FUJITSU

Apcoordinator 入門ガイド

Windows/Solaris/Linux

B1WS-1052-03Z0(00)
2014年9月

まえがき

本書は、Apcoordinatorの入門ガイドです。

本書の目的

本書は、Apcoordinatorを使用してWebアプリケーションを作成する人のための入門ガイドです。

本書の読者

本書は、以下の読者を対象としています。

- Apcoordinatorを利用してWebアプリケーションを開発する人
- Apcoordinatorを利用してWebシステムを構築する人

前提知識

本書を読むにあたっては、以下の知識が必要です。

- HTML/XHTML/JavaScript/Java/CSSに関する基本的な知識
- Webアプリケーションに関する基本的な知識

本書の構成

本書の構成は以下のとおりです。

第1章 Apcoordinatorアプリケーションの開発の概要

Apcoordinatorを利用したWebシステムの構築について説明しています。

第2章 簡単なApcoordinatorアプリケーションの作成

シンプルなアプリケーションの作成について説明しています。

第3章 Webアプリケーションの設計と作成

Webアプリケーションの作成について説明しています。

第4章 EJBを利用したアプリケーションの設計と作成

EJBを利用したアプリケーションの作成について説明しています。

付録A サンプル集

Apcoordinatorのサンプルです。

機能および製品の呼び名について

本書では、特に断りのない限り、本製品および関連製品が提供する機能を以下に示す名称で表記している場合があります。

機能	名称
以下の製品が提供するフレームワーク ・ Interstage Business Application Server ・ Interstage Application Server ・ Interstage Studio ・ Interstage Interaction Manager	Apcoordinator
以下の製品が提供するアプリケーションサーバ ・ Interstage Business Application Server ・ Interstage Application Server ・ Interstage Studio	Interstage Application Server または Interstage

また、本書では、Oracle Solarisオペレーティングシステム を Solaris と略記しています。

商標

Microsoft、Active Directory、ActiveX、Excel、Internet Explorer、MS-DOS、MSDN、 Visual Basic、Visual C++、Visual Studio、Windows、Windows NT、Windows Server、Win32 は、米国およびその他の国における 米国Microsoft Corporationの商標または登録商標です。

OracleとJavaは、Oracle Corporationおよびその子会社、関連会社の米国およびその他の国における登録商標です。文中の社名、商品名等は各社の商標または登録商標である場合があります。

その他の記載されている商標および登録商標については、一般に各社の商標または登録商標です。

輸出許可

本ドキュメントを輸出または第三者へ提供する場合は、お客様が居住する国および米国輸出管理関連法規等の規制をご確認のうえ、必要な手続きをおとりください。

出版年月

出版年月および版数	マニュアルコード
2014年9月 第3版	B1WS-1052-03Z0(00)/B1WS-1052-03Z2(00)
2013年6月 第2版	B1WS-1052-02Z0(00)/B1WS-1052-02Z2(00)
2012年8月 初版	B1WS-1052-01Z0(00)/B1WS-1052-01Z2(00)

お願い

本書を無断で他に転載しないようお願いします。
本書は予告なしに変更されることがあります。

著作権

変更履歴

変更内容	変更箇所	版数
輸出許可の記事を修正	まえがき	第3版
ターゲットランタイムをV11.1に修正	2.3.1, 3.3.1, 4.2.1, 4.3.1, 4.4.3	第2版
ビルドの完了により生成されるファイルの記事を削除	2.3.3, 4.2.5, 4.3.8	
ビルドの完了により生成されるファイルの記事を削除し、ビルドについて記事を追加	1.2.3, 4.4.3	
サンプルの削除	付録A	

目　次

第1章 Apcoordinatorアプリケーションの開発の概要....1
1.1 Apcoordinatorとは....1
1.2 Apcoodinatorの機能概要....1
1.2.1 ApcoordinatorによるWebアプリケーションの開発....1
1.2.2 入出力ページ、データBeanおよびビジネスクラスの連携....2
1.2.3 Webサーバへの配備....2

第2章 簡単なApcoordinatorアプリケーションの作成....3
2.1 HelloApcoordinatorの概要....3
2.2 HelloApcoordinatorの設計....5
2.2.1 HelloApcoordinatorアプリケーションの各種ファイル....5
2.2.2 HelloApcoordinatorの動作概要....6
2.3 HelloApcoordinatorの作成....8
2.3.1 Apcoordinatorアプリケーションのプロジェクト作成....8
2.3.2 HelloApcoordinatorアプリケーションの作成....9
2.3.3 ファイル情報の設定とビルド....9
2.3.4 実行....11

第3章 Webアプリケーションの設計と作成....12
3.1 行動予定表アプリケーションの概要....12
3.2 行動予定表アプリケーションの設計....14
3.2.1 構成の設計....14
3.2.2 行動予定表アプリケーション詳細....20
3.3 行動予定表アプリケーションの作成....23
3.3.1 Apcoordinatorアプリケーションのプロジェクト作成....23
3.3.2 データBeanの作成....24
3.3.3 ユーザ定義型とデータ保存クラス作成....30
3.3.4 入出力ページの作成....33
3.3.5 ビジネスクラスの生成....37
3.3.6 ビジネスロジックの追加....38
3.3.7 コンパイルとローカル実行の方法....43
3.3.8 マップファイルの編集....43
3.3.9 Webサーバへの配備....44
3.3.10 Webサーバ上での実行....45

第4章 EJBを利用したアプリケーションの設計と作成....48
4.1 会議室予約システムアプリケーションの概要....48
4.2 EJBの開発....51
4.2.1 プロジェクトの作成....51
4.2.2 ビジネスクラスの作成....53
4.2.3 EntityBeanの作成....54
4.2.4 デプロイメント記述子の編集....56
4.2.5 プロジェクトのビルド....57
4.3 Webアプリケーションの開発....57
4.3.1 プロジェクトの作成....57
4.3.2 データBeanの作成....58
4.3.3 入出力ページの作成....59
4.3.4 データ送受信のためのクラスの作成....60
4.3.5 ビジネスクラスの作成....61
4.3.6 リモートマップの作成....62
4.3.7 データBean変換マップの作成....62
4.3.8 プロジェクトのビルド....63
4.4 アプリケーションサーバでの実行準備....63
4.4.1 Symfowareへのテーブルの登録....63
4.4.2 JDBCデータソースの定義....65

4.4.3 EARファイルの作成....66
4.4.4 Webサーバへの配備....67
4.5 アプリケーションサーバでの実行と、ブラウザによる実行確認....68
4.5.1 Symfowareサービスの起動....68
4.5.2 ワークユニットの起動....68
4.5.3 ブラウザによる実行確認....68

付録A サンプル集....73
A.1 サンプルについて....73
A.1.1 サンプルの格納場所....73
A.1.2 サンプルの利用方法....73
A.1.3 コンパイル済みサンプルについて....74
A.2 基本的な構成のサンプル....74
A.3 中級的なサンプル....76
A.4 応用的なサンプル....86

索引....90

第1章 Apcoordinatorアプリケーションの開発の概要

1.1 Apcoordinatorとは

Apcoordinator（エー・ピー・コーディネーター）は、Java(TM) 2 Platform, Enterprise Edition（J2EE）およびJava(TM) Platform, Enterprise Edition(Java EE)に従ったアプリケーションの構築を支援するアプリケーションフレームワーク製品です。ApcoordinatorはWebアプリケーションなどの開発を支援します。

1.2 Apcoodinatorの機能概要

ここでは、Apcoodinatorを利用したWebアプリケーション作成の流れについて、説明します。

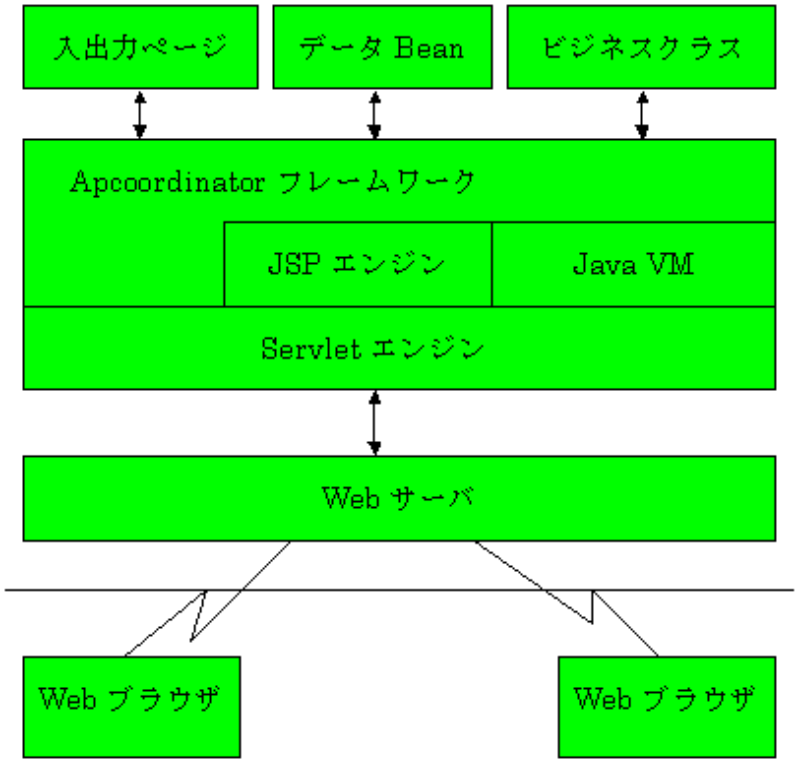
1.2.1 ApcoordinatorによるWebアプリケーションの開発

Webアプリケーションでは、Webブラウザ上に表示されるアプリケーションの画面を、Webページと呼びます。Webページ上では、HTMLのフォームを使用してデータを入力し、処理した結果を次のWebページに表示させたり、リンクを利用してほかのWebページを表示させたりすることができます。

Apcoordinatorでは上記の考えをさらにおしすすめ、拡張性、保守性を向上させる仕組みを提供します。

Apcoordinatorを使用したWebアプリケーションの構成を以下に示します。

図1.1 Apcoordinatorを使用したWebアプリケーションの構成



Interstage StudioのApcoordinatorアプリケーション開発機能では、複数の入出力ページ（いわゆるWebページ）から構成されるWebサイトの構築を支援します。

Apcoordinatorアプリケーション開発機能には、以下の機能があります。

- 入出力ページ作成機能
- データBean作成機能
- ビジネスクラス（ロジック）開発支援機能

・ デバッグ機能

Apcoordinatorアプリケーション開発機能では、画面設計からのソースプログラム自動生成とページベースの開発フレームワークの提供によって、画面設計部とデータ構造保持部とビジネスロジック（いわゆるアプリケーション処理部）の分離を可能にしています。画面設計はGUIベースのページエディタとApcoordinatorが提供する画面部品によって行うことができ、一方、ビジネスロジック処理部においては描画に関する記述がほぼ不要になります。

また、データ指向の構造を持つことが可能なため、複数のページと画面遷移を含むWebアプリケーションを効率よく構築することが可能です。画面遷移は、マッピングルールを記述するファイルにより、システムコンポーネントの依存関係の修正が容易に行えるようになっています。

デバッグ機能では、アプリケーションをInterstage上で動作させてデバッグするだけでなく、ローカルマシン上で動作させることも可能です。

1.2.2 入出力ページ、データBeanおよびビジネスクラスの連携

Apcoordinatorでは、Webアプリケーションの開発を以下の3つのコンポーネントに分離することにより、コンポーネントの再利用性と拡張の容易性を実現しています。

・ 入出力ページ

・ データBean

・ ビジネスクラス

入出力ページ

Apcoordinatorにおける入出力ページは、ページマップで関連付けられたBeanとデータをやり取りします。入出力ページとデータBean間のデータのやり取りは、Apcoordinatorが自動的に行います。Apcoordinatorでは、入出力ページ上にデータを直接操作するロジックを埋め込むことは、ロジックの分散化につながるので推奨されておりません。

データBean

Apcoordinatorでは入出力ページで使用されるデータは、データBeanに保持します。入力や表示に必要な一連のデータがデータBeanにまとめて保持されることにより、ビジネスクラスで扱うデータをBean単位で操作できるので、データ操作の方法がよりわかりやすいものになります。また入出力ページ側のデータアクセスも、データBeanとのやり取りのみを考慮するだけでよく、ビジネスクラスの実装に左右されない画面デザインが可能になります。

ビジネスクラス

Apcoordinatorのビジネスクラスは、ビジネスロジックを実装するクラスです。ビジネスクラスは、コマンドマップで定義されたアクションを処理するメソッドを実装します。各メソッドの中では、データBeanから値を取得し、その値を処理して別のデータBeanに書き込みます。ビジネスクラス側では、入出力ページでのデータの表示方法を考慮する必要はありません。

1.2.3 Webサーバへの配備

開発したアプリケーションを運用環境にインストールすることを配備と呼びます。Webアプリケーションの資産をWebサーバに配備するには、WARファイルを作成し、WARファイルをWebサーバに配備します。WARファイルは、Webアプリケーションを構成するファイルをすべてJAR形式でパッケージ化したファイルです。

WARファイルの作成

WARファイルは［ファイル］メニューの［エクスポート］により作成します。エクスポート前にアプリケーションをビルドする必要があります。［プロジェクト］メニューの［自動的にビルド］をチェック付きにすることで、自動的にビルドされます。チェックが入っていていない場合は、［プロジェクトのビルド］を選択します。

WARファイルの配備

作成したWARファイルをWebサーバに配備する方法は、ご使用になるアプリケーションサーバに依存します。Interstageをご使用の場合、Interstage Java EE管理コンソールで配備します。詳細については、Interstage Java EE管理コンソールのヘルプを参照してください。

第2章 簡単なApcoordinatorアプリケーションの作成

本章では、簡単なApcoordinatorアプリケーションを作成し、Apcoordinatorの動作の概要を説明します。入力フォームから入力されたデータ、アクセスカウンタおよび時刻を表示する簡単なApcoordinatorアプリケーションを作成します。

このアプリケーションの開発を通して、Interstage StudioでApcoordinatorアプリケーションを開発する際に必要な情報を紹介します。

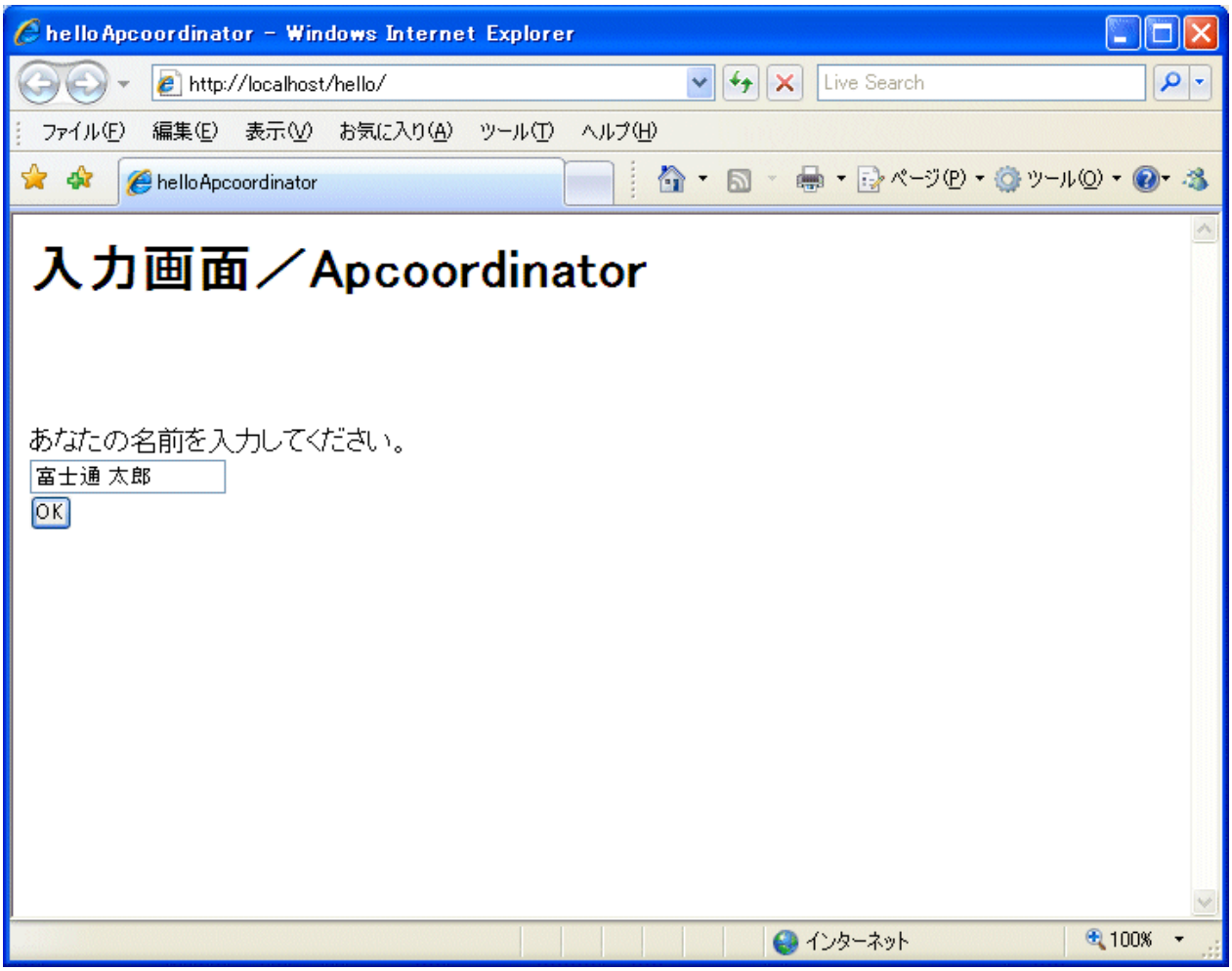
2.1 HelloApcoordinatorの概要

機能概要説明

ここではHelloApcoordinatorの機能を画面を見ながら説明していきます。

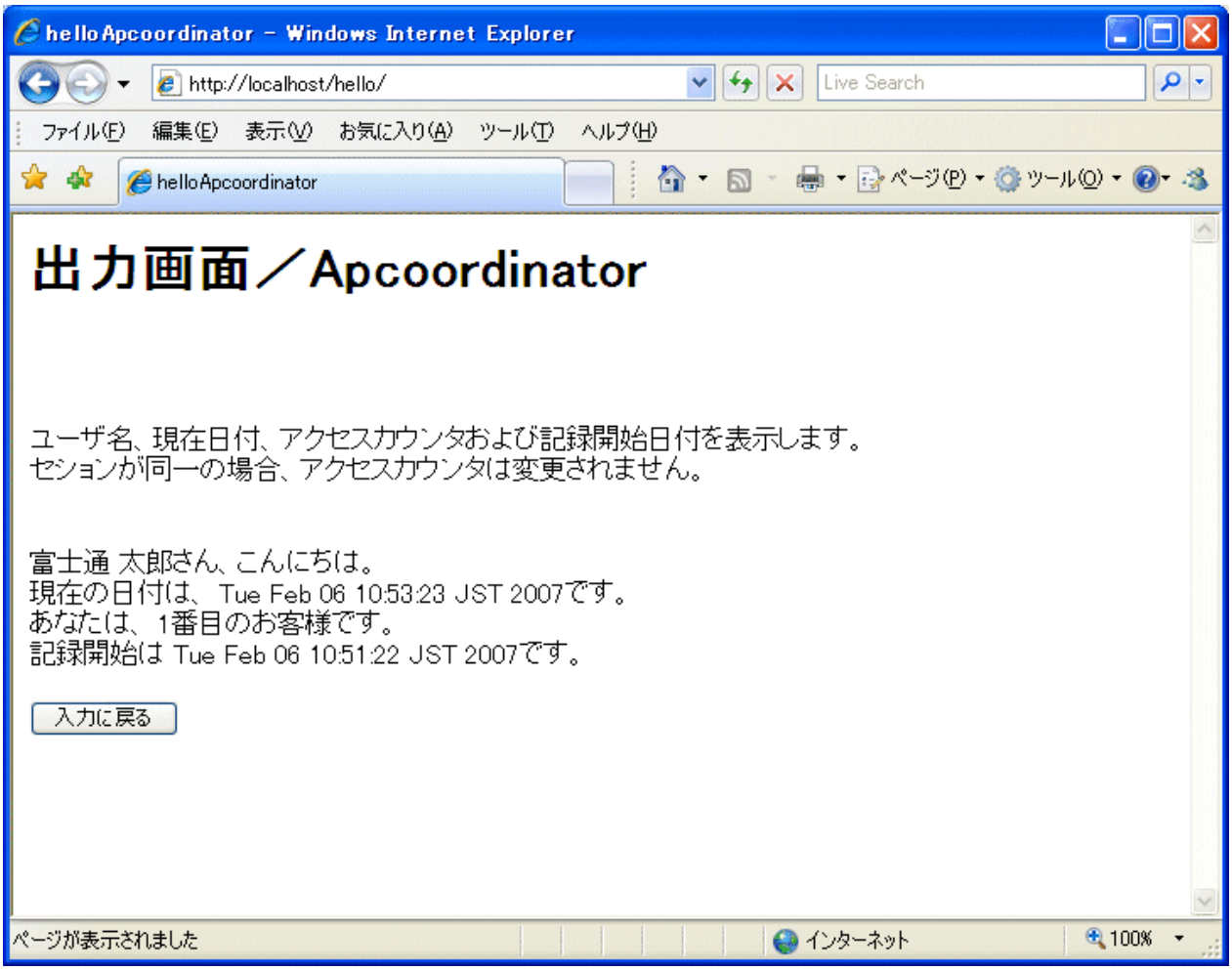
名前入力画面

名前を入力するテキストフィールドを表示します。



出力画面

入力された名前、現在の日付、アクセスカウンタを表示します。



画面項目

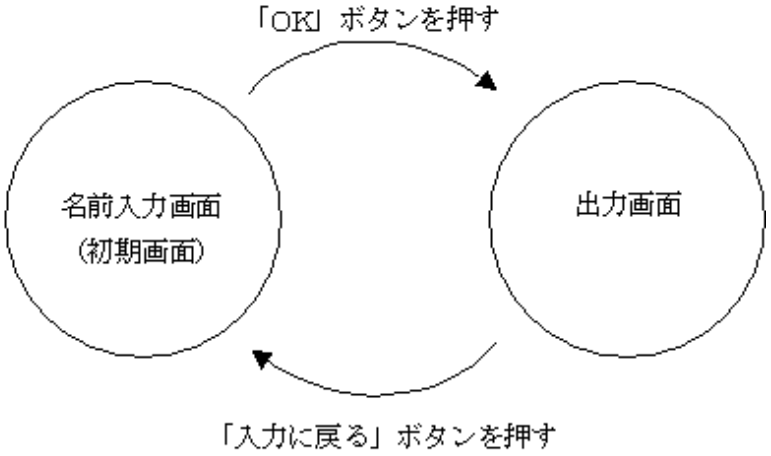
それぞれの画面では、以下に示された項目や部品を表示します。

- 名前入力画面
  - 名前を入力するためのテキストフィールド
- 出力画面
  - ユーザ名
  - 現在日付
  - アクセスカウンタ
  - 記録開始日付

画面遷移

以下に画面の遷移を表します。最初に表示する画面は、名前入力画面です。矢印の方向は画面の遷移を表します。矢印の上もしくは下に画面遷移のきっかけとなる動作を記述しています。

図2.1 HelloApcoordinatorアプリケーションの画面遷移図



画面構成

名前入力画面

最初に名前入力画面を表示します。名前入力画面では任意の文字列を入力します。

出力画面

出力画面では、名前入力画面から入力された文字列を受け取って表示します。その他に、現在の日付とApcoordinatorのセション管理機能を用いたアクセスカウンタと記録を開始した日付を表示します。

アプリケーションサーバとJava統合開発環境の実行画面は、提供する機能の違いにより、お使いのエディションや動作環境によって異なる場合があります。

2.2 HelloApcoordinatorの設計

ここでは、HelloApcoordinatorに基づいて、Apcoordinatorアプリケーションに必要な各種ファイルと動作の概要を説明します。

2.2.1 HelloApcoordinatorアプリケーションの各種ファイル

ここでは、HelloApcoordinatorアプリケーションの作成ウィザードによって作成されたファイルを見ていきます。

HelloApcoordinatorアプリケーションでは、大きく分けて7種類のファイルが生成されます。これら7種類16ファイルの概要は次の表のとおりです。

ファイル種別	ファイル名	内容
制御ページ	main.jsp	Apcoordinatorアプリケーションでは、クライアントからのリクエストを制御ページが受け取ります。そのため、アプリケーション内には必ず一つ以上の制御ページが必要です。
ファクトリクラスの拡張	HelloFactory.java HelloApplication.java HelloSession.java	ファクトリクラスを拡張することで、セション管理機能やDB連携機能を使用することができます。詳しくはApcoordinator ユーザーズガイドをご覧ください。
データBean	HelloHeadBean.java HelloBodyBean.java	入出力ページとビジネスクラスの間のデータ受渡しを行うJavaBeans形式のクラスです。
ビジネスクラス	HelloHandler.java	ビジネスロジックを記述するJavaのクラスです。
入出力ページ	helloHeadPage.jsp helloInputPage.jsp helloOutputPage.jsp helloError.jsp	入出力項目を持つJSPページを入出力ページと呼びます。一つの画面を複数の入出力ページで構成することができます。

ファイル種別	ファイル名	内容
関係定義ファイル	commands.map pages.map	Apcoordinatorでは、アプリケーション内でビジネスクラス名やJSPファイル名を直接指定せずに、関係定義ファイルを用いて間接的に指定します。関係定義ファイルにはコマンドマップとページマップがあります。
その他	web.xml ujiall.tld hello.txt	Interstage Studioは上記のファイルの他に3つのファイルをプロジェクト内に作成します。 web.xmlファイルはWebアプリケーションの環境定義ファイルです。アプリケーションのパラメタや、タグライブラリの場所など、Webアプリケーションの環境を定義します。 ujiall.tldはUJIタグのタグライブラリデスクリプタファイルです。UJIタグとは、Apcoordinatorが提供しているJSP拡張タグのことです。 hello.txtはウィザードで生成された各種ファイルの説明が記述されています。

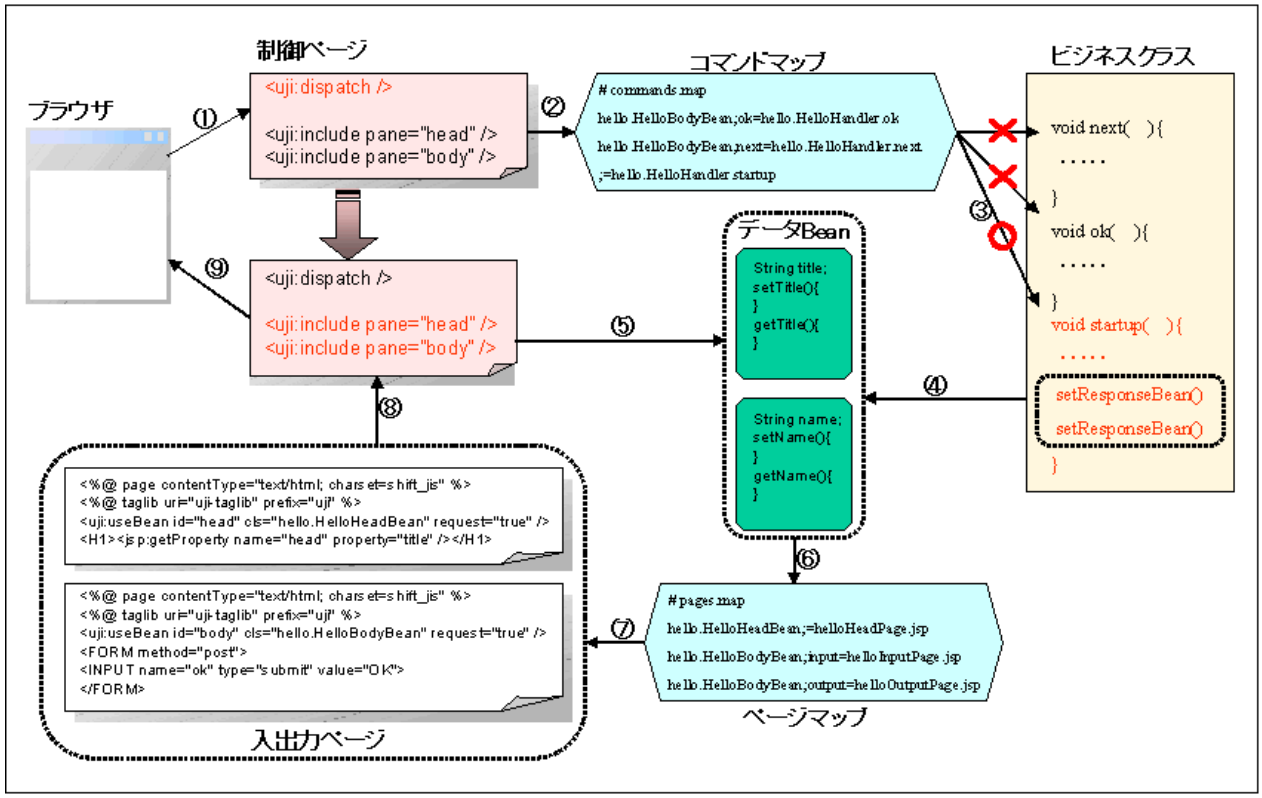
上記のファイルの中で重要なのは、入出力ページ、データBean、ビジネスクラスの3つです。これら3つのファイルによってWebアプリケーションの主な構成要素である 画面、データ、ロジックを分離して作成することが可能です。次節では、これら3つのファイルがアプリケーションとしてどのように動くのかを見ていきます。

