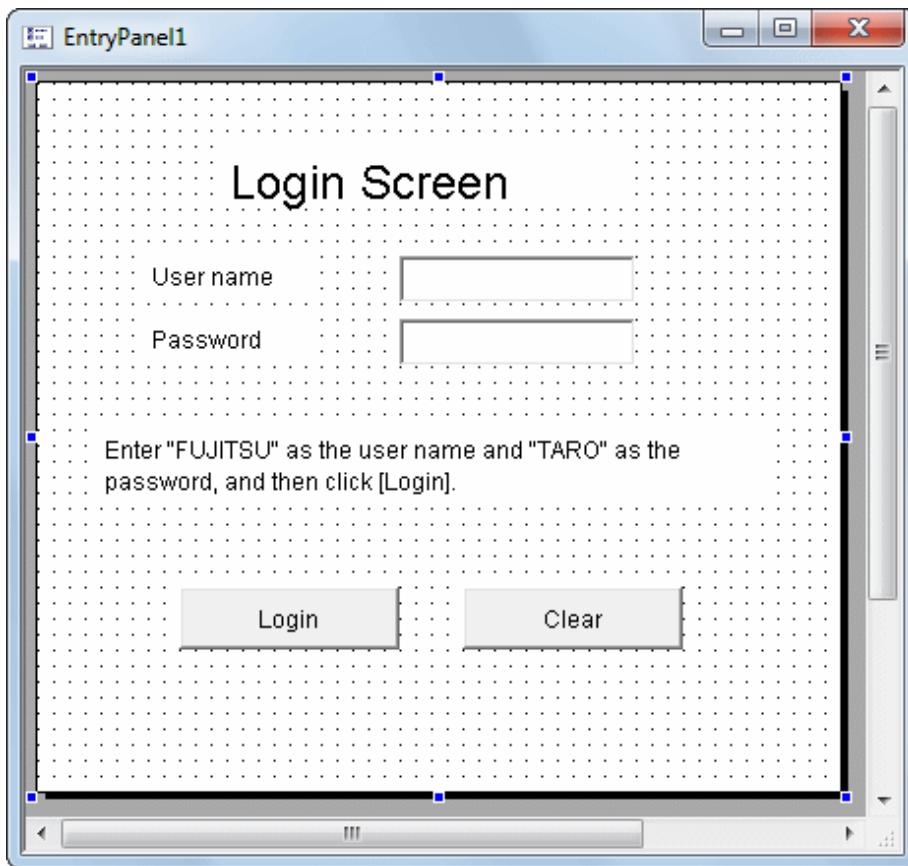


Enter "Login screen" for the text property value.



Place any other Beans as necessary, and define appropriate properties. Standard settings for a Bean rectangle are listed below. Your settings may be different, depending on the priority of the Beans.

Bean type		Bean name	Rectangle (left side, top, width, height)	Character String Field	Other
JBK	Multiple-Line Label	jFMultiLineLabel 1	96,32,208,32	Login screen	Font:Dialog,24
JBK	Multiple-Line Label	jFMultiLineLabel 2	56,88,88,24	User name	
JBK	String Field	username	184,88,120,24		
JBK	Multiple-Line Label	jFMultiLineLabel 3	56,120,88,24	Password	
JBK	String Field	password	184,120,120,24		echoChar: *
JBK	Multiple-Line Label	jFMultiLineLabel 4	32,176,344,40	Enter "FUJITSU" as the user name and "TARO" as the password, and then click [Login].	wordWrap: True
JBK	Image button	jFImageButton1	72,256,112,32	Login	
JBK	Image button	jFImageButton2	216,256,112,32	Clear	

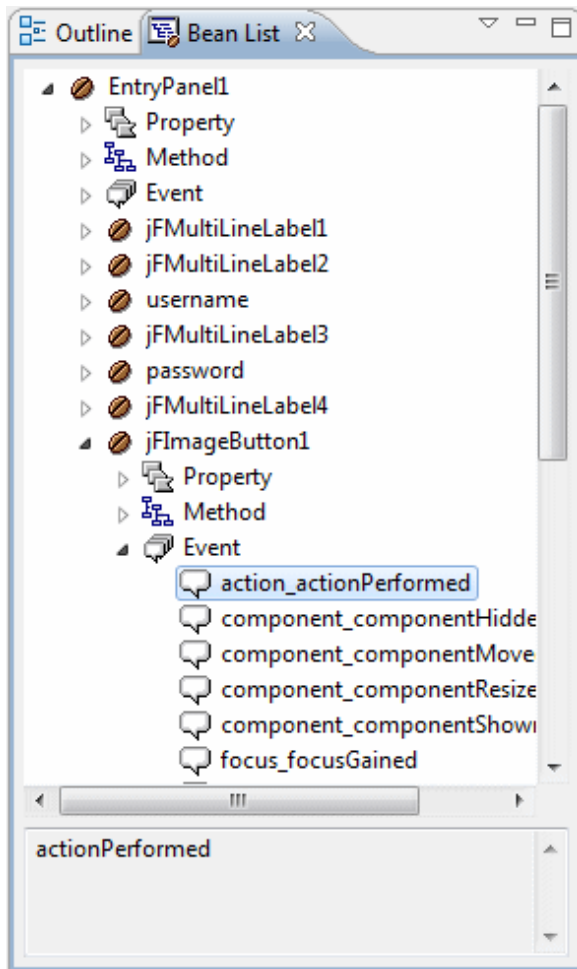


4. Describe each event process. Describe the processes by using Java editor in workbench. Click the [View] menu of Java Form Designer, and select [Java Editor] to open Java editor. Next, describe the event processes. The Hashtable class in the "java.util" package is used in the event processes, so add the import keyword at the beginning of the source. Add the text shown below in **red**.

```
import java.util.Hashtable;
```

5. For this example, describe the process that is executed when [Login] is clicked and when [Clear] is clicked. In the "action_actionPerformed" event, describe the process that is executed when a button is clicked. In order to describe the process in the "action_actionPerformed" of "jFImageButton1" expand [EntryPanel1] > [jFImageButton1] > [Event] in the Bean List view, and select "action_actionPerformed." Right-click "action_actionPerformed" to display the Context menu, and select [Insert]. Alternatively, double-click "action_actionPerformed." If the Bean List view is not displayed, select [Window] > [Show View] > [Other] on the workbench menu bar. Then, expand [Java]

from the [Show View] dialog box, and select [Bean List] to display it. A confirmation message appears. Click [Yes]. The event method is added to the source in Java editor.



Define the process in the added "jFImageButton1_action_actionPerformed\$" method when [Login] button is clicked. Use setData method of JFLEntryPanel to store screen data, and setPanel method to move to message screen. Add the text shown below in **red**.

```
private void jFImageButton1_action_actionPerformed$(java.awt.event.ActionEvent e) {
    if (!defaultEventProc$(e)) {
        // The process executed when the above event occurs is described here.
        Hashtable inputdata = new Hashtable();
        inputdata.put("username", new String(username.getText()));
        inputdata.put("password", new String(password.getText()));
        this.setData(inputdata);
        this.setPanel("MessageScreen");
    }
}
```

In the "jFImageButton2_action_actionPerformed\$" method, describe the process that is executed when [Clear] (jFImageButton2) is clicked.

Expand [jFImageButton2] > [Event] in the Bean List view. Select and right-click "action_actionPerformed." From the Context menu that is then displayed, select [Insert]. Alternatively, double-click "action_actionPerformed."

A confirmation message appears. Click [Yes].

Define the method so that when [Clear] is clicked, the two character string fields are cleared.

Add the text shown below in **red**.

```
private void jFImageButton2_action_actionPerformed$(java.awt.event.ActionEvent e) {
    if (!defaultEventProc$(e)) {
```

```

        // The procedure when the event is generated is described here.
        username.setText("");
        password.setText("");
    }
}

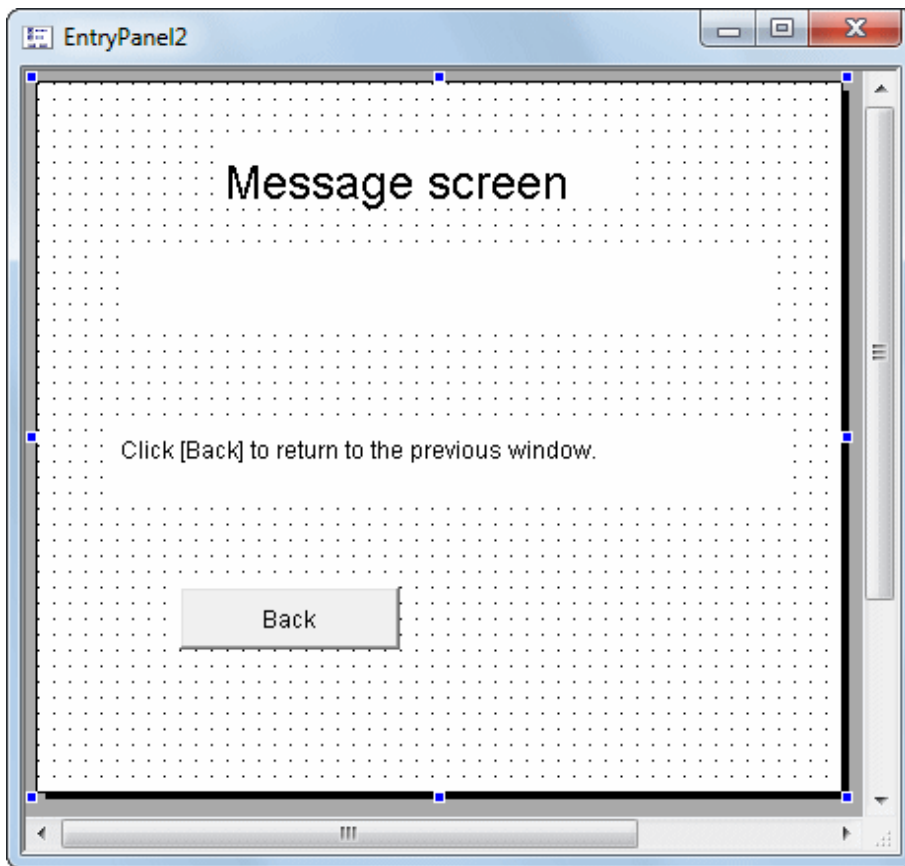
```

- This is the end of definition of the Login screen.
Select [File] > [Save] from the workbench menu bar.
To close the Java Form, select [File] > [Close] from the Java Form Designer menu bar.

Creating a Message Screen

- Next, design the Message screen. Select "MessageScreen" in Screen Control Panel Editor, and then click [Edit].
A confirmation message appears. Click [Yes].
- Java Form Designer starts, and EntryPanel2 is displayed.
Because you want to place Beans freely in this panel, release the Layout Manager setting. Change the Layout Manager setting from "FlowLayout" to "<None>."
- Place Beans, and define appropriate properties. Standard settings for a Bean rectangle are listed below. Your settings may be different, depending on the priority of the Beans.

Bean type		Bean name	Rectangle (left side, top, width, height)	Character String Field	Other
JBK	Multiple-Line Label	jFMultiLineLabel1	93,32,208,32	Message screen	Font:Dialog,24
JBK	Multiple-Line Label	msg	48,88,328,40		
JBK	Multiple-Line Label	jFMultiLineLabel2	40,176,344,40	Click [Back] to return to the previous window.	
JBK	Image button	jFImageButton1	72,256,112,32	Back	



4. Describe the process so that Login screen contents are read and a message is displayed at the time of a screen transition. The Hashtable class in the "java.util" package is used, so add the import keyword at the beginning of the source. Add the text shown below in **red**.

```
import java.util.Hashtable;
```

Next, use the JFLEntryPanel getData method to refer to the screen data corresponding to the screen.

Define the method so that the entered user name and password are verified and the appropriate message is defined in Multiple-Line Label.

Describe the process shown below at the end of the class file.

```
public void exposedPanel(String lastPanelName) {
    Hashtable inputdata = (Hashtable) this.getData("LoginScreen");
    String username = (String) inputdata.get("username");
    String password = (String) inputdata.get("password");
    if (username.equals("FUJITSU") && password.equals("TARO")) {
        msg.setText("Hello " + username + ". You have logged in successfully.");
    } else {
        msg.setText("Incorrect user name or password ");
    }
}
```



Point

Change prohibition part which is managed by Java Form Designer

Java forms include sources for calling screen (Bean) information and event processes. Because Java Form Designer is used to manage the sources, changing the sources is prohibited.

These sources that must not be changed are enclosed by comments in **blue**, as shown below.

```

public class EntryPanel2 extends com.fujitsu.jbk.gui.ctrl.JFLEntryPanel {
    //@Form Design Information start

    - Sources that are managed with Java Form Designer and that must not be changed -

    //@Form Design Information end

```

Next, in the "jFImageButton1_action_actionPerformed\$" method, describe the process that is executed when [Back] is clicked. Expand [EntryPanel2] > [jFImageButton1] > [Event] in the Bean List view, select and right-click "action_actionPerformed", and then select [Insert] from the Context menu that is displayed. Alternatively, double-click "action_actionPerformed." A confirmation message appears. Click [Yes]. Define the method so that when [Back] is clicked, the setPanel method is implemented for a screen transition to the Login screen. Add the text shown below in **red**.

```

private void jFImageButton1_action_actionPerformed$(java.awt.event.ActionEvent e) {
    if (!defaultEventProc$(e)) {
        // The process executed when the above event occurs is described here.
        this.setPanel("LoginScreen");
    }
}

```

- This is the end of definition of the Message screen. Select [File] > [Save] from the workbench menu bar. To close the Java Form, select [File] > [Close] from the Java Form Designer menu bar.
- Close Screen Control Panel Editor. Select [File] > [Exit] on the Screen Control Panel Editor menu bar.

Pasting a Screen Control Class into an Applet

- The screen control class (Screen Control Panel) is pasted into an applet by implementing the add method in the initUser\$ method of the LoginApplet class. Describe the process in Java editor in workbench. Add the text shown below in **red** in the initUser\$() method of the LoginApplet class.

```

protected void initUser$ () {
    // User definition initialization is described here.
    JFLCardPanell cardPanel = new JFLCardPanell();
    cardPanel.setBounds(0,0,400,400);
    getContentPane().add(cardPanel);
}

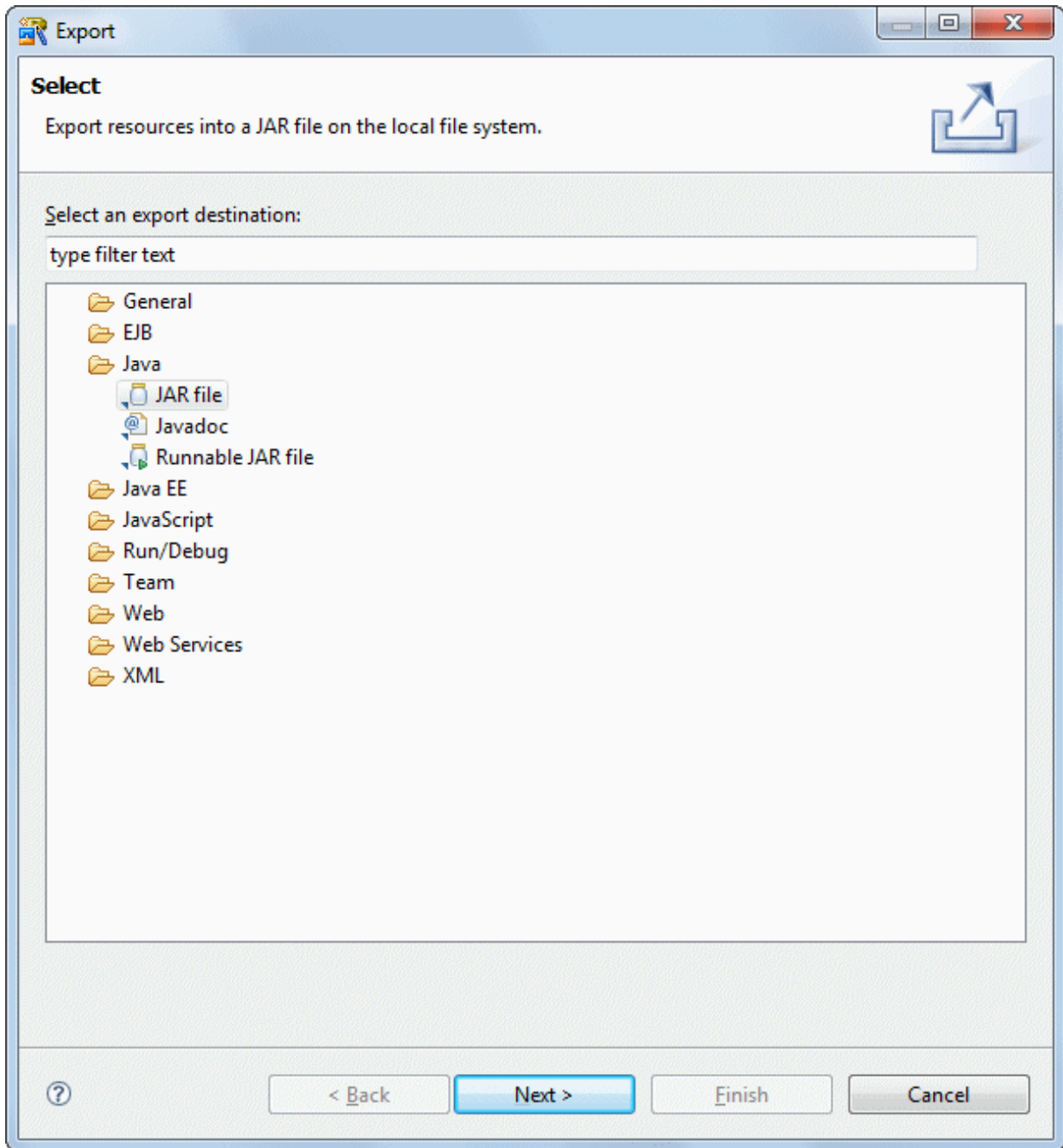
```

- Save the LoginApplet class. Select [File] > [Save] on the workbench menu bar.

Building

- Building is executed automatically when resource files (such as the Java files) are saved if the [Build Automatically] item is enabled in the [Project] menu. If [Build Automatically] is disabled, right-click on the project and select [Build Project] or select [Build Project] in the [Project] menu.
- A JAR file is required so create it. Use the export wizard to create the JAR file.

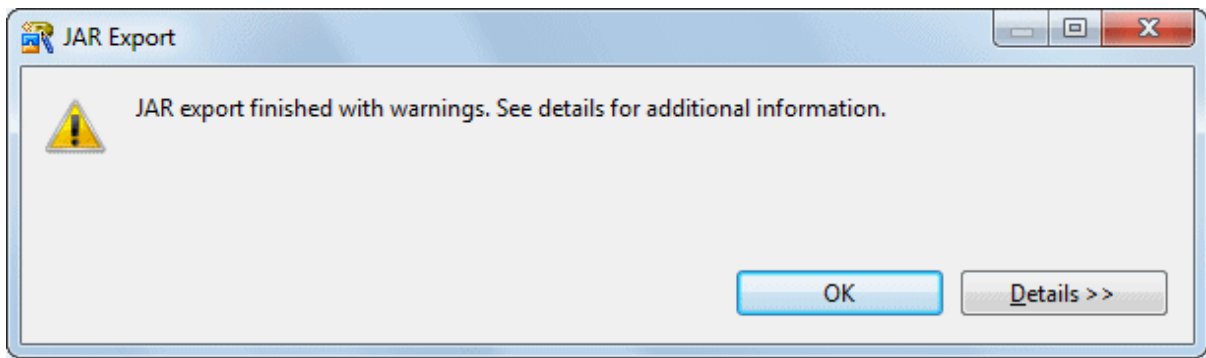
Select [File] > [Export] to start the export wizard.
 Select [Java] > [JAR file] in the export wizard.



- The JAR export wizard is displayed.
 Select [LoginApplet] > [src] folder in [Select the resources to export] and enter the following settings.
 After setting the information, click [Finish].

Setup Items	Setup Content
Select the resources to export	src/
Export generated class files and resources	Checked
JAR file	LoginApplet/LoginApplet.jar
Compress the contents of the JAR file	Checked

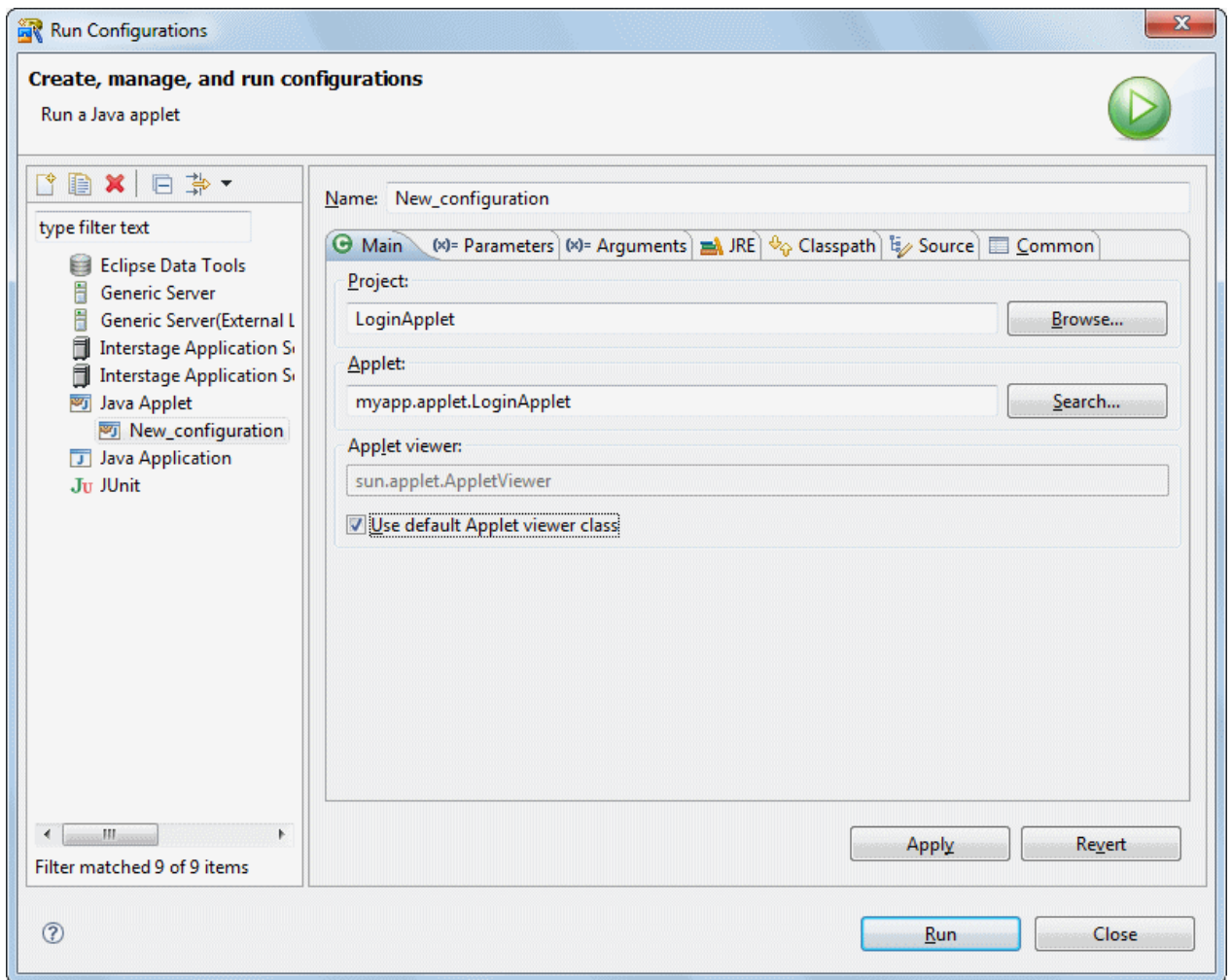
The following message is output when exporting JAR, but this does not indicate a problem.