2.2.2 HelloApcoordinatorの動作概要

ここでは、HelloApcoordinatorの動作概要を説明します。

まずは、初期起動時の動作です。

初期起動時の動作の概念図を下図に示します。下図のように初期起動は9つのフェーズに分かれています。



次に上記の図より、具体的な動きを追っていきます。図中の番号と、その動作を下記の表に示します。

1. リクエストの送信
   まず、ブラウザから制御ページにリクエストが送信されます。

2. uji:dispatchタグの起動
   制御ページに記述されてあるuji:dispatchタグから、Apcoordinatorが起動します。

3. コマンドマップから初期起動用のメソッドを検索
   コマンドマップ内から初期起動用のメソッドを検索します。

   初期起動では、下記の行が検索されます。

   ;="クラス名.メソッド名"

   HelloApcoordinatorではhello.HelloHandlerクラスのstartup()メソッドが呼ばれます。

4. データBeanを領域にセット
   HelloHandlerクラスのstartup()メソッド内でDispatchContextクラスのsetResponseBean()メソッドが呼ばれ、データBeanに領域名がセットされます。

   そのとき、HelloApcoordinatorではデータBeanに表示モードを設定しています。

   これは、複数の入出力ページに1つのデータBeanが対応している場合に用います。

   入出力ページに対応した表示モードをデータBeanに設定します。ここではinputが設定されています。

   これで、uji:dispatchタグの動作は終了です。

5. uji:includeタグの起動
   次に、制御ページに記述されているuji:includeタグが呼ばれ、Apcoordinatorが起動します。

6. 領域名からデータBean名を検索
   uji:includeタグのpaneアトリビュートに記述されている領域名からデータBeanを検索します。

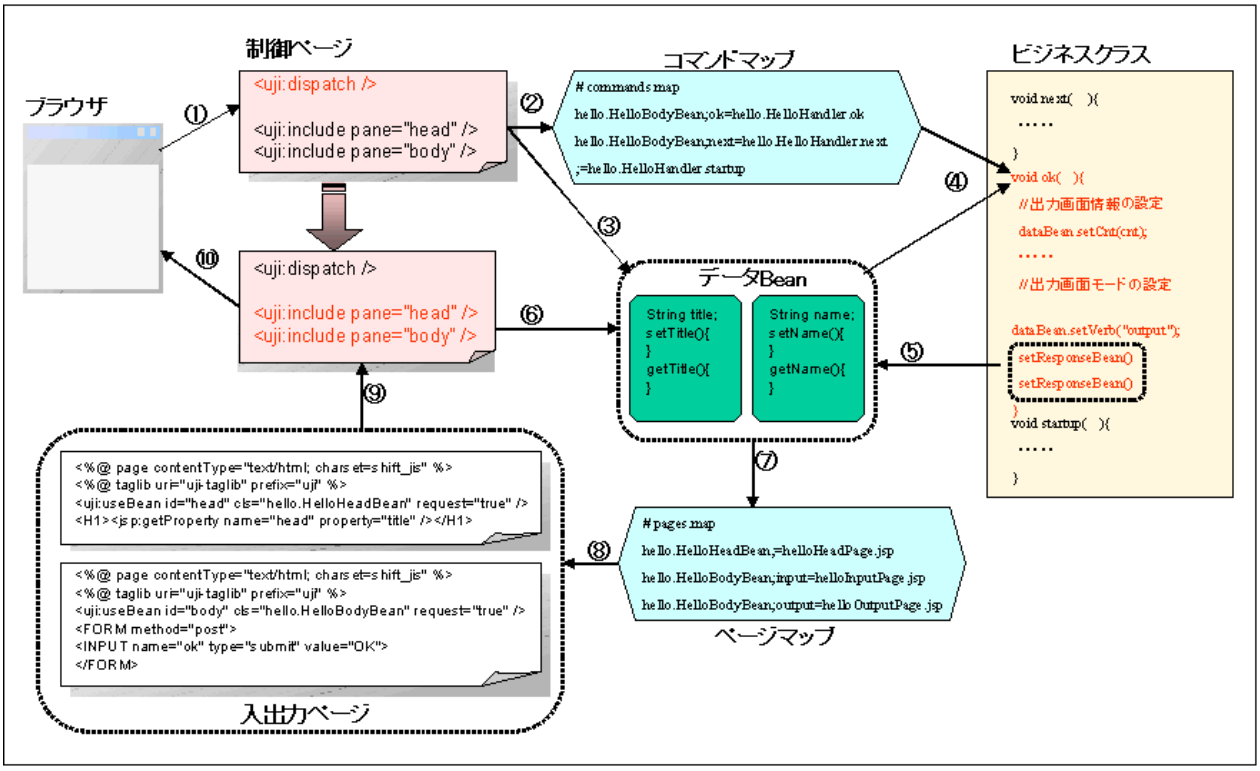
7. ページマップから入出力ページを検索
   検索されたデータBeanをもとに、ページマップから対応する入出力ページを検索します。

8. 入出力ページをuji:includeタグにセット
   入出力ページがuji:includeタグのある場所にセットされます。

9. レスポンスの送信
   最後に生成された表示用データをブラウザに送信します。

以上の9つのフェーズを経て、レスポンスが送信されます。ただし、5～8はuji:includeタグの数だけ繰り返されます。

次に、ブラウザからデータが送られて来る場合を考えます。この場合は、下図のように10のフェーズに分かれます。



先ほどと同様に、上記の図より、具体的な動きを追っていきます。図中の番号と、その動作を下記の表に示します。

初期起動時と異なるのは、リクエスト受信時にデータBeanにデータがセットされ、ビジネスクラスに渡されるところです。

1. リクエストの送信
   まず、ブラウザから制御ページにリクエストが送信されます。
2. uji:dispatchタグの起動
   制御ページに記述されてあるuji:dispatchタグから、Apcoordinatorが起動します。
3. データBeanにデータをセット
   ブラウザから送信されたデータをデータBeanにセットします。
4. コマンドマップからメソッドを検索
   コマンドマップ内からメソッドを検索します。

   ここでは、HelloApcoordinatorの名前入力画面からリクエストが送信されたとします。この場合、ブラウザからokボタンが押され、リクエスト内にHelloBodyBeanクラスのデータが格納されています。

   よって、下記の記述からHelloHandlerクラスのok()メソッドが呼ばれます。

   hello.HelloBodyBean;ok=hello.HelloHandler.ok

   このとき、3のデータBeanがok()メソッドに渡されます。
5. データBeanを領域にセット
   HelloHandlerクラスのok()メソッド内でDispatchContextクラスのsetResponseBean()メソッドが呼ばれ、データBeanに領域名がセットされます。

   そのとき、HelloApcoordinatorではデータBeanに表示モードを設定しています。

   これは、複数の入出力ページに1つのデータBeanが対応している場合に用います。

   入出力ページに対応した表示モードをデータBeanに設定します。ここではoutputが設定されています。

   これで、uji:dispatchタグの動作は終了です。
6. uji:includeタグの起動
   次に、制御ページに記述されているuji:includeタグが呼ばれ、Apcoordinatorが起動します。
7. 領域名からデータBean名を検索
   uji:includeタグのpaneアトリビュートに記述されている領域名からデータBeanを検索します。
8. ページマップから入出力ページを検索
   検索されたデータBeanをもとに、ページマップから対応する入出力ページを検索します。
9. 入出力ページをuji:includeタグにセット
   入出力ページがuji:includeタグのある場所にセットされます。
10. レスポンスの送信
    最後に生成された表示用データをブラウザに送信します。

2.3 HelloApcoordinatorの作成

2.3.1 Apcoordinatorアプリケーションのプロジェクト作成

まず、Interstage Studioを用いてApcoordinatorアプリケーションのプロジェクトを作成します。

1. Interstage Studioを起動します。
2. [ファイル]-[新規]メニューの[プロジェクト]を選択します。
3. 「新規プロジェクト」画面が表示されます。この画面の指示に従って必要な項目を入力すると、プロジェクトが作成されます。
   Apcoordinatorアプリケーションのプロジェクトを作成する場合、以下のようにします。

   [Apcoordinator] > [Webアプリケーションプロジェクト(Apcoordinator)]を選択し、[次へ]ボタンをクリックします。
4. 「新規動的 Web プロジェクト」の「動的 Web プロジェクト」入力画面が表示されます。

   設定する項目は以下のとおりです。

設定項目	設定内容	デフォルト設定値
プロジェクト名	"hello"を指定します。	(なし)
プロジェクトコンテンツ	[デフォルトの使用]をチェックします。	[デフォルトの使用]にチェック付き
ターゲットランタイム	[Interstage Application Server V11.1 IJServer Cluster (Java EE)]を選択します。	[なし]が選択
動的 Web モジュール バージョン	[2.5]を選択します。	[2.5]が選択
構成	[Interstage Application Server V11.1 IJServer Cluster (Java EE)デフォルト構成]を選択します。「ターゲットランタイム」を選択することで自動的に選択されます。	[Apcoordinator Webプロジェクトのデフォルト構成]が選択
EARにプロジェクトを追加	チェックしません。	チェックマークなし

[次へ]ボタンをクリックします。

5. 「新規動的 Web プロジェクト」の「Web モジュール」入力画面が表示されます。

   設定する項目は以下のとおりです。

設定項目	設定内容	デフォルト設定値
テンテキストルート	"hello"を指定します。	プロジェクト名
コンテンツフォルダ	"WebContent"を指定します。	"WebContent"
Javaソースフォルダ	"src"を指定します。	"src"
Deployment Descriptorの生成	チェックします。	チェックマーク付

[次へ]ボタンをクリックします。

これで、Apcoordinatorアプリケーションを作成する準備が整いました。

引き続き、Apcoordinatorアプリケーションの新規作成画面が起動されます。

2.3.2 HelloApcoordinatorアプリケーションの作成

[Hello Apcoordinatorアプリケーション]を選択し、[次へ] ボタンをクリックします。

次にApcoordinatorアプリケーションに必要なファイルを作成するウィザード画面が表示されます。

2.3.3 ファイル情報の設定とビルド

ここからは、Apcoordinatorアプリケーションに必要な各種ファイルの情報を設定していきます。

必要な情報は以下のとおりです。

1. 制御ページ情報
2. ファクトリ拡張情報
3. データBeanおよびプロパティの情報
4. ビジネスクラス情報
5. 入出力ページ情報

各ファイルの役割はHelloApcoordinatorの説明で説明します。ここではひとまず以下の説明に従って入力を行い、ウィザードの実行を完了させてください。

1 制御ページ情報の設定

まず、制御ページに関する情報を設定します。

デフォルトで下記の項目が設定されています。

[制御ページのタイトル]をhelloApcoordinatorに変更して、[次へ]ボタンをクリックします。

設定項目	設定内容	デフォルト設定値
ファイル名	"main"を指定します。 制御ページに指定するファイル名を拡張子なしで指定します。	main
タイトル	"helloApcoordinator"を指定します。 ここに入力した文字列は、ブラウザのタイトルバーに表示される名前となります。	プロジェクト名
エラーページを使用する	チェックします。 アプリケーションの実行途中にエラーが生じたときに、表示するページです。使用するときはチェックします。	チェックマーク付
エラーページファイル名	"helloError"を指定します。 例外が発生した時に、エラー出力を表示するページ名の名前を定義します。	プロジェクト名+Error

次のステップでは、ファクトリの拡張情報を定義します。

2 ファクトリ拡張情報の設定

設定する項目は以下のとおりです。

設定項目	設定内容	デフォルト設定値
パッケージ	"hello"を指定します。 アプリケーションのパッケージ名を指定します。この名前がパッケージ名となります。	プロジェクト名
ファクトリクラス	"HelloFactory"を指定します。 アプリケーション独自の拡張ファクトリクラスを作成するときのクラス名です。	プロジェクト名+Factory
アプリケーションクラス	"HelloApplication"を指定します。 アプリケーション独自の拡張アプリケーションクラスを作成するときのクラス名 です。	プロジェクト名 +Application
セションクラスを拡張する	チェックします。	チェックマーク付
セション管理機能を拡張する	チェックします。	チェックマーク付
セションクラス	"HelloSession"を指定します。 アプリケーション独自の拡張セションクラスを作成するときのクラス名です。	プロジェクト名+Session

[次へ] ボタンをクリックします。

3 データBeanおよびプロパティ情報の設定

次に、データBeanおよびプロパティの情報を設定します。

設定項目	設定内容	デフォルト設定値
ヘッダ領域用ファイル名	"HelloHeadBean"を指定します。	プロジェクト名＋HeadBean
画面タイトル用プロパティ名	"title"を指定します。	title
ボディ領域用ファイル名	"HelloBodyBean"を指定します。	プロジェクト名＋BodyBean
入出力用プロパティ名	"name"を指定します。	name

[次へ] ボタンをクリックします。

4 ビジネスクラス情報の設定

次に、ビジネスクラスの情報を設定します。

設定項目	設定内容	デフォルト設定値
ファイル名	"HelloHandler"を指定します。	プロジェクト名＋Handler
セションスコープ	チェックします。 HelloApcoordinatorでは必須です。	チェックマーク付

[次へ] ボタンをクリックします。

5 入出力ページ情報の設定

次に生成する各JSPファイルの名前を指定します。

設定項目	設定内容	デフォルト設定値
ヘッダ画面用ファイル名	"helloHeadPage"を指定します。	プロジェクト名＋HeadPage
入力画面用ファイル名	"helloInputPage"を指定します。	プロジェクト名＋InputPage
出力画面用ファイル名	"helloOutputPage"を指定します。	プロジェクト名＋OutputPage

[完了] ボタンをクリックします。

Apcoordinatorのパースペクティブを開いていない場合は、確認のダイアログが表示されます。

[はい]ボタンをクリックし、Apcoordinatorのパースペクティブを開きます。

以上の操作で新規プロジェクトが作成され、プロジェクト内にファイルが生成されます。

Interstage Studioの[プロジェクト]メニューの[自動的にビルド]は、デフォルトでチェックが入っています。このため、ファイルが生成されると同時にビルドも完了します。

問題ビューでエラーが発生していないことを確認します。

注意

問題ビューにserialVersionUIDに関する警告が表示されますが、問題はありません。

2.3.4 実行

次に、作成したプロジェクトを実行します。

1. Interstage Studioの[実行]メニューの[実行]から[サーバで実行]を選択します。

   新規にサーバを定義する場合は、[サーバで実行]画面で[サーバの選択方法]に[手動で新規サーバを定義]を選択し、サーバを作成します。詳細は「Interstage Studioユーザーズガイド」を参照ください。

2. [サーバで実行]画面が表示されます。使用するサーバを選択し、[次へ]ボタンをクリックします。

3. [プロジェクトの追加および除去]画面が表示されます。構成プロジェクトを確認し、[完了]ボタンをクリックします。作成したプロジェクトがローカルIJServerに配備されたあと、ローカルIJServerの起動が行われます。

4. ブラウザが起動され、名前の入力画面が表示されます。

5. 自分の名前を入力して、[OK]ボタンをクリックします。入力された名前とサーバの現在時刻および現在までのアクセス回数が表示されます。

第3章 Webアプリケーションの設計と作成

本章では、各人の行き先を管理する「行動予定表アプリケーション」を例に、Apcoordinatorアプリケーションの作成手順を説明します。

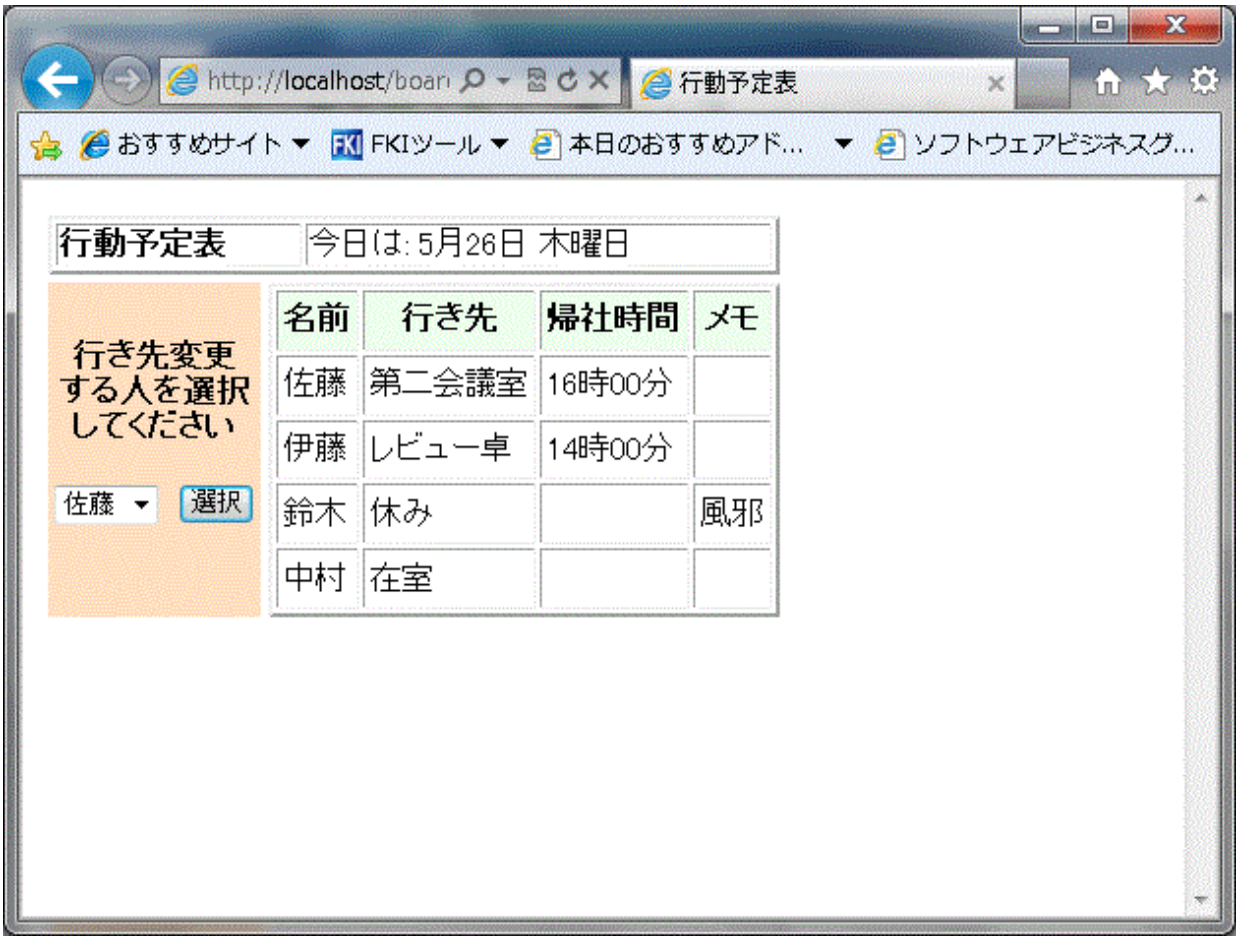
3.1 行動予定表アプリケーションの概要

機能概要説明

ここでは行動予定表アプリケーションの機能概要、画面構成、画面遷移、画面項目および内部構成の説明をします。行動予定表アプリケーションは2つの画面から構成されています。

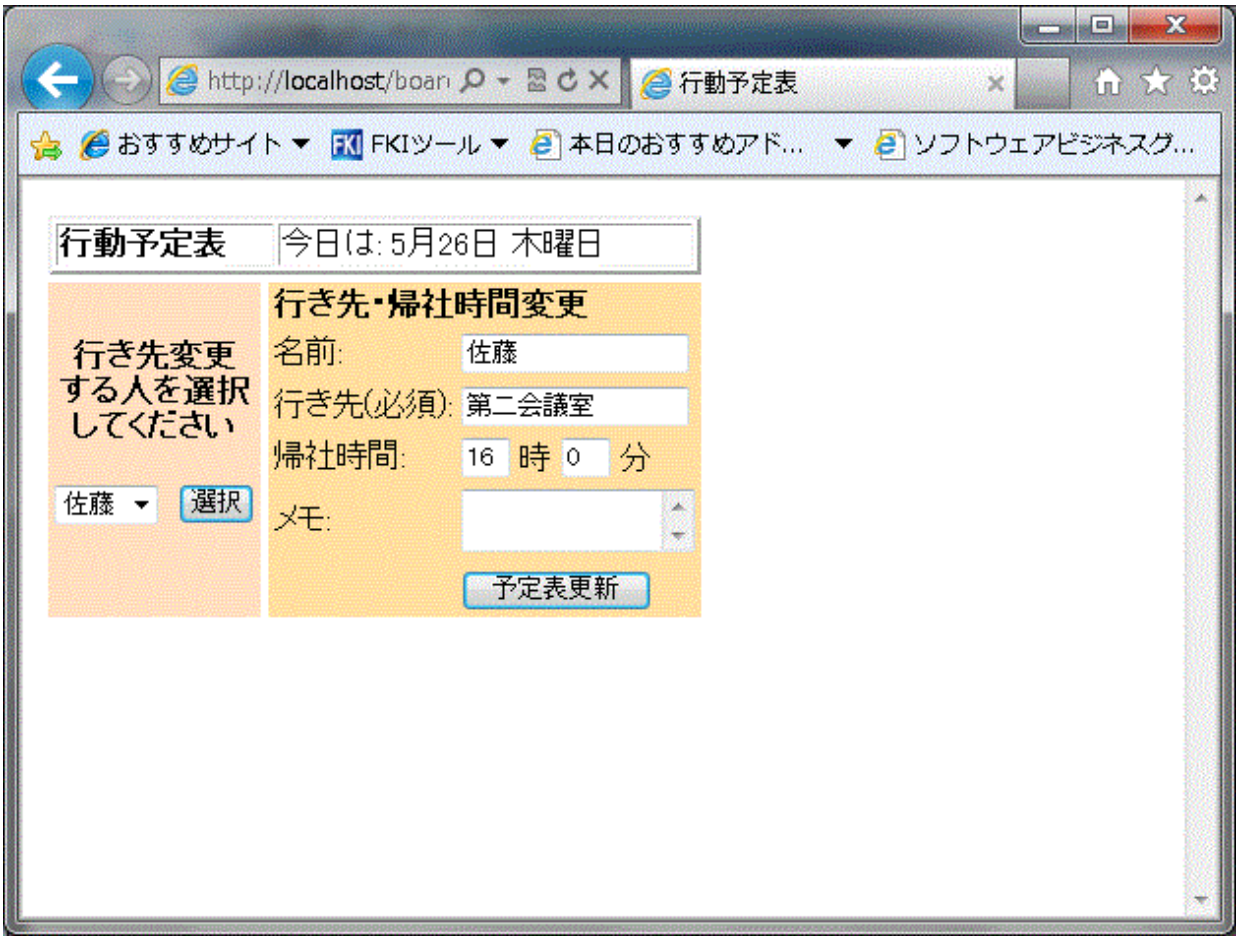
行き先一覧画面

すべての課員の行き先一覧を表示します。



行き先変更画面

課員ごとの行き先の変更を行う画面です。



画面項目

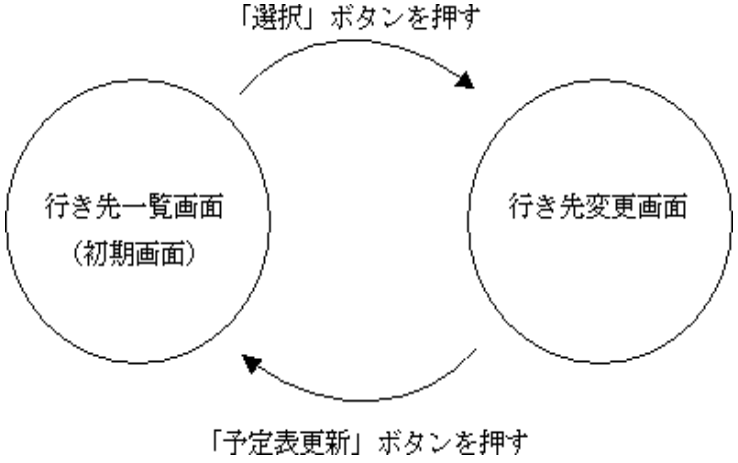
それぞれの画面では、以下に示された項目や部品を表示します。

- 行き先一覧画面
  - 今日の日付
  - 名前選択ドロップダウンリスト
  - 選択ボタン
  - 行動予定一覧(表形式)
- 行き先変更画面
  - 今日の日付
  - 名前選択ドロップダウンリスト
  - 選択ボタン
  - 名前
  - 行き先
  - 帰社時間
  - メモ
  - 予定表更新ボタン

画面遷移

以下に画面の遷移を表します。最初に表示する画面は、行き先一覧表示から始まります。矢印の方向は画面の遷移を表します。矢印の上もしくは下に画面遷移のきっかけとなる動作を記述しています。

図3.1 行動予定表アプリケーションの画面遷移図



画面構成

行き先一覧画面

最初に行き先一覧画面を表示します。行き先一覧画面では、上にタイトルと今日の日付を表示します。タイトルの下では、左側に名前の選択画面、右側に全員の「名前」、「行き先」、「帰社時間」、「メモ」を表形式で表示します。この画面では、文字の入力はできません。行き先情報を変更したい場合は、左側のドロップダウンメニューで、名前を選択してから「選択」ボタンを押すと、行き先変更画面で変更が可能です。

行き先変更画面

行き先一覧画面から名前を選択して選択ボタンを押すことにより、行き先変更画面に移ります。この画面では、選択した人の情報を表示・更新することができます。選択された人の「名前」、「行き先」、「帰社時間」、「メモ」が、各項目の入力フィールドに表示されます。これらの情報を編集する場合は、編集したい項目の上に新しい情報を入力します。入力後は、「予定表更新」ボタンを押すことにより、更新された情報が「行き先一覧画面」に表示されます。(ただし、名前は入力できないようになっています)

アプリケーションサーバとJava統合開発環境の実行画面は、提供する機能の違いにより、お使いのエディションや動作環境によって異なる場合があります。

3.2 行動予定表アプリケーションの設計

3.2.1 構成の設計

ここでは、行動予定表アプリケーションをApcoordinatorアプリケーションとして作成する前に、以下の手順でアプリケーション設計を行います。

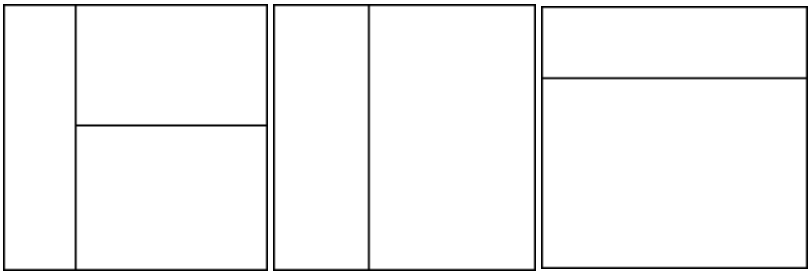
入出力ページ、データBean、およびビジネスクラスと項目の関係は次のとおりです。

- 入出力ページに関する項目
  - 画面構成の決定
  - 画面項目の整理
  - コマンドの整理
- データBeanに関する項目
  - データBeanの設計
  - 画面項目の整理

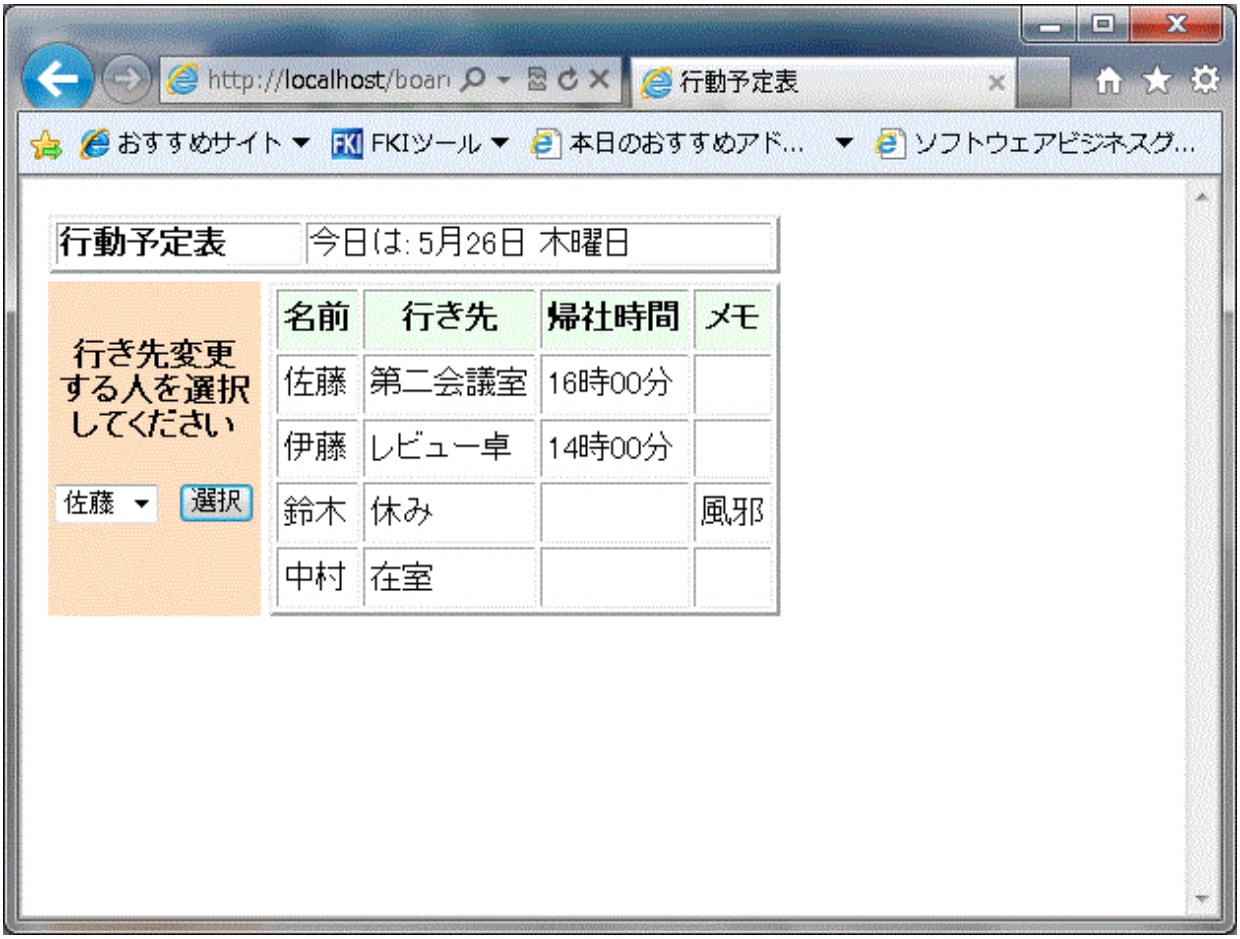
- ビジネスクラスに関する項目
  - ビジネスクラスの設計
  - コマンドの整理

1 画面構成の決定

Apcoordinatorアプリケーションを作成する場合には、まず部品構成を考えます。部品構成とは、画面を分割してどのような部品で構成すればよいかということです。Apcoordinatorアプリケーションでの画面構成は、特定の機能を持つ複数の画面の組み合わせで実現します。例えば、画面分割は下のようなパターンが考えられます。他にもいろいろなパターンが考えられるでしょう。



Apcoordinatorではそれぞれの分割された領域ごとに部品を定義して、HTTPリクエストに対して動的に組み合わせて表示させることが可能です。それでは、行動予定表アプリケーションの画面分割をどのようにしたら良いか考えます。



行動予定表アプリケーションの画面構成を考え、具体的に画面分割を行います。以降の画面要素の決定や分割の方法は、ひとつの例であって、これが絶対的なものではありません。

「行き先一覧画面」で表示する項目は6つあります。

- 「行動予定表」という文字列

- 今日の日付を表す文字列
- 「行き先変更する人を選択してください」という文字列
- 行き先を変更する人を選択するドロップダウンリスト
- 選択するときに押す「選択」ボタン
- 全員の行動予定表

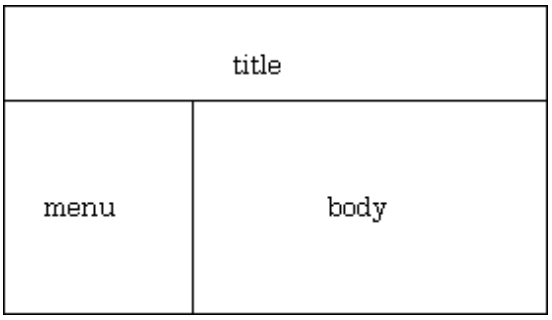
「行動予定表」の文字と今日の日付を画面上部に表示させます。行き先変更する人の選択はひとつの単位として作成します。Apcoordinatorでは繰り返し構造をもつデータをテーブル形式、リスト形式、およびツリー形式で表示できる画面部品があります。そこで、テーブルは一つの画面要素として考えます。表示する画面を分割するときは、分割する画面同士がお互いに関係の深いデータを持たないようにします。

「行き先変更画面」についても表示する項目を考えます。

- 「行動予定表」という文字列
- 今日の日付を表す文字列
- 「行き先変更する人を選択してください」という文字列
- 行き先を変更する人を選択するドロップダウンリスト
- 選択するときに押す「選択」ボタン
- 選択された人の行き先変更画面

「行き先一覧画面」と異なるところは、行き先一覧が、選択された人の行き先を編集する部分に変わった点です。よって、「行き先一覧画面」と同じ画面構成で設計するように考えます。

上の方針に基づき画面構成の設計と分割をします。分割をするときには、分割された領域を示す文字列（領域名）が必要になります。この文字列は任意の文字列で定義します。このアプリケーションでは以下のように作成します。



それぞれの領域名について説明します。

title:

画面の一番上にある「行動予定表」の文字と「今日の日付」が書いてある部分です。

menu:

課員の名前を書いたコンボボックスと選択ボタンを表示する領域です。コンボボックスには現在登録されている課員の名前がすべて出てきます。各課員は、名前を選択したあと「選択」ボタンを押すことによって、今日の行き先、帰社時間、メモを変更することができます。

body:

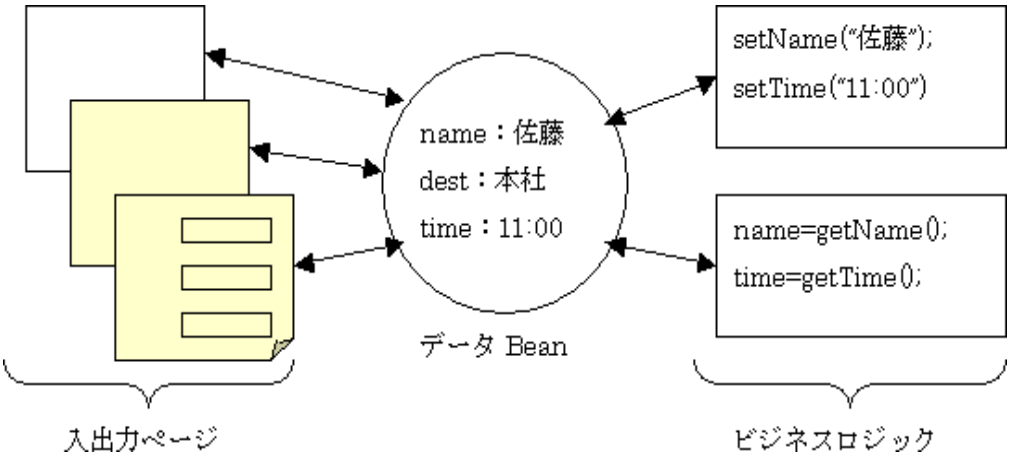
この領域では、以下の2種類の画面を切り替えて表示します。

1. 課員のスケジュールがテーブル形式で表示される画面です。各課員の「名前」、「行き先」、「帰社時間」、「メモ」が表形式で表示されます。
2. 「行き先」、「帰社時間」、「メモ」を入力・更新する画面です。「名前」フィールドはボタンで選択した名前がそのまま使われます（名前は変更できません）。また、入力情報を反映する「予定表更新」ボタンが画面下にあります。

上で決定したそれぞれの領域の配置は、main.jsp内で記述します。

2 データBeanの設計

上のそれぞれの画面領域に対してデータBean、入出力項目、およびアクション項目の整理を行います。データBeanは、入出力ページとビジネスクラス（与えられたデータに対する処理を記述する部分）の間で、データのやりとりを仲介します。入出力ページ、データBean、ビジネスクラスの関係は以下の図のようになります。



アプリケーション内で関係の深いデータを一つのデータBeanでまとめて保存することにより、それらのデータの扱いが容易になります。画面構成のみ変えたい場合は入出力ページのHTMLの記述を変えるだけでよく、データBeanには手を加える必要はありません。また、ビジネスクラスの変更も、データBeanの項目が変わらない限りは、入出力ページ側に手が入ることもありません。

まず、どのような種類のデータBeanを用意するかを考えます。データBeanを作成するときには以下のことに注意します。

- 画面分割に従って、Beanを作成する。
- 関連の深い項目は一つのBeanにする。

"1 画面構成の決定"で定義した各領域で表示するデータを考えます。

title:行動予定表という文字列と今日の日付

上記の今日の日付データの内容は、行動予定表のデータとは独立して表示されるので、タイトル部分を表示するデータBeanとして作成します。変更のあるデータは「今日の日付」のみなので、一つのプロパティ名（データBeanが保存するデータ名につける名前）として用意します。

menu: 選択可能課員一覧と「選択」ボタン

選択可能な課員一覧は、登録されている課員の名前から作成します。課員一覧の要素は行動予定一覧にも含まれているので、先に説明したbody領域で一覧を表示するものと同じデータBeanでもかまわないのですが、この例では別のデータBeanとして定義します。

body: 行動予定一覧と各課員の項目一覧

ここでは2つの異なる表示形式が入ります。

a. 行動予定一覧

b. 各課員の項目一覧

a,b両方とも、「名前」「行き先」「帰社時間」「メモ」と同じ内容を表示しています。ただし、今回は行動予定の一覧には、コンポーネントタグのモデルクラスを使用し、各課員の項目一覧には使用しません。したがって、ここでは異なるデータBeanを定義します。

もし、同じデータBeanを使用して異なる表示を行う場合は「表示モード」を使用してください。「表示モード」とは、同じ表示領域に表示する場合に、どちらの画面を表示させるか識別するための名前です。

上記からデータBeanの扱うデータの種類と名前を決めます。

- TitleBean（タイトル情報用Bean）
- ScheduleBean（課員行動予定表用Bean）
- ProfileBean（課員選択一覧用Bean）
- UserScheduleBean（各課員の項目一覧用Bean）

また、それぞれの部品の入出力ページとデータBeanの関係をまとめます。入出力ページ名は表示内容や保持するデータが容易に想像できる名前をつけます。入出力ページ名はすべて小文字にします。

入出力ページ名	データBean名	ページで表示する内容の説明
title.jsp	TitleBean	タイトルと今日の日付の表示
usermenu.jsp	ProfileBean	課員選択画面の表示
table.jsp	ScheduleBean	行動予定表一覧の表示（表形式）
userschedule.jsp	UserScheduleBean	課員別行動予定の変更画面の表示

上記4つのデータBeanに関しては、各データ項目に対するメソッドを実装します。データBean作成の際には保存されるデータの項目の組み合わせについて、十分に吟味する必要があります。

次に各データBean内のプロパティ設計を行います。「プロパティ」とは、データBeanが扱う各項目のデータ名に相当するものです。入出力ページは、関連するデータBeanからこのプロパティ名を使い、各項目の値を取得します。各データBeanで扱うデータについては、以下のような種類があります。

データBean名	扱うデータ項目名
TitleBean	「今日の日付」
ProfileBean	「課員名一覧」「選択された課員名」
ScheduleBean	「スケジュール一覧」
UserScheduleBean	「名前」「行き先」「帰社時間」「メモ」

上記データ項目ごとにプロパティ名を決めます。

「プロパティ名」は入出力ページがデータBeanから特定のデータを取り出すときの名前として使われます。ここでは以下のようにプロパティ名を定義します。プロパティ名は、項目から連想しやすい名前にします。

「データ型」は、実際にデータBeanの中に保存されるデータ型です。データ型の中のComboBoxやTableViewは、Apcoordinatorでコンボボックス形式のデータや表形式のデータを表示するための特別なクラス名です。ここではコンボボックス形式や表形式のデータを表示する際に使われるものであると考えてください。

データ項目名	プロパティ名	データ型
今日の日付	today	String
課員名一覧	username	ComboBox
スケジュール一覧	scheduleTable	TableView
名前	name	FieldString
行き先	destination	FieldString
帰社時間（時）	hour	FieldLong
帰社時間（分）	minute	FieldLong
メモ	memo	FieldTextArea

注：日付はDate型ではなく、String型（文字列）で渡してください。理由は「行動予定表」では日付の形式をアプリケーション自身で定義しているからです。

3 画面項目の整理

画面入出力項目に関しての整理を行います。各入出力ページの各入出力項目（フォームまたはApcoordinatorでサポートしているGUI部品）について以下の事柄について一覧を作成します。

- プロパティ名
- データBean
- I/O

- データ型
- データの意味

「プロパティ名」は、“2 データBeanの設計”で定義したものを使用します。入出力ページは、関連するデータBeanからこのプロパティ名を使い、各項目の値を取得します。

「データBean名」は各入出力ページに対応するデータBean名を表します。

「I/O」は、ページの各項目について、入力のみであれば"I"、出力(表示)のみであれば"O"、入力出力の両方であれば"I/O"となります。

「データ型」はプロパティ名に対応するデータ型を記述します。

「データの意味」はプロパティ名に対応するデータの意味です。

title.jsp

プロパティ名	データBean名	I/O	データ型	データの意味
today	TitleBean	O	String	今日の日付

usermenu.jsp

プロパティ名	データBean名	I/O	データ型	データの意味
username	ProfileBean	I/O	ComboBox	課員一覧

table.jsp

プロパティ名	データBean名	I/O	データ型	データの意味
scheduleTable	ScheduleBean	O	TableView	スケジュール一覧

userschedule.jsp

プロパティ名	データBean名	I/O	データ型	データの意味
name	UserScheduleBean	O	FieldString	名前
destination	UserScheduleBean	I/O	FieldString	行き先
hour	UserScheduleBean	I/O	FieldLong	帰社時間(時)
minute	UserScheduleBean	I/O	FieldLong	帰社時間(分)
memo	UserScheduleBean	I/O	FieldTextArea	メモ

4 ビジネスクラスの設計

ビジネスクラスは、業務単位で作成します。例をあげると、「見積」という業務を考え、その単位で一つのビジネスクラスに割り当てます。「見積表示」、「見積依頼」などの同系統のデータを扱うときは、一つのビジネスクラスで処理します。ビジネスクラスで保持するものとして、一覧表のデータを一つ保持する必要があります。よって、この一覧データを扱うクラスのインスタンスは、クラス内に静的変数として定義します。ビジネスクラス内では、入出力ページにあるボタンを押されたときに行う処理を記述する必要があります。行動予定表アプリケーションでは、2つの場合があります。

- 課員選択後に「選択」ボタンを押したあとの処理
- 課員が予定の変更を反映するときに「予定表更新」ボタンを押したあとの処理

また、最初の画面を表示するときに、その前処理(初期化)も必要になります。

したがって、あわせて3つの処理を実装する必要があります。3つの処理はそれぞれ、ビジネスクラス内の別のメソッド内で定義します。ここでは、メソッド名とその処理の内容を以下のように定義します。

メソッド名	処理の内容
showUserSchedule	このメソッド内では、行動予定を変更する課員が選択されたあとの処理を行います。
updateUserSchedule	このメソッド内では、変更した課員の行動予定を更新する時の処理を行います。
startup	このメソッド内ではデータの初期化(ファイルから一覧データの読み込み)を行います。

5 コマンドの整理

次に、コマンドについての整理を行います。コマンドとは、ボタンを押した時に呼ばれるメソッド名の一覧を記述します。ボタンを押した以外にも、初期画面を表示する時にも用いられます。

ボタンを表示する入出力ページに対して以下の事柄について一覧を作成します。

- コマンド
- ビジネスクラス
- メソッド
- コマンドの意味

「コマンド」は、ボタンを押したときに実行されるコマンドを表します。

「ビジネスクラス」は、実際のデータ処理を行うクラスです。この例では一つのビジネスクラスを使います。

「メソッド」は対応するコマンドが実行されたときに、ビジネスクラスの中で実行されるメソッド名です。

「コマンドの意味」は実行されるコマンドの意味を表します。ただし、一番最初の画面の表示の際には、コマンドはどこからも与えられないので、空白になっています。

usermenu.jsp

コマンド	ビジネスクラス	メソッド	コマンドの意味
selectuser	BoardHandler	showUserSchedule	選択課員の予定表示

userschedule.jsp

コマンド	ビジネスクラス	メソッド	コマンドの意味
update	BoardHandler	updateUserSchedule	予定表更新

最初の画面表示時

コマンド	ビジネスクラス	メソッド	コマンドの意味
	BoardHandler	startup	初期画面表示

ビジネスクラスBoardHandler内の3つのメソッド(startup,showUserSchedule,updateUserSchedule)についての処理詳細は“3.3 行動予定表アプリケーションの作成”の“3.3.6 ビジネスロジックの追加”を参照してください。

3.2.2 行動予定表アプリケーション詳細

ここでは、行動予定表アプリケーションの実装の詳細に関する説明を行っています。

1 表やコンボボックス形式のデータを扱うクラスの作成

行動予定表アプリケーションでは、行動予定一覧を表示していますが、Apcoordinatorでは表形式、コンボボックス形式、ツリー形式などのデータを表示するための仕組みを用意しています。これらをコンポーネントタグと呼んでいます。コンポーネントタグを使用する手順は以下のとおりです。

1. コンポーネントの表示に必要なクラスを作成します。
2. データBeanに上記クラスのインスタンスを設定(set)および取得(get)できるように、内部変数とメソッドの実装をします。
3. 入出力ページ側にApcoordinatorでサポートしている"uji:comboBox"や"uji:tableView"タグを実装します。

ここでは、ComboBox、TableView、FieldString、FieldLong、FieldTextAreaクラスを使用します。これらのクラスは、それぞれApcoordinatorでサポートしている項目クラスのインタフェースをインプリメントしています。

コンポーネントタグを使用することにより以下の利点があります。

- 定型的な表現形式(表やツリー)を簡単に表示することができます。また、色や大きさなどの外観も簡単に変更できます。
- 入力データ値のチェックが簡単に行えます。また、数値項目のフォーマットが簡単に指定できます。

・ 表の中である特定の列や行、または特定のセルにしるしをつけ、入出力ページ側で色を変えるなどの異なる表示が可能になります。

この行動予定表アプリケーションにおいては、次の目的でコンポーネントタグを使用しています。

項目クラス名	使用する利点
ComboBox	コンボボックス形式の表現を容易に実現するためです。
TableView	表形式のデータを容易に容易に実現するためです。
FieldString	文字列入力フィールドを容易に実現するためです。
FieldLong	数値入力フィールドを容易に実現し、値の下限、上限を設定するためです。
FieldTextArea	複数行テキスト入力フィールドを容易に実現するためです。

2 行動予定データを保持するクラスの設計

行動予定表アプリケーションでは課員の行動予定を格納するクラスを考えます。以下の2つのクラスを用います。

・ DataManager

・ UserRecord

DataManagerは、課員情報を管理するクラスです。内部のVector変数で一つ以上のUserRecordクラス（課員一人の予定データを格納するクラス）のインスタンスを保持します。任意のUserRecordのインスタンス書き換えが可能です。

UserRecordは、「名前」、「行き先」、「帰社時間」、「メモ」の一組のデータを保持するクラスです。すべての項目の書き換えが可能です。

DataManagerクラスが各レコード(UserRecord)を要素とするVector型で管理しています。データ保持の主体はUserRecordです。ビジネスクラスでは、全課員の行動予定データの参照・更新はDataManagerに対して行います。

3 行動予定表データの復元と保存

行動予定表アプリケーションは、初期化時に、ファイルからシリアライズ（オブジェクトをストリームに変換する動作のこと）された行動予定表データを読み込み、復元を行います。行動予定表アプリケーションは、「予定表更新」ボタンを押すごとにDataManagerのインスタンスをシリアライズしてファイルに保存します。

この動作により、常に最新の予定が保存されます。上記の処理は、以下のタイミングで行われます。

処理内容	処理が行われるタイミング
行動予定表データの読み込み	ビジネスクラスのstartupメソッドが呼ばれたとき
行動予定表データの保存	ビジネスクラスのupdateUserScheduleメソッドが呼ばれたとき

4 データBeanとScheduleModel, UserModelクラスの関係

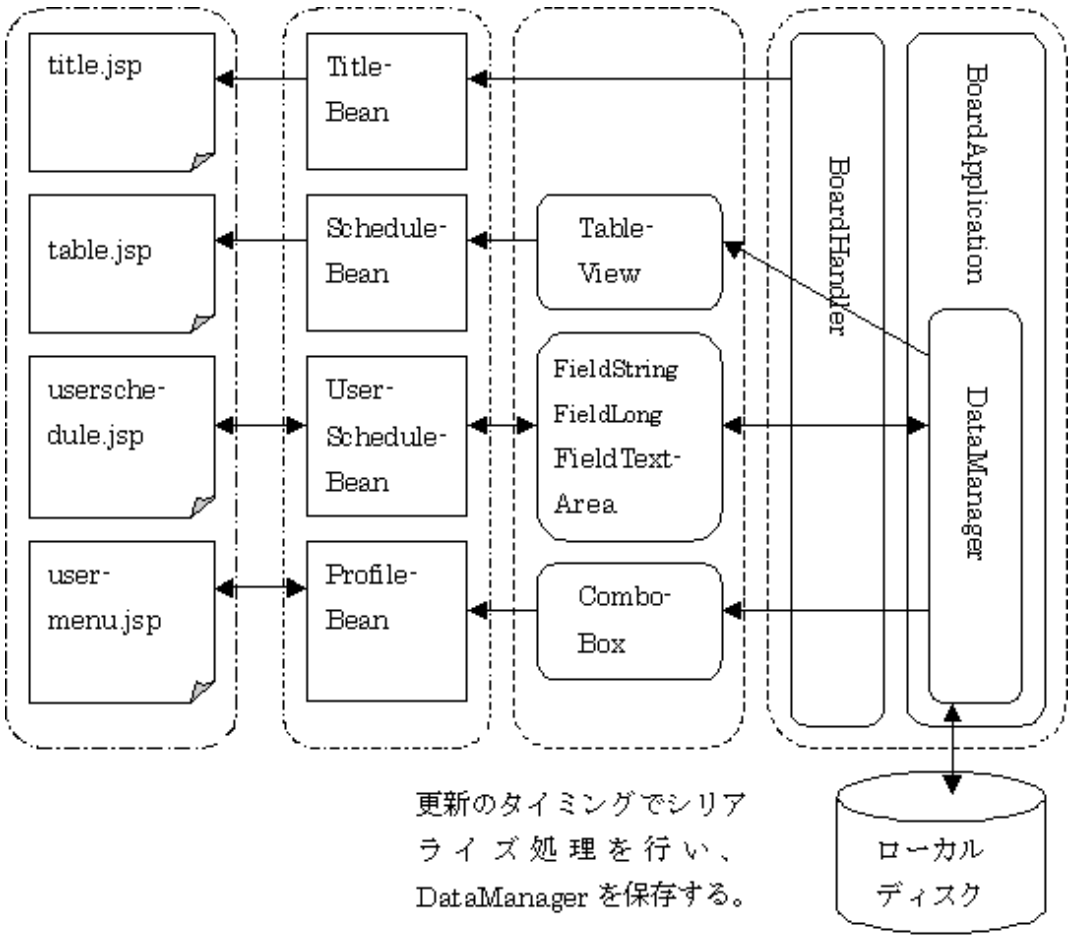
データBeanは入出力ページが一覧を表示できるように、ScheduleModelとUserModelを直接入出力ページに渡します。データBeanはこれらクラスをset,getするメソッドを実装する必要があります。データBeanとそれぞれの扱うクラスの関係は以下のとおりです。

データBean	扱うクラス	クラスの説明
ScheduleBean	TableView	行動予定一覧を表示するためのクラス
UserScheduleBean	FieldString,FieldLong,FieldTextArea	行動予定一覧を表示するためのクラス
ProfileBean	ComboBox	課員選択を表示するためのクラス

上記表に基づきデータBeanに対応するクラスの入出力メソッドを実装します。実装方法については、“3.3.4 入出力ページの作成”を参照してください。

5 内部構造の説明

ここでは簡単にデータの流れとデータ形式の説明します。各オブジェクトの関係図は以下のとおりです。矢印の向きはデータの流れをあらわします。



シリアライズはビジネスクラスが保持するDataManagerのインスタンスについて、更新のたびに行われます。以下は上記の図に出てくる各コンポーネントの説明です。

入出力ページの機能説明

入出力ページ名	説明
title.jsp	画面の一番上のタイトルを表示する入出力ページ
usermenu.jsp	課員一覧を表示して選択する入出力ページ
table.jsp	行動予定一覧を表示する入出力ページ
userschedule.jsp	課員ごとの行動予定を表示・編集できる入出力ページ

データBeanの機能説明

データBean名	説明
TitleBean	タイトルに表示する今日の日付を保存するBean
ProfileBean	課員一覧情報を保存するBean
ScheduleBean	行動予定表情報を保存するBean
UserScheduleBean	各課員の項目情報を保存するBean

データの流れは、矢印で表しています。矢印の向きはデータの流れる方向を表しています。両方向の矢印は、データの流れが双方向であることを示し、単方向の矢印は、入力か出力かのどちらかを示しています。

ビジネスクラスである、BoardHandler内では、startup, showUserSchedule,updateUserScheduleの各メソッドが呼ばれるたびに、BoardApplication内部に格納しているDataManagerを用いてデータBeanに値を設定しています。データBeanと入出力ページ間のデータのやり取りは、ページの遷移とともに自動的に行われます。

3.3 行動予定表アプリケーションの作成

ここでは、「行動予定表アプリケーション」の開発過程を紹介しながら説明します。

3.3.1 Apcoordinatorアプリケーションのプロジェクト作成

設計を行い、仕様がある程度決定した段階で、プロジェクトを作成します。以下の手順で行います。

1. [ファイル] > [新規]メニューの[プロジェクト]を選択します。
2. [新規プロジェクト]画面が表示されます。
3. [Apcoordinator] > [Webアプリケーションプロジェクト(Apcoordinator)]を選択し、[次へ]ボタンをクリックします。
4. [動的Webプロジェクト]画面では、以下を設定します。