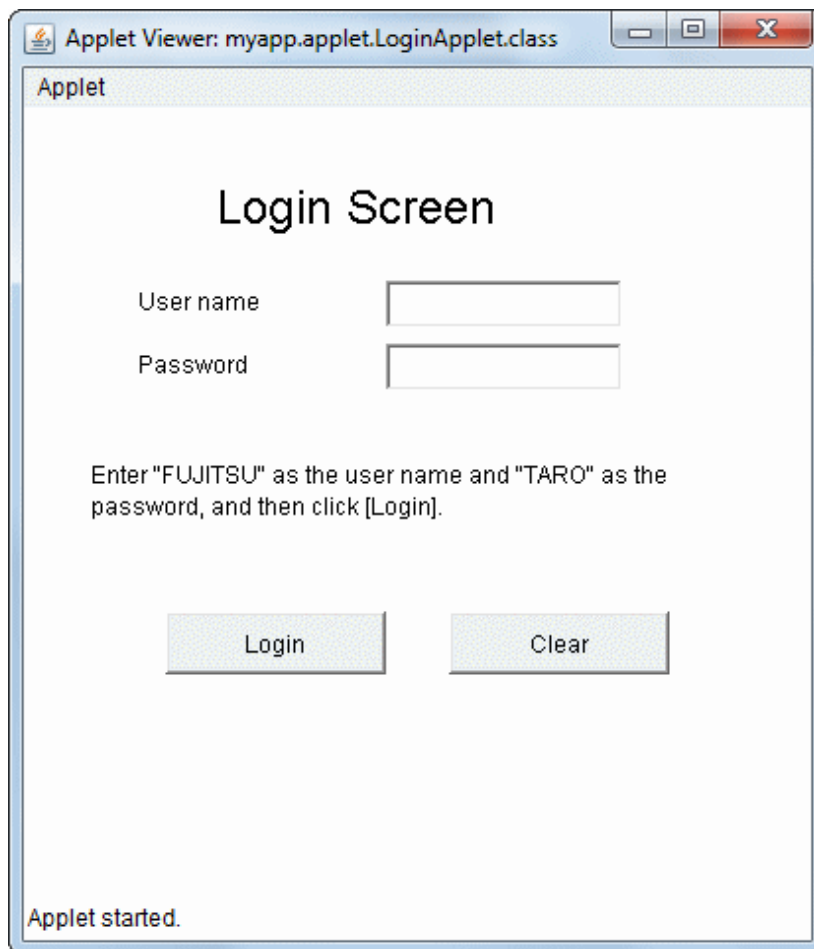
Running

1. Select the file (class) to be run. Then, click [LoginApplet] > [src] > [myapp.applet] > [LoginApplet.java] in the [Package Explorer] view in workbench.
2. Create a launch configuration in the same way as described in Running of "F.3.1 Developing Applet", and run the project. Enter the following for settings:

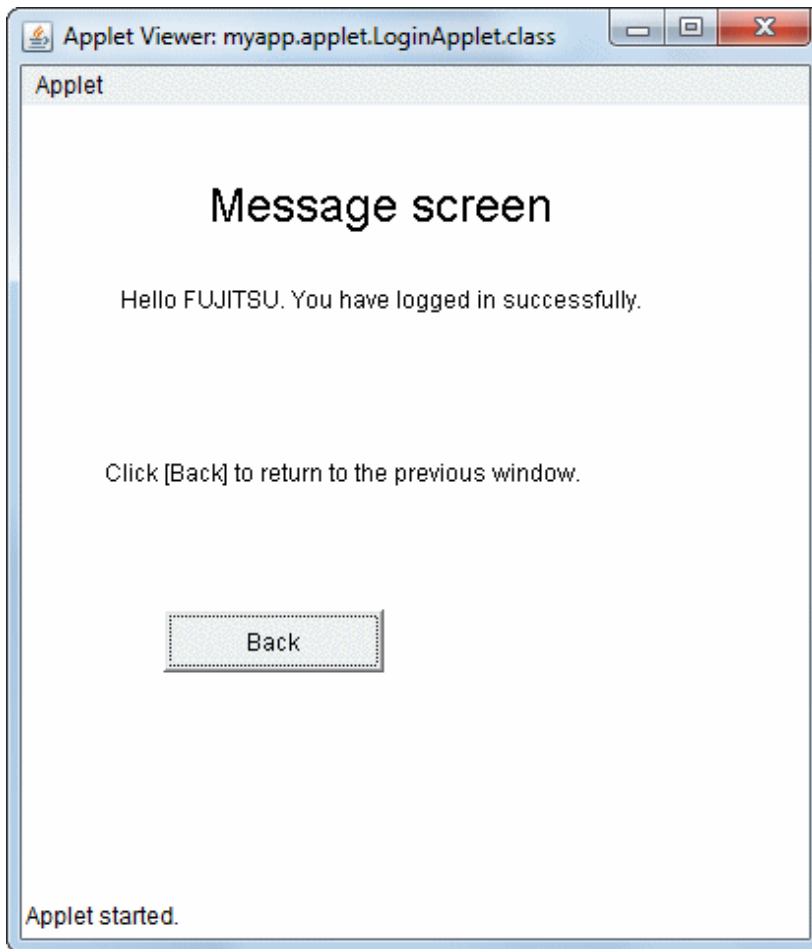
Setup Items	Setup Content
Project	LoginApplet
Applet	myapp.applet.LoginApplet
Parameters Width	400
Parameters Height	400



3. Clicking [Run] causes Applet Viewer to start, and the applet runs in it.

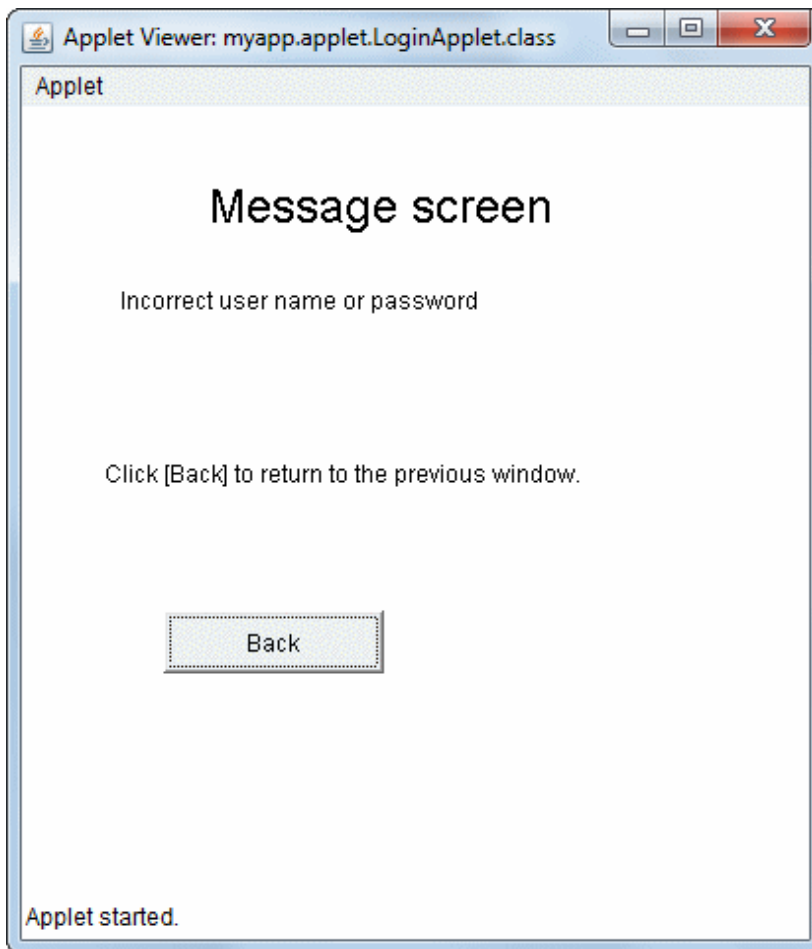


4. Enter "FUJITSU" as the user name and "TARO" as the password, and then click [Login]. The Message screen is displayed.



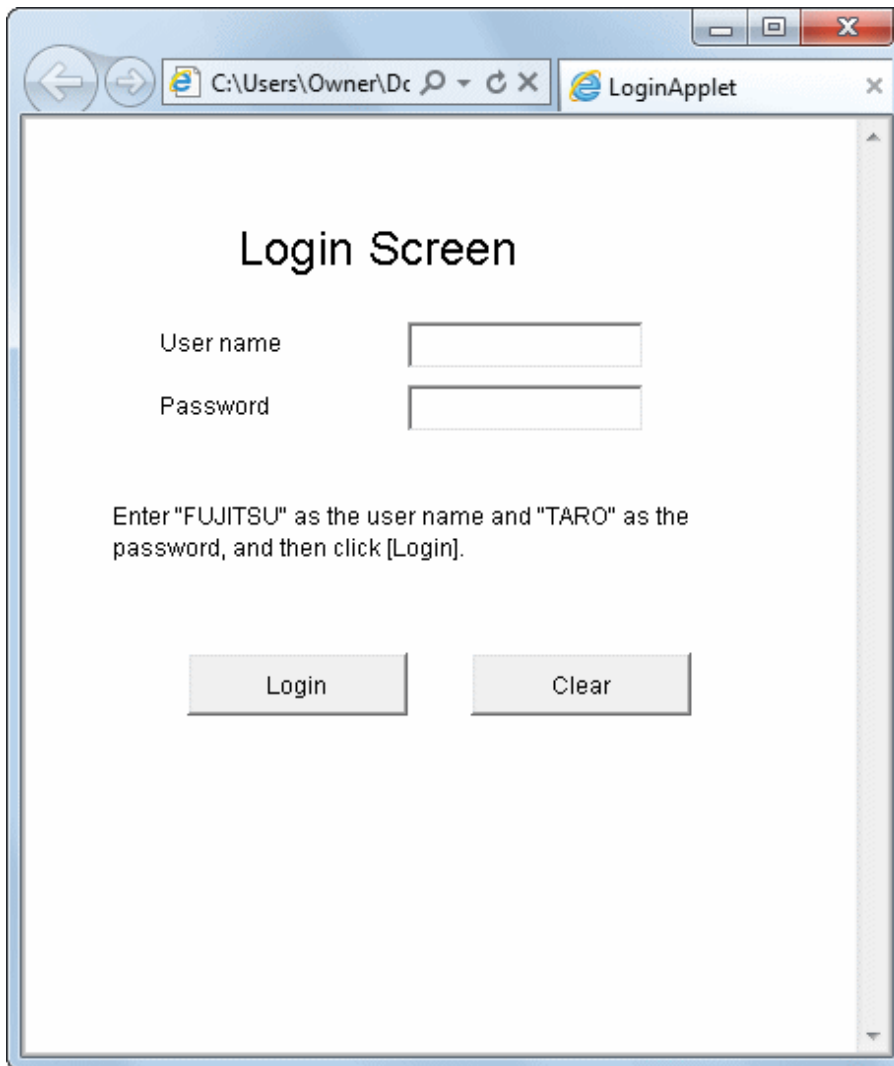
You can return to the Login screen by clicking [Back].

The following message is displayed if a user name other than "FUJITSU" and password other than "TARO" are input.



5. Close Applet Viewer by clicking the [Close (X)] button on the title bar.

- Next, confirm that the applet runs correctly in a Web browser.
To do this, open the "LoginApplet-JBKPlugin.html" file stored in the project folder in Internet Explorer.



F.4 JavaBeans

You will learn how to develop JavaBeans, by creating a component (JavaBeans) that is a combination of the JBK-provided integer field Bean and spin button Bean.

F.4.1 Developing JavaBeans

In workbench, you can develop JavaBeans, which is a type of Java application.

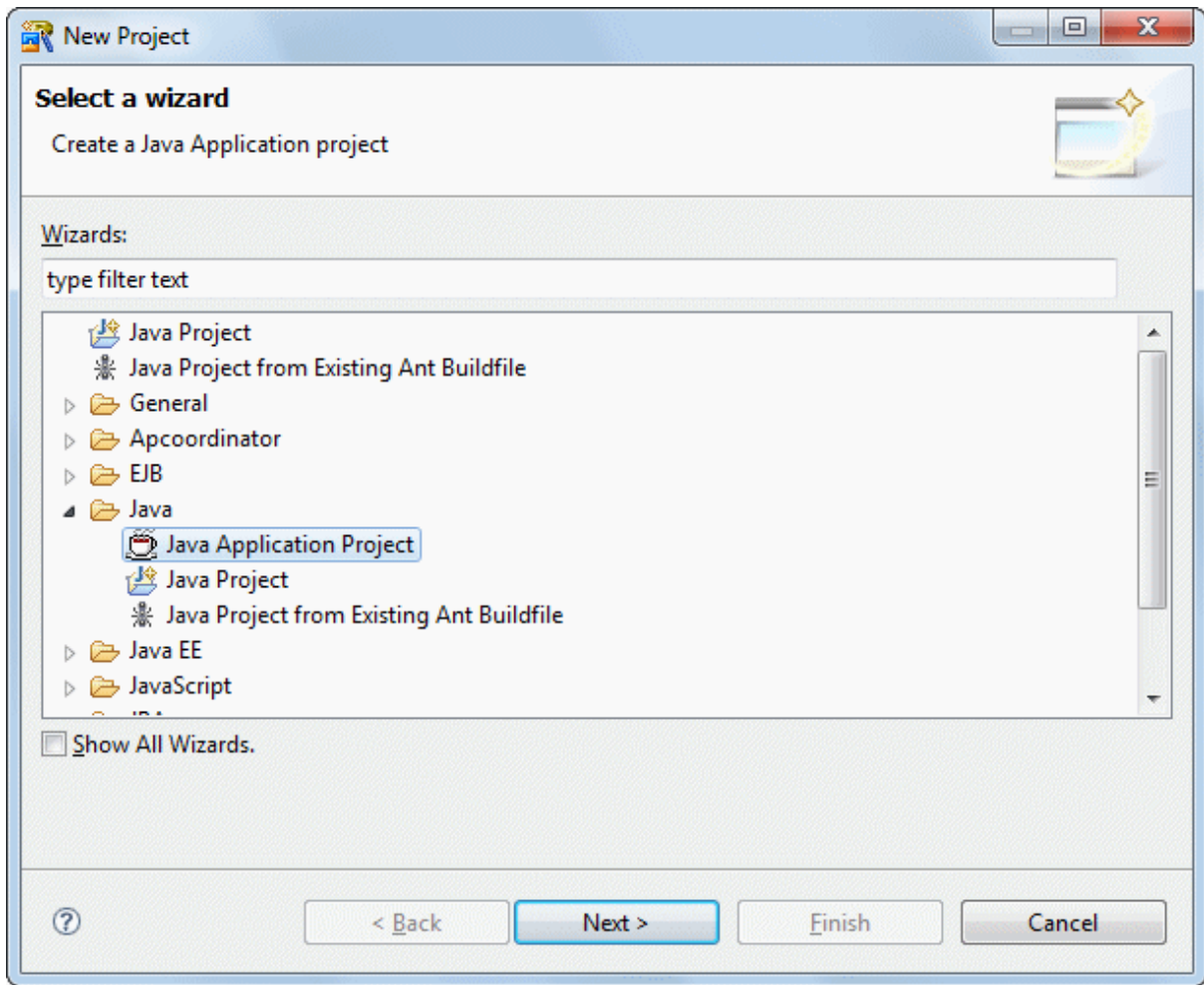
JavaBeans is a standard that is the basis for creating components of Java classes, and it is a generic name for these components. Each of these components is referred to as a Bean.

This lesson describes how to combine a JBK integer field and spin button to develop a new Bean.

Creating a Java Application Project

- Start the workbench.
- Select [File] > [New] > [Project] from the menu bar.

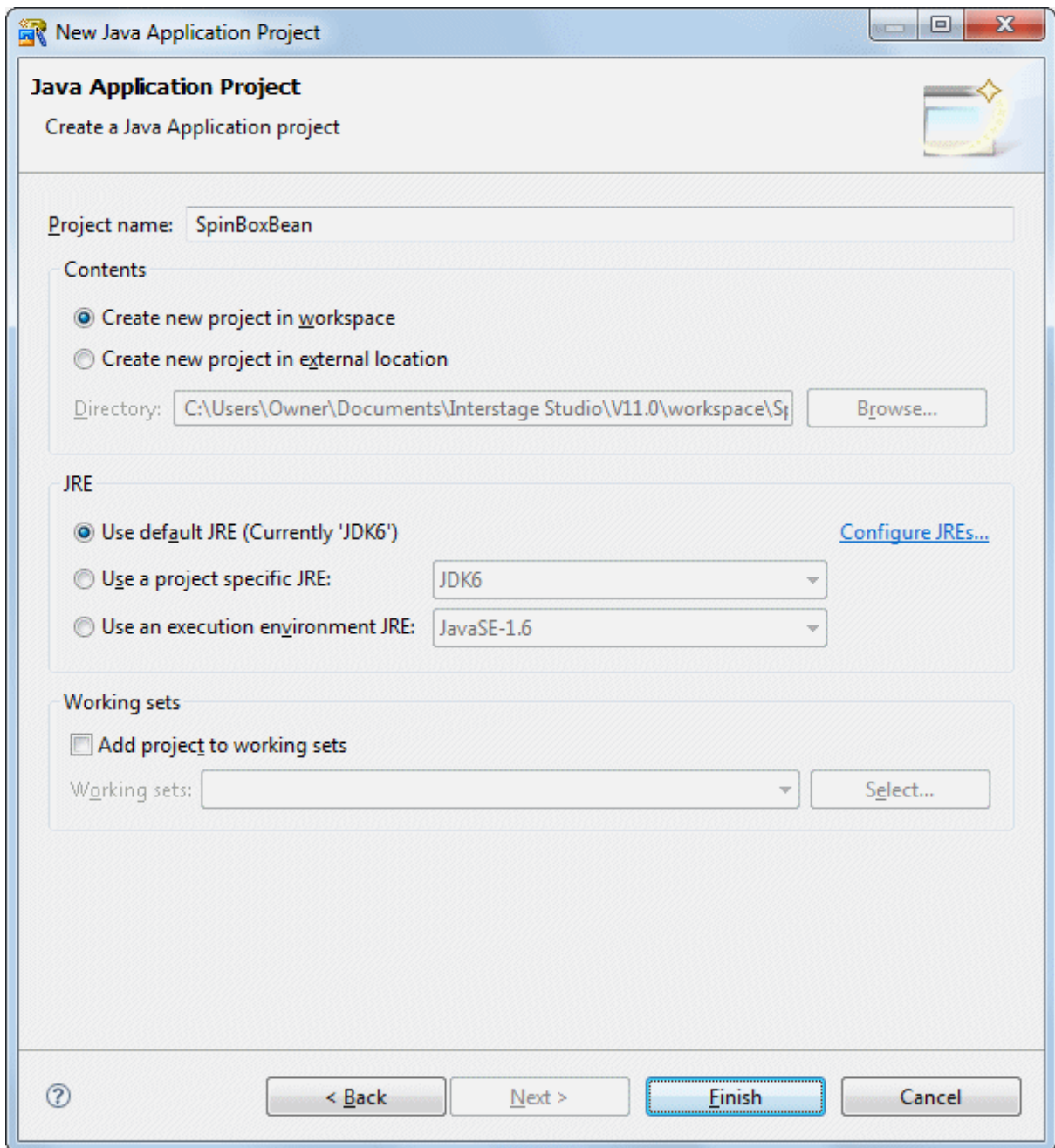
3. The [New Project] wizard is displayed. Select [Java Application Project] from the tree.



Click [Next].

4. The [Java Application Project] page is displayed.
Enter the following project information:

Setup Items	Setup Content
Project name	SpinBoxBean



Click [Finish].

Point

About Bean types

Interstage Studio classifies Beans into the two types described below. Both can be defined in workbench.

- Uni Bean

This Bean has simple functions, such as for text fields and buttons. When a Uni Bean is created, an existing Uni Bean, such as one for a text field (JTextField), is inherited, and additional functions are added to it.

If you specify a class that has no view for the base class, the Bean is called an Invisible Bean. The Bean inherits java.lang.Object and others. Because an Invisible Bean has no view, it can be edited in an editor in the same way as for ordinary source.