設定項目	設定内容	デフォルト設定値
プロジェクト名	"board"を指定します。	(なし)
ターゲットランタイム	[Interstage Application Server V11.1 IJServer Cluster (Java EE)]を選択します。	[なし]が選択
動的 Web モジュール バージョン	[2.5]を選択します。	[2.5]が選択
構成	[Interstage Application Server V11.1 IJServer Cluster (Java EE)デフォルト構成]を選択します。「ターゲットランタイム」を選択することで自動的に選択されます。	[Apcoordinator Webプロジェクトのデフォルト構成]が選択
EARにプロジェクトを追加	チェックしません。	チェックマークなし

入力完了後、[次へ]ボタンをクリックします。

5. [Webモジュール]画面では、以下を設定します。

設定項目	設定内容	デフォルト設定値
コンテキストルート	"board"を指定します。	プロジェクト名
コンテンツフォルダ	"WebContent"を指定します。	"WebContent"
Javaソースフォルダ	"src"を指定します。	"src"
Deployment Descriptrotの生成	チェックします。	チェックマーク付

入力完了後、[次へ]ボタンをクリックします。

6. [テンプレートを選択]画面では、[標準Apcoordinatorアプリケーション]を選択し、[次へ]ボタンをクリックします。
7. [制御ページ情報]画面では、以下を設定します。

設定項目	設定内容	デフォルト設定値
ファイル名	"main"を指定します。 制御ページに指定するファイル名を拡張子なしで指定します。	main
タイトル	"行動予定表"を指定します。 ここに入力した文字列は、ブラウザのタイトルバーに表示される名前となります。	プロジェクト名
エラーページを使用する	チェックします。 アプリケーションの実行途中にエラーが生じたときに、表示するページです。使用するときはチェックします。	チェックマーク付

設定項目	設定内容	デフォルト設定値
エラーページファイル名	"boardError"を指定します。 例外が発生した時に、エラー出力を表示するページ名の名前を定義します。	プロジェクト名＋Error

それぞれの項目を入力したあと、[次へ]ボタンを押します。

8. 次のステップでは、ファクトリの拡張情報を定義します。

アプリケーションクラスはアプリケーション単位（アプリケーションが動いている間）での処理を拡張するときに用いられ、セションクラスはセション単位（セションが行われている間）での処理を拡張するときに用いられます。

ファクトリクラスは、アプリケーションクラスおよびセションクラスの拡張を行った際に、拡張する必要があります。これら3つのクラスはユーザ固有の処理を記述するときに用いられます。

3つのクラスの詳細はApcoordinatorのマニュアルを参照してください。行動予定表アプリケーションでは、「ファクトリクラス」と「アプリケーションクラス」の拡張を行います。

設定する項目は以下のとおりです。

設定項目	設定内容	デフォルト設定値
ファクトリクラスを拡張する	チェックします。 拡張するときはチェックします。	チェックマーク付
パッケージ	"board"を指定します。ファクトリクラスのパッケージ名を指定します。	プロジェクト名
ファクトリクラス	"BoardFactory"を指定します。アプリケーション独自の拡張ファクトリクラスを作成するときのクラス名です。	プロジェクト名+Factory
アプリケーションクラスを拡張する	チェックします。 拡張するときはチェックします。	チェックマーク付
DB連携機能を使用する	チェックしません。 データベース連携機能を使用するときはチェックします。	チェックマークなし
アプリケーションクラス	"BoardApplication"を指定します。 アプリケーション独自の拡張アプリケーションクラスを作成するときのクラス名です。	プロジェクト名 +Application
セションクラスを拡張する	チェックしません。 拡張するときはチェックします。	チェックマーク付
セション管理機能を使用する	"セションクラスを拡張する"のチェックを外すことで、選択できなくなります。	チェックマーク付
セションクラス	"BoardSession"を指定します。 アプリケーション独自の拡張セションクラスを作成するときのクラス名です。	プロジェクト名+Session

9. [完了]ボタンをクリックします。

Apcoordinatorのパースペクティブを開いていない場合は、確認のダイアログが表示されます。

[はい]ボタンをクリックし、Apcoordinatorのパースペクティブを開きます。

以上でひな型となるプロジェクト「board」ができました。以降は、このプロジェクトにファイルを追加していきます。

3.3.2 データBeanの作成

画面を構成する入出力ページにあわせて以下のデータBeanを作成します。これらは直接入出力ページと関連付けがされます。データBean作成においては、データBean生成ウィザードを用います。この項では以下の4つのデータBeanを作成します。

- TitleBean
- ScheduleBean

- UserScheduleBean
- ProfileBean

最初にScheduleBeanを例にとり、以下の手順で作成します。

Interstage Studioのメイン画面で、[ファイル]メニューの[新規] > [データBean(Apcoordinator)]を選択します。

[データBeanクラス]画面で設定する項目は以下のとおりです。

設定項目	設定内容
プロジェクト名	"board"（必須・デフォルトで設定されています）
参照ファイル名	設定しません
パッケージ名	"board"データBeanのパッケージ名
クラス名	"ScheduleBean"データBeanになるクラス名（必須）
項目名	入出力ページが表示に用いる項目名(入力フィールドの横でラベルとなる)
プロパティ名	データBeanおよび入出力ページがデータ入出力に使用するプロパティ名（必須）
型	実際に入出力を行うデータ型（必須）ユーザ定義型も扱うことができます

次にScheduleBeanに関するプロパティ名一覧を作成します。

プロパティ名	データBean名	I/O	データ型	データの意味
scheduleTable	ScheduleBean	O	TableView	スケジュール一覧

上表に基づきScheduleBeanの項目の定義を行います。

右側にある[追加]ボタンを押すことにより、項目名、プロパティ名、型のそれぞれを追加します。項目名に関しては、あとで入出力ページのひな型作成時に、ここで入力した名前がそのまま項目になります。項目名は、入出力ページの項目名にそのまま使われますが、あとで入出力ページを編集して修正できます。

[プロパティの追加]ダイアログで設定する項目は以下のとおりです。

項目	入力する文字列
項目名	スケジュール一覧
プロパティ名	scheduleTable（最初の文字は小文字で始まること）
型	TableView
新しいインスタンスを作る	チェックしてください。（デフォルトではチェック状態です）

プロパティ名を入力したあとに[追加]を押すとプロパティが追加されます。

[プロパティの追加]ダイアログの[閉じる]を押します。

[データBeanクラス]画面の[完了]ボタンを押してひな型を作成します。Interstage Studioの左側に、"ScheduleBean.java"が追加されていることを確認してください。

上記を残り3つのBean（TitleBean, ProfileBean, UserScheduleBean）に対して行い、データBeanを作成します。

設定する項目は以下のとおりです。プロパティの項目名には、以下の表の「データの意味」欄の内容を指定してください。

プロパティ名	データBean名	I/O	データ型	データの意味
today	TitleBean	O	String	今日の日付
username	ProfileBean	I/O	ComboBox	課員一覧
name	UserScheduleBean	O	FieldString	課員名
destination	UserScheduleBean	I/O	FieldString	行き先
hour	UserScheduleBean	I/O	FieldLong	帰社時間（時）

プロパティ名	データBean名	I/O	データ型	データの意味
minute	UserScheduleBean	I/O	FieldLong	帰社時間(分)
memo	UserScheduleBean	I/O	FieldTextArea	メモ

各データBeanのソースコードは以下のようになります。赤字はひな型作成後に追加した部分です。

・ TitleBean.java

```
/*
 * TitleBean.java
 *
 * Copyright(c)
 */

package board;

import java.text.SimpleDateFormat;
import java.util.Date;

import com.fujitsu.uji.DataBean;

/**
 * データBeanクラスです。
 *
 * @author
 * @version
 */
public class TitleBean extends DataBean {
          //{{UJIWIZ_GENERATE(DATADEF)
          //@@UJI-Wizard Information(Bean)
          //@@今日の日付,today,String
          //}}UJIWIZ_GENERATE

          //内部変数
          //{{UJIWIZ_GENERATE(DATA)
          /** todayの変数です。 */
          protected String today;
          private static SimpleDateFormat df = new SimpleDateFormat("M月d日 E曜日");
          //}}UJIWIZ_GENERATE

          //
          //{{UJIWIZ_GENERATE(GETSET)
          /**
           * todayを取得します。
           * @return            today
           */
          public String getToday() {
                    //return today;
                    synchronized(df) {
                              return df.format(new Date());
                    }
          }
          /**
           * todayを設定します。
           * @param todya       today
           */
          public void setToday(String today) {
                    //this.today = today;
                    throw new UnsupportedOperationException();
          }

          //}}UJIWIZ_GENERATE
```

```
}
```

- ScheduleBean.java

```
/*
 * ScheduleBean.java
 *
 * Copyright(c)
 */

package board;

import com.fujitsu.uji.DataBean;
import com.fujitsu.uji.compo.*;

/**
 * データBeanクラスです。
 *
 * @author
 * @version
 */
public class ScheduleBean extends DataBean
{
	//{{UJIWIZ_GENERATE(DATADEF)
	//@@UJI-Wizard Information(Bean)
	//@@スケジュール一覧,scheduleTable,TableView
	//}}UJIWIZ_GENERATE

	//
	//{{UJIWIZ_GENERATE(DATA)
	/** scheduleTableの変数です。*/
	protected TableView scheduleTable = new TableView();
	//}}UJIWIZ_GENERATE

	// コンストラクタ
	public ScheduleBean(DataManager dm){
		update(dm);
	}

	public void setScheduleTable(TableView scheduleTable){
		this.scheduleTable = scheduleTable;
	}

	//
	//{{UJIWIZ_GENERATE(GETSET)
	/**
	 * scheduleTableを取得します。
	 * @return		scheduleTable
	 */
	public TableView getScheduleTable() {
		return scheduleTable;
	}

	//}}UJIWIZ_GENERATE

	public void update(DataManager dm){
		scheduleTable = new TableView();
		scheduleTable.setHeader(new String[]{"名前","行き先","帰社時間","メモ"});
		for(int i=0; i < dm.getRecordCount() ; i++){
			scheduleTable.setValueAt(dm.getRecord(i).getName(),i,0);
			scheduleTable.setValueAt(dm.getRecord(i).getDestination(),i,1);
			String time = getTime(dm.getRecord(i));
```

```
                        if(time == null){
                                scheduleTable.setValueAt("",i,2);
                        } else {
                                scheduleTable.setValueAt(time.toString(),i,2);
                        }
                        scheduleTable.setValueAt(dm.getRecord(i).getMemo(),i,3);
                }
        }

        private String getTime(UserRecord ur){
                StringBuffer buf = new StringBuffer();
                if(ur.getHour().equals("")){
                        return null;
                }
                buf.append(ur.getHour());
                buf.append("時");
                if(ur.getMinute().equals("") || ur.getMinute().equals("0")){
                        buf.append("00");
                } else {
                        buf.append(ur.getMinute());
                }
                buf.append("分");
                return buf.toString();
        }
}
```

- ProfileBean.java

```
/*
 * ProfileBean.java
 *
 * Copyright(c)
 */

package board;

import com.fujitsu.uji.DataBean;
import com.fujitsu.uji.compo.*;

public class ProfileBean extends DataBean
{
        //{{UJIWIZ_GENERATE(DATADEF)
        //@@UJI-Wizard Information(Bean)
        //@@課員一覧,username,ComboBox
        //}}UJIWIZ_GENERATE

        //
        //{{UJIWIZ_GENERATE(DATA)
        protected ComboBox username = new ComboBox();
        //}}UJIWIZ_GENERATE

        //
        //{{UJIWIZ_GENERATE(GETSET)
        public ComboBox getUsername() {
                return username;
        }

        //}}UJIWIZ_GENERATE

        // コンストラクタ
        public ProfileBean(DataManager dm){
                int userCount = dm.getRecordCount();
                String[] users = new String[userCount];
```

```
                for(int i=0; i < userCount; i++){
                        users[i] = dm.getRecord(i).getName();
                }
                username = new ComboBox(users);
        }

        public void setUsername(ComboBox username){
                this.username = username;
        }
}
```

・ UserScheduleBean.java（変更なし）

```
/*
 * UserScheduleBean.java
 *
 * Copyright(c)
 */

package board;

import com.fujitsu.uji.DataBean;
import com.fujitsu.uji.compo.*;

/**
 * データBeanクラスです。
 *
 * @author
 * @version
 */
public class UserScheduleBean extends DataBean
{
   //{{UJIWIZ_GENERATE(DATADEF)
   //@@UJI-Wizard Information(Bean)
   //@@課員名,name,FieldString
   //@@行き先,destination,FieldString
   //@@帰社時間（時）,hour,FieldLong
   //@@帰社時間（分）,minute,FieldLong
   //@@メモ,memo,FieldTextArea
   //}}UJIWIZ_GENERATE

   //
   //{{UJIWIZ_GENERATE(DATA)
  /** nameの変数です。 */
   protected FieldString name = new FieldString();
  /** destinationの変数です。 */
   protected FieldString destination = new FieldString();
  /** hourの変数です。 */
   protected FieldLong hour = new FieldLong();
  /** minuteの変数です。 */
   protected FieldLong minute = new FieldLong();
  /** memoの変数です。 */
   protected FieldTextArea memo = new FieldTextArea();
   //}}UJIWIZ_GENERATE

   //
   //{{UJIWIZ_GENERATE(GETSET)
  /**
   * nameを取得します。
   * @return           name
   */
   public FieldString getName() {
      return name;
```

```
    }

    /**
     * destinationを取得します。
     * @return           destination
     */
    public FieldString getDestination() {
        return destination;
    }

    /**
     * hourを取得します。
     * @return           hour
     */
    public FieldLong getHour() {
        return hour;
    }

    /**
     * minuteを取得します。
     * @return           minute
     */
    public FieldLong getMinute() {
        return minute;
    }

    /**
     * memoを取得します。
     * @return           memo
     */
    public FieldTextArea getMemo() {
        return memo;
    }

    //}}UJIWIZ_GENERATE
}
```

3.3.3 ユーザ定義型とデータ保存クラス作成

内部のデータ形式にあわせて以下のクラスを作成します。

- DataManager(行動予定表データを保持するクラス)
- UserRecord(1レコード情報を保持するクラス)

上記2つのクラスの雛型生成は、クラス生成ウィザードを用いて行います。以下のように行います。

最初にユーザ定義型"DataManager"を作成します。この"DataManager"では、シリアライズを使用するので、java.io.Serializableをインプリメントする必要があります。

Interstage Studioのメイン画面で、[ファイル]メニューの[新規] > [クラス]を選択します。[新規]画面が表示されます。

[Javaクラス]画面で設定する項目は以下のとおりです。(インターフェース追加についての説明はこの下です)

項目	説明
ソースフォルダ	board/src(デフォルト値)
パッケージ名	board(パッケージ名を入力する)
エンクロージング型	チェックなし(デフォルト値) 入れ子のクラスにするどうか(当サンプルではチェックは入れません)
名前	DataManager(作成するクラス名を入力する)
修飾子	publicにチェックをいれる(デフォルト値)

項目	説明
スーパークラス	java.lang.Object (デフォルト値)
インターフェース	インプリメントするインターフェースを追加する Serializableを一覧より選択する
作成するメソッドスタブの選択	継承された抽象メソッドにチェックをいれる(デフォルト値) 各項目にチェックを入れるとそれに沿ったソースコードが生成されます
コメントの生成	チェックなし(デフォルト値)

インプリメントするインタフェースを追加するため、[追加]ボタンを押します。[実装されたインタフェースの選択]画面が表示されます。

画面が表示されたら入力フィールドに、"Serializable"と入力してください。

インプリメントするインタフェースを入力するか[一致する項目]より選択します。

クラス名(実際はインタフェース名)が確定したら、[OK]を押します。

[完了]ボタンをクリックします。Javaソースが生成されます。

生成されたソース"DataManager.java"を、Interstage Studioで編集を行い、赤字の個所を追加します。

- DataManager.java

```
package board;
import java.io.Serializable;
import java.util.Vector;
import com.fujitsu.uji.compo.ComboBox;

public class DataManager implements Serializable {

        Vector data; //課員情報（レコード）を要素とするVectorで保存

        public DataManager() {
                // 以下に初期化する手続きを記述してください。
                prepareData();
        }

        //課員レコード数の取得（課員数）
        public int getRecordCount() {
                return data.size();
        }

        //指定した課員の情報を取得する
        public UserRecord getRecord(int index) {
                return (UserRecord)data.elementAt(index);
        }

        //データが保存されたファイルが存在しないときに、
        //内部のstatic変数で一覧表データを初期化するメソッド
        private void prepareData() {
                data = new Vector();
                for(int i = 0; i < initialData.length; i++) {
                        UserRecord bean = new UserRecord(initialData[i]);
                        data.addElement(bean);
                }
        }

        //課員名をキーとしてdm内を検索して、新しいデータで更新する
        public void update(UserRecord record){
                String key = record.getName();
                for(int i=0; i < getRecordCount(); i++ ){
                        String name = getRecord(i).getName();
                        if(name.equals(key)){
                                data.set(i, record);
```

```
                        }
                }
        }

        //課員名をキーとしてdm内を検索してUserScheduleBeanにデータを設定する
        public UserScheduleBean obtainUserSchedule(ProfileBean dataBean){
                //課員名の取得
                ComboBox username = dataBean.getUsername();
                int selected = username.getSelectedIndex();
                String user = username.getTextAt(selected);
                //空のBeanを作成して、データをセット
                UserScheduleBean userScheBean = new UserScheduleBean();
                for(int i=0; i < getRecordCount();i++ ){
                      UserRecord curRecord = getRecord(i);
                      if(curRecord.getName().equals(user)){
                            userScheBean.getName().setText(curRecord.getName());
                            userScheBean.getDestination().setText(curRecord.getDestination());
                            userScheBean.getHour().setText(curRecord.getHour());
                            userScheBean.getMinute().setText(curRecord.getMinute());
                            userScheBean.getMemo().setText(curRecord.getMemo());
                      }
                }
                return userScheBean;
        }

        //一覧表データファイルが存在しないときの初期化用データ
        private static final String initialData[][] = {
                { "佐藤","第二会議室","16","00",""},
                { "伊藤","レビュー卓","14","00",""},
                { "鈴木","休み","","","風邪"},
                { "中村","在室","","",""},
        };
}
```

UserRecord.javaに関しては、下記のコードをそのまま利用してください。DataManagerと同様にして空のクラスを作成後に、コピーアンドペーストで貼りつけてください。

- UserRecord.java

```
package board;

import java.io.Serializable;

public class UserRecord implements Serializable{

  //
  protected String name;
  protected String destination;
  protected String hour;
  protected String minute;
  protected String memo;

  //レコード情報の初期化
  public UserRecord(String[] data){
    name = data[0];
    destination = data[1];
    hour = data[2];
    minute = data[3];
    memo = data[4];
  }

  public String getName() {
    return name;
  }
```

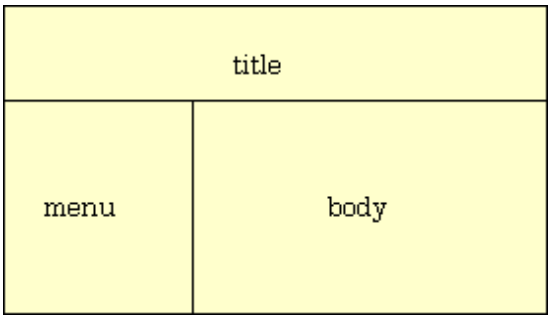
```
  public void setName(String name) {
    this.name = name;
  }

  public String getDestination() {
    return destination;
  }
  public void setDestination(String destination) {
    this.destination = destination;
  }

  public String getHour() {
    return hour;
  }
  public void setHour(String hour) {
    this.hour = hour;
  }

  public String getMinute() {
    return minute;
  }
  public void setMinute(String minute) {
    this.minute = minute;
  }

  public String getMemo() {
    return memo;
  }
  public void setMemo(String memo) {
    this.memo = memo;
  }
}
```

以上でユーザ定義クラスと、表一覧データを保存するクラスができました。

3.3.4 入出力ページの作成

作成したデータBeanに対して、それらに対応する入出力ページを作成します。

作成する入出力ページ名と対応するデータBean名を関連付けます。画面領域名は以下の様に分割されています。



"body"領域は、データBeanによって画面を切り替えます。

表示する内容	データBean
行動予定一覧表	ScheduleBean
課員の予定項目一覧	UserScheduleBean

入出力ページと対応するデータBeanの関係は以下のようになります。

入出力ページ名	データBean名	ページで表示する内容の説明
title.jsp	TitleBean	タイトルと今日の日付の表示
table.jsp	ScheduleBean	行動予定表一覧の表示(表形式)
userschedule.jsp	UserScheduleBean	課員別行動予定の変更画面の表示
usermenu.jsp	ProfileBean	課員選択画面の表示

上の情報より入力すべき情報の一覧をまとめます。「ラベル名」は入出力ページに表示されるボタンのラベル名です。このラベル名はあとで変更が可能です。

入出力ページ名	データBean	表示領域名	表示モード	コマンド	ラベル名	説明
title.jsp	board.TitleBean	title	(なし)	(なし)	(なし)	タイトル
table.jsp	board.ScheduleBean	body	(なし)	(なし)	(なし)	予定一覧
userschedule.jsp	board.UserScheduleBean	body	(なし)	update	予定表更新	課員別予定
usermenu.jsp	board.ProfileBean	menu	(なし)	selectuser	選択	課員選択

注:このサンプルでは表示モードは使用していません。1つのデータBeanで複数の画面を扱うときは表示モードを使用してください。

定義に必要な情報を整理後、ページを作成します。最初にmain.jspに記述を追加します。このページはすべてのHTTPリクエストに対して、必ず呼び出されるページです。このページで画面上の領域を定義します。"title","menu","body"の配置はT字型になるので、1行目をひとつのセルにして、テーブルを使います。

main.jspのソースは以下のとおりです。赤字が追加・修正したところです。

- main.jsp

```
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=windows-31j">
<title>行動予定表</title>

<%@ page contentType= "text/html; charset=windows-31j" %>
<%@ taglib uri="uji-taglib" prefix="uji" %>
<%@ page errorPage="boardError.jsp" %>
</head>
<body>

<uji:dispatch />
<table>
    <tbody>
    <tr>
    <td colspan="2">
    <uji:include pane="title" /></td>
    </tr>
    <tr>
    <td valign="top" height="100%">
    <uji:include pane="menu" /></td>
    <td valign="top" height="100%">
    <uji:include pane="body" /></td>
    </tr>
    </tbody>
</table>
</body>
</html>
```

次に他の入出力ページを作成します。まずusermenu.jspを作成します。以下の手順で行います。

Interstage Studioのメイン画面で、[ファイル]メニューの[新規] > [入出力ページ(Apcoordinator)]を選択します。

[入出力ページ]画面が表示されます。

それぞれの項目の意味は以下のとおりです。

項目	意味
プロジェクト名	入出力ページの追加先プロジェクト
ファイル名	入出力ページのファイル名（拡張子jspを除いた名前）
入力データ	入出力ページが使用するBeanの種別を指定します。データBeanかXMLデータBeanが選択できます。
データBean	入出力ページに対応するデータBean名(パッケージ名を付けた名前)
表示領域名	画面上で表示される領域名
表示モード	同じ表示領域で表示を切り替えるときに用いる
コマンド（ボタン名）	ボタンに割り付けられているコマンド名
ラベル	ボタンのラベルとなる文字列
uji:getPropertyを利用して入出力ページを生成	jsp:getPropertyを使用する場合はチェックを外してください。

表示モードは、複数の入出力ページで同じデータBeanを使う場合は、入出力ページ切り替えのためモード名を別にする必要があります。

追加ボタンを押すことにより、ボタンの追加ができます。コマンドは前表で用いたものと同じものを使います。ラベル欄にはボタン上に表示される文字列を入力します。

usermenu.jspに関して入力する項目は以下のとおりです。

項目	意味
プロジェクト名	board
ファイル名	usermenu
入力データ	データBean
データBean	board.ProfileBean
表示領域名	menu
表示モード	（なし）
コマンド（ボタン名）	selectuser
ラベル	選択

データBeanの指定は、[参照]ボタンを使って、[ファイルを開く]ダイアログから選択することが可能です。

コマンドとラベル名の入力は、右側にある[追加]ボタンを押すと以下のようになります。[コマンド]と[ラベル]を入力します。

[完了]ボタンを押すと、usermenu.jspのひな型が生成されます。以下のソースになります。

- usermenu.jsp（ひな型）

```
<%@ page contentType="text/html; charset=windows-31j" %>
<%@ taglib uri="uji-taglib" prefix="uji" %>
<uji:useBean id="menu" cls="board.ProfileBean" request="true" />
<uji:form name="mainForm" method="post" verbs="selectuser" beanId="menu" beanCls="board.ProfileBean>

    <%-- {{UJIWIZ_GENERATE(ITEM) --%><br>
  課員一覧<uji:comboBox bean="menu" property="username"/><br>
    <%-- }}UJIWIZ_GENERATE --%>

    <%-- {{UJIWIZ_GENERATE(SUBMIT) --%>
  <input type="submit" name="selectuser" value="選択" >
    <%-- }}UJIWIZ_GENERATE --%>
</uji:form>
```

usremenu.jspのひな型を以下のように変更します。

・ usermenu.jsp

```
<%@ page contentType="text/html; charset=windows-31j" %>
<%@ taglib uri="uji-taglib" prefix="uji" %>
<uji:useBean id="menu" cls="board.ProfileBean" request="true" />

<table height="170" border="0" bgcolor="#ffe0c0">
<tr><td>
<h4 align="center">行き先変更<br />する人を選択<br />してください</h4>

<uji:form name="myform" method="post" verbs="selectuser" beanId="menu">
  <uji:comboBox bean="menu" property="username" />
  <input type="submit" name="selectuser" value="選択" >
</uji:form>

 </td></tr>
</table>
```

上記のJSP内で最も重要な個所は、"uji:comboBox"タグです。

"uji:comboBox"にはコンボボックス表示に使用するデータBeanと、そのBeanの内部で定義されている変数の組を記述します。

タグの属性として、"bean"と"property"がありますが、"bean"は、"uji:useBean"のidで定義されている領域名（この場合はmenu）を入れます。実際は"cls"属性に記述されている"ProfileBean"が"usermenu.jsp"との間でデータをやり取りします。

"property"は"ProfileBean"の中に定義されているプロパティ名を書きます。この場合はコンボボックスタグを使うので、プロパティusernameはComboBoxクラスのインスタンス（実体）でなければなりません。

その他の入出力ページ(userschedule.jsp, title.jsp, table.jsp)に関しても、usermenu.jspと同様に入力を行います。ソースは以下のとおりです。ウィザードで雛型を作り、以下のソースをコピーしてください。なお、userschedule.jspのボタンのラベルは「予定表更新」、コマンド（ボタン名）は「update」とします。

・ userschedule.jsp

```
<%@ page contentType="text/html;charset=windows-31j" %>
<%@ taglib uri="uji-taglib" prefix="uji" %>
<uji:useBean id="body" request="true" cls="board.UserScheduleBean" />
<uji:form method="post" name="update" verbs="update" beanId="body" beanCls="board.UserScheduleBean">
        <table bgcolor="#ffe0a0" border="0" height="170">
        <tbody>
        <tr>
        <td colspan="2">
        <b>行き先・帰社時間変更</b></td>
        </tr>
        <tr>
        <td>名前:</td>
        <td>
        <uji:fieldString bean="body" property="name" editable="false" />
        <tr>
        <td>行き先(必須):</td>
        <td>
        <uji:fieldString bean="body" property="destination" indispensableField="true" />
        </tr>
        <tr>
        <td>帰社時間:</td>
        <td>
        <uji:fieldLong bean="body" property="hour" width="25"
                maxLength="2" minimumValue="0" maximumValue="24" />時
        <uji:fieldLong bean="body" property="minute" width="25"
                maxLength="2" minimumValue="0" maximumValue="59" />分
        </tr>
        <tr>
        <td>メモ:</td>
```

```
        <td>
        <uji:fieldTextArea bean="body" property="memo" columns="14" rows="2" />
        </tr>
        <tr>
        <td></td>
        <td></td>
        </tr>
        <tr>
        <td></td>
        <td>
        <input type="submit" value="予定表更新" name="update" />
        </td>
        </tr>
        </tbody>
        </table>
        <br>
</uji:form>
```

・ title.jsp

```
<%@ page contentType="text/html;charset=windows-31j" %>
<%@ taglib uri="uji-taglib" prefix="uji" %>
<uji:useBean id="title" cls="board.TitleBean" request="true" />
<table width="100%" border="2">
        <tbody>
        <tr>
        <td>
        <b>行動予定表</b></td>
        <td>今日は:
        <uji:getProperty bean="title" property="today" /></td>
        </tr>
        </tbody>
</table>
```

・ table.jsp

```
<%@ page contentType="text/html; charset=windows-31j" %>
<%@ taglib uri="uji-taglib" prefix="uji" %>
<uji:useBean id="body" request="true" cls="board.ScheduleBean" />

<uji:tableView bean="body" property="scheduleTable" borderWidth="2"
        height="170" width="100%" ruleWidth="1" headerBackground="#eeffee"/>
```

これで、すべての入出力ページができました。

3.3.5 ビジネスクラスの生成

ビジネスクラスは、以下の手順で作成します。

Interstage Studioのメイン画面で、[ファイル]メニューの[新規] > [ビジネスクラス(Apcoordinator)]を選択します。

[ビジネスクラス情報]画面で設定する項目は以下のとおりです。

設定項目	設定内容
プロジェクト名	"board"が指定されていることを確認してください。
参照ファイル名	何も指定しません。(デフォルト)
パッケージ名	"board"を指定します。 生成するビジネスクラスのパッケージ名を入力します。
クラス名	"BoardHandler"を指定します。 生成するビジネスクラスのクラス名を入力します。

設定項目	設定内容
入力データ種別	メソッドの入力データ型を選択します。
入力情報	データBeanのクラス名(パッケージ名付き)を入力します。
コマンド	コマンドマップに設定するコマンドを指定します。
メソッド	生成するメソッド名を指定します。
復帰値型	メソッドの復帰値型を選択します。
セションスコープとして生成	チェックを入れる(デフォルト値) 同一セションの間同じインスタンスを使用する場合にチェックします。
コマンドスクリプティングとして生成する	チェックを入れない(デフォルト値) ビジネスクラス生成ウィザードで指定した内容をコマンドスクリプティングとして生成する場合にチェックします。

次に追加ボタンを押して、実装するメソッドの雛型を作成します。[クラス名]で指定したデータBeanは[メソッド]の引数に与えられます。

[メソッドの追加]ダイアログで設定する項目は以下のとおりです。

入力データ	クラス名	コマンド	メソッド	復帰値型	コマンドの意味
データBean	board.UserScheduleBean	update	updateUserSchedule	void	予定表更新
データBean	board.ProfileBean	selectuser	showUserSchedule	void	選択課員の予定表示
入力データなし	(なし)	(なし)	startup	void	初期画面表示

データBeanはボタンのある入出力ページとデータをやり取りしているBeanである必要があります。

[追加]ボタンで次のメソッドの追加画面になります。上の表のすべてのメソッドについて繰り返し追加を行います。"startup"メソッドは、メソッド名のみの追加になります。追加が終わったら「閉じる」ボタンを押します。

メソッドの追加後、[完了]ボタンを押して終了します。BoardHandler.javaの名前でビジネスクラスのファイルが生成されます。次節でビジネスロジックの追加をします。

3.3.6 ビジネスロジックの追加

“3.3.5 ビジネスクラスの生成”で、ビジネスクラスとして、BoardHandler.javaを生成しましたが、具体的にロジックを実装する作業に入ります。ソースコードは以下のとおりです。赤字が修正・追加した個所です。

- BoardHandler.java

```
/*
 * BoardHandler.java
 * author:
 * Copyright(c)
 */

package board;

import javax.servlet.ServletContext;

import com.fujitsu.uji.DispatchContext;
import com.fujitsu.uji.GenericHandler;
import com.fujitsu.uji.http.HttpDispatchContext;
import board.*;
```

```
/**
 * ビジネスクラスです。
 *
 * @author
 * @version
 */
public class BoardHandler extends GenericHandler
{
    //{{UJIWIZ_GENERATE(METHODDEF)
    //@@UJI-Wizard Information(Class)
    //@@,,startup,void
    //@@board.UserScheduleBean,update,updateUserSchedule,void
    //@@board.ProfileBean,selectuser,showUserSchedule,void
    //}}UJIWIZ_GENERATE

    //内部変数
    TitleBean titleBean;   //タイトル情報を保持するデータBean
    ProfileBean proBean;   //課員メニュー情報を保持するデータBean
    ScheduleBean scheBean; //行動予定表情報を保持するデータBean

   /**
    * 新しいビジネスクラスのオブジェクトを作成します。
    */
    public BoardHandler() {
        // 以下に初期化する手続きを記述して下さい。
        titleBean = new TitleBean();
    }

   /**
    * クラスが初期化される時に呼び出されます。
    * trueを返却するとインスタンスはセションスコープになります。
    * falseを返却するとインスタンスはリクエストスコープになります
    * @return セションスコープとする場合true
    */
    public boolean init() {
        // セッションスコープとする場合はtrue、しない場合はfalseを返します。
        return true;
    }

    //{{UJIWIZ_GENERATE(METHOD)
    /**
     * startupの処理を行います。
     * @param context 呼び出し情報を保持するクラス
     */
    public void startup(DispatchContext context) {
        // 以下に必要な手続きを記述して下さい。
        // 例 :
        //    // Beanを生成し、データを設定します。
        //    SampleBean dataBean = new SampleBean();
        //    dataBean.setXXX(...);
        //    // 表示に使うBeanに表示モードを設定します。
        //    dataBean.setVerb("mode1");
        //    // 各画面領域対してBeanを設定します。
        //    context.setResponseBean("body", dataBean);

        //行動予定表データのロード
        BoardApplication ba = (BoardApplication)context.getApplicationProfile();

        //main.jspの絶対パスを取得する
        HttpDispatchContext hdc = (HttpDispatchContext)context;
        ServletContext sc = hdc.getServletContext();
        String path = sc.getRealPath("/");
        ba.setPath(path);
```

```
    //一覧表データの取得
    DataManager dm = ba.loadData();

    //課員の行動予定一覧を作成
    scheBean = new ScheduleBean(dm);
    context.setResponseBean("body",scheBean);

    //課員選択メニューの作成
    proBean = new ProfileBean(dm);
    context.setResponseBean("menu",proBean, true);

    //タイトルの作成
    context.setResponseBean("title",titleBean, true);
}

/**
 * updateの処理を行います。
 * @param context 呼び出し情報を保持するクラス
 * @ぱらm だたBean データBeanクラス
 */
public void updateUserSchedule(DispatchContext context, board.UserScheduleBean dataBean) {
    // 「update」の処理を記述します。
    // 例 :
    //    // 次画面用のBeanを生成し、データを設定します。
    //    YYYBean bean = new YYYBean();
    //    bean.setZZZ(...);
    //    // 表示に使うBeanに表示モードを設定します。
    //    bean.setVerb("mode2");
    //    // 各画面領域対してBeanを設定します。
    //    context.setResponseBean("body", bean);

    //アプリケーションクラスで保存されている予定表一覧データの取得
    BoardApplication ba = (BoardApplication)context.getApplicationProfile();
    DataManager dm = ba.loadData();

    //課員名をキーとしてdm内を検索して、新しいデータを設定する
    String[] record = new String[5];
    record[0] = dataBean.getName().getRawText();
    record[1] = dataBean.getDestination().getRawText();
    record[2] = dataBean.getHour().getRawText();
    record[3] = dataBean.getMinute().getRawText();
    record[4] = dataBean.getMemo().getRawText();
    UserRecord ur = new UserRecord(record);
    dm.update(ur);
    //dm.updateDataManager(dataBean);

    //アプリケーションクラスにある一覧表データの更新
    ba.saveData(dm);

    //課員の行動予定一覧を作成
    ScheduleBean scheBean = new ScheduleBean(dm);
    context.setResponseBean("body",scheBean);
}

/**
 * selectuserの処理を行います。
 * @param context 呼び出し情報を保持するクラス
 * @param data Bean データBeanクラス
 */
public void showUserSchedule(DispatchContext context, board.ProfileBean dataBean) {
    // 「selectuser」の処理を記述します。
    // 例 :
```

```
        //    // 次画面用のBeanを生成し、データを設定します。
        //    YYYBean bean = new YYYBean();
        //    bean.setZZZ(...);
        //    // 表示に使うBeanに表示モードを設定します。
        //    bean.setVerb("mode2");
        //    // 各画面領域対してBeanを設定します。
        //    context.setResponseBean("body", bean);

        //アプリケーションクラスで保存されている予定表一覧データの取得
        BoardApplication ba = (BoardApplication)context.getApplicationProfile();
        DataManager dm = ba.loadData();

        //選択された課員の予定を表示する
        UserScheduleBean userScheBean = dm.obtainUserSchedule(dataBean);
        context.setResponseBean("body",userScheBean);
    }

    //}}UJIWIZ_GENERATE

}
```

以下は主なメソッド内での処理内容です。

startup:

このメソッド内ではデータの初期化を行います。処理の流れとしては以下のとおりです。

1. 保存されている表データを読み込み、アプリケーションクラス内で定義している静的変数に保存します。
2. それぞれのデータBeanに対してset(プロパティ名)メソッドを使い表構成に必要なデータを設定します。
3. 表示する表示領域とデータBeanの対を設定します。

上記処理のうち、2,3は必要最小限の処理です。1は、このアプリケーション特有の処理です。

showUserSchedule:

このメソッド内では、行動予定を変更する課員が選択されたあとの処理を記述します。

1. 選択された課員名をキーとして、startupの解説の1で読み込まれた表データから、選択された課員情報を取り出します。
2. どの課員が選択されたかは、ProfileBeanの中のComboBoxのインスタンスが保持しています。
3. 取り出した課員情報を、UserScheduleBeanに設定します。
4. 表示する表示領域とデータBeanの対を設定します。

updateUserSchedule:

このメソッド内では、変更した課員の行動予定を更新する時の処理を記述します。

1. UserScheduleBeanから更新されたデータを取得します。
2. 表データを管理している内部変数に上書きします。
3. 表データをファイルに保存します。
4. ScheduleBean のインスタンスを新たに生成します。
5. 表示する表示領域とデータBeanの対を設定します。

次にアプリケーションクラスの拡張を行ないます。このクラスでは、行動予定表作成に必要なデータを保持します。赤字が追加した個所です。

- BoardApplication.java

```
/*
 * BoardApplication.java
 *
 * Copyright (c)
 */
```

```
package board;

import java.io.File;
import java.io.FileInputStream;
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.ObjectInputStream;
import java.io.ObjectOutputStream;

import com.fujitsu.uji.ApplicationProfile;

/**
 * アプリケーションクラス
 *
 * @author
 * @version
 */

public class BoardApplication extends ApplicationProfile {

    //内部変数
    private static String datafile = "DataManager.tmp"; //一覧表データが格納される
    private String filepath = ""; //メインページのある絶対パス
    protected DataManager dm; //データが入る
    /**
     * このクラスのコンストラクタです。
     */
    public BoardApplication(){
    }

    /**
     * uji.methodリクエストパラメタによるメソッド指定を有効にするかどうか指定します。
     * trueを返すとuji.methodリクエストパラメタを処理します。
     * falseの場合uji.methodリクエストパラメタを無視します。
     * @return trueの場合有効になります。falseの場合無視します。
     */
    public boolean isUjiMethodAvailable() {
        return false;
    }

    public DataManager loadData(){
        if(dm==null){
            this.dm = loadDataManager(new File(getPath(), datafile));
        }
        return this.dm;
    }

    public synchronized void saveData(DataManager data){
        this.dm = data;
        //更新ごとに最新データのセーブを行う
        saveDataManager(this.dm, new File(getPath(), datafile));
    }

    public String getPath(){
        return this.filepath;
    }

    public void setPath(String path){
        this.filepath = path;
    }

    //シリアライズされた予定表データを元に戻すためのメソッド
```

```
    private DataManager loadDataManager(File infile){
        try{
            FileInputStream fis = new FileInputStream(infile);
            ObjectInputStream ois = new ObjectInputStream(fis);
            return (DataManager)ois.readObject();
        } catch(IOException ioe) {
            //シリアライズされたファイルから読み込めない
            DataManager dm = new DataManager();
            return dm;
        } catch (ClassNotFoundException cnex){
            //ファイルは読み込めたが該当するクラスが無い場合
            DataManager dm = new DataManager();
            return dm;
        }
    }

    //予定表データをシリアライズするためのメソッド
    private synchronized void saveDataManager(DataManager dmr, File outfile){
        try{
            FileOutputStream fos = new FileOutputStream(outfile);
            ObjectOutputStream oos = new ObjectOutputStream(fos);
            oos.writeObject(dmr);
        } catch (IOException ioe2 ){}
    }
}
```

これで、行動予定表アプリケーションに必要なソースコードが全部そろいました。

3.3.7 コンパイルとローカル実行の方法

コンパイルの方法は、Interstage Studioの説明を参照してください。以下では簡単な流れと注意点を説明します。

1. Interstage Studioを起動して、Apcoordinatorアプリケーションのプロジェクトを開きます。
2. Interstage Studioの[プロジェクト]メニューの[自動的にビルド]にチェックが入っていない場合は[プロジェクトのビルド]を選択します。チェックが入っている場合はプロジェクト作成後、自動的にビルドされるため操作は不要です。

   注意

   問題ビューにserialVersionUIDに関する警告が表示されますが、問題はありません。
3. [実行]メニューの[デバッグ]から[サーバでデバッグ]を選択します。
4. サーバを選択し、[次へ]ボタンをクリックすると、サーバ上の構成するプロジェクトの選択画面が表示されます。
5. [完了]ボタンをクリックすると、サーバ上へのプロジェクトの構成とサーバ起動が実行され、デバッグパースペクティブへの切り替え確認ダイアログが表示されます。
6. パースペクティブへの切り替え確認に[はい]をクリックするとデバッグが開始されます。

3.3.8 マップファイルの編集

通常はビジネスクラスの作成時にマップファイルは自動的に作成されますが、Interstage Studioでビジネスクラスを生成したあとにマップファイルを修正する場合は、以下の記述を参考にしてください。Interstage Studioでビジネスクラスを作成したが、あとで使わなくなったメソッドが出てきたとか、メソッドを手で追加した場合が当てはまります。ここではcommands.mapとpages.mapの書式の説明を行動予定表アプリケーションを例にとり、説明しています。

commands.mapの書式は以下のとおりです。

```
（データBean名）；（ボタンを押されたときに送られるコマンド）＝ （パッケージ名）.（ビジネスクラス名）.（メソッド名）
```

データBean名は、コマンドを送るボタンが付いている入出力ページがデータを送る先のデータBeanを指定します。これは[入出力ページの作成]で指定したBean名と同じである必要があります。その他の項目については、[ビジネスクラスの生成]の時に入力した値をそのまま使用してください。

例として、行動予定表アプリケーションでは以下の組み合わせができます。

userschedule.jspに関して:

(データBean);update=(作ったパッケージ名).BoardHandler.updateUserSchedule

usermenu.jspに関して:

(データBean);selectuser=(作ったパッケージ名).BoardHandler.showUserSchedule

最初に呼ばれるとき:

(データBeanなし);(コマンドなし) =(作ったパッケージ名).BoardHandler.startup

パッケージ名はプロジェクトの作成で入力したものを使います。ここではサンプルと同様に"board"にします。データBeanと入出力ページの間には以下の関係があります。

```
userschedule.jsp<->UserScheduleBean
usermenu.jsp<->ProfileBean
```

かっこの中を埋めると、以下のcommands.mapが導き出されます。

- commands.map

```
# commands.map
board.UserScheduleBean;update=board.BoardHandler.updateUserSchedule
board.ProfileBean;selectuser=board.BoardHandler.showUserSchedule
;=board.BoardHandler.startup
```

pages.mapの書式は以下のとおりです。

```
(データBean名) ;(表示モード)=(対応する入出力ページ名)
```

行動予定表アプリケーションを例にとると以下のとおりです。

- pages.mapの途中の状態

```
# pages.map
board.ProfileBean;(表示モード)=usermenu.jsp
board.UserScheduleBean;(表示モード)=userschedule.jsp
board.TitleBean;(表示モード)=title.jsp
board.ScheduleBean;(表示モード)=table.jsp
```

表示モードは同じBeanに複数の入出力ページが割り当てられているときの識別子に相当します。ビジネスクラスのメソッド内でデータBeanに設定する表示モードと同じものをここに指定してください。ビジネスクラスをまだ作成していない場合は、pages.mapファイルで表示モードを適当な名前で定義し、あとでビジネスクラスの方であらかじめ定義してある表示モードをsetVerb()で指定してください。対応する入出力ページが一種類しかない場合は、表示モードは省略することもできます。このアプリケーションでは表示モードを使用いないので、下記のようになります。

- pages.map(最終形)

```
# pages.map
board.ProfileBean;=usermenu.jsp
board.UserScheduleBean;=userschedule.jsp
board.TitleBean;=title.jsp
board.ScheduleBean;=table.jsp
```

マップするBeanや入出力ページを変更する場合には、以上の手順で、マップファイルの手動編集を行います。

3.3.9 Webサーバへの配備

運用環境に配布するには、WARファイルを作成する必要があります。作成したWARファイルは、Interstage Java EE管理コンソールの配備機能を利用して、サーバに配備します。

参照

運用環境へのWARファイルの配布は、「Interstage Studio ユーザーズガイド」を参照ください。

ポイント

Interstage Java EE管理コンソールを使用して必要なjarファイルをクラスパスへ設定します。

[設定] > [クラスタ名-config] または [server-config] > [JVM 設定] > [パス設定]タブ

Windowsの場合

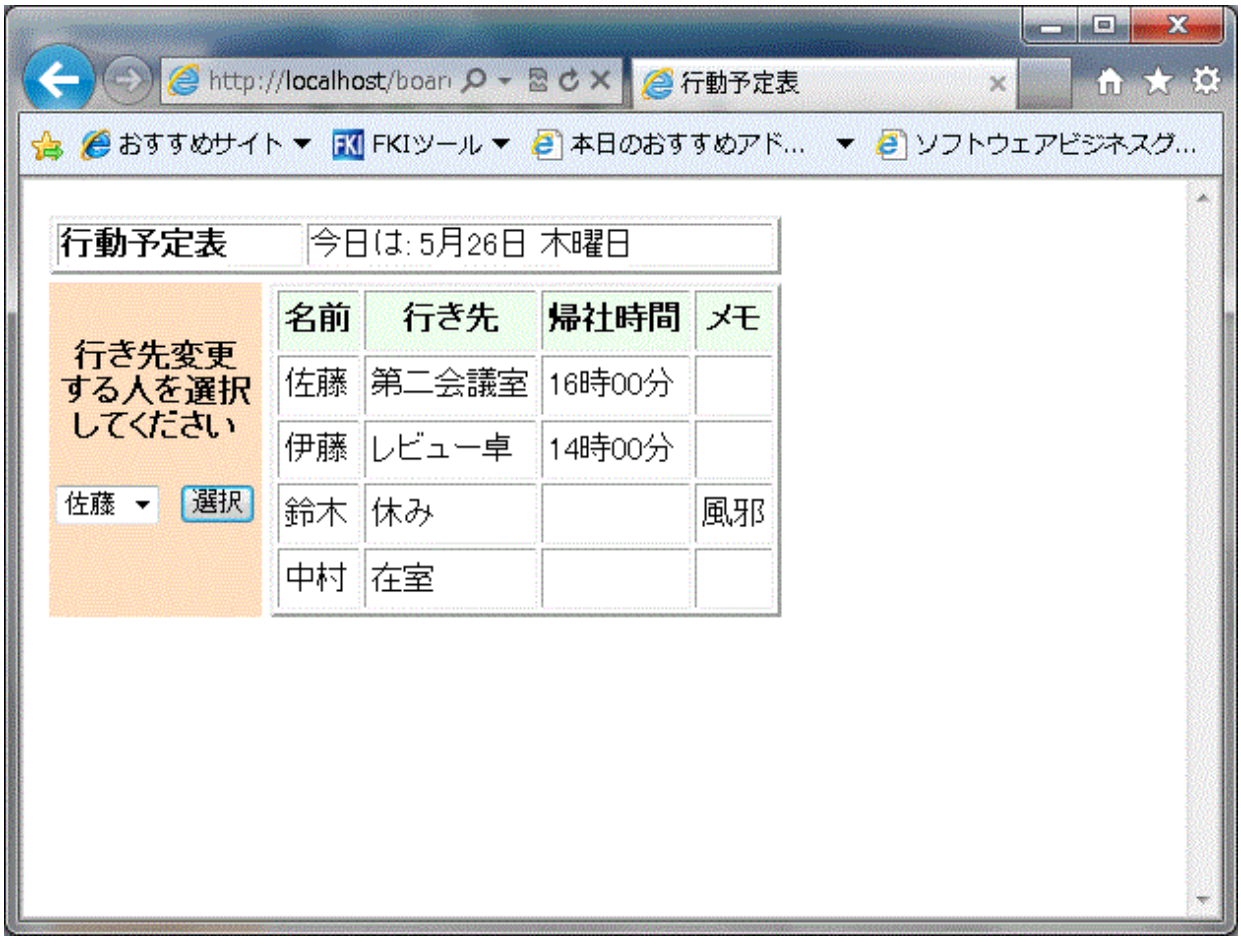
設定項目	設定内容
クラスパスのサフィックス	[Apcoordinatorのインストールフォルダ]¥lib¥uji.jar

Solaris,Linuxの場合

設定項目	設定内容
クラスパスのサフィックス	[Apcoordinatorのインストールフォルダ] /FJSVwebc/lib/uji.jar

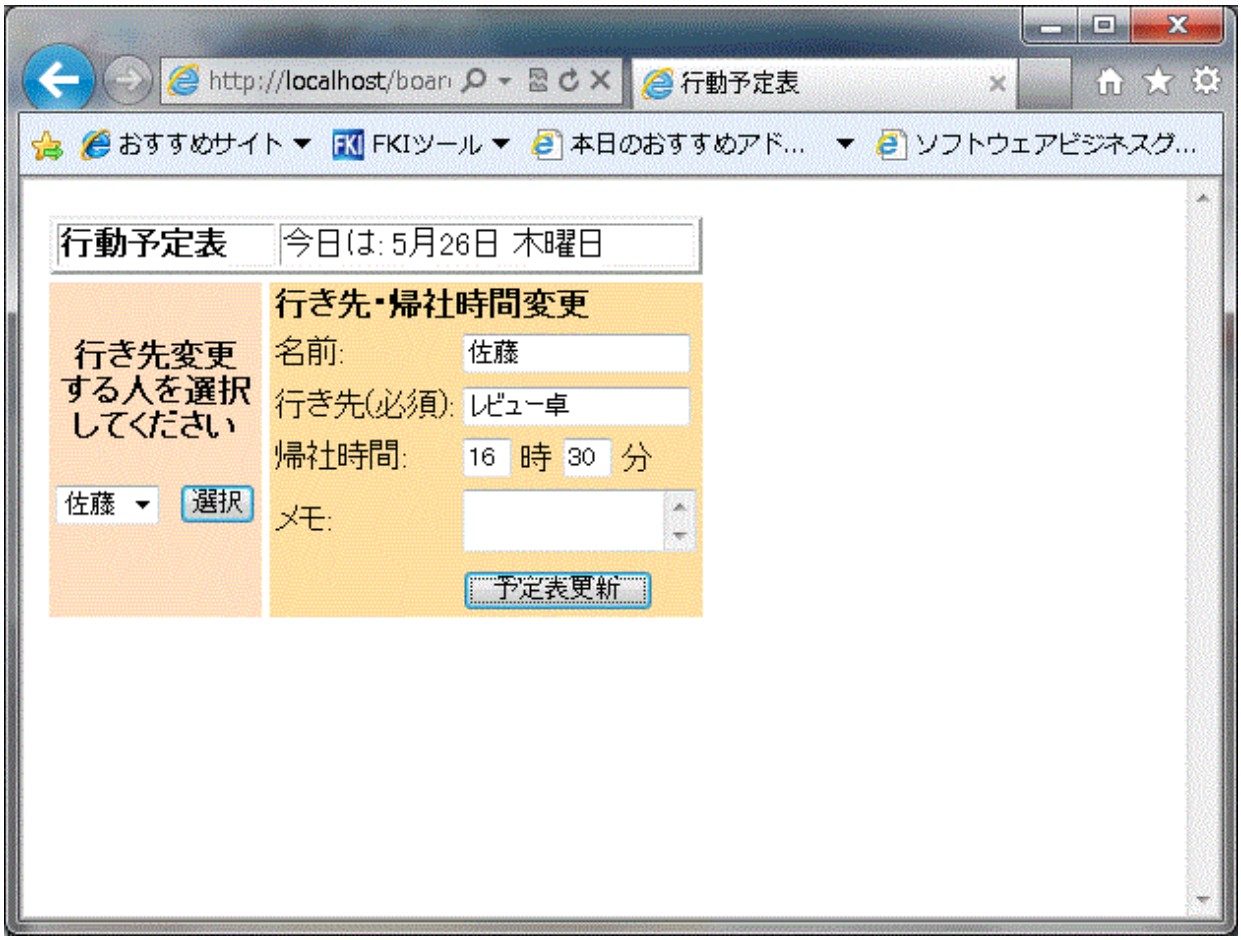
3.3.10 Webサーバ上での実行

1. Webブラウザを起動して、"http://Webサーバの名前/board/main.jsp"を開きます。

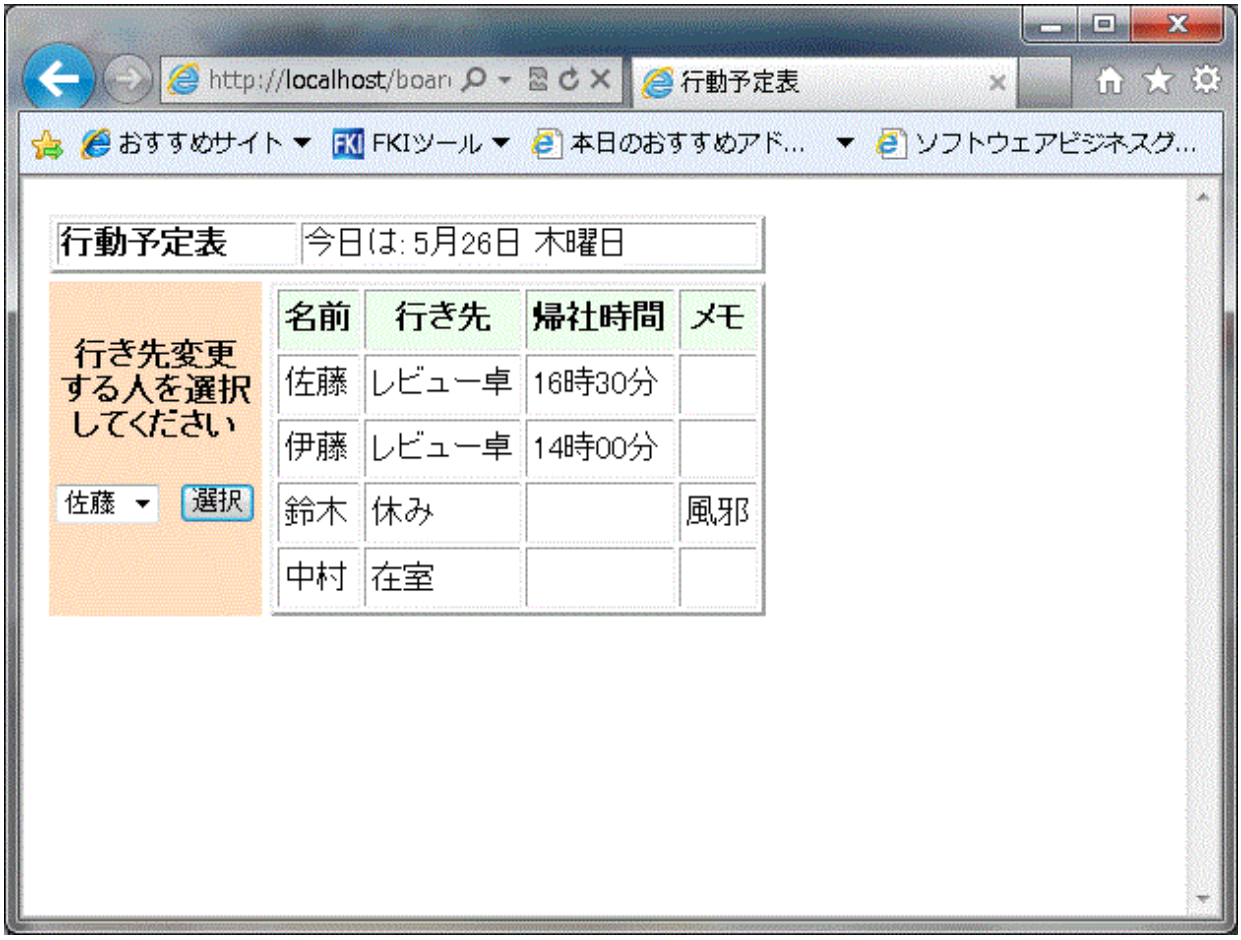


「佐藤」さんを選択して、「選択」ボタンをクリックします。

2. 佐藤さんの個人の行動予定が表示されます。行き先を「第二会議室」から「レビュー卓」へ変更し、帰社時間を「16時00分」から「16時30分」に変更してから、「予定表更新」ボタンを押します。