- Compound Bean

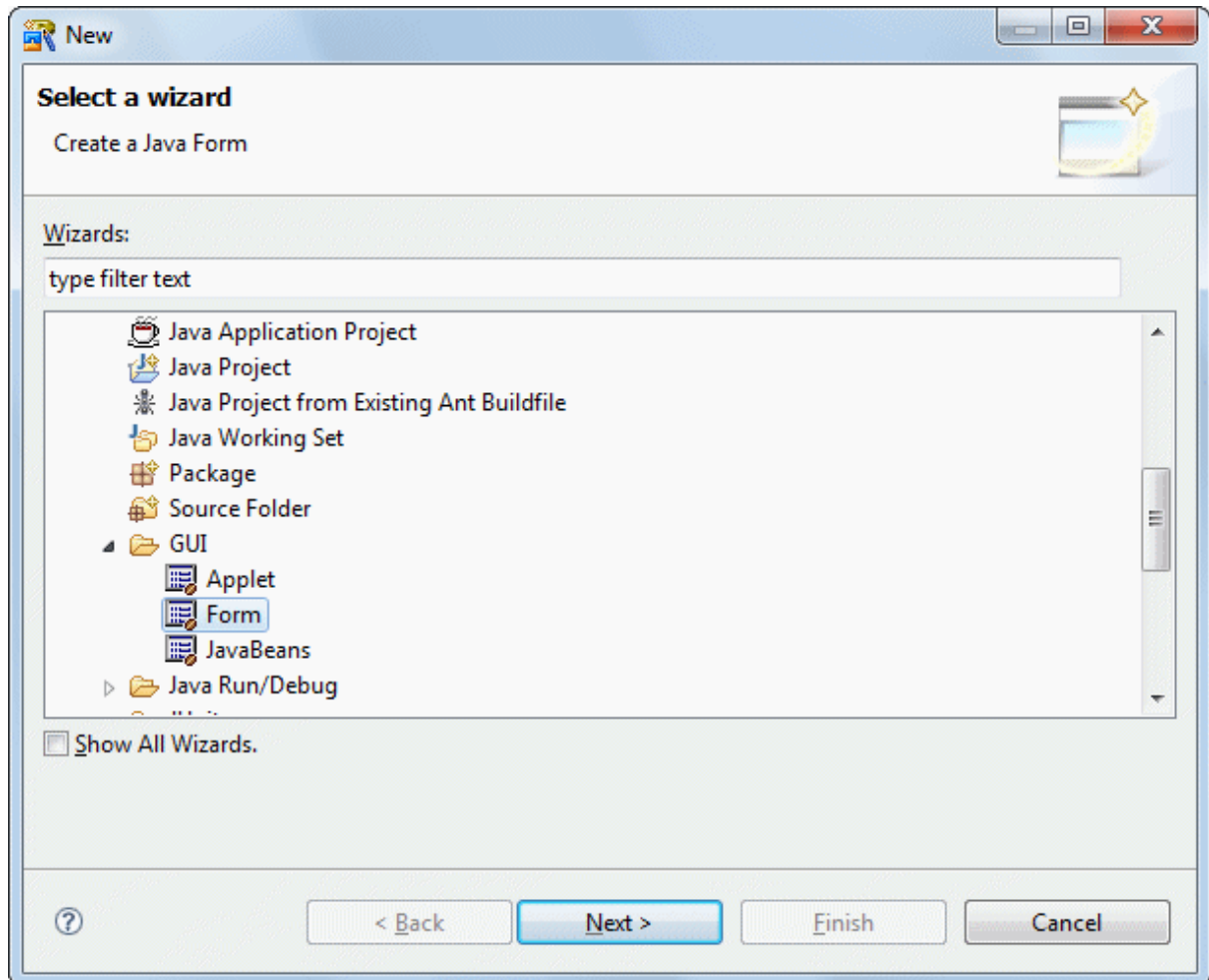
This Bean consists of two or more Beans. You can create a Compound Bean in Java Form Designer by pasting Beans into panels

(JPanel or Panel) for the form layout. Depending on the container type, use JPanel for lightweight components, and use Panel for heavyweight components.

Creating a Java Form

1. The next step is to create a Compound Bean. Select [File] > [New] > [Other] from the menu bar in workbench. Then, select [Java] > [GUI] > [Form] from the tree in the [New] wizard that appears.

If you are creating a Uni Bean or Invisible Bean, select [JavaBeans] here. In this example, however, select [Form] because you are creating a Compound Bean.



Click [Next].

2. The [Java Form Information] page is displayed.
Enter information as follows.

Setup Items	Setup Content
Package	myapp.javabeans

New Form

Java Form Information

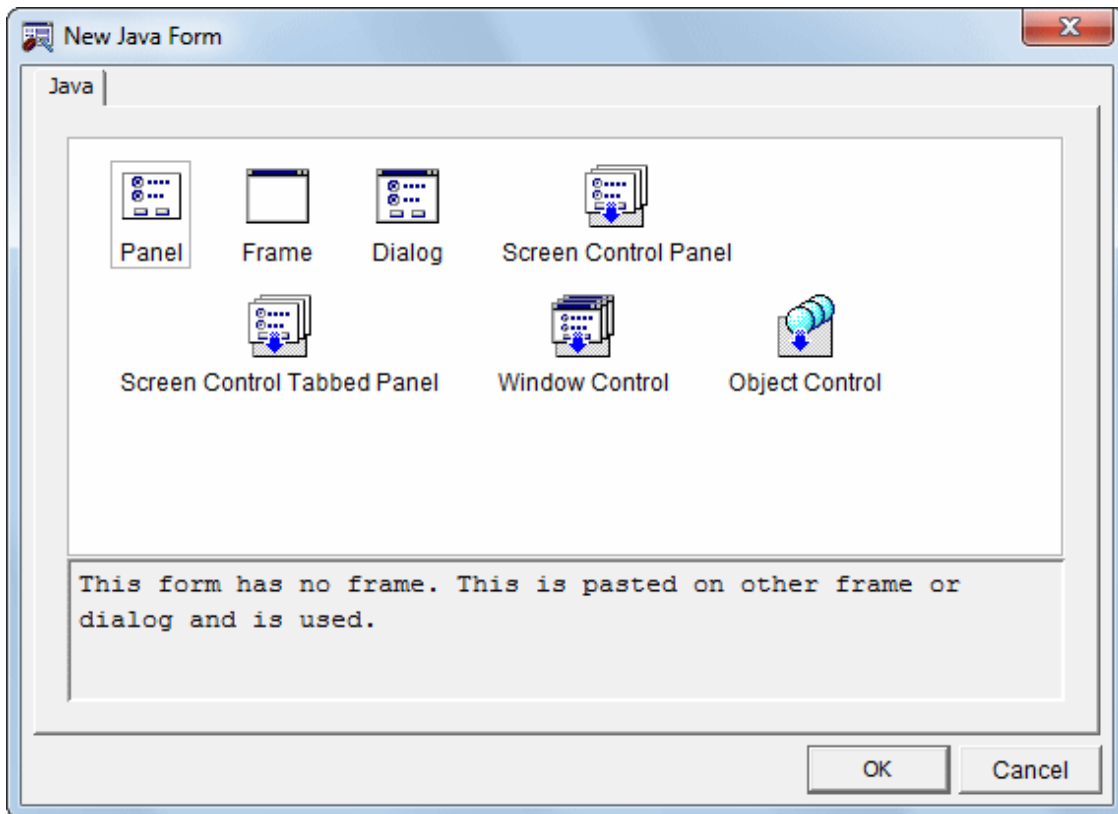
Please input the Java Form information.

Source Folder:

Package:

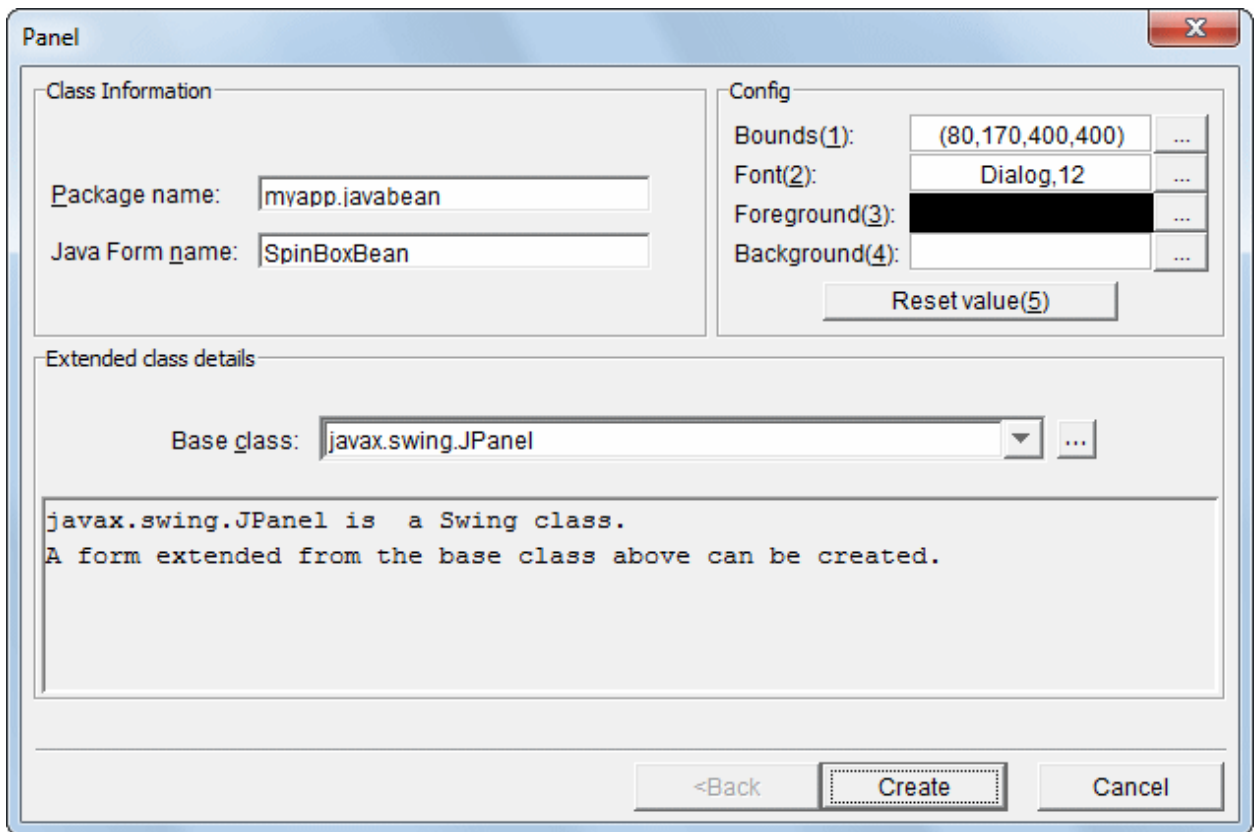
Click [Next].

- The [New Java Form] wizard is displayed.
Select the type of form that you want to create. Select [Panel] on the [Java] tab, then click [OK].



- Create a new panel format form.
Enter information as follows.

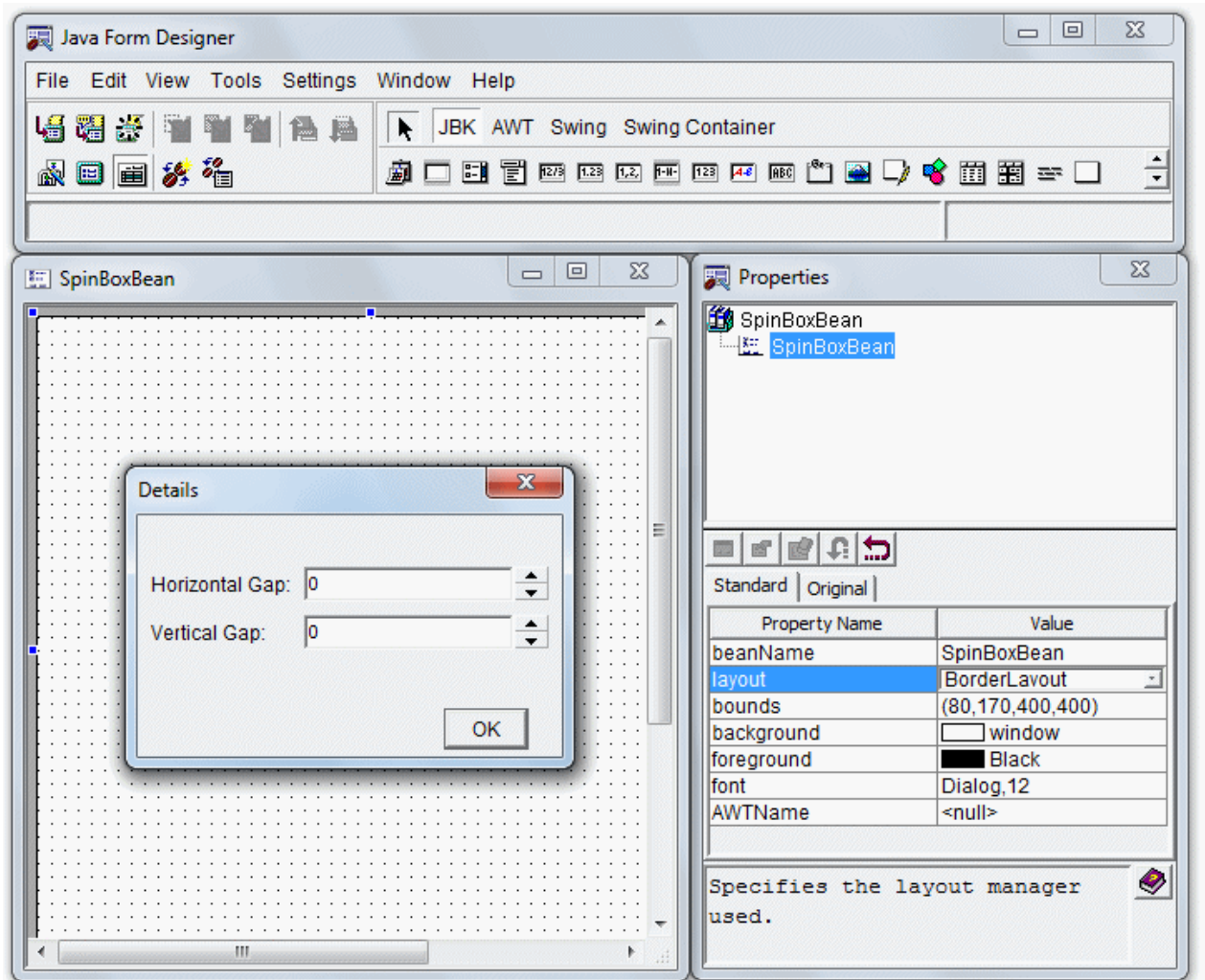
Setup Items	Setup Content
Java Form name	SpinBoxBean
Base class	javax.swing.JPanel



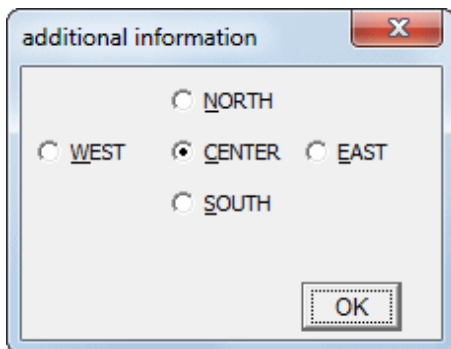
Click [Create].

5. Java Form Designer starts. Using Java Form Designer, place Beans, set properties (attributes), and code the flow of processes to complete the form.

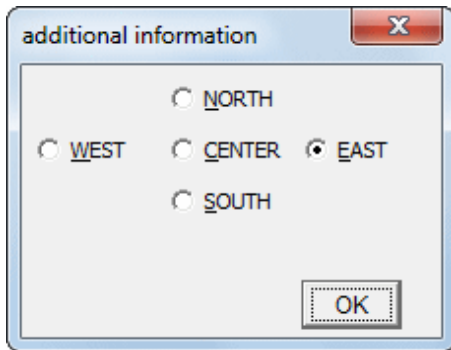
- Modify frame properties.
Specify "BorderLayout" for the property of layout.
Specify 0 as the horizontal gap and vertical intervals, then click [OK].



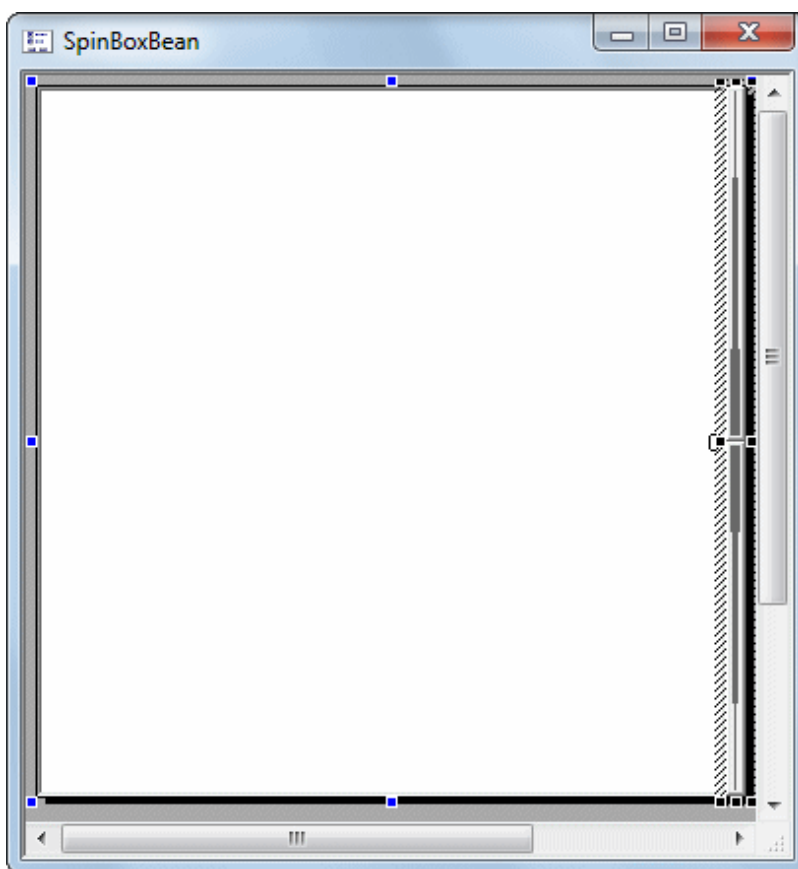
- Place the integer field Bean.
If [JBK] is not selected on the Object Palette, click [JBK].
Click [[Integer Field] on the Object Palette to select it. Place it on the Java Form by dragging it with the mouse.
Use "BorderLayout" to place the Bean.
The Bean can be placed in any direction with "BorderLayout". The directions are called "NORTH" for up, "SOUTH" for down, "WEST" for left, "EAST" for right, and "CENTER" for the center.
Select [CENTER], then click [OK].



- Place the spin button Bean.
Click [Spin Button] on the Object Palette to select it. Place it on the Java Form by dragging it with the mouse.
Select [EAST], then click [OK].



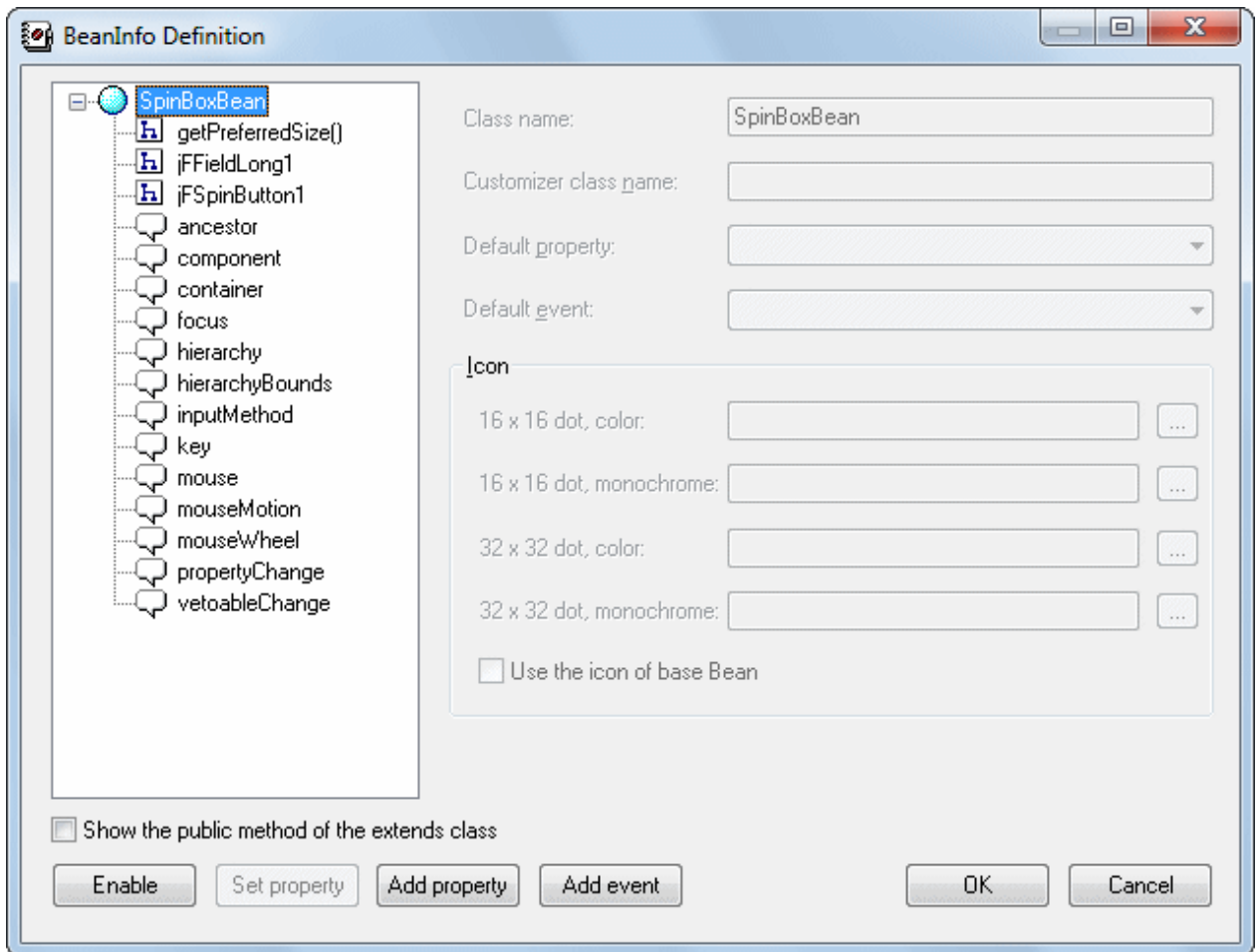
Two Beans are pasted as shown below.



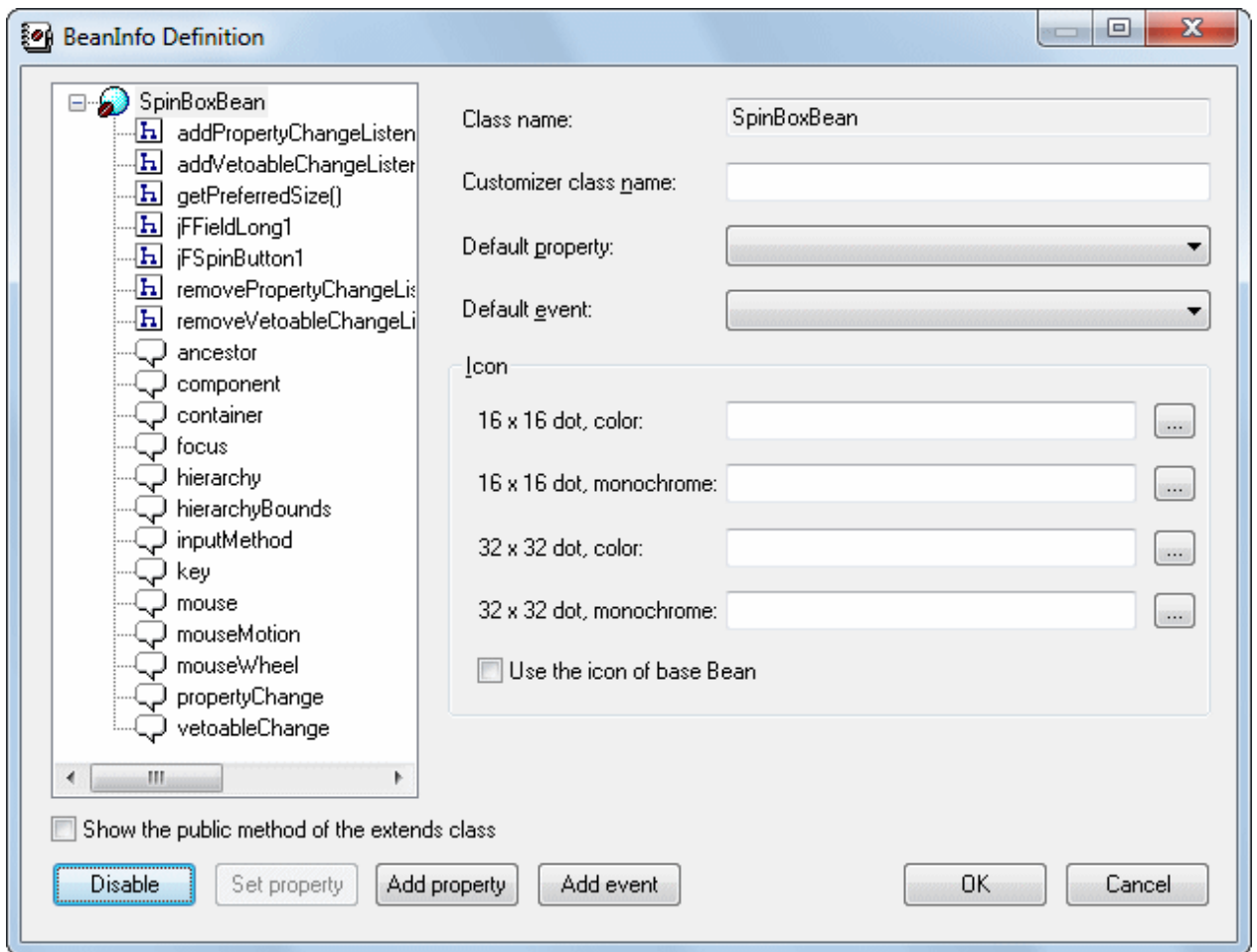
Defining BeanInfo

- BeanInfo is external interface information for programs that use Beans as components. Using BeanInfo Definition, define such information as properties, methods, and events.
Start BeanInfo Definition from workbench. To start workbench, select [View] > [Java Editor] from the Java Form Designer menu bar.

2. Select [Edit] > [Define BeanInfo] > [BeanInfo Definition] from the workbench menu bar to call BeanInfo Definition.



With "SpinBoxBean" selected, click [Enable].



Point

About icons

Using BeanInfo Definition, you can specify icons for created Beans.

Create an image file with the icon in advance, using any paint tool. The icon image file must be in the GIF or JPEG format. Add it to the project as follows: Click [File] > [Import] from the workbench menu bar. From [File system] in the [Import] dialog, import the image file to the project.

In order to have Bean icons displayed on the Object Palette and in the Properties window component tree in workbench, use the image file specified as "16 x 16 dot, Color."

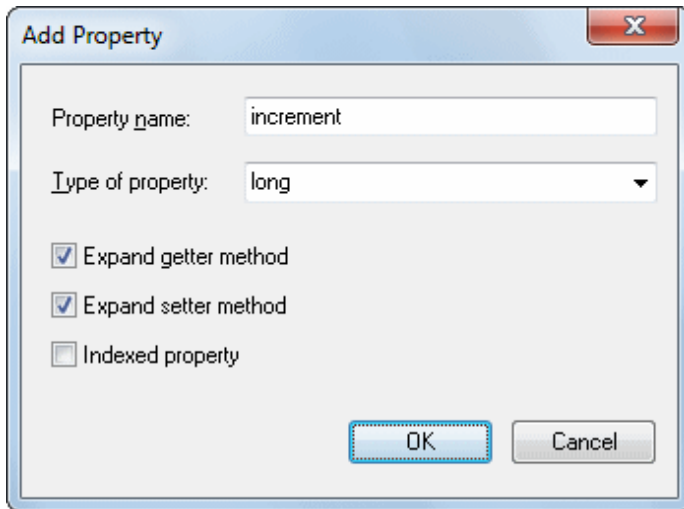
Specification of the icon image file can be omitted. If you do not specify an icon image file, default icons provided by workbench are used.

3. Next, define properties.

Prepare the following two properties:

Property Name	Type	Explanation
increment	long	Spin button incremental value
value	long	Integer field value

Click [Add property] in the BeanInfo Definition dialog.
Specify as shown in the following dialog, then click [OK]:

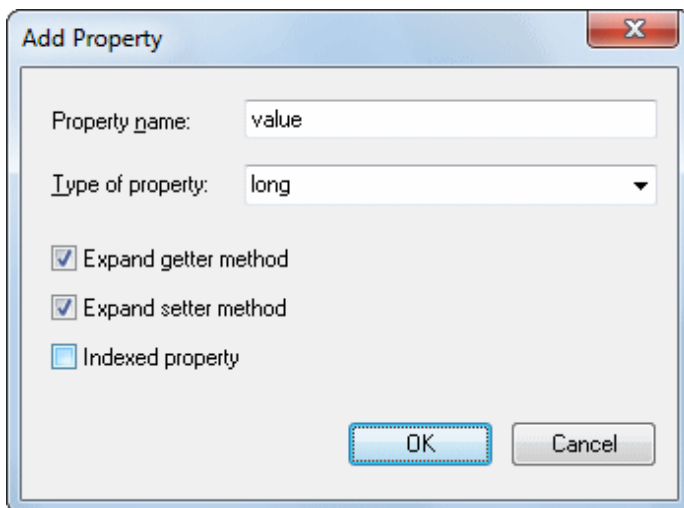


The screenshot shows a dialog box titled "Add Property" with a close button (X) in the top right corner. It contains the following fields and options:

- Property name: increment
- Type of property: long
- Expand getter method
- Expand setter method
- Indexed property

At the bottom, there are two buttons: "OK" (highlighted with a blue border) and "Cancel".

Add value properties in the same way.
Click [Add property] in the BeanInfo Definition dialog.
Specify as shown in the following dialog, then click [OK]:

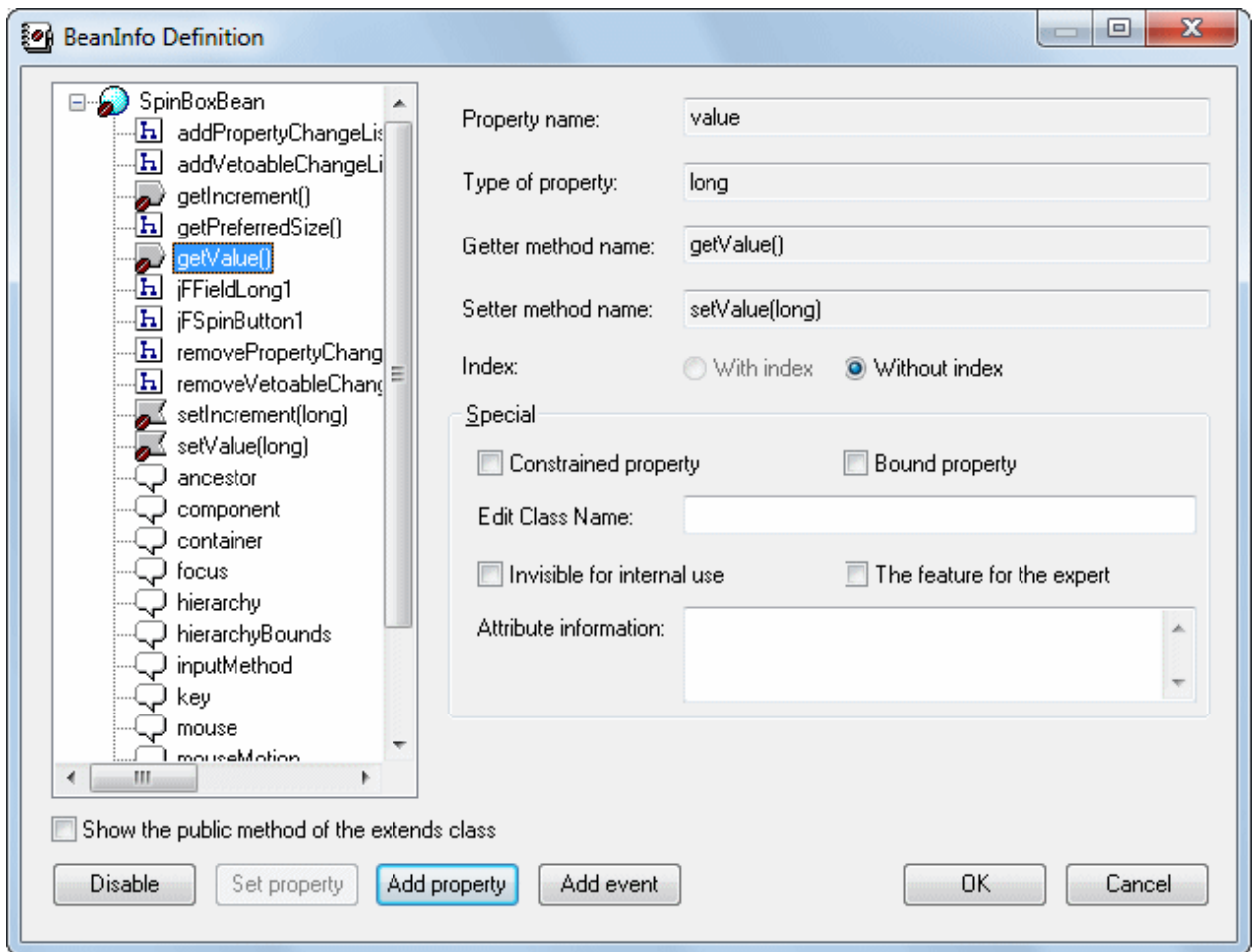


The screenshot shows a dialog box titled "Add Property" with a close button (X) in the top right corner. It contains the following fields and options:

- Property name: value
- Type of property: long
- Expand getter method
- Expand setter method
- Indexed property

At the bottom, there are two buttons: "OK" (highlighted with a blue border) and "Cancel".

The properties are added as shown below.



4. Click [OK]. The BeanInfo Definition dialog closes.

Coding an event process

1. Click the [SpinBoxBean.java] tab in the editor area to make the Java editor active.
2. Since the JBK class is used, add the import keyword.
Add the text shown below in **red**.

```
import com.fujitsu.jbk.gui.JFSpinButton;
```

3. Add a field that contains the incremental value to the class.
Add the increment field above the constructor of the SpinBoxBean class. Add the text shown below in **red**.

```
/**  
 * The form is constructed.  
 */  
public SpinBoxBean() {  
}
```

Point

Change prohibition part which is managed by Java Form Designer

Java Forms include sources for calling screen (Bean) information and event processes. Because Java Form Designer is used to manage the sources, changing the sources is prohibited.

These sources that must not be changed are enclosed by comments in **blue**, as shown below.

```
public class SpinBoxBean extends javax.swing.JPanel {  
    /*@Form Design Information start  
  
    - Sources that are managed with Java Form Designer and that must not be changed -  
  
    /*@Form Design Information end  
}
```

4. Code the getIncrement method process.

Add the process to getIncrement, which is the increment property getter method. Add the text shown below in **red**.

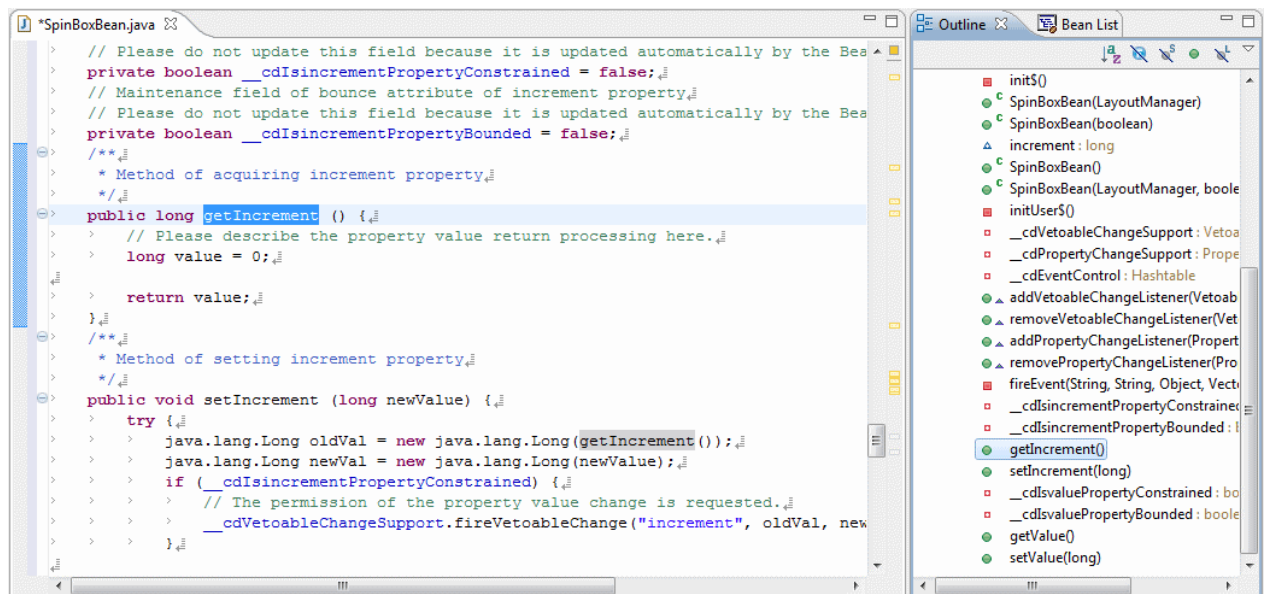
```
public long getIncrement() {  
    // Please describe the property value return processing here.  
    long value = 0;  
    value = increment;  
    return value;  
}
```

Point

Scrolling in Java editor

To easily scroll to a method or field you want to edit in Java editor, you can use the [Outline] view.

Clicking "getIncrement()" in the [Outline] view causes the Java editor display to scroll to the getIncrement method.



5. Code the setIncrement method process.

Add the process to setIncrement, which is the increment property setter method. Add the text shown below in **red**.

```
public void setIncrement(long newValue) {  
    try {  
        java.lang.Long oldVal = new java.lang.Long(getIncrement());  
        java.lang.Long newVal = new java.lang.Long(newValue);  
        if( __cdIsIncrementPropertyConstrained ) {  
            // The permission of the property value change is requested.  
            __cdVetoableChangeSupport.fireVetoableChange("increment", oldVal, newVal);  
        }  
    }  
}
```

```

        // Please describe the property value set processing here.
        increment = newValue;

        if( __cdIsincrementPropertyBounded ) {
            // The property value change is notified.
            __cdPropertyChangeSupport.firePropertyChange("increment",oldVal,newVal);
        }
    }
    catch(java.beans.PropertyVetoException e) {
    }
}

```

6. Code the getValue method process.

Add the process to getValue, which is the value property getter method. Add the text shown below in **red**.

```

public long getValue() {
    // Please describe the property value return processing here.
    long value = 0;
    value = jFFieldLong1.getValue();
    return value;
}

```

7. Code the setValue method process.

Add the process to setValue, which is the value property setter method. Add the text shown below in **red**.

```

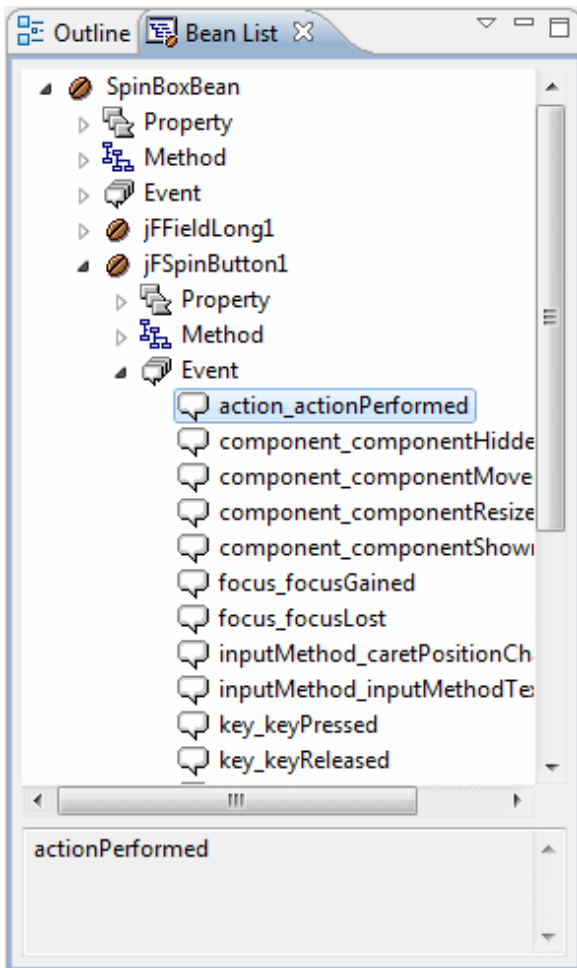
public void setValue(long newValue) {
    try {
        java.lang.Long oldVal = new java.lang.Long(getValue());
        java.lang.Long newVal = new java.lang.Long(newValue);
        if( __cdIsvaluePropertyConstrained ) {
            // The permission of the property value change is requested.
            __cdVetoableChangeSupport.fireVetoableChange("value",oldVal,newVal);
        }

        // Please describe the property value set processing here.
        jFFieldLong1.setValue(newValue);

        if( __cdIsvaluePropertyBounded ) {
            // The property value change is notified.
            __cdPropertyChangeSupport.firePropertyChange("value",oldVal,newVal);
        }
    }
    catch(java.beans.PropertyVetoException e) {
    }
}

```

- To code the process in which a specific event occurs, display the [Bean List] view. Select [Window] > [Show View] > [Other] from the menu bar. The [Show View] dialog is displayed. Select [Java] > [Bean List], then click [OK].