3. 佐藤さんの予定が更新されました。



第4章 EJBを利用したアプリケーションの設計と作成

本章では、会議室の予約状況を管理する「会議室予約システム」アプリケーションを例に、Apcoordinatorによるリモート共通インタフェースを使用したEJBアプリケーションの作成手順を説明します。

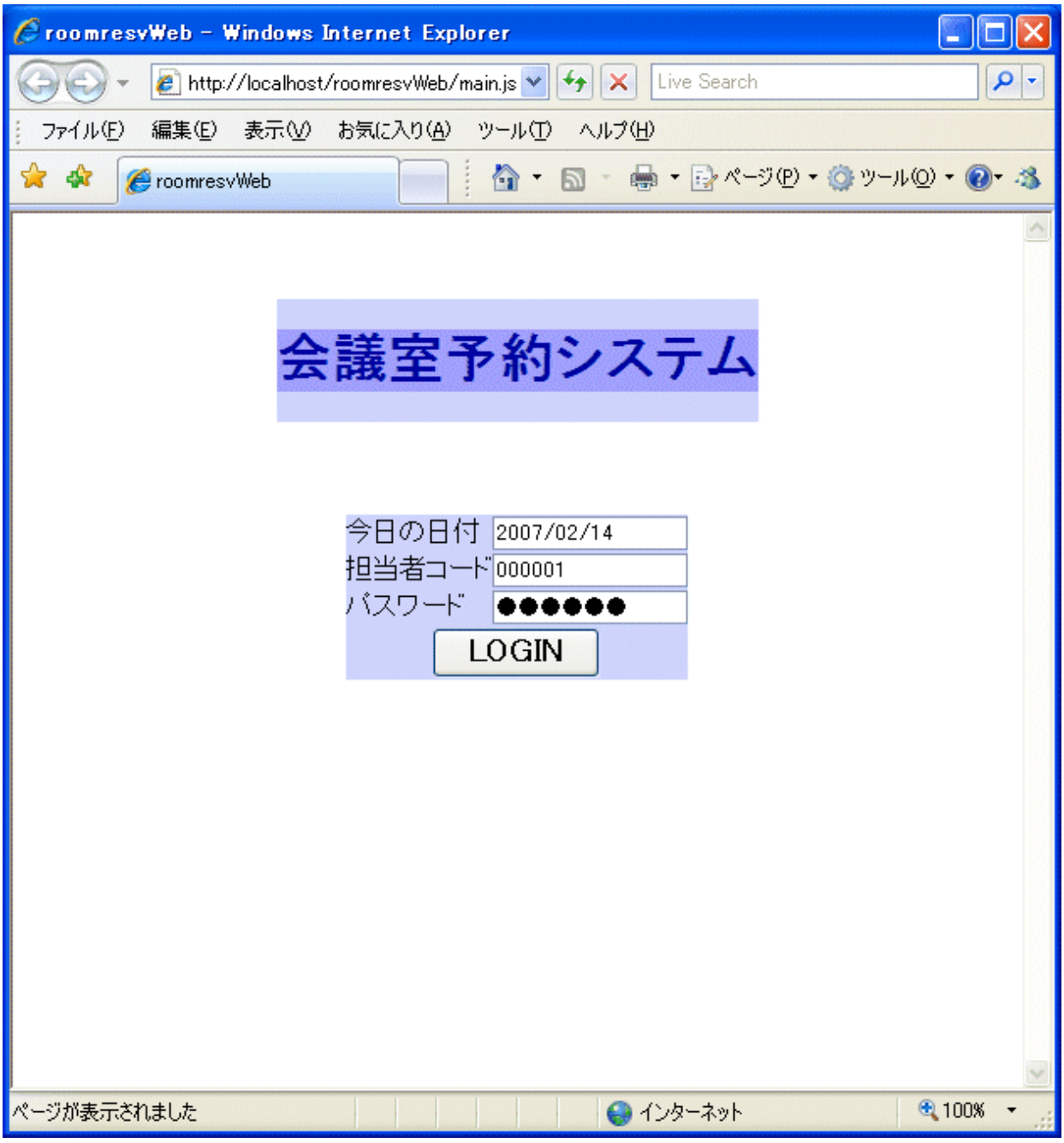
4.1 会議室予約システムアプリケーションの概要

機能概要説明

ここでは会議室予約システムアプリケーションの機能概要、画面構成および内部構成を説明します。

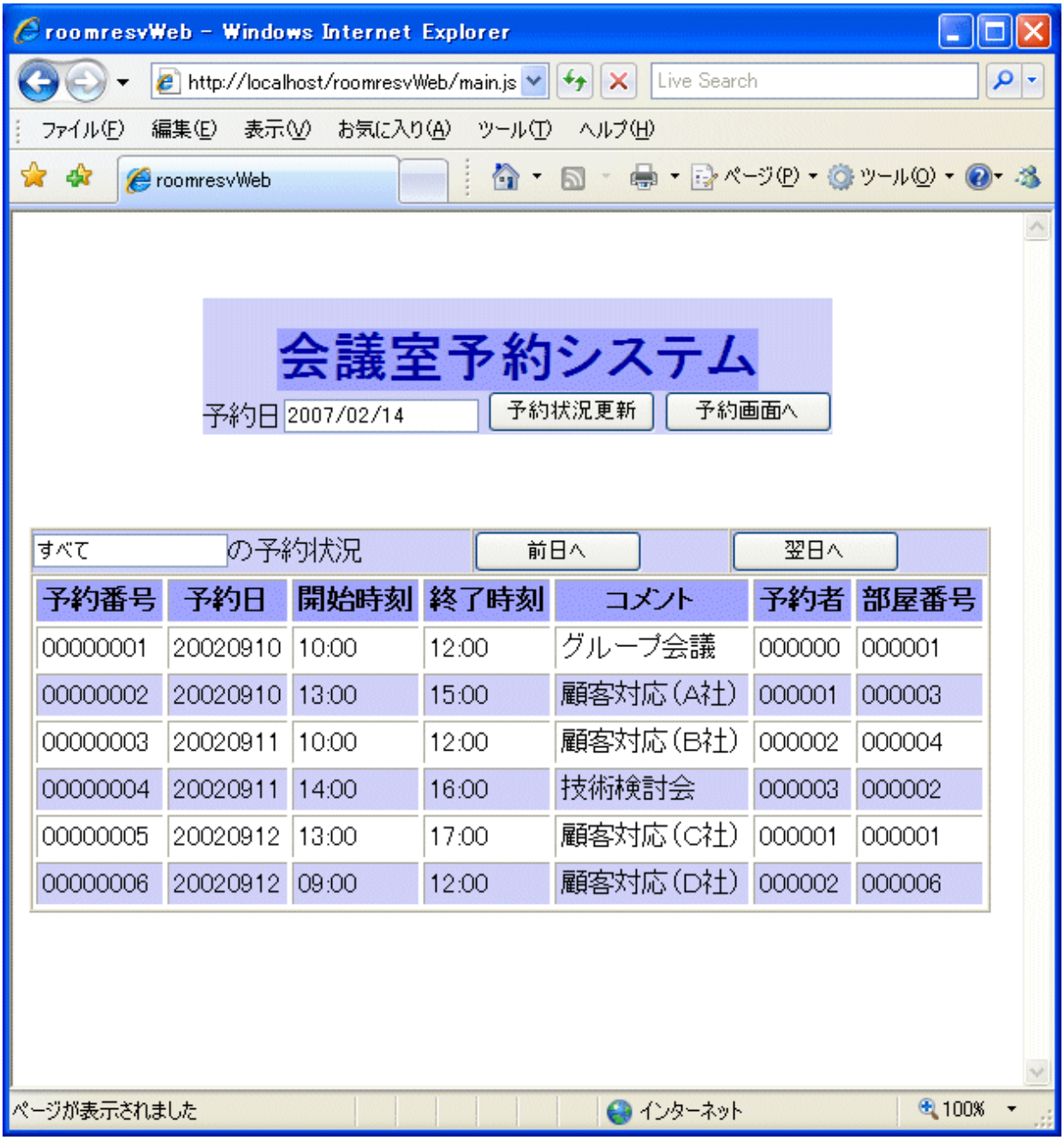
ログイン画面

アプリケーションへログインします。



予約日の指定画面

予約状況を確認し、予約日を入力します。



予約情報の入力画面

予約する会議室・時刻・予約者の情報を入力し、登録します。予約番号を入力し、予約を取り消すこともできます。

アプリケーションの構成

このチュートリアルのアプリケーションは、以下のように構成されています。

- EJBアプリケーション
  - SessionBean x 1
  - EntityBean x 3
- Webアプリケーション

アプリケーションの動作環境

このチュートリアルのアプリケーションの開発、動作に必要なソフトウェア環境は以下です。

以下のいずれかの製品が提供するフレームワークおよびアプリケーションサーバ

- Interstage Business Application Server

・ Interstage Application Server

・ Interstage Studio

以下の製品が提供するJava統合開発環境

・ Interstage Studio

上記に加え、以下のソフトウェアが必要です。

・ Symfoware(R) Server

なお、このチュートリアルでは、開発・実行環境の状態として以下を仮定しています。

開発環境

― Java統合開発環境がC:¥Interstage¥IDEフォルダへインストールされていること

― フレームワークがC:¥Interstage¥APCフォルダへインストールされていること

― Symfoware JDBCドライバがC:¥SFWCLNT¥JDBC¥fjjdbcフォルダへインストールされていること

― Symfoware JDBCのネーミングサービスが動作し、JDBCデータソースが登録されていること

JDBCのネーミングサービスおよびJDBCデータソースについては、以下をご参照ください。

― “4.5.1 Symfowareサービスの起動”

― “4.4.2 JDBCデータソースの定義”

実行環境

― Symfoware ServerをWindows上で動作させていること

― フレームワークがC:¥Interstage¥APCフォルダ(Windowsの場合)または/optディレクトリ(Solaris, Linuxの場合)へインストールされていること

― アプリケーションサーバがC:¥Interstageフォルダ(Windowsの場合)または/optディレクトリ(Solaris, Linuxの場合)へインストールされていること

― Symfoware JDBCドライバがC:¥SFWCLNT¥JDBC¥fjjdbcフォルダ(Windowsの場合)または/opt/FJSVsymjd/fjjdbcディレクトリ(Solaris, Linuxの場合)へインストールされていること

― Symfoware ServerのSymfoware Server RDBサービスおよびSymfoware RDAサービスが動作すること

― Symfoware JDBCのネーミングサービスが動作し、JDBCデータソースが登録されていること

JDBCのネーミングサービスおよびJDBCデータソースについては、以下をご参照ください。

― “4.5.1 Symfowareサービスの起動”

― “4.4.2 JDBCデータソースの定義”

これらを変更された場合は、お使いの環境に合わせてチュートリアルの記事を読み替えてください。

アプリケーションサーバとJava統合開発環境の実行画面は、提供する機能の違いにより、お使いのエディションや動作環境によって異なる場合があります。

4.2 EJBの開発

4.2.1 プロジェクトの作成

以下の手順で Enterprise JavaBeansプロジェクトを新規作成します。

1. Interstage Studioを起動します。
2. [ファイル] > [新規]メニューの[プロジェクト]を選択します。
3. [Apcoordinator] > [Enterprise JaveBeansプロジェクト(Apcoordinator)]を選択し、[次へ]をクリックします。

4. [EJBプロジェクト]画面では、以下を入力し、[次へ]をクリックします。

項目	設定内容	デフォルト設定値
プロジェクト名	"roomresvEJB"を指定します。	(なし)
プロジェクトコンテンツ	(デフォルト設定値)	チェックマーク付
ターゲットランタイム	[Interstage Application Server V11.1 IJServer Cluster(Java EE)]	(なし)
EJBモジュールバージョン	[2.1]を選択します。	3.0
構成	[EJBモジュールバージョン]を選択することで[<カスタム>]が自動的に選択されます。	Apcoordinator EJBプロジェクトのデフォルト構成
EARにプロジェクトを追加	(デフォルト設定値)	チェックマークなし

5. [EJBモジュール]画面では、以下を入力し、[次へ]をクリックします。

項目	設定内容	デフォルト設定値
ソースフォルダ	(デフォルト設定値)	ejbModule

6. [テンプレートを選択]画面では、以下を入力し、[次へ]をクリックします。

項目	設定内容	デフォルト設定値
テンプレートを使用してサンプルコードを作成	(デフォルト設定値)	チェックマーク付
使用可能なテンプレート	(デフォルト設定値)	"Enterprise Bean"

7. [Enterprise Bean(Apcoordinator)生成(1/3)]画面では、以下を入力し、[次へ]をクリックします。

項目	設定内容	デフォルト設定値
パッケージ名	"roomresv.ejb"を指定します。	"roomresvEJB"
名前	(デフォルト設定値)	"RoomresvEJB"

8. [Enterprise Bean(Apcoordinator)生成(2/3)]画面では、以下を入力し、[次へ]をクリックします。

項目	設定内容	デフォルト設定値
Session Bean種別	"Statefull Session Bean"を選択	"Stateless Session Bean"が選択
トランザクション管理種別	"Bean"を選択	"Container"が選択

9. [Enterprise Bean(Apcoordinator)生成(3/3)]画面では、表示内容を確認し、[完了]をクリックします。

項目	設定内容	デフォルト設定値
ファクトリクラスを拡張する	(デフォルト設定値)	チェックマーク付
ファクトリクラス名	(デフォルト設定値)	"RoomresvEJBFactory"
アプリケーションクラスを拡張する	(デフォルト設定値)	チェックマーク付
アプリケーションクラス名	(デフォルト設定値)	"RoomresvEJBApplication"
セションクラスを拡張する	(デフォルト設定値)	チェックマーク付
セションクラス名	(デフォルト設定値)	"RoomresvEJBSession"

10. Apcoordinatorのパースペクティブを開いていない場合は、確認のダイアログが表示されます。
[はい]をクリックします。

プロジェクトが作成されます。

4.2.2 ビジネスクラスの作成

1 ビジネスクラスの作成

以下の手順でビジネスクラスを作成し、メソッドを追加します。

1. Interstage Studioの[ファイル] > [新規]メニューの[ビジネスクラス(Apcoordinator)]を選択します。
2. プロジェクト名に[roomresvEJB]が選択されていることを確認し、クラス名およびメソッドを定義します。
   以下のようにクラス名を入力し、[追加]をクリックすると、メソッドの追加ダイアログが開きます。

項目	設定内容	デフォルト設定値
プロジェクト名	(デフォルト設定値)	roomresvEJB
クラス名	"RoomresvEJBHandler"を指定	(なし)
格納先EJBアプリケーション	(デフォルト設定値)	RoomresvEJB
セションスコープとして生成	(デフォルト設定値)	チェックマーク付
コマンドスクリプティングとして生成する	(デフォルト設定値)	チェックマークなし

[メソッドの追加]ダイアログでは、以下のすべてのメソッドを追加します。

入力データ	データ型	コマンド	メソッド	復帰値の型
HashMap/Map	UserRecord	findUser	findUser	Object
HashMap/Map	UserRecord	getResvTable	getResvTable	Object
HashMap/Map	UserRecord	getResvTableDate	getResvTableDate	Object
HashMap/Map	UserRecord	addResvTable	addResvTable	Object
HashMap/Map	UserRecord	delResvTable	delResvTable	Object
HashMap/Map	UserRecord	getRoomTable	getRoomTable	Object

すべて追加したら[閉じる]をクリックします。

[完了]をクリックするとビジネスクラスが作成されます。

2 EJBクライアントに対して送受信するデータの設計

EJBクライアントに対して送受信するデータとして、HashMapのキーを設計します。

ビジネスロジックはこのキー値により、クライアントからのデータを参照・設定します。

HashMapのキー値	データ型	クライアントでのデータ型
userID	String	String
userPass	String	String
userName	String	String
userPhone	String	String
resvID	String	String
resvDate	String	String
resvStart	String	String

HashMapのキー値	データ型	クライアントでのデータ型
resvEnd	String	String
resvComment	String	String
resvTable	Object[][]	TableView
roomID	String	String
roomTable	Object[][]	TableView

3 ビジネスクラス処理の実装

“1 ビジネスクラスの作成”で、作成された「RoomresvEJBHandler.java」ファイルに対し、メソッドの処理を実装します。

完成されたファイルが以下のフォルダにあります。参考にしてファイルを編集してください。

「C:¥Interstage¥APC¥sample¥javaee¥roomresv¥roomresvEJB¥ejbModule¥roomresv¥ejb¥RoomresvEJBHandler.java」

4.2.3 EntityBeanの作成

ここでは以下の3つのEntitiyBeanを作成します。

- RoomresvEJBCMPreserve
- RoomresvEJBCMProom
- RoomresvEJBCMPuser

注意

このチュートリアルの手順でEntityBeanを作成するには、あらかじめデータベースにテーブルが作成されており、開発環境からアクセスできるようになっている必要があります。

Symfoware(R) Serverを使用してデータベースを登録する手順は、以下に記載されています。

“4.4.1 Symfowareへのテーブルの登録”

以下の手順は、開発環境からデータベースのテーブルにアクセスできるようになったあとで実施してください。

また、プロジェクトのビルドパスに[Symfoware JDBCドライバインストールフォルダ]¥lib¥fjsymjdbc2.jarが必要です。

Interstage Studioの[ファイル]メニューの[プロパティ] > [Javaのビルドパス]を選択し、[ライブラリタブ]の[外部JARの追加]をクリックしビルドパスに追加してください。

RoomresvEJBCMPreserveの作成

RoomresvEJBCMPreserveは、データベースの「TRESERVE」テーブルを使用するEntityBeanです。

1. Interstage Studioの[ファイル] > [新規]メニューの[その他]を選択します。

   [EJB] > [J2EE] > [Enterprise Bean]を選択し、[次へ]をクリックします。

2. [Enterprise Bean]画面では、以下を入力し、[次へ]をクリックします。

項目	設定内容	デフォルト設定値
ソースフォルダ	(デフォルト設定値)	roomresvEJB/ejbModule
パッケージ	(デフォルト設定値)	roomresv.ejb
名前	"RoomresvEJBCMPreserve"を指定	(なし)
Enterprise Bean種別	"EJB2.x Container-managed persistence Entity Bean"を選択	"Stateless Session Bean"が選択

3. [Container-managed persistence Entity Bean]画面では、以下を入力し、[次へ]をクリックします。

項目	設定内容	デフォルト設定値
トランザクション管理種別	(デフォルト設定値)	"Container"が選択
リエントラントにする	(デフォルト設定値)	チェックマークなし
Recordクラスでデータの取得/設定を行う	(デフォルト設定値)	チェックマーク付
データソース名	jdbc/roomresvCMP	(なし)
Home/Remoteインタフェースを生成する	チェックマーク付	チェックマークなし
LocalHome/Localインタフェースを生成する	チェックマークなし	チェックマーク付

4. [永続化フィールド定義]画面では、永続化するフィールドを指定します。ここで、データベースに接続してテーブルの情報を取得することができます。[DB参照]をクリックします。

5. 接続データベースを選択します。

   データベース開発パースペクティブで作成済の一覧から選択します。新規に接続データベースを追加する場合は、[接続の追加]より接続データベースを追加作成します。

   [OK]をクリックします。

6. 永続化するフィールドを選択します。

   データベースに接続が成功すると、スキーマ・テーブル・フィールドの情報が表示されます。

   ここでは「ROOMRSDB」データベースの「TRESERVE」テーブルのすべてのフィールドを選択するよう、「TRESERVE」テーブルにチェックします。

   [OK]をクリックします。

7. Primary Keyとget/setメソッドの対象フィールドを選択し、[次へ]をクリックします。

型	フィールド名	Primary Key
String	resvid	チェックマーク付
String	resvdate	チェックマークなし
String	resvstart	チェックマークなし
String	resvend	チェックマークなし
String	resvcomment	チェックマークなし
String	userid	チェックマークなし
String	roomid	チェックマークなし

8. [ejbCreateメソッド定義]画面のメソッド一覧に「ejbCreate(RoomresvEJBCMPreserveRecord)」が表示されていることを確認し、[次へ]をクリックします。

9. [Finderメソッド定義]画面のメソッド一覧に「findByPrimaryKey(RoomresvEJBCMPreservePrimaryKey)」が表示されています。これに以下の2つのメソッドを追加します。

メソッド名	戻り値の型	パラメタリスト		検索条件
		変数名	型	
findAll	java.util.Collection			
findByDate	java.util.Collection	param1	String	SELECT OBJECT(r) FROM RoomresvEJBCMPreserve AS r WHERE r.resvdate = ?1

注) 検索条件は折り返さずに一行で記入してください。

追加の手順は以下のとおりです。

1. [追加]をクリックします。
2. メソッド名、戻り値の型、パラメタリスト、検索条件を指定します。

   [パラメタ追加]をクリックすると、パラメタリストに変数を追加することができます。

   入力後、[OK]をクリックします。

   表示内容を確認し、[次へ]をクリックします。

10. [Homeメソッド定義画面]のメソッド一覧が空欄であることを確認し、[次へ]をクリックします。
11. [ejbSelectメソッド定義]画面のメソッド一覧が空欄であることを確認し、[次へ]をクリックします。
12. [ビジネスメソッド定義]画面のメソッド一覧が空欄であることを確認し、[完了]をクリックします。

注意

RoomresvEJBCMPreserve_ROOMRSDB.dbschema 、 RoomresvEJBCMProom_ROOMRSDB.dbschema、RoomresvEJBCMPuser_ROOMRSDB.dbschemaファイルを作成する必要があります。

- データベースの種類がOracleの場合、EJBコンテナは配備時にデータベースにアクセスして、テーブルのスキーマ情報を自動的に取得します。プロジェクトのソースフォルダに自動的に生成されたdbschemaファイルを利用してください。
- データベースの種類がSymfowareの場合、以下のコマンドを使用して、このファイルを取得してください。

```
capture-schema -username ユーザ名 -password パスワード
 -dburl jdbc:symford://ホスト名:ポート番号/データベース名
 -driver com.fujitsu.symfoware.jdbc.SYMDriver -schemaname スキーマ名
 -table テーブル名 -out dbschemaファイル名
```

RoomresvEJBCMProomの作成

RoomresvEJBCMProomは、データベースの「TROOM」テーブルを使用するEntityBeanです。

RoomresvEJBCMPreserveの作成手順と同様に、作成してください。

手順の変更点は以下です。

- 手順6で選択するテーブルは、「TROOM」です。
- 手順9で追加するfinderメソッドは以下のとおりです。

メソッド名	戻り値の型	パラメタリスト		検索条件
		変数名	型	
findAll	java.util.Collection			

RoomresvEJBCMPuserの作成

RoomresvEJBCMPuserは、データベースの「TUSER」テーブルを使用するEntityBeanです。

RoomresvEJBCMPreserveの作成手順と同様に、作成してください。

手順の変更点は以下です。

- 手順6で選択するテーブルは、「TUSER」です。
- 手順9で追加するfinderメソッドはありません。

4.2.4 デプロイメント記述子の編集

Interstage Studioの[プロジェクトエクスプローラ]に表示されている、"デプロイメント記述子: roomresvEJB"をダブルクリックしてください。[XMLエディタ]が起動します。

[ソース]タブを選択し、デプロイメント記述子を編集します。

- <assembly-descriptor>タグの<ejb-name>が"RoomresvEJBCMPreserve"に対して、<trans-attribute>を [Mandatory]から[Required]に変更します。
- 同様に、以下の2つのEntity Beanについても、<trans-attribute>を[Required]に変更します。
  - RoomresvEJBCMProom
  - RoomresvEJBCMPuser

変更が完了したら、[ファイル]メニューの[保存]を選択し保存してください。

4.2.5 プロジェクトのビルド

以下の手順でプロジェクトをビルドします。

1. Interstage Studioの[プロジェクト]メニューの[自動的にビルド]にチェックが入っていない場合は[プロジェクトのビルド]を選択します。

   チェックが入っている場合はプロジェクト作成後、自動的にビルドされるため操作は不要です。

   注意

   問題ビューに警告が表示されますが、問題はありません。

4.3 Webアプリケーションの開発

4.3.1 プロジェクトの作成

以下の手順で Webアプリケーション(Apcoordinator)のプロジェクトを新規作成します。

1. [ファイル] > [新規]メニューの[プロジェクト]を選択します。

   [Apcoordinator] > [Webアプリケーションプロジェクト(Apcoordinator)]を選択し、[次へ]をクリックします。

2. [動的Webプロジェクト画面]では、以下を入力し、[次へ]をクリックします。

項目	設定内容	デフォルト設定
プロジェクト名	roomresvWeb	(なし)
デフォルトの使用	(デフォルト設定値)	チェックマーク付
ターゲットランタイム	(デフォルト設定値)	"Interstage Application Server V11.1 IJServer Cluster(Java EE)"が選択
動的Webモジュールバージョン	(デフォルト設定値)	2.5
構成	(デフォルト設定値)	"Apcoordinator Webプロジェクトのデフォルト構成"が選択
EARにプロジェクトを追加	(デフォルト設定値)	チェックマークなし

3. [Webモジュール]画面では、以下を入力し、[次へ]をクリックします。

項目	設定内容	デフォルト設定
コンテキストルート	(デフォルト設定値)	roomresvWeb
コンテンツフォルダ	(デフォルト設定値)	WebContent
Javaソースフォルダ	(デフォルト設定値)	src
Deployment Descriptorの生成	(デフォルト設定値)	チェックマーク付

4. [テンプレートを選択]画面では、[標準Apcoordinatorアプリケーション]を選択し、[次へ]をクリックします。

5. [制御ページ情報]画面では、以下を入力し、[次へ]をクリックします。

項目	設定内容	デフォルト設定
ファイル名	(デフォルト設定値)	main
タイトル	(デフォルト設定値)	roomresvWeb
エラーページを使用する	(デフォルト設定値)	チェックマーク付
エラーページファイル名	(デフォルト設定値)	roomresvWebError

6. [ファクトリ拡張情報]画面では、以下を入力し、[完了]をクリックします。

項目	設定内容	デフォルト設定
ファクトリクラスを拡張する	(デフォルト設定値)	チェックマーク付
パッケージ名	roomresv.web	roomresvWeb
ファクトリクラス	(デフォルト設定値)	RoomresvWebFactory
アプリケーションクラスを拡張する	(デフォルト設定値)	チェックマーク付
アプリケーションクラス	(デフォルト設定値)	RoomresvWebApplication
セションクラスを拡張する	(デフォルト設定値)	チェックマーク付
セション管理機能を使用する	(デフォルト設定値)	チェックマーク付
セションクラス	(デフォルト設定値)	RoomresvWebSession