- Code the process in which the spin button is clicked. Double-click [SpinBoxBean] > [jFSpinButton1] > [Event] > [action_actionPerformed] in the [Bean List] view. An event process creation confirmation dialog is displayed. Click [Yes]. When the processing procedure of "jFSpinButton1_action_actionPerformed\$" is displayed, add the text shown below in **red**.

```

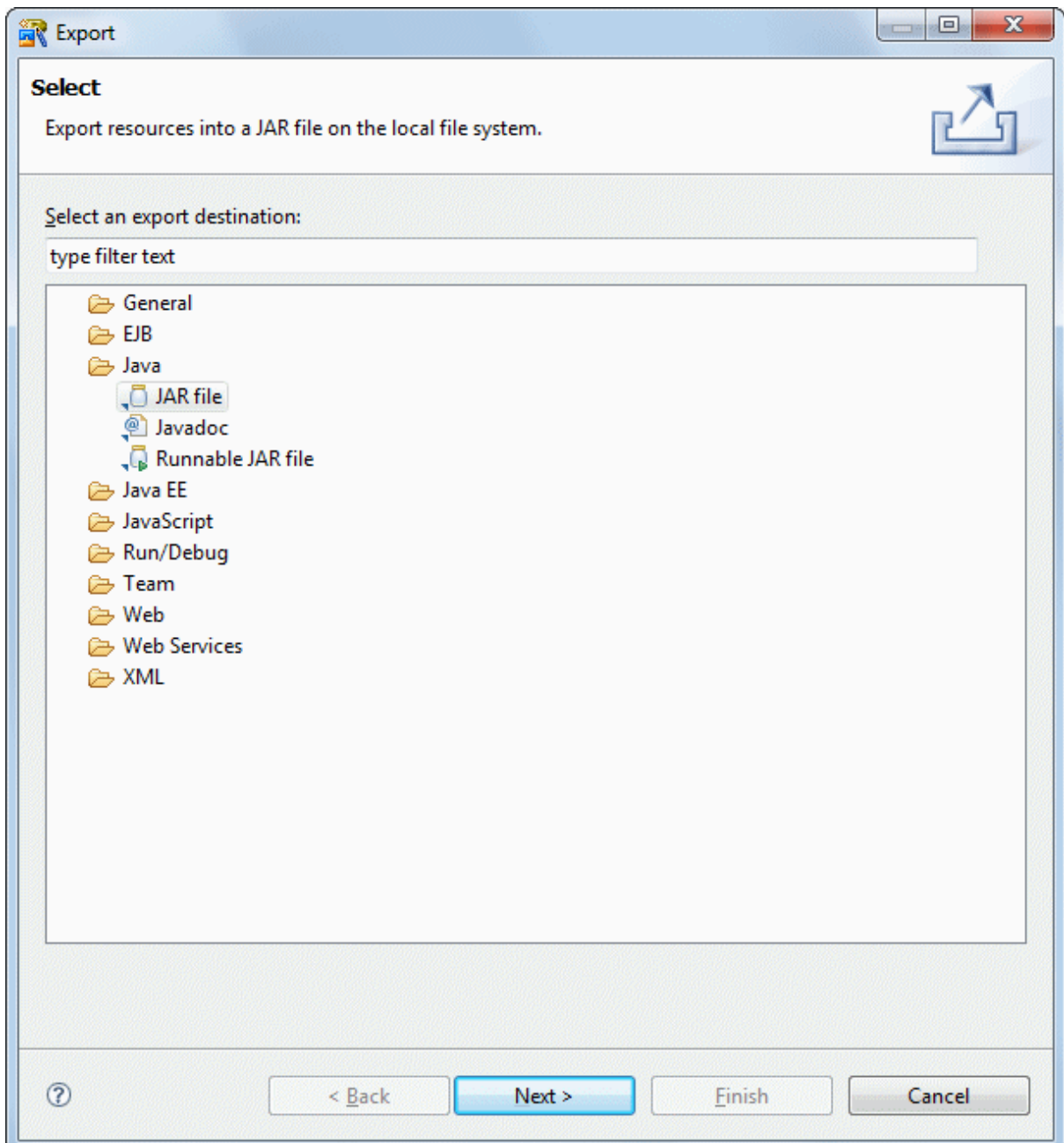
public void jFSpinButton1_action_actionPerformed$(java.awt.event.ActionEvent e) {
    if (!defaultEventProc(e)) {
        // The procedure when the event is generated is described below.
        if (e.getActionCommand() == JFSpinButton.UP){
            // If the Up button is clicked.
            jFFieldLong1.setValue(jFFieldLong1.getValue() + increment);
        }
        if (e.getActionCommand() == JFSpinButton.DOWN){
            // If the Down button is clicked.
            jFFieldLong1.setValue(jFFieldLong1.getValue() - increment);
        }
    }
}

```

- Save the Java Form. Select [File] > [Save] from the workbench menu bar. Select [File] > [Exit] from the Java Form Designer menu to close the Java Form.

Building

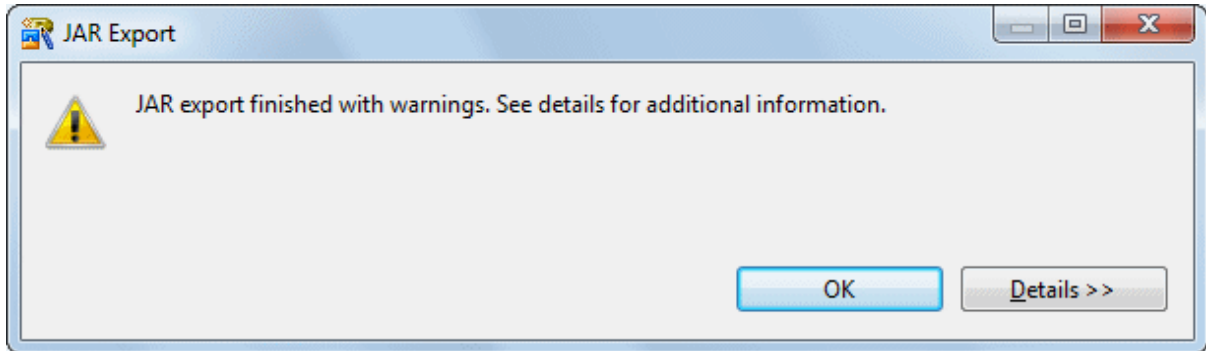
1. Building is executed automatically when resource files (such as the Java files) are saved if the [Build Automatically] item is enabled in the [Project] menu.
If [Build Automatically] is disabled, right-click on the project and select [Build Project] or select [Build Project] in the [Project] menu.
2. A JAR file is required so create it.
Use the export wizard to create the JAR file.
Select [File] > [Export] to start the export wizard.
Select [Java] > [JAR file] in the export wizard.



3. The JAR export wizard is displayed.
Select [SpinBoxBean] > [src] and [_beaninfo_src] folders in [Select the resources to export] and enter the following settings.
After setting the information, click [Finish].

Setup Items	Setup Content
Select the resources to export	src/ _beaninfo_src/
Export generated class files and resources	Checked
JAR file	SpinBoxBean/SpinBoxBean.jar
Compress the contents of the JAR file	Checked

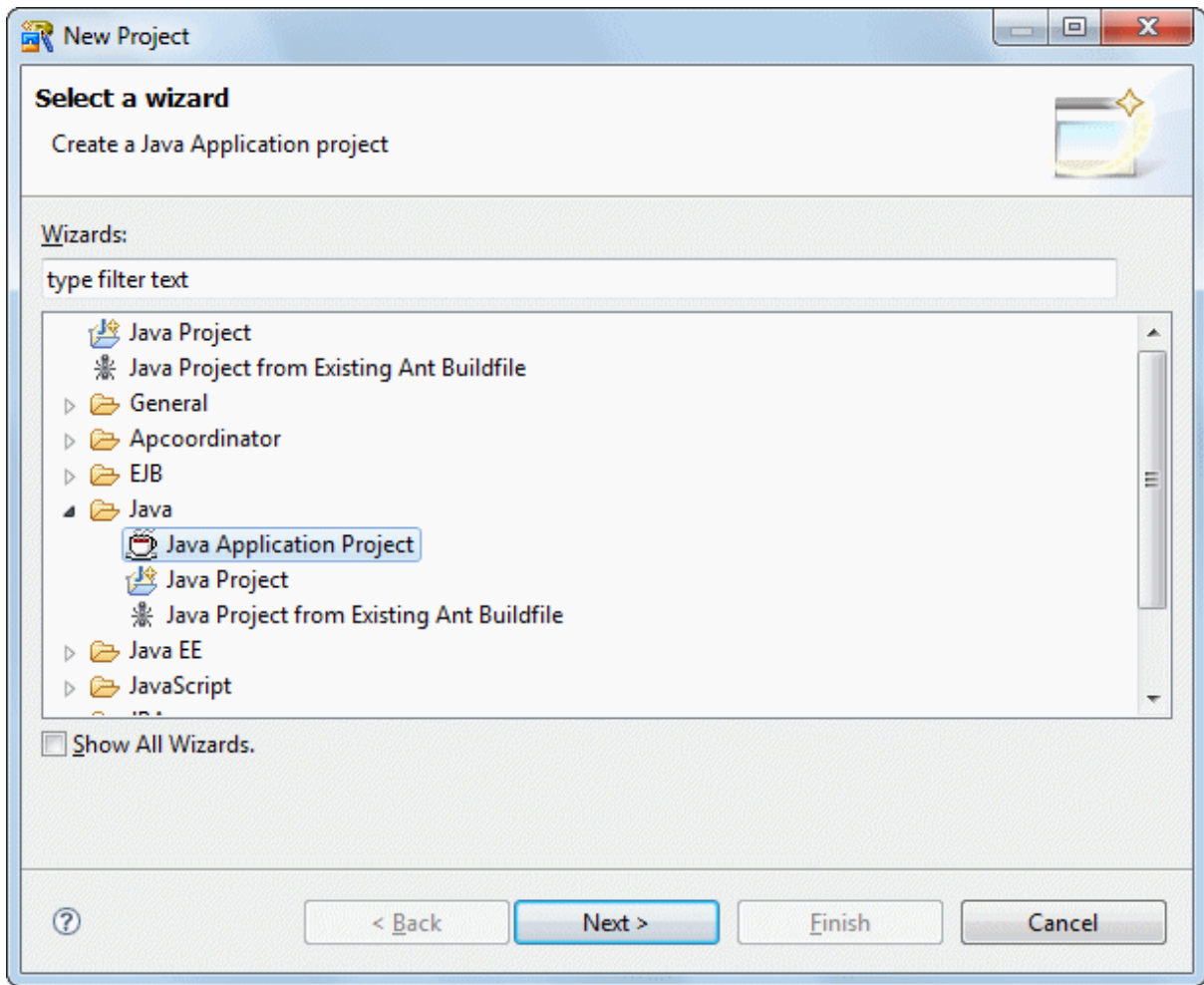
The following message is output when exporting JAR, but this does not indicate a problem.



Creating a Java Application Project (creating a program to test the spin button)

1. Create a separate project to test the created SpinBoxBean.
Select [File] > [New] > [Project] from the menu bar.

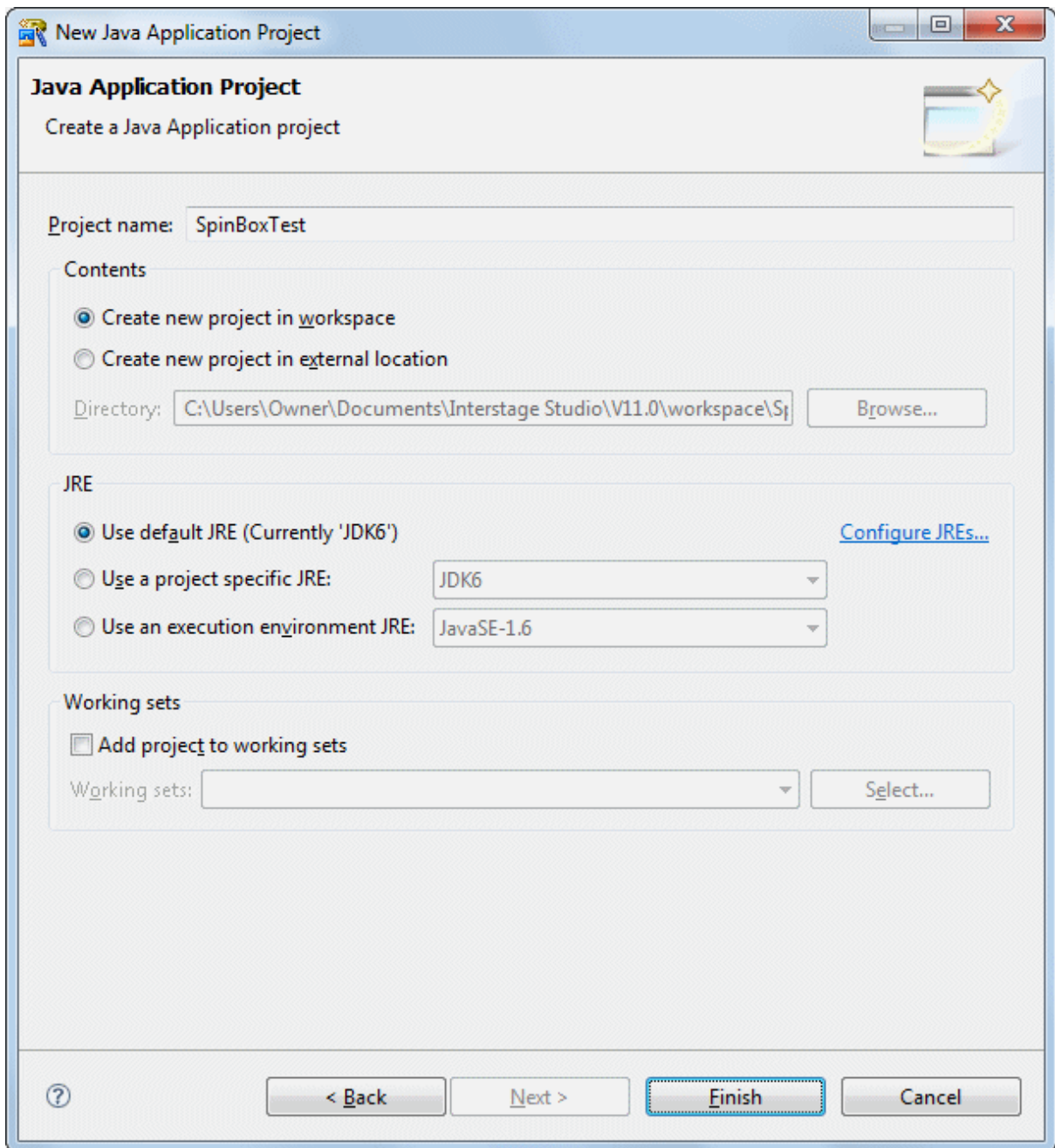
2. The [New Project] wizard is displayed. Select [Java Application Project] from the tree.



Click [Next].

3. The [Java Application Project] page is displayed. Enter the following project information:

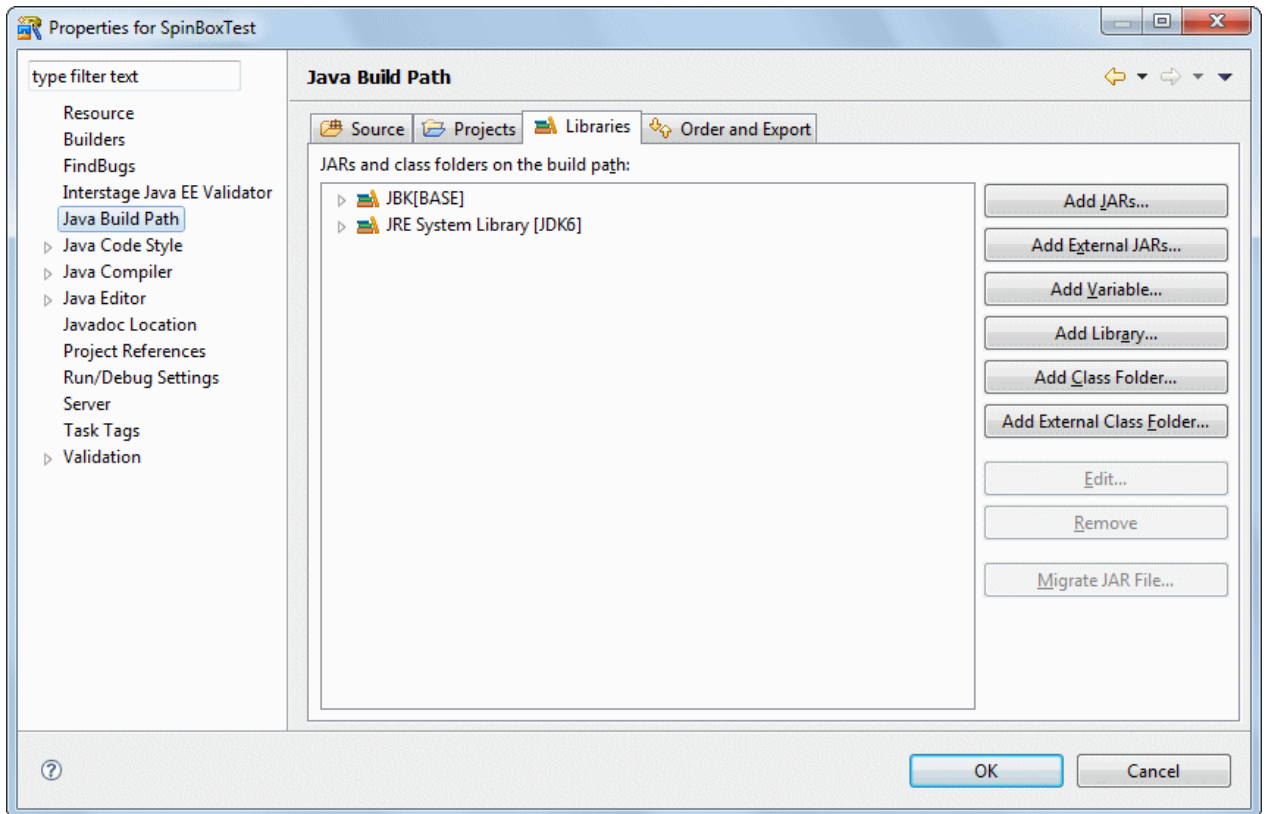
Setup Items	Setup Content
Project name	SpinBoxTest



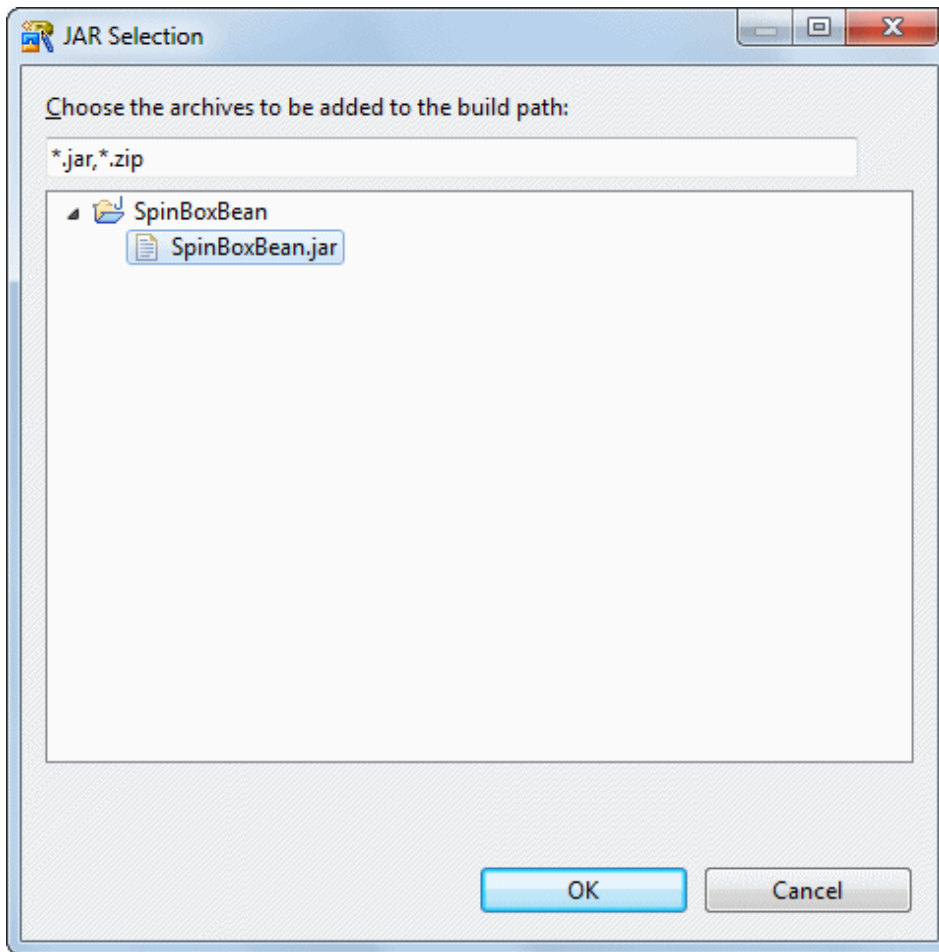
Click [Finish]

4. Select and right-click the created project, and then select [Properties] from the context menu that appears. When the [Properties] dialog box appears, select [Java Build Path] from the tree in the left pane. The [Java Build Path] page is displayed.

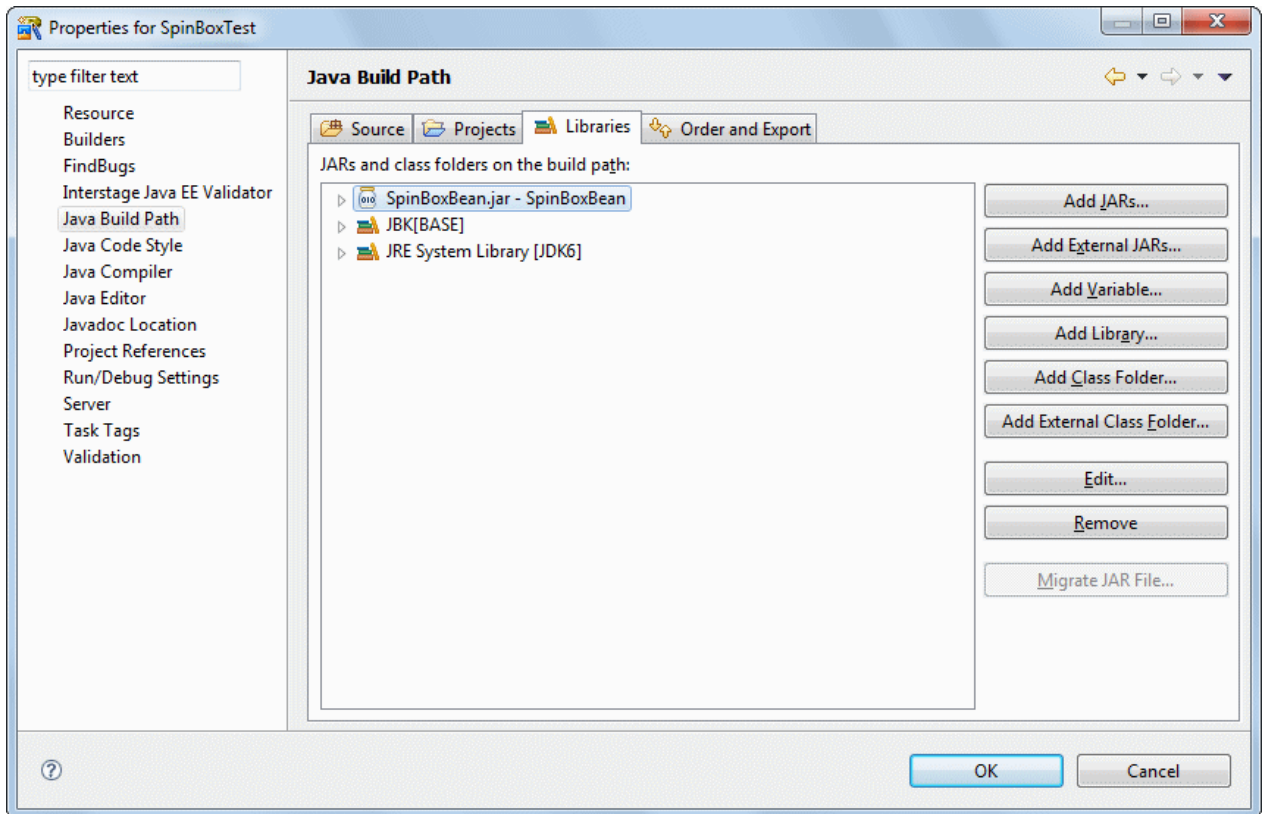
Because this project uses the Bean that you created, add the "SpinBoxBean.jar" file in the "SpinBoxBean" project to the build path. Click the [Libraries] tab.



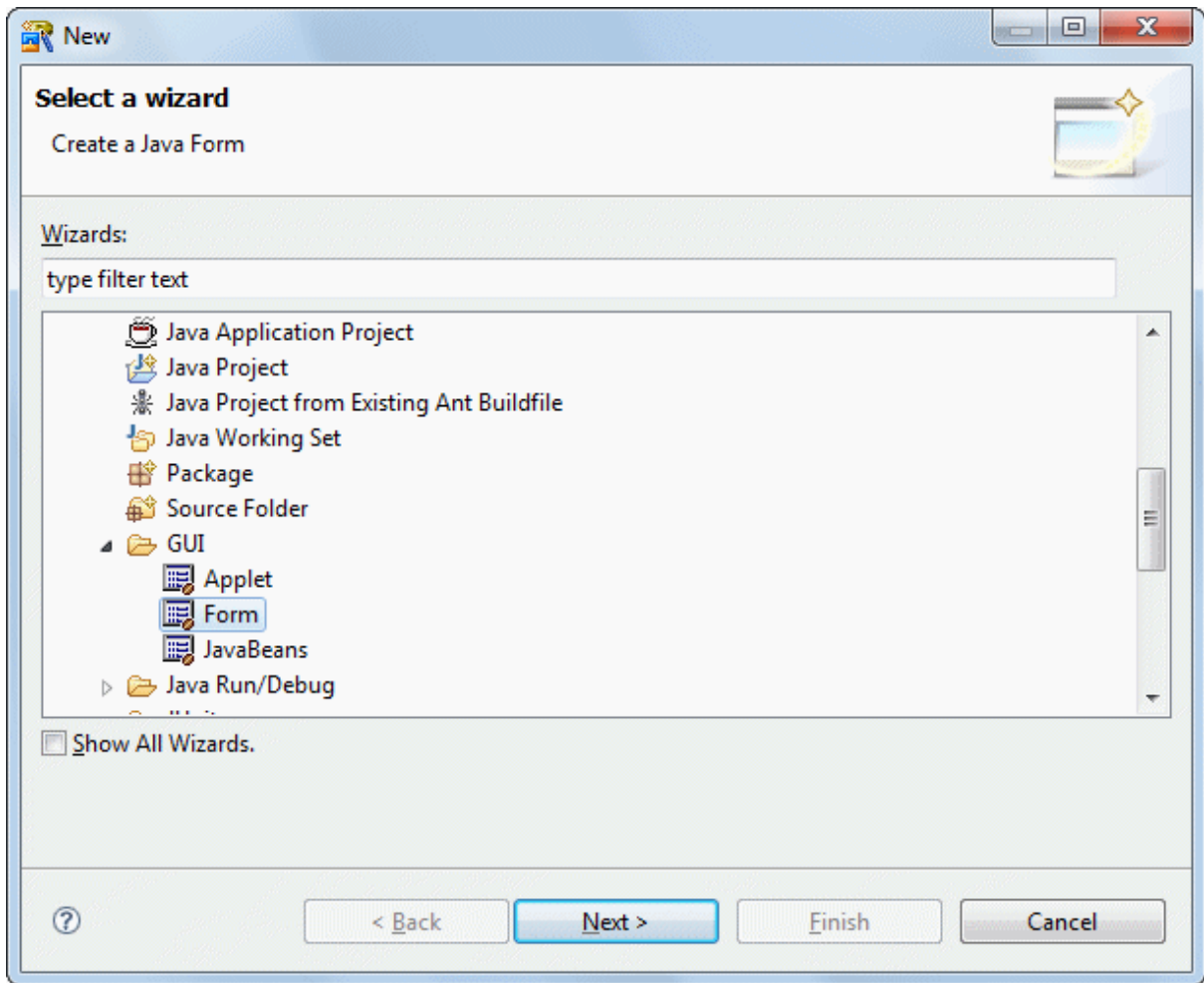
Clicking [Add JARs] displays the [JAR Selection] dialog. Click [SpinBoxBean] > [SpinBoxBean.jar] in [Choose the archives to be added to the build path].
Click [OK].



[SpinBoxBean.jar] is added.
Click [OK].



- The next step is to create a class of the form and an executable class that will display the form. Select [File] > [New] > [Other] from the menu bar in workbench. Then, select [Java] > [GUI] > [Form] from the tree in the [New] wizard that appears.



Click [Next].

- The [Java Form Information] page is displayed. Enter information as follow.

Setup Items	Setup Content
Source folder	SpinBoxTest/src
Package	mayapp.javabeans

New Form

Java Form Information

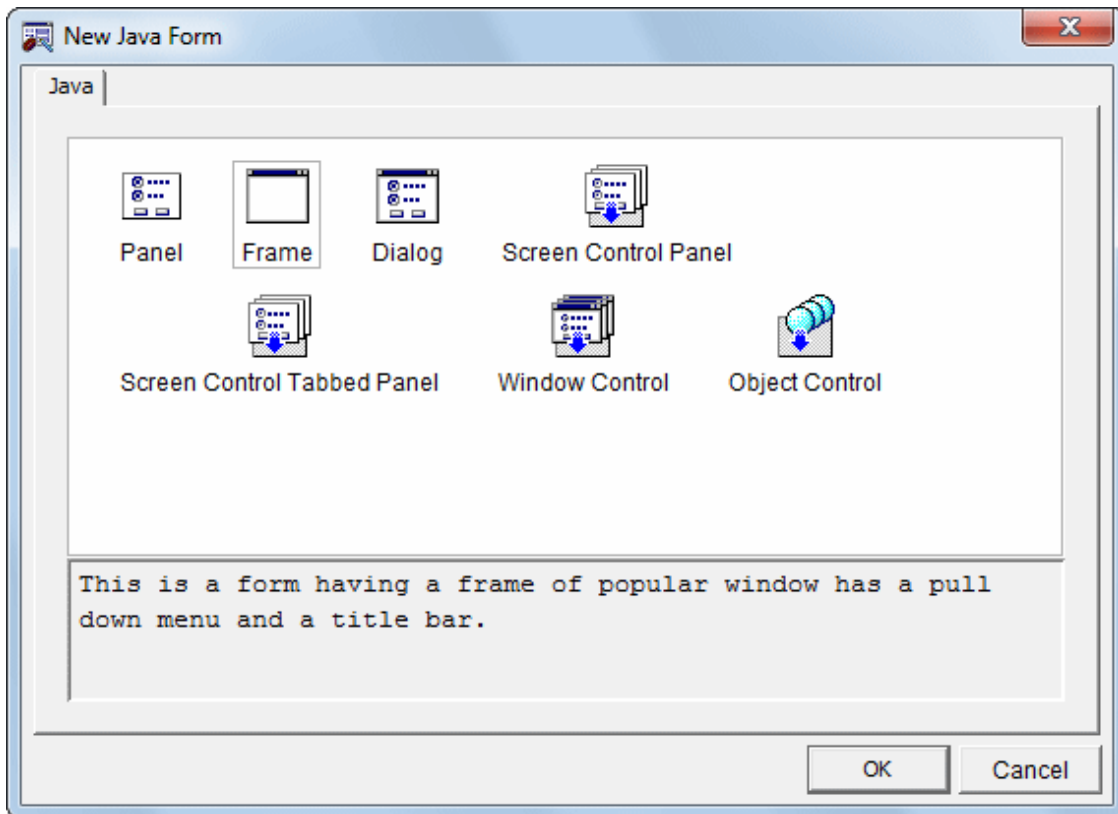
Please input the Java Form information.

Source Folder:

Package:

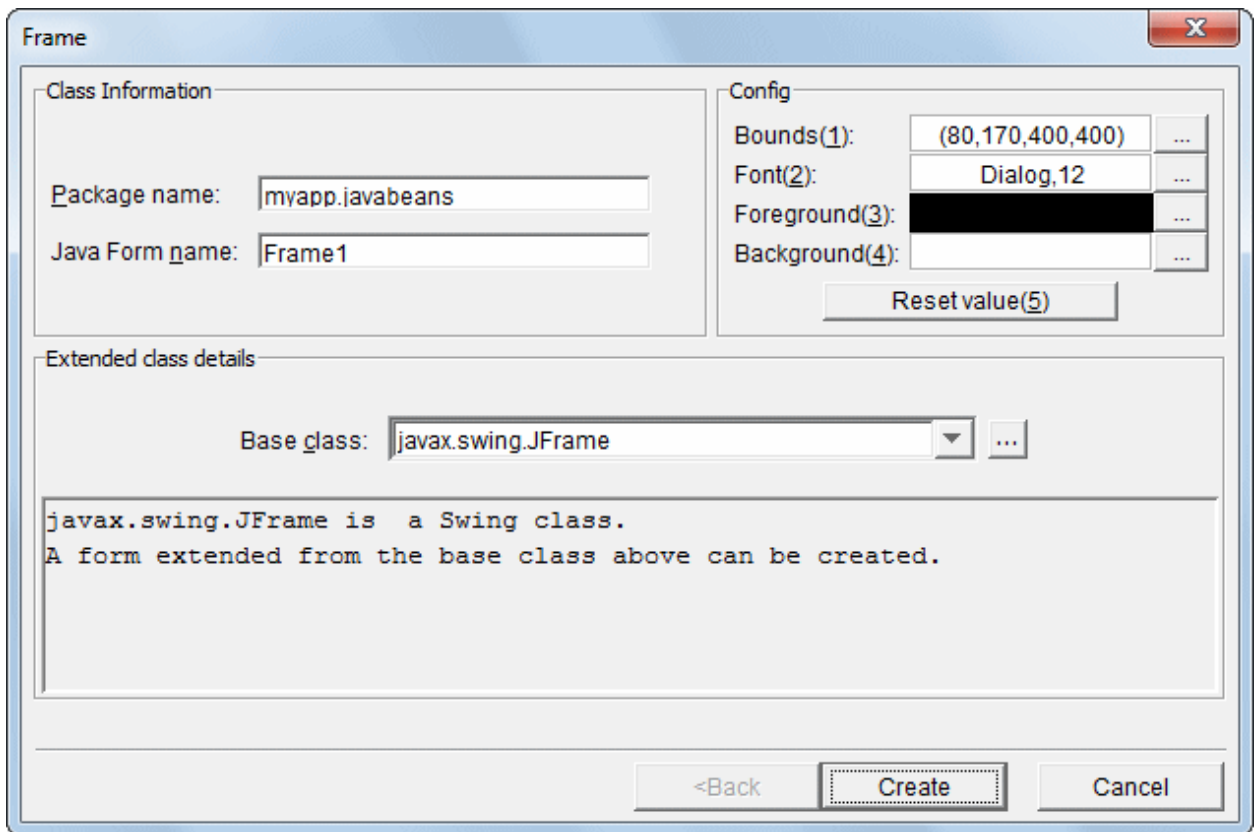
Click [Next].

7. The [New Java Form] dialog box is displayed.
 Select [Frame], and Click [Next].



8. The [Frame] dialog is displayed.
 On this page, specify the required information to create the frame.
 Enter information as follow.

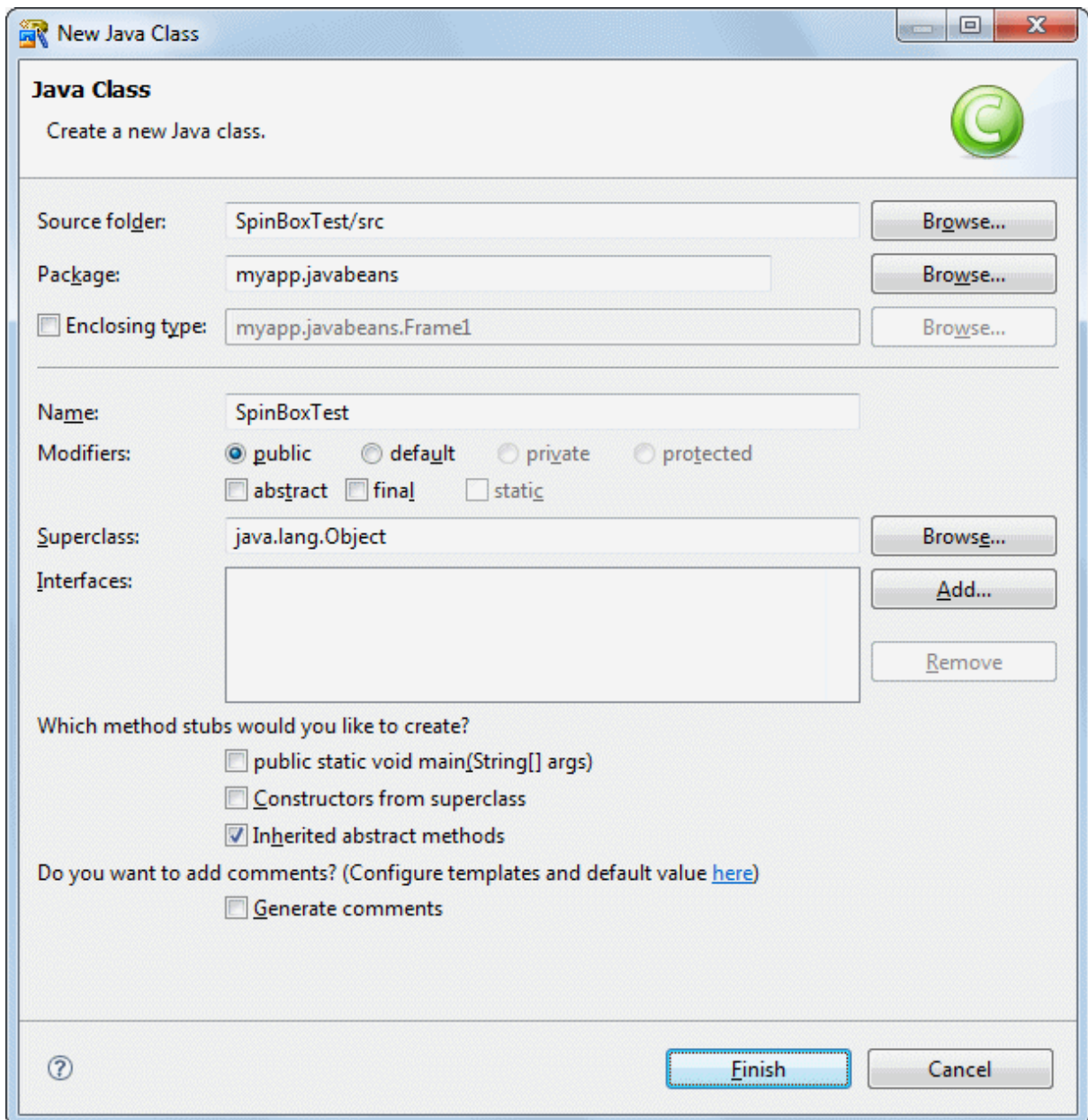
Setup Items	Setup Content
Package name	myapp.javabeans
Java Form name	Frame1
Base class	Javax.swing.JFrame



Click [Create].

9. Create a class of the form and an executable class that will display the form.
Select [File] > [New] > [Class] from the menu bar.
10. The [Java Class] page is displayed.
On this page, specify the basic information to create the Java class.
Enter information as follow.

Setup Items	Setup Content
Source folder	SpinBoxTest/src
Package	myapp.javabeans
Name	SpinBoxTest
Inherited abstract methods	Checked



Click [Finish].

11. The source created in the [Java Editor] is displayed.
Edit the source that was created. Add the parts in **red**.

```
package myapp.javabeans;

public class SpinBoxTest {
    //Constructor
    public SpinBoxTest() {
    }

    public void run(String[] args) {
        //Create new form project
        Frame1 form = new Frame1();
        //Show form
        form.setVisible(true);
    }
}
```

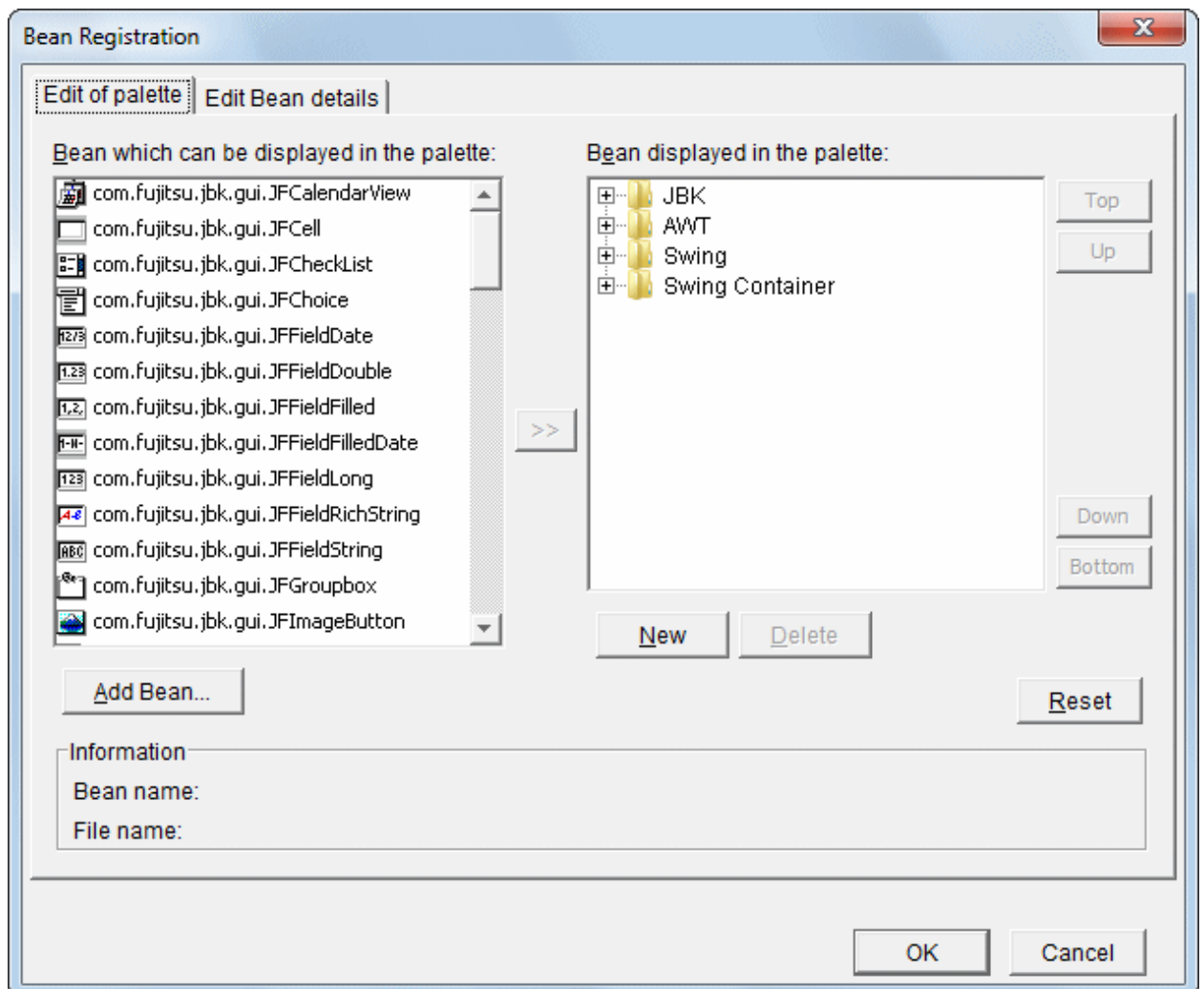
```

//Main process
public static void main(String[] args) {
    // Create SpinBoxTest class instance
    SpinBoxTest object = new SpinBoxTest();
    // Call run method of the SpinBoxTest class instance
    object.run(args);
}
}

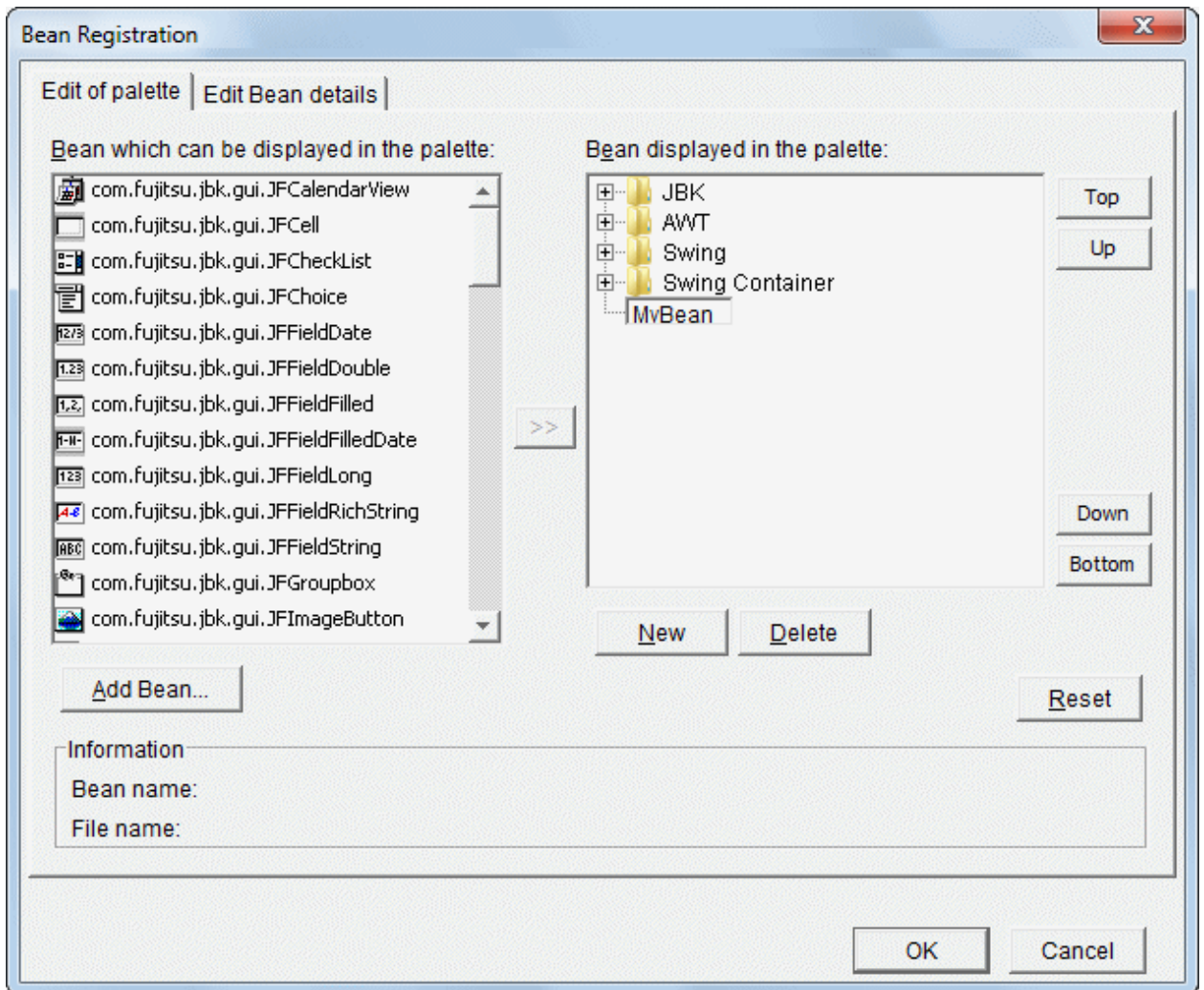
```

Registering a Bean

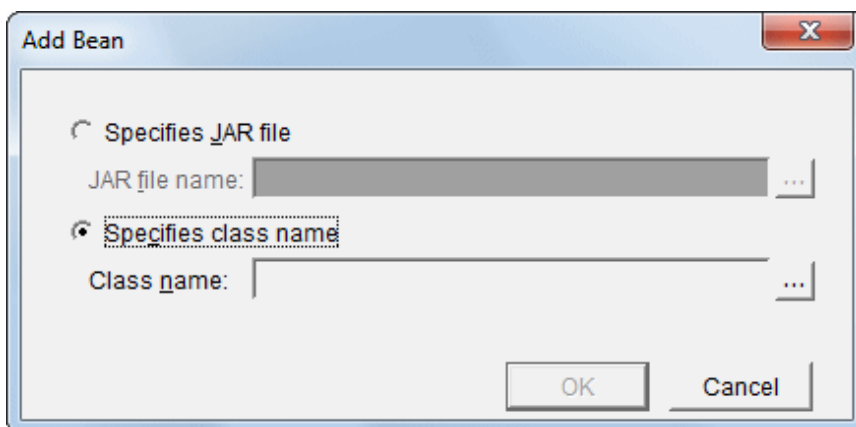
1. Open [Frame1.java] in Java Form Designer.
To display the context menu, right-click [SpinBoxTest] > [src] > [myapp.javabeans] > [Frame1.java] in the [Package Explorer] view of workbench. Then, select [Open With] > [Graphical Editor].
2. In Java Form Designer, register the Bean that you created. This enables you to paste the new Bean into a Java Form in the same way as a JBK Bean and other Beans.
Select [Settings] > [Bean Registration] from the Java Form Designer menu bar.
3. The [Bean Registration] dialog box is displayed.