7. Apcoordinatorのパースペクティブを開いていない場合は、確認のダイアログが表示されます。
   [はい]をクリックします。

4.3.2 データBeanの作成

1 データBeanの設計

本章のWebアプリケーションでは、以下の5つのデータBeanを設計します。

パッケージ名	クラス名	用途
roomresv.web	Head1Bean	ログイン画面のタイトル
	Head2Bean	予約状況参照画面のタイトル
	Head3Bean	予約登録・取消画面のタイトル
	Body1Bean	ログイン画面の入力域
	Body2Bean	予約状況の一覧表示域

それぞれのデータBeanは、以下のプロパティを持ちます。

パッケージ名	クラス名	項目名	プロパティ名	型
roomresv.web	Head1Bean	(なし)		
	Head2Bean	予約日	rsvdate	String
	Head3Bean	会議室名	romname	ComboBox
		開始時	bgnhour	ComboBox
		開始分	bgnmin	ComboBox
		終了時	endhour	ComboBox
		終了分	endmin	ComboBox
		氏名	tanname	String

パッケージ名	クラス名	項目名	プロパティ名	型
		連絡先	telnum	String
		用件	comment	String
		取消し予約番号	rsvid	String
	Body1Bean	今日の日付	curdate	String
		担当者コード	tancode	String
		パスワード	passwd	FieldString
	Body2Bean	予約日	vewdate	String
		予約状況	dsptable	TableView

2 データBeanの作成

以下の手順でデータBeanを作成し、項目を追加します。

1. [ファイル] > [新規]メニューの[データBean(Apcoordinator)]を選択します。
2. [データBeanクラス]画面では、プロジェクト名に[roomresvWeb]が選択されていることを確認し、パッケージ名およびクラス名を入力します。

   プロパティを追加するには、[追加]をクリックします。プロパティの追加ダイアログに、項目名、プロパティ名、型を入力し、[追加]をクリックします。

1)～2)の手順を繰り返し、設計した5つのデータBeanをそれぞれ作成します。

4.3.3 入出力ページの作成

1 入出力ページの設計

データBeanから入出力ページを作成します。入出力ページに配置する「送信ボタン」も同時に生成することができます。

本章のWebアプリケーションでは、以下のように入出力ページと送信ボタンを設計します。

ファイル名	入力データ	データBean	表示領域名	送信ボタン	
				コマンド	ラベル
Head1	データBean	roomresv.web.Head1Bean	head		
Head2	データBean	roomresv.web.Head2Bean	head	exView	予約状況更新
				exReserve	予約画面へ
Head3	データBean	roomresv.web.Head3Bean	head	exMenu	メニューへ
				exaddRecord	登録
				exdelRecord	取消
Body1	データBean	roomresv.web.Body1Bean	body	exLogin	LOGIN
Body2	データBean	roomresv.web.Body2Bean	body	exBackward	前日へ
				exForward	翌日へ

2 入出力ページの作成

以下の手順で入出力ページを作成します。

1. [ファイル] > [新規]メニューの[入出力ページ(Apcoordinator)]を選択します。
2. プロジェクト名に[roomresvWeb]が選択されていることを確認し入出力ページの設計に基づいて、設定します。

3. 送信ボタンは、[追加]をクリックして入力します。

   コマンド(ボタン名)とラベルを入力し、[追加]をクリックします。

   全てのボタンを追加したら、[閉じる]をクリックします。

1. 入出力ページ(Apcoordinator)生成ウィザード画面で[完了]をクリックします。

1)～4)の手順を繰り返し、設計した5つの入出力ページをそれぞれ作成します。

3 入出力ページの編集

“2 入出力ページの作成”で作成されたファイルは、以下の5つになります。

- Head1.jsp
- Head2.jsp
- Head3.jsp
- Body1.jsp
- Body2.jsp

それぞれの.jspファイルに対し、画面の見栄えを整えます。

完成されたファイルが以下のフォルダにあります。参考にしてファイルを編集してください。

「C:¥Interstage¥APC¥sample¥javaee¥roomresv¥roomresvWeb¥WebContent¥」

4.3.4 データ送受信のためのクラスの作成

EJB側の「EJBクライアントに対して送受信するデータの設計」で以下のようにデータを設計しました。

HashMapのキー値	クライアントでのデータ型
userID	String
userPass	String
userName	String
userPhone	String
resvID	String
resvDate	String
resvStart	String
resvEnd	String
resvComment	String
resvTable	TableView
roomID	String
roomTable	TableView

このチュートリアルのWebアプリケーションでは、この送受信されるデータを管理するためにクラスを作成します。

1. [ファイル] > [新規]メニューの[クラス]を選択します。

   [Javaクラス]画面では、以下を入力します。[完了]をクリックするとJavaソースが生成されます。

項目	設定内容	デフォルト設定
ソースフォルダ	roomresvWeb/src	(なし)
パッケージ	roomresv.web	(なし)
エンクロージング型	(デフォルト設定値)	チェックマークなし

項目	設定内容	デフォルト設定
名前	UserRecord	(なし)
修飾子	(デフォルト設定値)	"public"が選択
abstract	(デフォルト設定値)	チェックマークなし
final	(デフォルト設定値)	チェックマークなし
static	(デフォルト設定値)	チェックマークなし
スーパークラス	(デフォルト設定値)	java.lang.Object
インタフェース	(デフォルト設定値)	(なし)
public static void main(String[] args)	(デフォルト設定値)	チェックマークなし
スーパークラスからコンストラクタ	(デフォルト設定値)	チェックマークなし
敬称された抽象メソッド	チェックマークなし	チェックマーク付
コメントの生成	(デフォルト設定値)	チェックマークなし

2. 作成された「UserRecord.java」ファイルに対し、データの定義をします。

完成されたファイルが以下のフォルダにあります。参考にしてファイルを編集してください。

「C:¥Interstage¥APC¥sample¥javaee¥roomresv¥roomresvWeb¥src¥roomresv¥web¥UserRecord.java」

4.3.5 ビジネスクラスの作成

1 ビジネスクラスの設計

入出力ページで定義した「送信ボタン」のコマンドに対応して処理されるメソッドを作成します。

このチュートリアルのWebアプリケーションでは、以下のようにメソッドを設計します。

入力データ種別	クラス名	コマンド	メソッド	復帰値型
入力データなし			startup	void
データBean	roomresv.web.Head2Bean	exView	exView	void
データBean	roomresv.web.Head2Bean	exReserve	exReserve	void
データBean	roomresv.web.Head3Bean	exMenu	exMenu	void
データBean	roomresv.web.Head3Bean	exaddRecord	exaddRecord	void
データBean	roomresv.web.Head3Bean	exdelRecord	exdelRecord	void
データBean	roomresv.web.Body1Bean	exLogin	exLogin	void
データBean	roomresv.web.Body2Bean	exBackward	exBackward	void
データBean	roomresv.web.Body2Bean	exForward	exForward	void

2 ビジネスクラスの作成

以下の手順でビジネスクラスを作成します。

1. [ファイル] > [新規]メニューの[ビジネスクラス(Apcoordinator)]を選択します。
2. プロジェクト名に[roomresvWeb]が選択されていることを確認し、パッケージ名およびクラス名を入力します。

項目	設定内容	デフォルト設定
プロジェクト名	(デフォルト設定値)	roomresvWeb
パッケージ名	(デフォルト設定値)	roomresv.web
クラス名	Handler	(なし)

項目	設定内容	デフォルト設定
セションスコープとして生成	(デフォルト設定値)	チェックマーク付
コマンドスクリプティングとして生成	(デフォルト設定値)	チェックマークなし

メソッドを追加するには、[追加]をクリックします。

3. 全てのメソッドを入力したら、[完了]をクリックします。

3 ビジネスクラスの編集

“2 ビジネスクラスの作成”で作成されたファイルは「Handler.java」です。

このファイルのメソッドに対して、ビジネスロジックを記述します。

完成されたファイルが以下のフォルダにあります。参考にしてファイルを編集してください。

「C:¥Interstage¥APC¥sample¥javaee¥roomresv¥roomresvWeb¥src¥roomresv¥web¥Handler.java」

4.3.6 リモートマップの作成

以下の手順でリモートマップを作成します。

1. [ファイル] > [新規]メニューの[その他]を選択します。

   [Apcoordinator] > [リモートマップ(Apcoordinator)]を選択し、[次へ]をクリックします。

2. 表示内容を確認し、[完了]をクリックします。

項目	表示内容
プロジェクト名	roomresvWeb
ファイル名	remote

3. 作成された"remote.xml"が開きます。

   以下のように、"ejb"タグを入力します。"ejb"タグのパラメタは以下のようになります。

   - nameはWebアプリケーションからEJBを呼び出す時に使用する名前
   - applicationは対応するEJBの名前

```
<?xml version="1.0" encoding="windows-31j"?>
<remoteMap
    xmlns="http://interstage.fujitsu.com/schemas/uji/remoteMap"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://interstage.fujitsu.com/schemas/uji/remoteMap
                        http://interstage.fujitsu.com/schemas/uji/remotemap.xsd">
    <config>
        <version>5.0</version>
    </config>

    <!-- 以下は記述例です。-->

    <!-- EJBの呼び出しを定義します。-->
    <!--
    <ejb name="RoomresvEJB"
        application="java:comp/env/ejb/ RoomresvEJB "/>
    -->

</remoteMap>
```

4.3.7 データBean変換マップの作成

以下の手順でデータBean変換マップを作成します。

1. [ファイル] > [新規]メニューの[その他]を選択します。

   [Apcoordinator] > [データBean変換マップ(Apcoordinator)]を選択し、[次へ]をクリックします。

2. 表示内容を確認し、[完了]をクリックします。

項目	表示内容
プロジェクト名	roomresvWeb
ファイル名	conv

3. 作成された「conv.xml」が開きます。

   データ変換を定義します。

   完成されたファイルが以下のフォルダにあります。参考にしてファイルを編集してください。

   「C:¥Interstage¥APC¥sample¥javaee¥roomresv¥roomresvWeb¥WebContent¥WEB-INF¥conv.xml」

4.3.8 プロジェクトのビルド

以下の手順でプロジェクトをビルドします。

1. Interstage Studioの[プロジェクト]メニューの[自動的にビルド]にチェックが入っていない場合は[プロジェクトのビルド]を選択します。

   チェックが入っている場合はプロジェクト作成後、自動的にビルドされるため操作は不要です。

   注意

   問題ビューに警告が表示されますが、問題はありません。

4.4 アプリケーションサーバでの実行準備

4.4.1 Symfowareへのテーブルの登録

1 テーブルの設計

このWebアプリケーションで使用するデータベースのテーブルを設計します。

テーブルは以下の3つです。

DATABASE	SCHEMA	TABLE	FIELD	
ROOMRSDB	ROOMRSDB	TRESERVE	RESVID	CHAR(8)
			RESVDATE	CHAR(8)
			RESVSTART	CHAR(5)
			RESVEND	CHAR(5)
			RESVCOMMENT	VARCHAR(256)
			USERID	CHAR(8)
			ROOMID	CHAR(6)
		TUSER	USERID	CHAR(8)
			USERPASS	CHAR(32)
			USERNAME	VARCHAR(32)
			USERPHONE	CHAR(9)

DATABASE	SCHEMA	TABLE	FIELD	
		TROOM	ROOMID	CHAR(6)
			ROOMNAME	VARCHAR(16)
			ROOMFLOOR	CHAR(4)

注意

このチュートリアルの手順で「JDBCデータソースの登録」および、データベースに「テーブルを登録」するには、Symfoware JDBCのネーミングサービスが起動されている必要があります。

Symfoware JDBCのネーミングサービスを起動する手順は、以下に記載されています。

“4.5.1 Symfowareサービスの起動”

以下の手順は、ネーミングサービスが起動したあとで実施してください。

2 JDBCデータソースの登録

JDBCデータソースの登録は、開発環境・運用環境両方で必要です。

1. スタートメニューの「ファイル名を指定して実行」から以下のコマンドを実行します。

```
java com.fujitsu.symfoware.jdbc2.tool.FJJdbcTool <host> <port_no>
```

<host>

- ネーミングサービスを運用しているホスト名またはIPアドレスを指定します。
- 別のマシン上で動作しているネーミングサービスに対して操作を行う場合に指定します。
- 省略した場合はlocalhostとなります。

<port_no>

- ネーミングサービスのポート番号を指定します。
- ネーミングサービスの起動時に指定したポート番号を指定してください。

「JDBCデータソース登録ツール」が起動したら、「追加」をクリックします。

2. データソースの定義を入力します。

「JDBCデータソース情報設定」が起動します。以下を入力します。

項目	設定内容
データソース名	roomresvCMP
データ資源名	ROOMRSDB

以下の情報はお使いのSymfoware(R)Serverの設定に合わせて変更してください。

- ホスト名
- ポート番号
- ユーザー名
- パスワード

入力後、「閉じる」をクリックします。

3 テーブルの登録

本章の手順では以下の作業をSymfoware(R) Serverがインストールされたサーバ上で行ってください。

“1 テーブルの設計”で設計したテーブルをデータベースに構築し、初期データを登録します。

Windows向けには、テーブルの構築と初期データの登録を行うバッチファイルがチュートリアルに用意されています。

「C:¥Interstage¥APC¥sample¥javaee¥roomresv¥database¥Symfoware¥CreateTable¥」

- PrepareTable.bat (テーブルの構築と初期データの登録を行うバッチファイル)
- PrepareTable.sql (テーブルの構築を行うSQL文)
- dreserveTable.dat (TRESERVEテーブル初期データ)
- droomTable.dat (TROOMテーブル初期データ)
- duserTable.dat (TUSERテーブル初期データ)

バッチファイルは、以下の情報をお使いの環境に合わせて変更することで使用することができます。

- PrepareTable.sql
  以下の文のDBファイルの位置を修正してください。

  CREATE DBSPACE ROOMRSDB_DBSP ALLOCATE FILE **C:¥sfwd¥RDB¥USR¥DBSP¥ROOMRSDB** ATTRIBUTE SPACE(5M);

- PrepareTable.bat
  以下の文のデータファイルの位置をお使いの環境に合わせて修正してください。

  rdbsloader -mi -i ROOMRSDB.#ROOMRSDB#TUSER -t
  C:¥Interstage¥APC¥sample¥javaee¥roomresv¥database¥Symfoware¥CreateTable¥duserTable.dat

  rdbsloader -mi -i ROOMRSDB.#ROOMRSDB#TROOM -t
  C:¥Interstage¥APC¥sample¥javaee¥roomresv¥database¥Symfoware¥CreateTable¥droomTable.dat

  rdbsloader -mi -i ROOMRSDB.#ROOMRSDB#TRESERVE -t
  C:¥Interstage¥APC¥sample¥javaee¥roomresv¥database¥Symfoware¥CreateTable¥dreserveTable.dat

修正が完了したらバッチファイル「PrepareTable.bat」を実行し、データベースにテーブルとデータを登録します。

4.4.2 JDBCデータソースの定義

Interstage Java EE管理コンソールを使い、JDBCのデータソースを定義します。

Interstage Java EE管理コンソール画面でユーザ名、パスワードを入力してログインします。

ポイント

ログイン時に入力するユーザ名は、Interstageを運用しているサーバのオペレーティングシステムに登録されているユーザ名です。以降の操作は管理者権限が必要なため、ここでは管理者権限のあるユーザ名でログインします。

1 JDBC接続プールの作成

Interstage Java EE管理コンソールを使い、JDBCの接続プールを定義します。

1. 画面左側のツリーから、[リソース]内の[JDBC]の[接続プール]を選択します。
2. [新規作成]タブを選択し、以下の情報を入力します。

項目	入力内容
名前	roomresvCMPPool
リソースタイプ	javax.sql.ConnectionPoolDataSource
データベースベンダー	Symfoware

3. [次へ]タブを選択し、以下の情報を入力します。

項目	入力内容
schema	スキーマ名
databaseName	データベース名
networkProtocol	symford
serverName	ホスト名
portNumber	ポート番号
user	ユーザ名
password	パスワード

4. 入力後、[作成]をクリックします。

2 JDBCリソースの作成

1. 画面左側のツリーから、[リソース]内の[JDBC]の[JDBC リソース]を選択します。
2. [新規]タブを選択し、以下の情報を入力します。

項目	入力内容
JNDI名	jdbc/roomresvCMP
プール名	roomresvCMPPool

3. 入力後、[了解]をクリックします。

4.4.3 EARファイルの作成

EARファイルはejb-jarファイルとWARファイルをまとめたファイルです。開発環境上で以下の手順で作成します。

Interstage Studioでは、EARファイルを作成するためのエンタープライズアプリケーションプロジェクトウィザードを提供しています。作成したエンタープライズアプリケーションプロジェクトをビルドしてEARファイルを作成します。

1. Interstage Studioのメニューから、[ファイル] > [新規] > [プロジェクト]を選択します。

   [Java EE] > [エンタープライズアプリケーションプロジェクト]を選択し、[次へ]をクリックします。

2. [EARアプリケーションプロジェクト]画面では、以下を入力し、[次へ]をクリックします。

項目	設定内容	デフォルト設定
プロジェクト	roomresv	(なし)
デフォルトの使用	(デフォルト設定値)	チェックマーク付
ターゲットランタイム	(デフォルト設定値)	"Interstage Application Server V11.1 IJServer Cluster(Java EE)"が選択
EARバージョン	(デフォルト設定値)	"5.0"が選択
構成	(デフォルト設定値)	"Interstage Application Server V11.1 IJServer Cluster(Java EE)デフォルト構成"が選択

3. [エンタープライズアプリケーション]画面では、以下を入力し、[完了]をクリックします。

項目	設定内容	デフォルト設定
roomresvEJB	チェックマーク付	チェックマークなし
roomresvWeb	チェックマーク付	チェックマークなし
コンテンツフォルダ	(デフォルト設定値)	EarContent
Deployment Descriptorの生成	チェックマーク付	チェックマークなし

4. 作成したプロジェクトを以下の手順でビルドします。

   - Interstage Studioの[プロジェクト]メニューの[自動的にビルド]にチェックが入っていない場合は[プロジェクトのビルド]を選択します。
   - チェックが入っている場合はプロジェクト作成後、自動的にビルドされるため操作は不要です。

   ビルドの完了後、Interstage Studioの[ファイル]メニューの[エクスポート]からEARファイルを作成してください。

注意

問題ビューに警告が表示されますが、問題はありません。

4.4.4 Webサーバへの配備

運用環境に配布するには、EARファイルを作成する必要があります。EARファイルを配備することにより、EJBアプリケーションとWebアプリケーションをインストールします。作成したEARファイルは、Interstage Java EE管理コンソールの配備機能を利用して、サーバに配備します。

参照

運用環境へのEARファイルの配布は、「Interstage Studio ユーザーズガイド」を参照ください。

ポイント

Interstage Java EE管理コンソールを使用して以下を追加します。

[設定] > [クラスタ名-config] または [server-config] > [JVM 設定] > [パス設定]タブ

Windowsの場合

項目	入力内容
クラスパスのサフィックス	C:¥Interstage¥APC¥lib¥uji.jar C:¥Interstage¥APC¥lib¥ujiejb.jar C:¥SFWCLNT¥JDBC¥fjjdbc¥lib¥fjsymjdbc2.jar
ネイティブライブラリパスのサフィックス	C:¥SFWCLNT¥JDBC¥fjjdbc¥bin

Solaris,Linuxの場合

項目	入力内容
クラスパスのサフィックス	/opt/FJSVwebc/lib/uji.jar /opt/FJSVbcco/lib/ujiejb.jar /opt/FJSVsymjd/fjjdbc/lib/fjsymjdbc2.jar
ネイティブライブラリパスのサフィックス	/opt/FJSVsymjd/fjjdbc/bin

また、JDBCリソースの[roomresvCMP]の[選択したターゲット]に「roomresv」を追加します。

4.5 アプリケーションサーバでの実行と、ブラウザによる実行確認

4.5.1 Symfowareサービスの起動

以下の手順でSymfoware(R) Serverデータベースにアクセスできるようにします。

なお、各手順の詳細については、お使いのSymfoware(R) Serverのドキュメントをご参照ください。

1. Symfoware Server RDBサービスの起動

   Symfoware Serverのプログラムフォルダから「RDB 起動・停止」ツールを起動します。

   Symfoware/RDB Service Controllerの「起動」をクリックします。

   サービスが動作中になったことを確認します。

2. Symfoware RDAサービスの起動

   コントロールパネルのサービスから「Symfoware RDA」を選択します。

   「開始」をクリックし、Symfoware RDAサービスを開始します。

3. Symfoware JDBCネーミングサービスの起動

   スタートメニューの[ファイル名を指定して実行]から以下のコマンドを実行します。

```
java com.fujitsu.symfoware.jdbc2.naming.SYMNameService
```

開いたコマンドプロンプトウインドウは閉じずにそのままにしておきます。

4.5.2 ワークユニットの起動

1. Interstage Java EE管理コンソールにログインし、Interstageのシステム状態が「起動」になっていることを確認します。Interstageのシステム状態が「停止」になっている場合は、[起動]をクリックします。
2. 対象のIJServerクラスタ名を選択し、状態が「起動」になっていることを確認します。「停止」になっている場合は、[起動]をクリックします。

4.5.3 ブラウザによる実行確認

以下の手順でWebブラウザから動作を確認します。

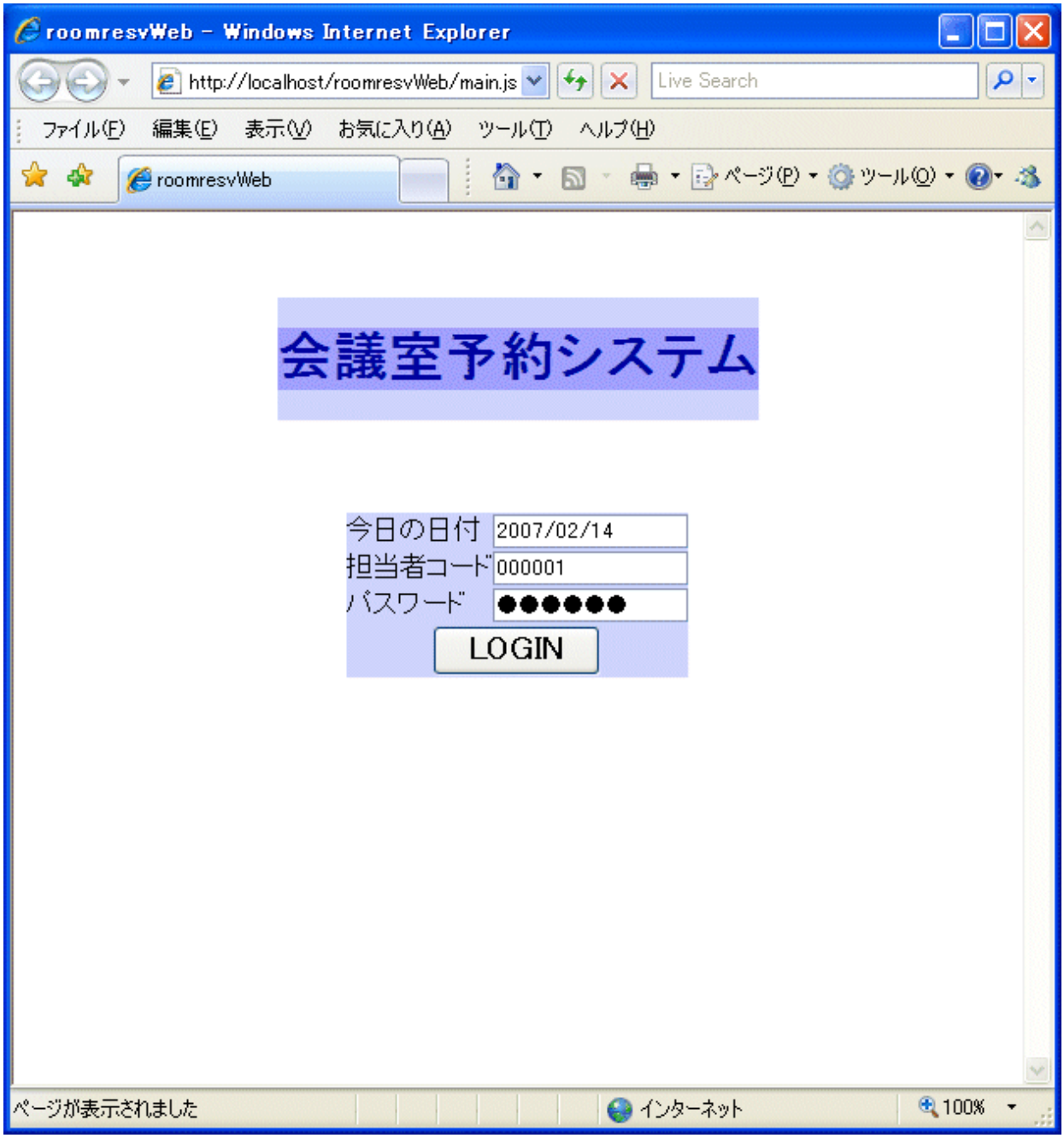
Webブラウザを起動し、以下のURLを入力します。

```
http://Webサーバの名前/roomresvWeb/main.jsp
```

例: Webサーバの名前が "www.tutorialserver.domain"の場合

```
http://www.tutorialserver.domain/roomresvWeb/main.jsp
```

1. 起動画面



以下の値を入力し、"LOGIN"をクリックします。

- 担当者コード: 000001
- パスワード: passwd

2. 全予約状況の表示

すべての予約が表示されています。

予約日に予約したい日を入力し、"予約画面へ"をクリックします。

例:2002/09/10

3. 指定日の予約状況の表示

指定日の予約状況が表示されています。

会議室名,開始時,終了時を入力します。

氏名,連絡先を確認し、用件を入力します。

「登録」をクリックすると、予約が入力されます。

予約を取り消す場合は、予約番号を取り消し予約番号へ入力し、「取消」をクリックします。

付録A　サンプル集

Apcoodinatorでは、さまざまなサンプルを用意しています。本付録では、どのようなサンプルがあるかを示します。

A.1　サンプルについて

A.1.1　サンプルの格納場所

サンプルが格納されているフォルダは以下のとおりです。

1. Interstage Application Serverの場合
2. 1)以外の場合

1) Interstage Application Serverの場合

サンプルのファイルは圧縮されて提供されます。以下のように展開して使用してください。

1. Interstageオンラインマニュアル CD-ROMの以下のファイルを展開先のフォルダにコピーします。

   ```
   ¥Framework¥sample.exe
   ```

2. 展開先にコピーした「sample.exe」を実行します。展開先にsampleフォルダが作成され、その中にサンプルのファイルが展開されます。

各サンプルのプロジェクトがそれぞれsampleフォルダ内に以下のように展開されます。

```
sample¥javaee¥［プロジェクト名］¥
```

2) 1)以外の場合

サンプルは、次のいずれかの条件の場合にインストールされます。

- Interstage Business Application Serverで「サンプルプロジェクト」をインストールした場合

  (開発環境パッケージを標準インストールした場合はインストールされています。)

- Interstage Studioで「フレームワーク」をインストールした場合
- Interstage Interaction Managerの開発環境パッケージでApcoordinatorをインストールした場合

```
［製品のインストールフォルダ］¥APC¥sample¥javaee¥［プロジェクト名］¥
```

A.1.2　サンプルの利用方法

各サンプルはInterstage Studioのプロジェクト形式で格納されています。Interstage Studioでは以下の手順で使用できます。

1. Interstage Studioを起動します。起動時の環境設定でサンプル用のワークスペースを指定し、そのワークスペースでInterstage Studioを起動します。
2. 各サンプルの注意事項が、プロジェクト名.txtファイルに記述されていますのでご覧ください。
3. Interstage Studioのメニューで[ファイル] > [インポート]を選択します。
4. [インポート]ダイアログボックスで、[一般] > [既存プロジェクトをワークスペースへ]を選択し、[次へ]をクリックします。
5. [ルートフォルダの選択]にサンプルアプリケーションの格納場所を指定し、[プロジェクトをワークスペースにコピー]をチェックして、インポートを実行します。
6. Interstage Studioでプロジェクトをビルドし、デバッグ実行をするとサンプルアプリケーションの動作を確認することができます。