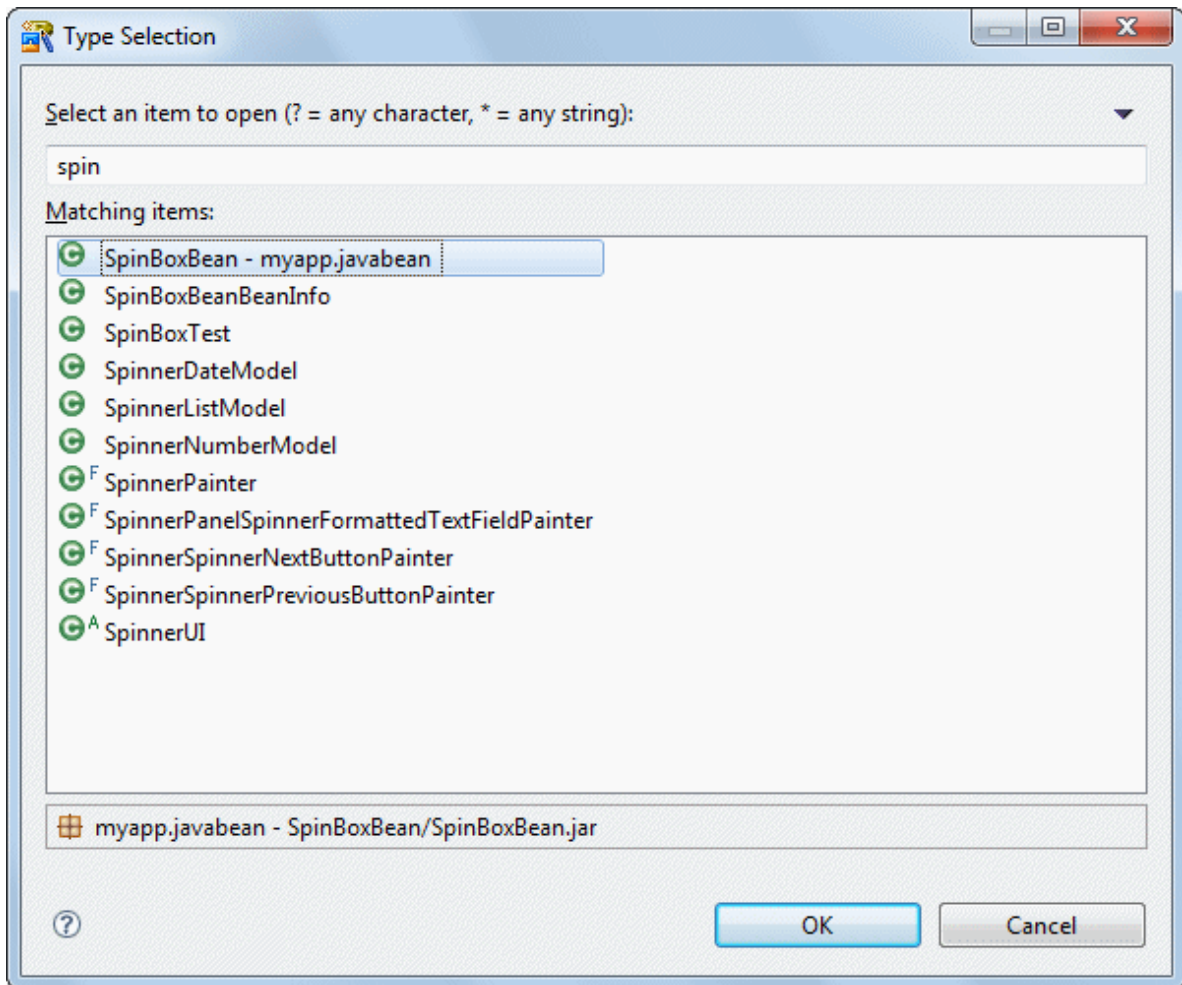
4. Click [New]. Enter "MyBean".



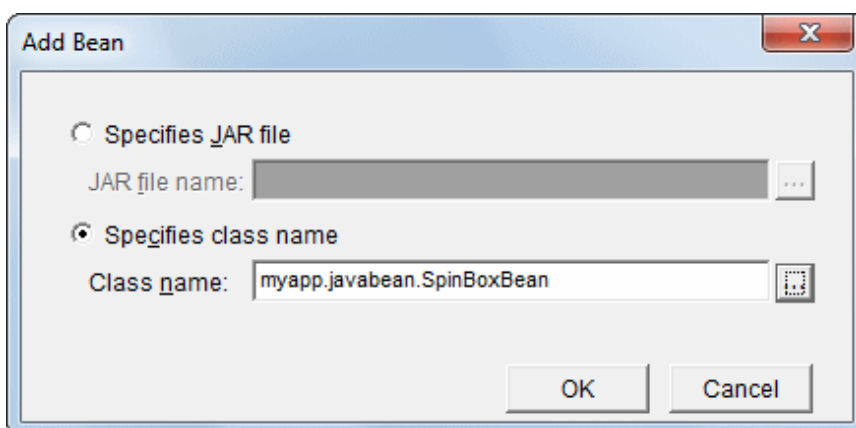
5. Next, click [Add Bean]. The [Add Bean] dialog box is displayed. Specify "SpinBoxBean," which you created. Select [Specifies class name], and click the [...] button.



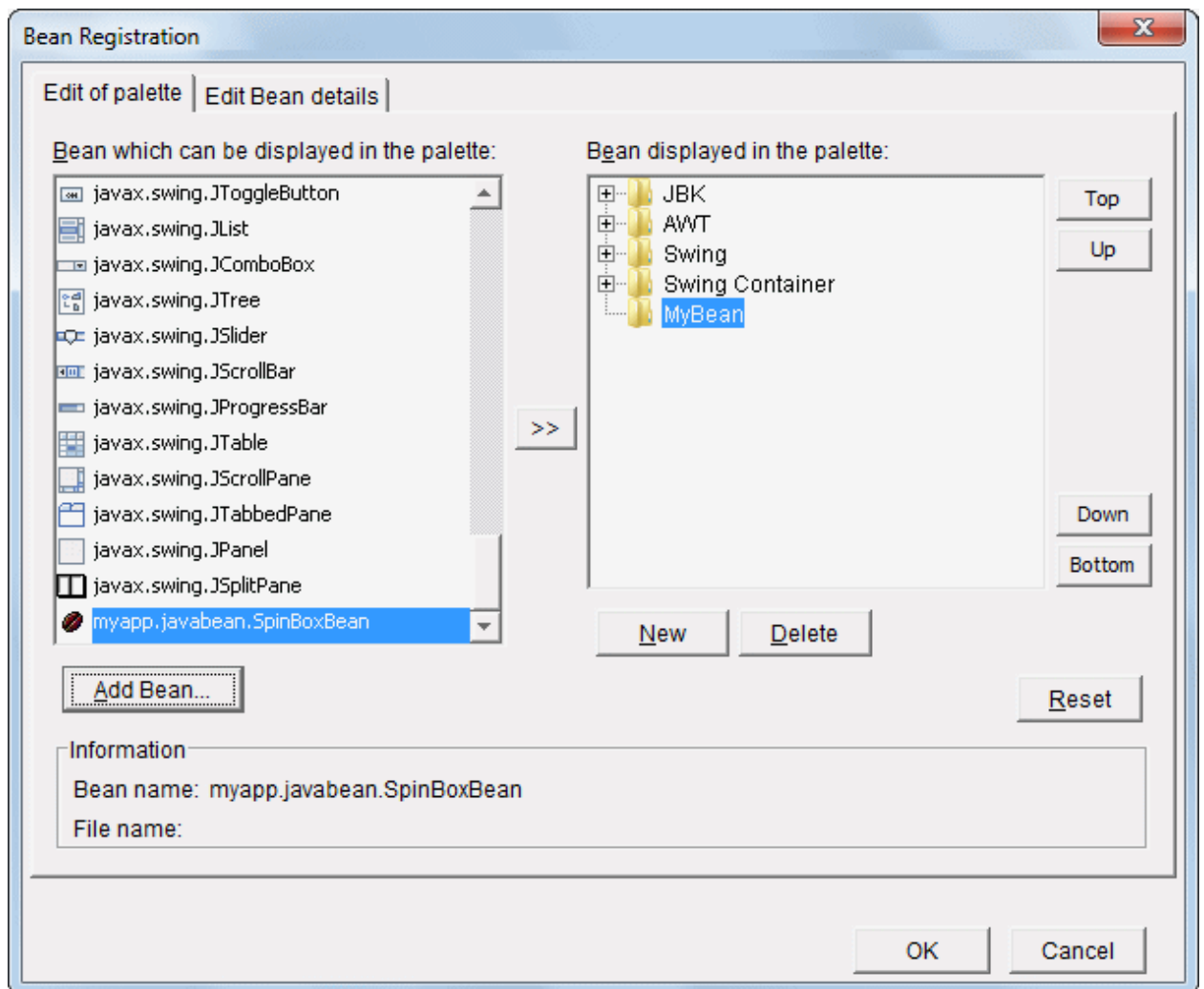
6. The [Reference to class] dialog box is displayed.
Enter "spin" in the [Select an item to open] list box to bring up [SpinBoxBean] in [Matching items] and select it.



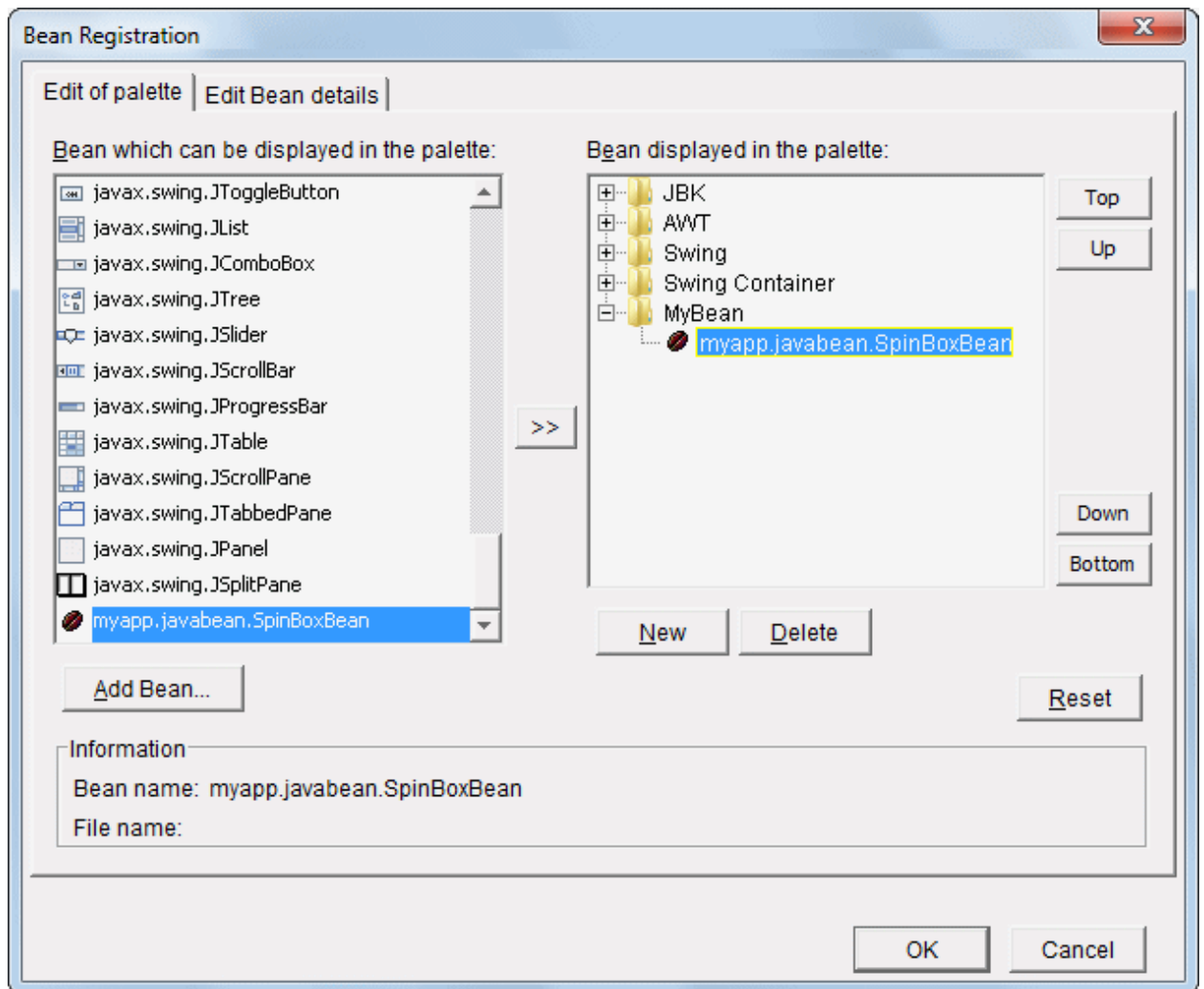
7. "SpinBoxBean" is set in [Class name].
Click [OK].



8. Select the added "myapp.javabeans.SpinBoxBean" in [Bean which can be displayed in the palette] and "MyBean" in [Bean displayed in the palette], then click [>>].



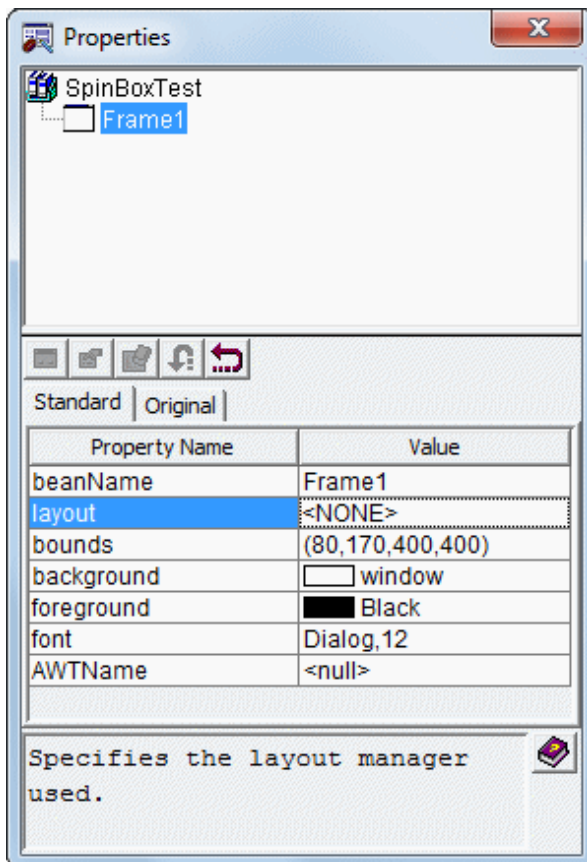
SpinBoxBean is registered in MyBean.
After this, you can paste SpinBoxBean into Java Forms.



9. Click [OK] to close the Registration Bean dialog.

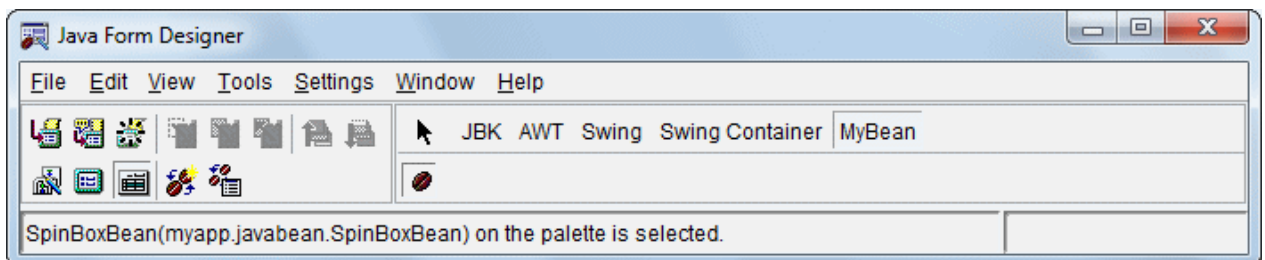
Editing a Java Form

1. Change the [layout] property of [Frame1] to [<NONE>].

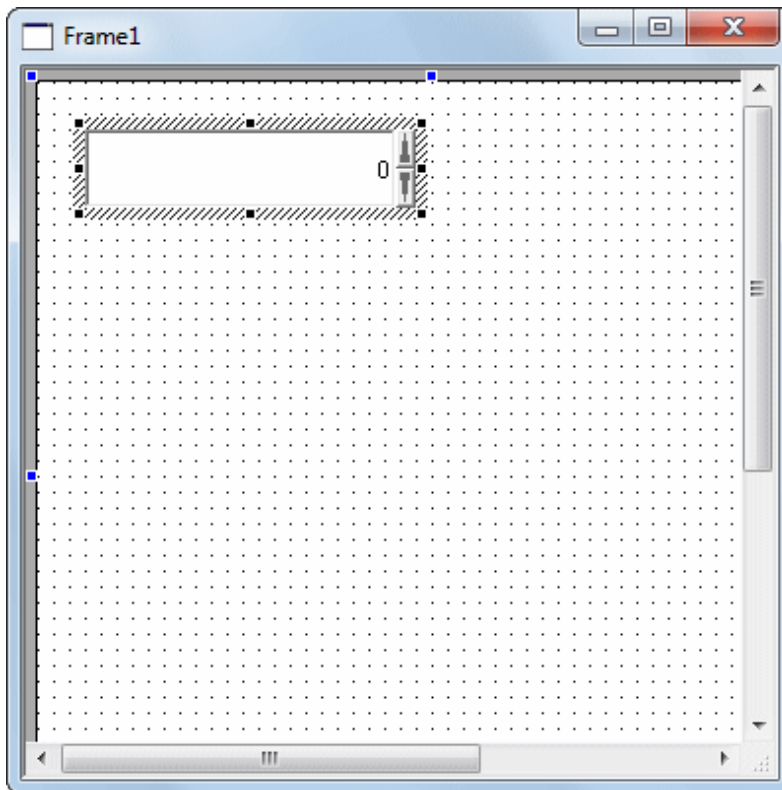


2. Paste the SpinBoxBean.

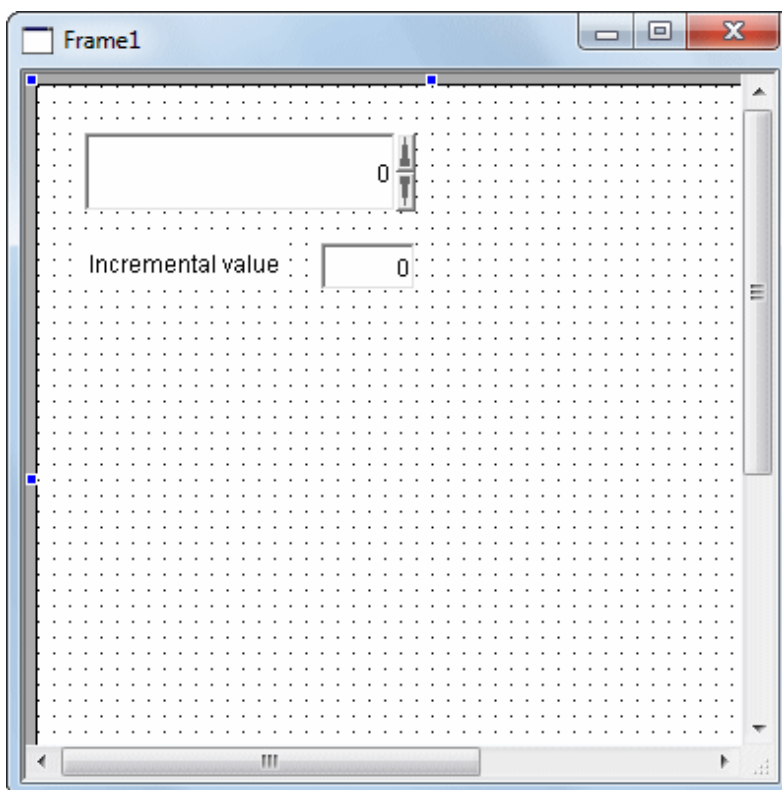
Click [MyBean] on the Object Palette to select it, then click [SpinBoxBean]. Note that if you specified an icon in its BeanInfo definition, that icon is displayed on the Object Palette. In this tutorial, you did not specify any additional icons, so only default icons are displayed on the Object Palette.



- Place the Bean on the Java Form by dragging it with the mouse.



- Paste JBK [Multiple-Line Label]. On the property sheet, specify "Incremental value" for the property of text.
- Paste JBK [Integer Field].
The screen has the following appearance:



Coding an event process

1. Modify the Frame1 initUser\$ method. Click the [Frame1.java] tab in the workbench editor area to make [Frame1.java] active.

```
> > // initialize form
> > init$();
...
> > // call user definition initialization
> > initUser$();
}
...
/**
 * The form is constructed.
 */
public Frame1() {
    > // initialize form
    > init$();
    > // call user definition initialization
    > initUser$();
}
...
/**
 * User definition initialization
 */
private void initUser$() {
    > // The user definition initialization is described at this position.
}
}
```

2. Code the initUser\$ method process.

Add the process for setting initial values to an integer field (jFFieldLong1). Add the text shown in **red**.

```
protected void initUser$() {
    // The user definition initialization is described at this position.
    jFFieldLong1.setValue(SpinBoxBean1.getIncrement());
}
```

3. Code the process in the action event of the integer field (jFFieldLong1).

Double-click [Frame1] > [jFFieldLong1] > [Event] > [action_actionPerformed] in the [Bean List] view.

An event process creation confirmation dialog is displayed. Click "Yes".

The processing procedure of "jFFieldLong1_action_actionPerformed\$" is displayed. Add the text shown in **red**.

```
public void jFFieldLong1_action_actionPerformed$(java.awt.event.ActionEvent e) {
    if (!defaultEventProc(e)) {
        // The procedure when the event is generated is described below.
        SpinBoxBean1.setIncrement(jFFieldLong1.getValue());
    }
}
```

4. Code the process such that the application ends when the frame closes.

When the frame closes, a window event occurs. In the "window_windowClosed" event, code a process corresponding to the window event.

Double-click [Frame1] > [Event] > [window_windowClosed] in the [Bean List] view.

An event process creation confirmation dialog is displayed. Click "Yes."

The processing procedure of "Frame1_window_windowClosed\$" is displayed. Add the text shown in **red**.

```
public void Frame1_window_windowClosed$(java.awt.event.WindowEvent e) {
    if (!defaultEventProc(e)) {
        // The procedure when the event is generated is described below
    }
}
```

```
        System.exit(0);
    }
}
```

5. Save the Java Form.
Select [File] > [Save] from the Java Form Designer menu.
Select [File] > [Exit] from the menu to close the Java Form.

Building

1. Building is executed automatically when resource files (such as the Java files) are saved if the [Build Automatically] item is enabled in the [Project] menu.
If [Build Automatically] is disabled, right-click on the project and select [Build Project] or select [Build Project] in the [Project] menu.

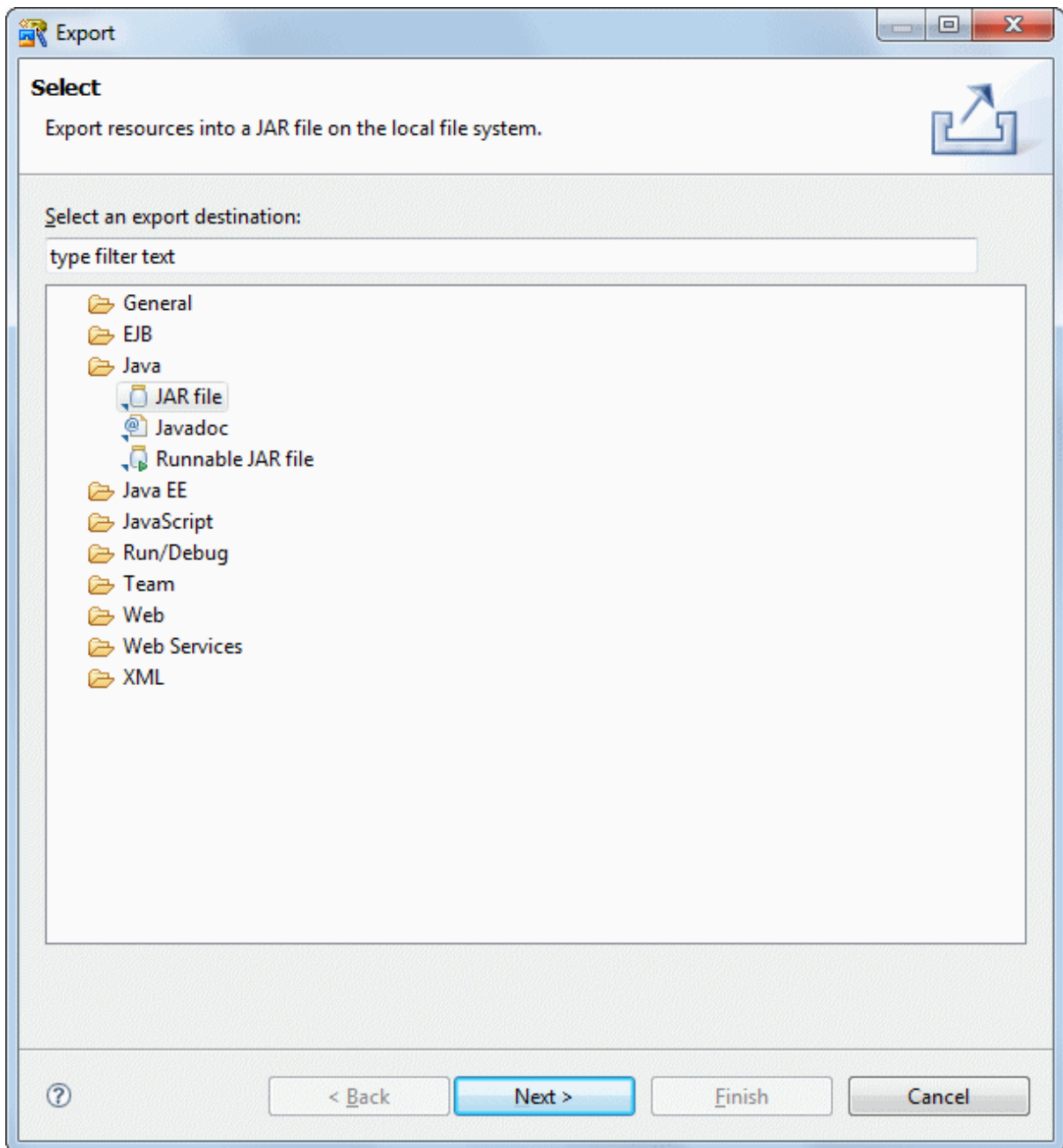


Cleaning of a SpinBoxBean project

An attempt to clean and build a SpinBoxBean project may fail. This is because SpinBoxBean.jar cannot be deleted since Java Form Designer is using this file. In such cases, exit Java Form Designer and close the SpinBoxTest project.

2. A JAR file is required so create it.
Use the export wizard to create the JAR file.

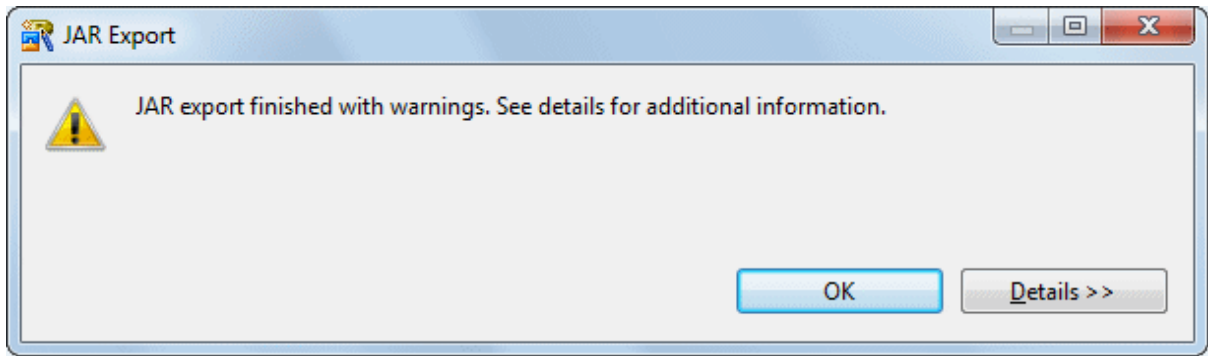
Select [File] > [Export] to start the export wizard.
 Select [Java] > [JAR file] in the export wizard.



- The JAR export wizard is displayed.
 Select [SpinBoxTest] > [src] folder in [Select the resources to export] and enter the following settings.
 After setting the information, click [Finish].

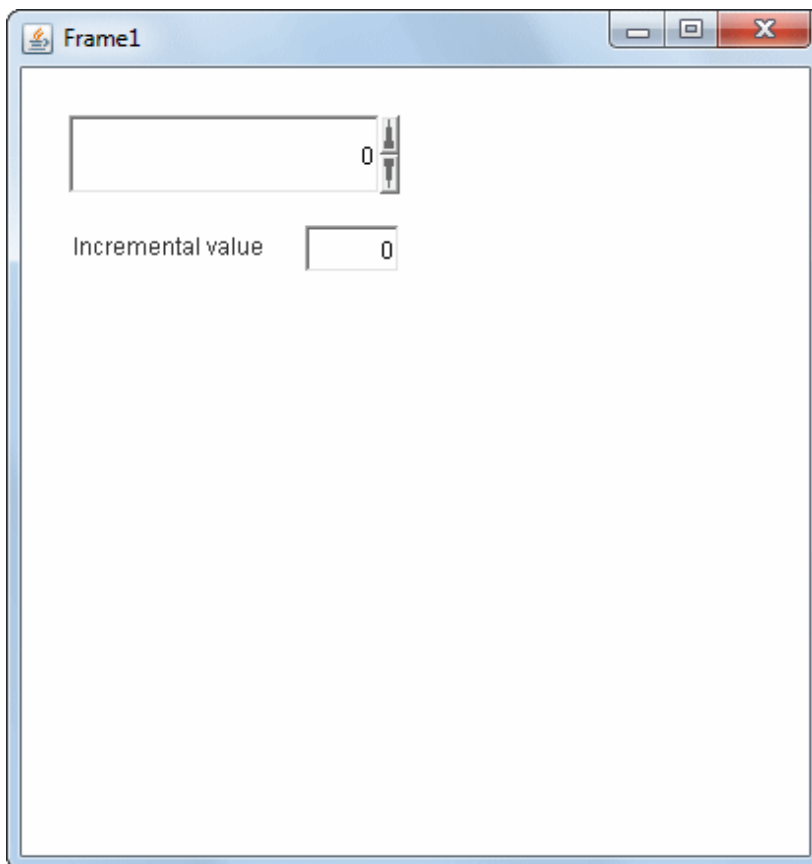
Setup Items	Setup Content
Select the resources to export	src/
Export generated class files and resources	Checked
JAR file	SpinBoxTest/SpinBoxTest.jar
Compress the contents of the JAR file	Checked

The following message is output when exporting JAR, but this does not indicate a problem.

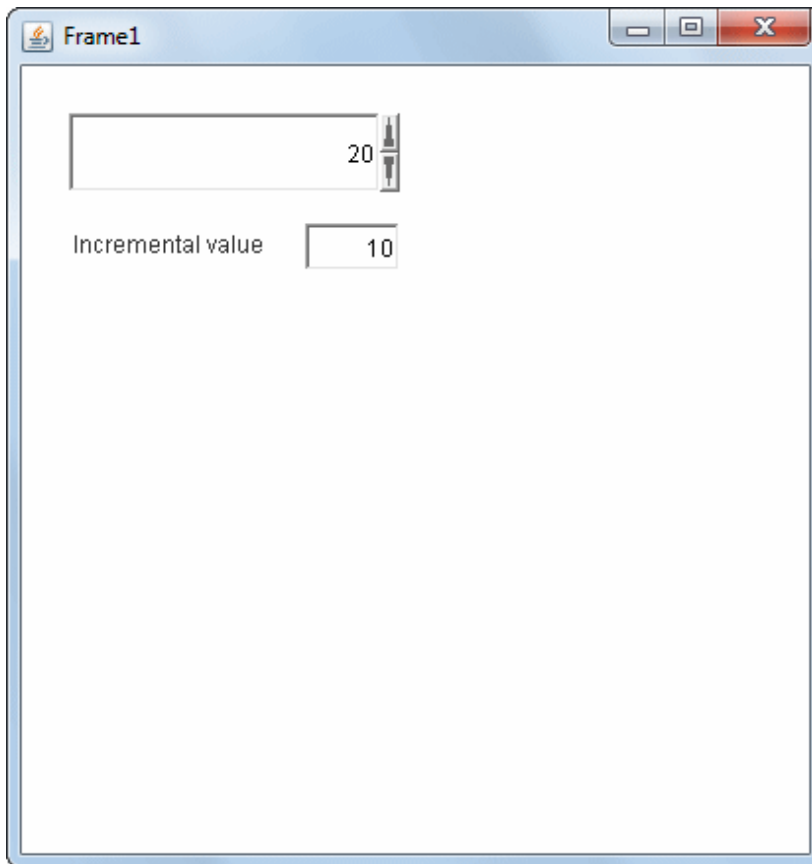


Running

1. Select the file (class) to be executed. Click [SpinBoxTest] > [src] > [myapp.javabeans] > [SpinBoxTest.java] in the [Package Explorer] view of workbench.
2. Select [Run] > [Run As] > [Java Application] from the menu bar. As a result, the application starts, and the Java Form is displayed.



- Specify "10" as the incremental value, and press the [Enter] key.
Click the spin button, and confirm that the value changes by 10 units.



- Click the Close [X] button on the title bar to close the application.

Index

- [A]
- Adding a Try/Catch Block.....192
 - Adding Classpaths.....195
 - Adding Documents to Help.....199
 - Adding Import Statements.....192
 - Annotation.....7,33,53,82
 - Ant Invocation During Build.....195
 - Applet.....123,124,125
 - Archive file.....121,187,217
- [B]
- Back Function.....192
 - Batched import of multiple projects.....224
 - BeanInfo.....143
 - Bean Relationships.....139
 - Breakpoints.....120,186
 - Bug category.....114
 - Bug pattern.....114
 - Build.....3,4
 - build path.....109,184
- [C]
- Cascading Style Sheet.....6
 - Changing the Default Editor.....190
 - Changing the execution environment from IJServer(J2EE) to IJServer Cluster.....217
 - Checking Differences.....201
 - Classpath variable.....195
 - Cleanup.....191
 - Client application.....123
 - Colors and Fonts.....190
 - Confirming file locations.....224
 - Confirming the build path.....224
 - connection profile.....159
 - Constraints.....153
 - Contents Assist.....190
 - Creating an EAR file (earbuild.xml).....215
 - CSS editor.....27
 - CSS file.....27
- [D]
- database.....49,57,153,154,155,159,160,164
 - Dependency Injection.....202
 - Derby.....166,168,171
 - Developing J2EE applications that operate under older versions of Interstage Application Server.....217
- [E]
- EAR Project.....40,61
 - Editor.....1,4
 - Editor settings.....215
 - EJB.....31,34,47,51
 - EJB application.....35,37,46
 - EJB Client.....40
 - Enabling automatic build.....214
 - Enabling Functions not Visible by Default.....190
- Encoding.....190
- enterprise application.....107,181
 - Enterprise Bean.....31
 - Entity.....52
 - Entity class.....72
 - Entity manager.....52,78
 - Exception Breakpoints.....197
 - Expression Evaluation.....197
- [F]
- Fetch.....76
 - File Comments and Type Comments.....193
 - filter file.....116
 - FindBugs.....113
 - Form.....123,133
 - Format.....191
- [G]
- Generating a Getter and Setter.....192
 - Graphical Editor.....128,136,152
- [H]
- HA Database Ready (NativeSQL).....166,168,171
 - HA Database Ready (OpenSQL).....167,168,172
 - Histories used for JDBC driver names and connection destination URLs.....215
 - HTML.....6
 - HTML editor.....22
 - HTML file.....21,25
 - HTTP.....6
- [I]
- IDL compiler build tool.....214
 - IJServer Launch Configuration.....213
 - Index.....153
 - Infopop Help.....200
 - Initializing a Java EE 6 runtime environment that uses JDK 6229
 - Installation procedure.....204
 - IPv6 environment.....206
- [J]
- J2EE 1.4 application.....231
 - JAR package build tool settings.....214
 - Java Applet launch configuration.....215
 - Java application.....123,131,132
 - JavaBeans.....123,135
 - Java Class.....11,125
 - Java compiler build tool settings.....215
 - Javadoc.....201
 - Java editor.....110
 - Java EE.....105,175,202
 - Java EE application.....80,104,105,117,121,175,181,185,187
 - Java file.....111
 - Java Form.....136
 - Java Persistence AP.....52
 - Java Persistence Query Language.....52

JavaScript.....	6,26,28
JavaScript editor.....	27
Java source file.....	109
Java versions.....	216
JAX-WS.....	81
JPA.....	53,54,57,69,78,80
JSF.....	7
JSP.....	7
JSP editor.....	24
JSP Extended Tags.....	25
JSP file.....	15,23,25,111
JSTL.....	7
JUnit classpath setting.....	215
[K]	
Key Binding.....	189
[L]	
launch configuration.....	3
layout manager.....	137
Library.....	123
Logic Class.....	60
[M]	
Manifest classpath.....	109
Maximizing the Editor Area.....	190
Message-driven Bean.....	32,48,49
Migrating J2EE Associated Templates.....	211
Migrating Templates.....	211
Migration from a J2EE application to a Java EE application.....	218
Migration from EJB1.0 or EJB1.1 to EJB2.0.....	221
Migration from EJB2.0 to EJB2.1.....	218
Migration from J2EE Application Client1.3 to J2EE Application Client1.4.....	222
Migration procedure.....	210
Modifications since J2EE1.4.....	105
Modifying existing workspace settings.....	229
Multiple Workbench Display.....	200
[N]	
Notes on Editing the Screen Control Panel.....	152
Notes on Java forms.....	151
Notes on WAR/EJB-JAR/EAR File Creation.....	212
[O]	
O/R mapping.....	52
Opening a Java Declaration.....	194
Optimistic Locking.....	76
Oracle.....	165,167,169
orm.xml.....	52
[P]	
Performing Build using isstudiobld.exe.....	195
persistence.xml.....	52
Persistence context.....	52
Persistence unit.....	52,71
Perspective.....	1,4
Perspective settings.....	214
Point-To-Point model.....	32
Points to consider on applets.....	150
Points to consider on Beans.....	150
PowerGres Plus.....	165,168,170
Predefined Libraries.....	216
Preferences and Properties Page Filters.....	201
Problems View Filter Function.....	194
Project.....	2,3
Projects that can be migrated.....	223
Publish/Subscribe model.....	32
[Q]	
Quick Diff.....	191
[R]	
Refactoring.....	201
relational database.....	153
Resume.....	120,187
[S]	
Schema.....	153
Screen Control Panel.....	141
Searching for a Corresponding Bracket.....	191
Searching Java Reference Locations.....	194
Search Results History.....	194
Service Endpoint Interface.....	100
Servlet.....	7,20
Servlet Class.....	13
Session Bean.....	31,37,47,49
Setting Automatic Execution at Save Time.....	192
Snippets.....	193
Snippets View Tags.....	212
SOAP.....	81
SOAP application of up to V7.....	215
Specifying Syntax Color.....	191
Specifying the Java Version Used for Build and Debugging.....	197
Spell Check.....	191
SQL.....	153
SQL file.....	160
SQL Server.....	165,167,170
Stack Trace View.....	198
Starting a Java EE 6 workbench that uses JDK 6.....	229
Stateful.....	31
Stateless.....	32
Stateless Session Bean.....	98
Step-in to selected Method.....	197
Step Into.....	120,187
Step Over.....	120,187
Step Return.....	120,187
String Externalization.....	194
Switching Files in the Editor.....	192
Symfoware Server.....	164,167,169
Synchronizing Editor and View.....	192
[T]	
table.....	162,163
Table Data editor.....	163
Targeted runtime.....	184
Target runtime.....	109

TCP/IP monitor.....	103
Templates.....	193
Temporary Disabling of Breakpoints.....	196
thin client environment.....	206
TODO Comments.....	193
Tomcat launch configuration.....	216

[U]

Unified EL.....	8
Uninstall procedure.....	205
Updating a workspace.....	214
User library.....	195
Using a Web application project.....	216
Using Proxies.....	189

[V]

validators.....	112
Variable and Return Highlighting.....	193
Variable Name Suggestions.....	193
View.....	1,4,153
View Display Method.....	200

[W]

Warning priority.....	114
web.xml.....	8,28
Web application.....	6,7,8,9,10,19,28,30,62,201
Web Page editor.....	25
Web service.....	81,83,96,97,98,100,101,102
Web service application.....	84,85,96
Window Control Panel.....	142
Workbench.....	1
Workspace.....	2
WS-I Attachments Profile 1.0.....	81
WS-I Basic Profile 1.1.....	81
WSDL.....	81,99,100
WSDL editor.....	99
WSDL Validator.....	100

[X]

XML editor.....	110
XML files.....	110