注意

サンプルのビルド時に警告が表示されますが、問題はありません。

A.1.3 コンパイル済みサンプルについて

アプリケーションサーバのサンプル統合環境がインストールされている場合は、以下のサンプルがコンパイル済みの形式で提供されます。(*)

サンプル	格納場所	ファイル名
meetingroom	(Windows版の場合) ［製品のインストールフォルダ］¥sample¥integrate¥ja¥framework¥meetingroom (Solaris版、Linux版の場合) /opt/FJSVisspl/integrate/ja/framework/meetingroom	meetingroom.war
ejbOffice	(Windows版の場合) ［製品のインストールフォルダ］¥sample¥integrate¥ja¥framework¥ejboffice (Solaris版、Linux版の場合) /opt/FJSVisspl/integrate/ja/framework/ejboffice	ejbOffice.ear

サンプル統合環境は、標準インストールでインストールされます。カスタムインストールでは選択してインストールすることが可能です。

(*) 以下の製品で提供されます。

- Interstage Business Application Server Standard Edition / Enterprise Edition
- Interstage Studio Standard-J Edition

注意

コンパイル済みサンプルは、J2EEのサンプルプログラムです。

A.2 基本的な構成のサンプル

基本的な構成のサンプルとして、以下のものがあります。

使用機能	プロジェクト名	エディション		
		(1)	(2)	(3)
inputタグによる入力、uji:getPropertyタグによる表示	sample	○	○	○
uji:fieldString、uji:fieldLong、uji:fieldTextタグによる入力、 uji:getProperty、uji:tableViewタグによる表示	board	○	○	○
ログ作成処理	logTest	○	○	○

表中の番号に対応する製品とエディションは以下のとおりです。

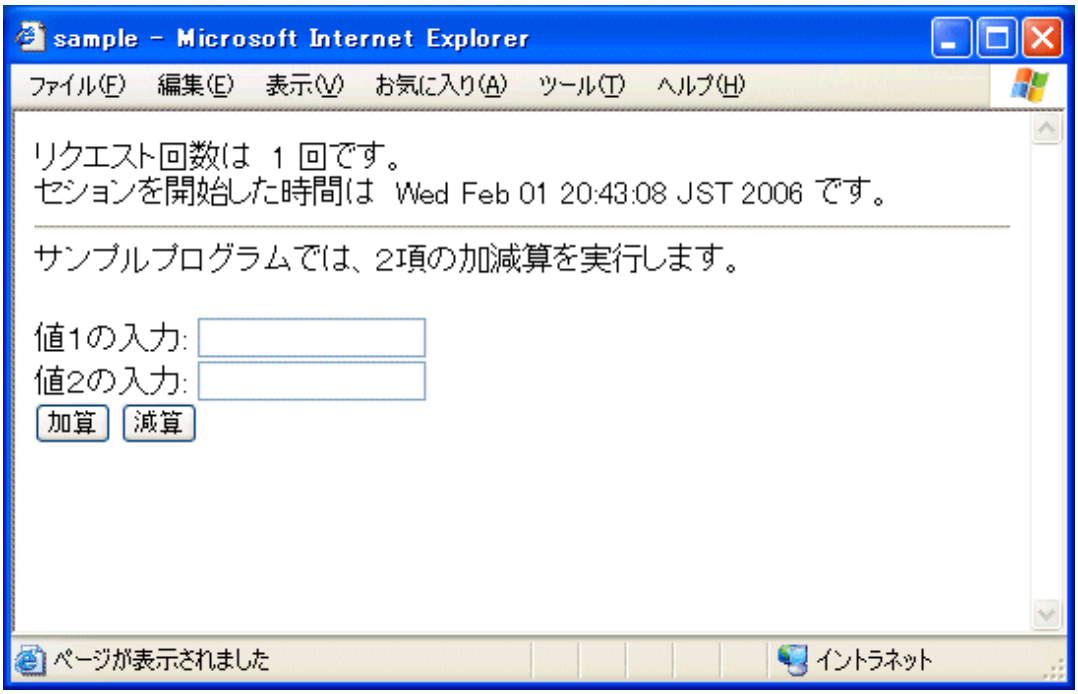
番号	製品名とエディション
(1)	Interstage Interaction Manager
(2)	Interstage Studio Standard-J Edition
(3)	Interstage Business Application Server Standard Edition Interstage Business Application Server Enterprise Edition

sampleプロジェクト

“Apcoordinatorユーザーズガイド”の付録A.1のサンプルです。Apcoordinatorの基礎を使用しています。

使用機能は以下のとおりです。

- inputタグによる入力
- uji:getPropertyタグによる表示



boardプロジェクト

“第3章 Webアプリケーションの設計と作成”で解説している行動予定表サンプルです。

使用機能は以下のとおりです。

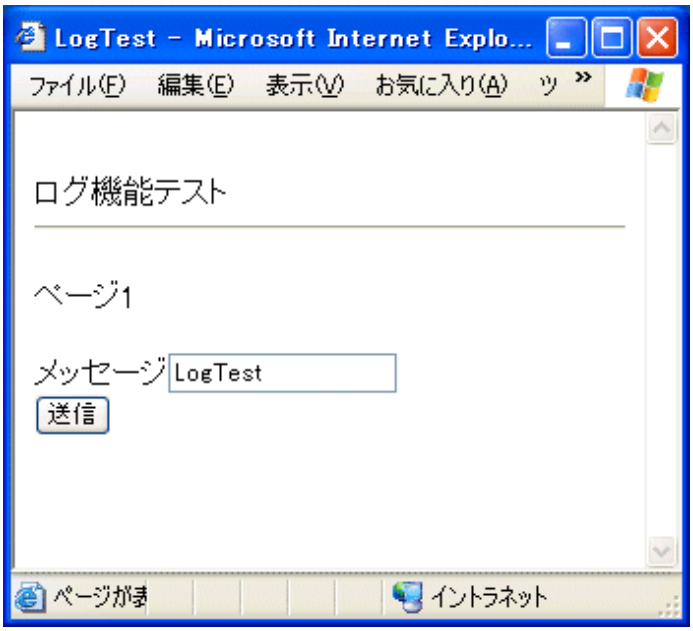
- uji:fieldString
- uji:fieldLong
- uji:fieldTextタグによる入力
- uji:getProperty
- uji:tableViewタグによる表示

logTestプロジェクト

アプリケーションログの採取を制御するサンプルです。

使用機能は以下のとおりです。

- ログ作成処理



A.3 中級的なサンプル

中級的な構成のサンプルとして、以下のものがあります。

使用機能	プロジェクト名	エディション		
		(1)	(2)	(3)
バイナリファイルの送受信	updown	○	○	○
サブウィンドウの利用	subwin	○	○	○
フレームの利用	frame	○	○	○
EJBの利用(1)	ejbOffice	-	○	○
EJBの利用(2)	roomresv	-	○	○
XMLデータ仕様記述の利用(1)	carrentc	○	○	○
XMLデータ仕様記述の利用(2)	eSpecCalcc	○	○	○
アプレット連携のアプリケーション	applet	○	○	○
コマンドスクリプティングの利用	commandMap	○	○	○
リソースファイルの利用	meetingroom	○	○	○

表中の番号に対応する製品とエディションは以下のとおりです。

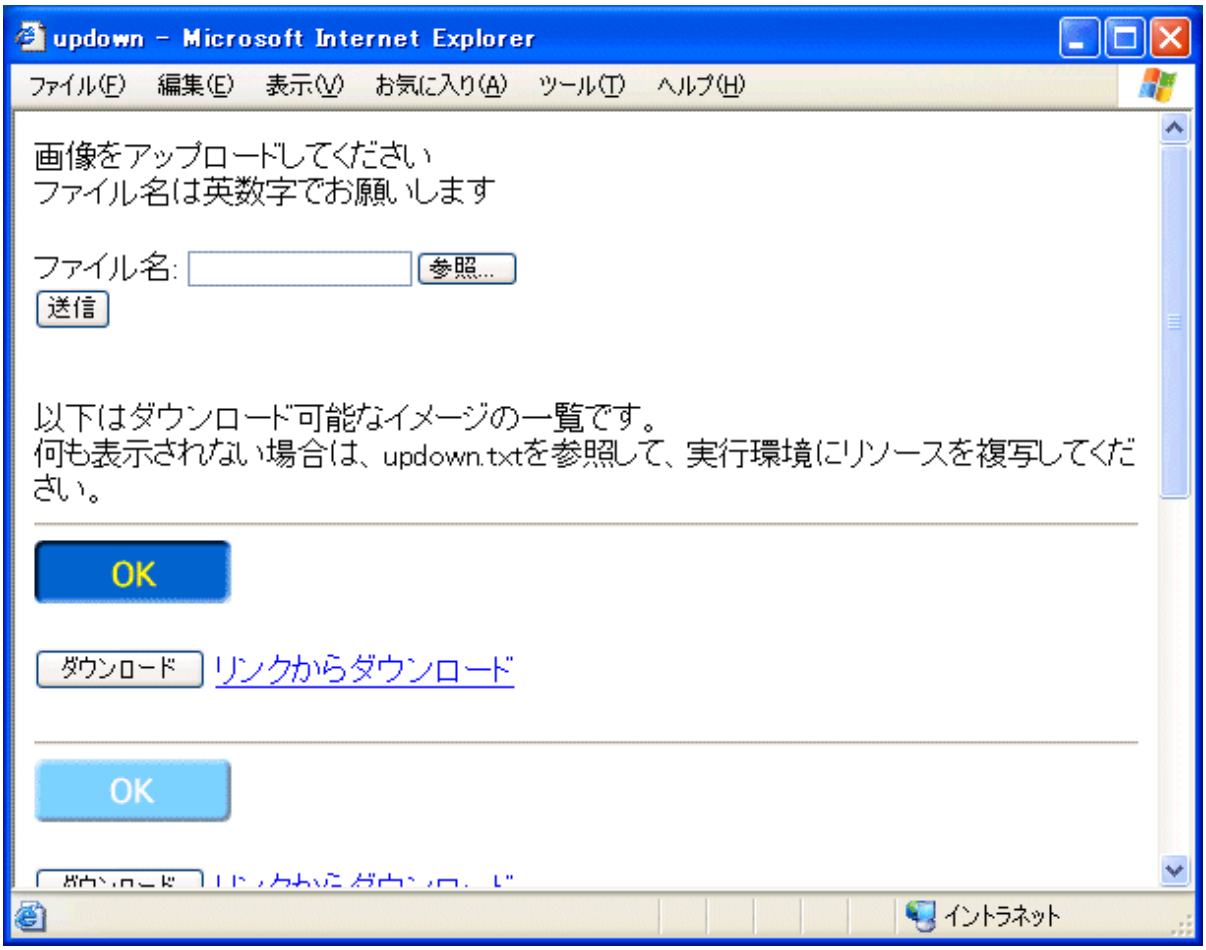
番号	製品名とエディション
(1)	Interstage Interaction Manager
(2)	Interstage Studio Standard-J Edition
(3)	Interstage Business Application Server Standard Edition Interstage Business Application Server Enterprise Edition

updownプロジェクト

ファイルのアップロード・ダウンロード処理のサンプルです。

使用機能は以下のとおりです。

- バイナリファイルの送受信

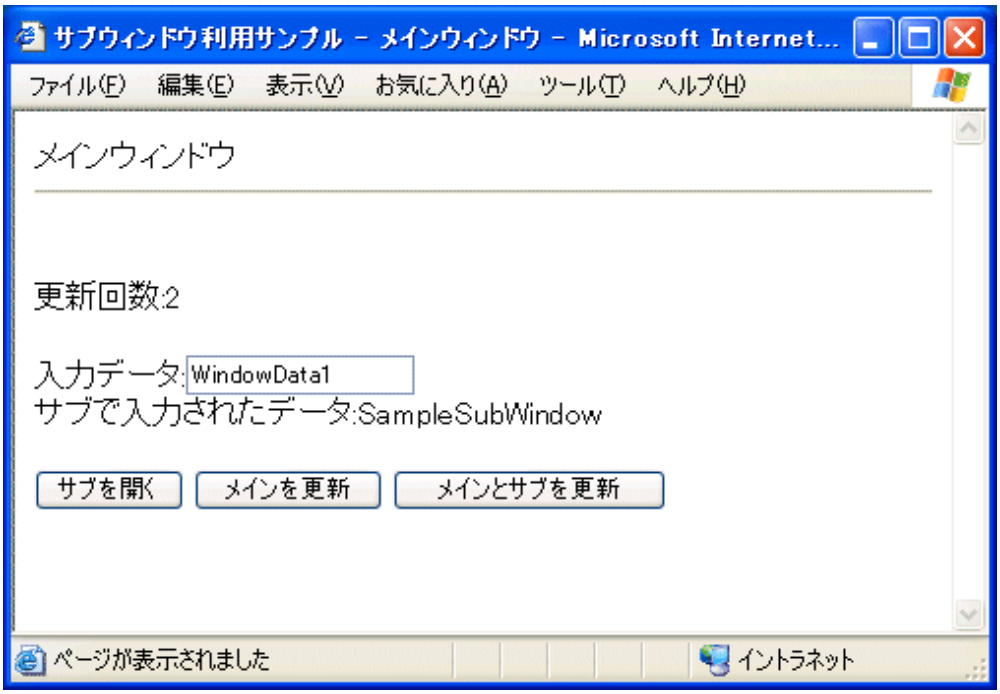


subwinプロジェクト

サブウィンドウを利用したアプリケーションのサンプルです。

使用機能は以下のとおりです。

- サブウィンドウの利用

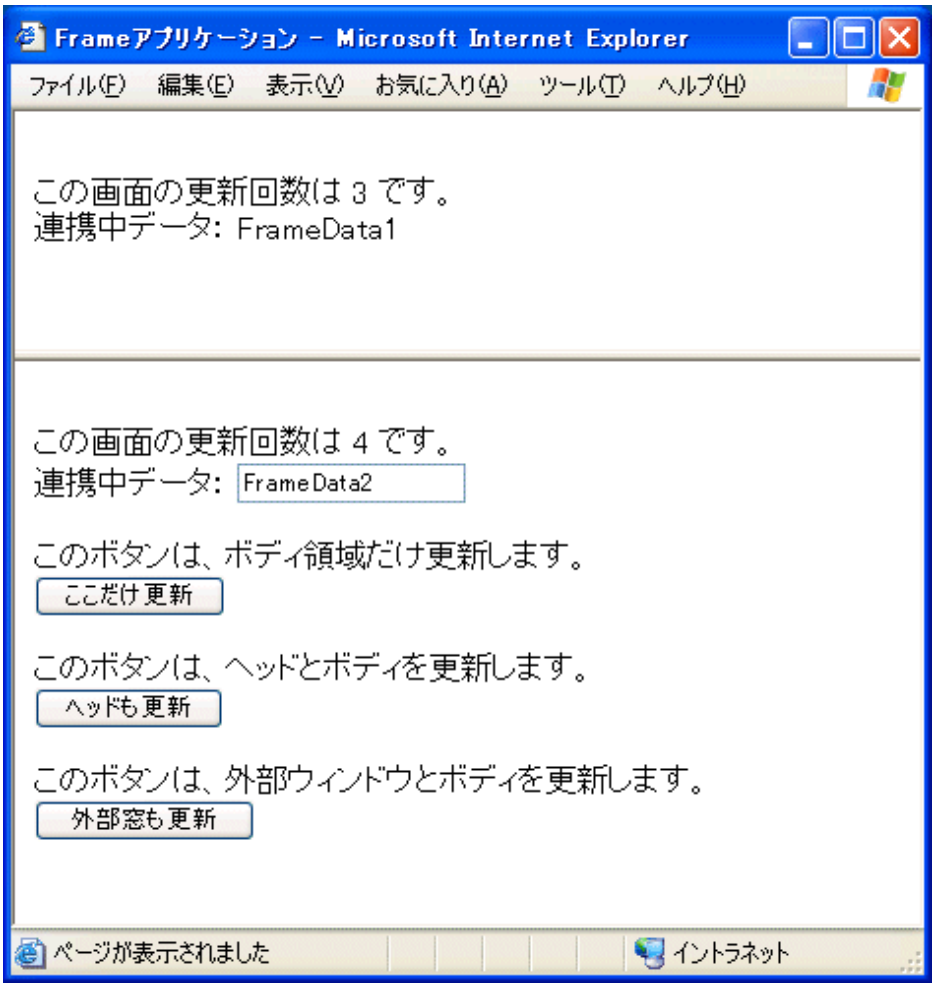


frameプロジェクト

フレームを使ったサンプルです。

使用機能は以下のとおりです。

- フレームの利用



ejbOfficeプロジェクト

EJBセションBeanの作成と呼び出しのサンプルです。

使用機能は以下のとおりです。

- EJBの利用(1)

roomresvプロジェクト

“第4章 EJBを利用したアプリケーションの設計と作成”で解説している会議室予約システムのサンプルです。

使用機能は以下のとおりです。

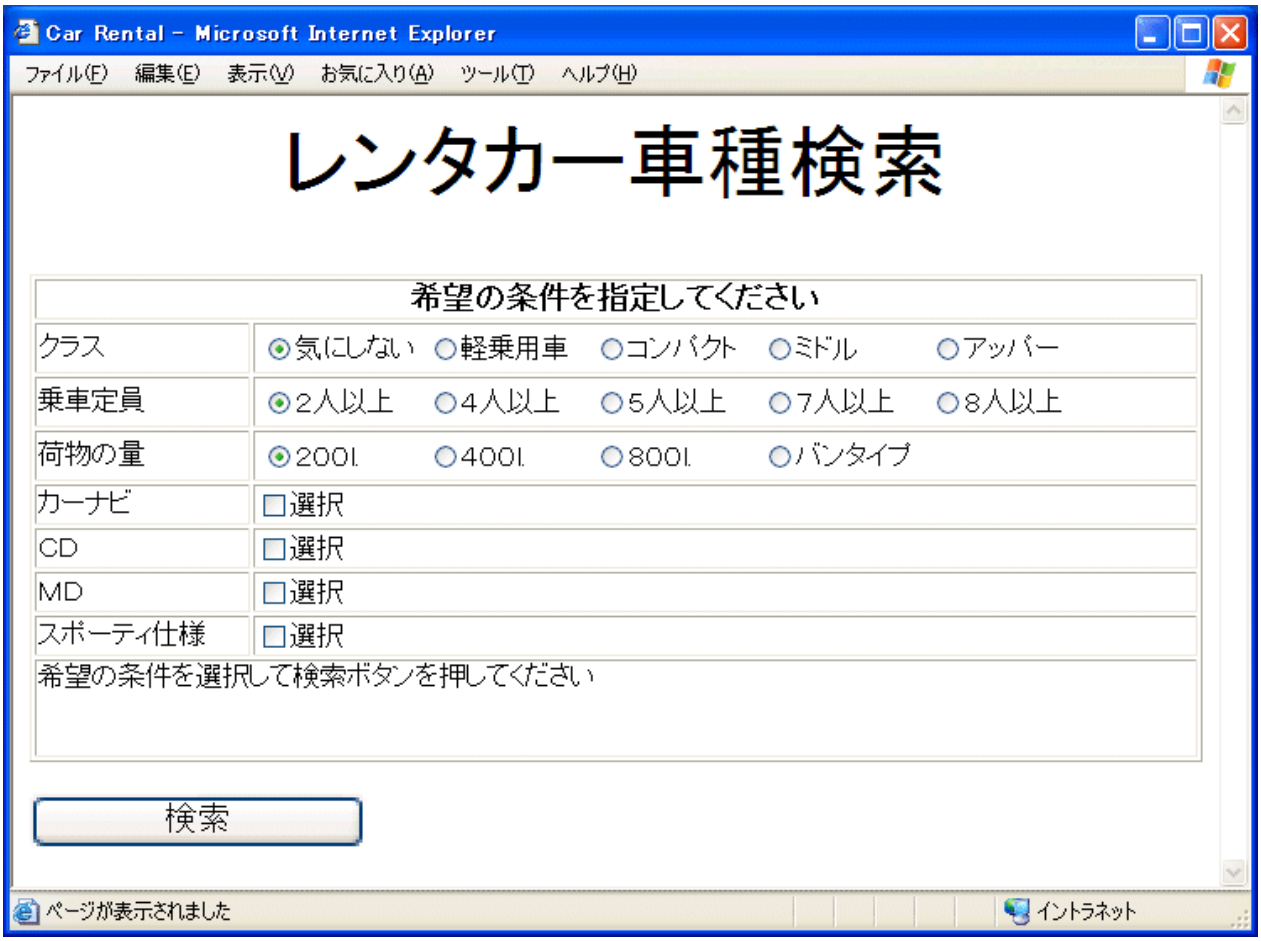
- EJBの利用(2)

carrentcプロジェクト

XMLデータ仕様記述の相関チェックサンプルです。(コンパイラ生成ソース使用版)

使用機能は以下のとおりです。

- XMLデータ仕様記述の利用(1)

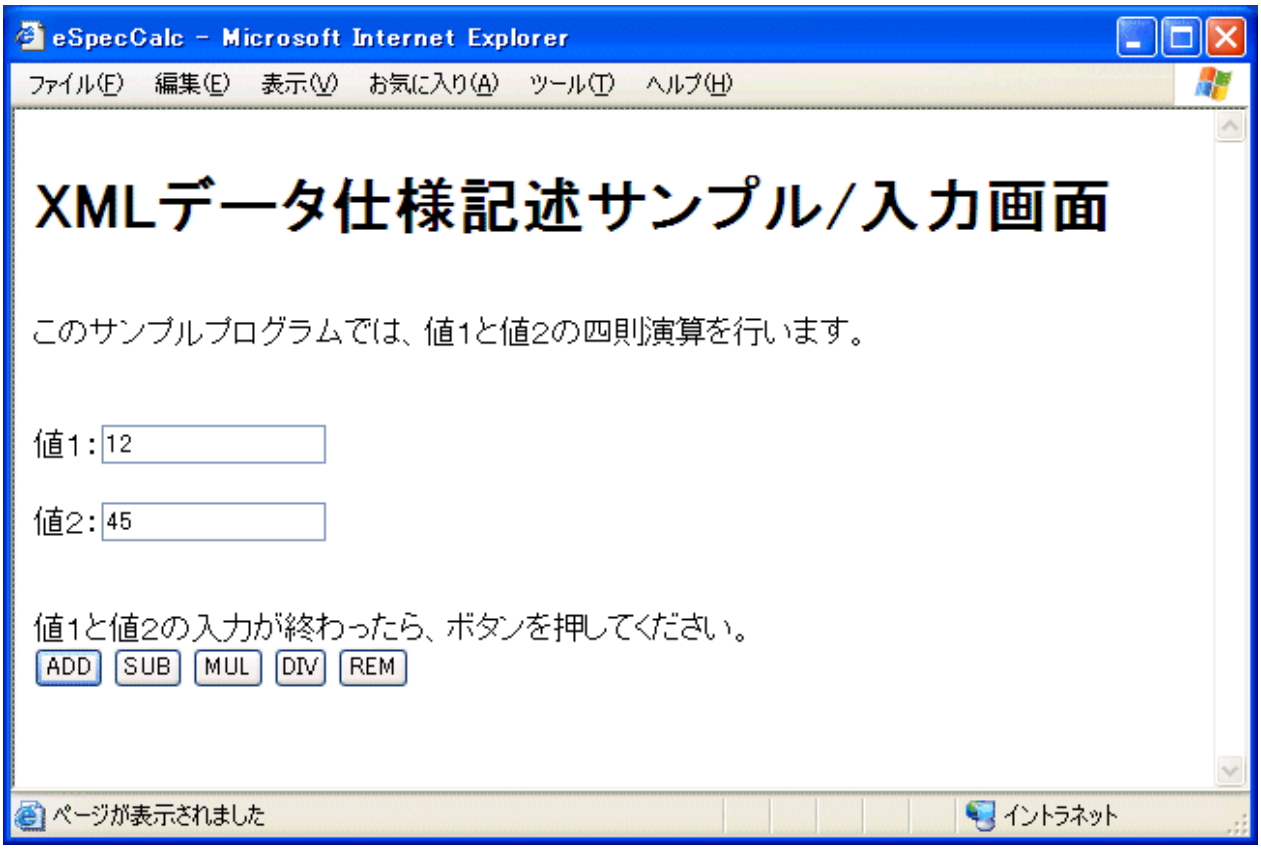


eSpecCalccプロジェクト

XMLデータ仕様記述を利用した四則演算のサンプルです。(コンパイラ生成ソース使用版)

使用機能は以下のとおりです。

- XMLデータ仕様記述の利用(2)

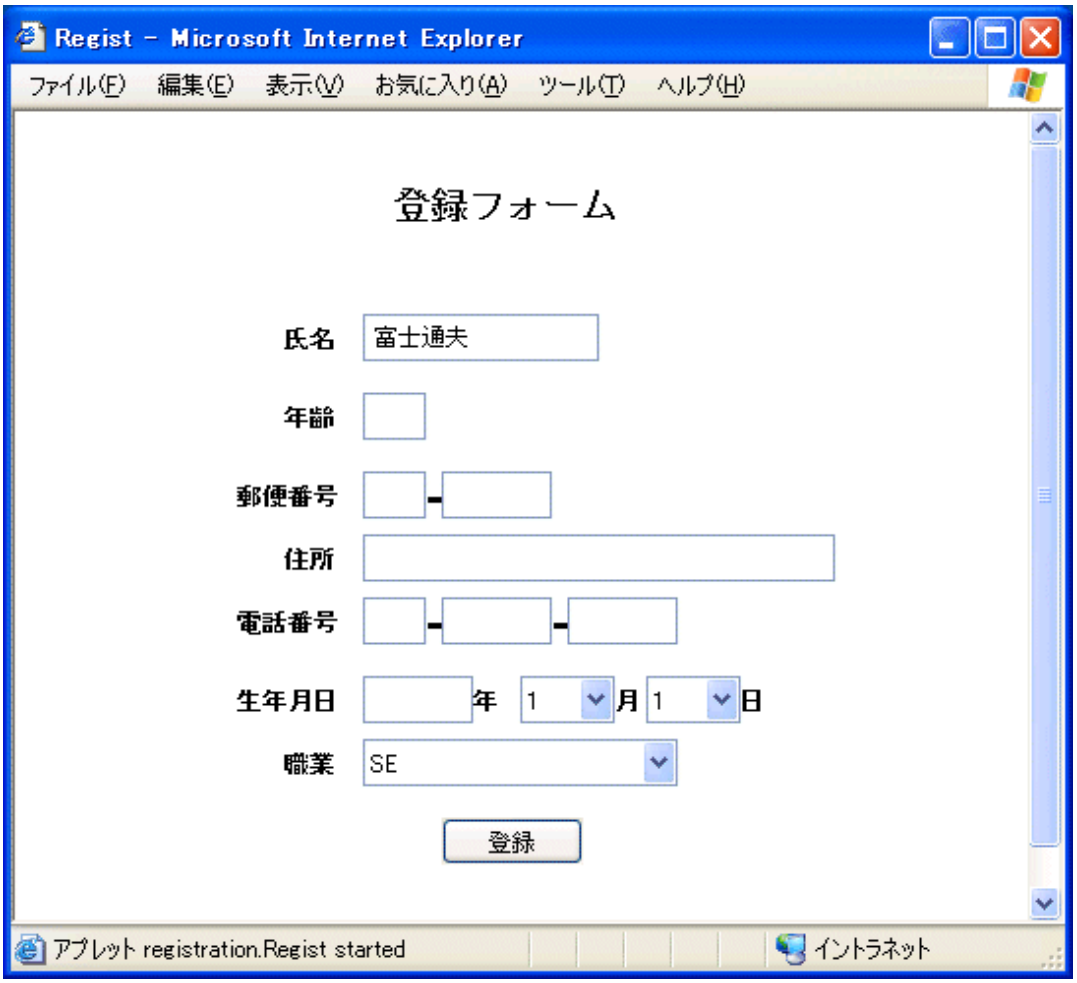


appletプロジェクト

アプレット連携機能を使用した、ユーザ登録のサンプルです。

使用機能は以下のとおりです。

- アプレット連携のアプリケーション

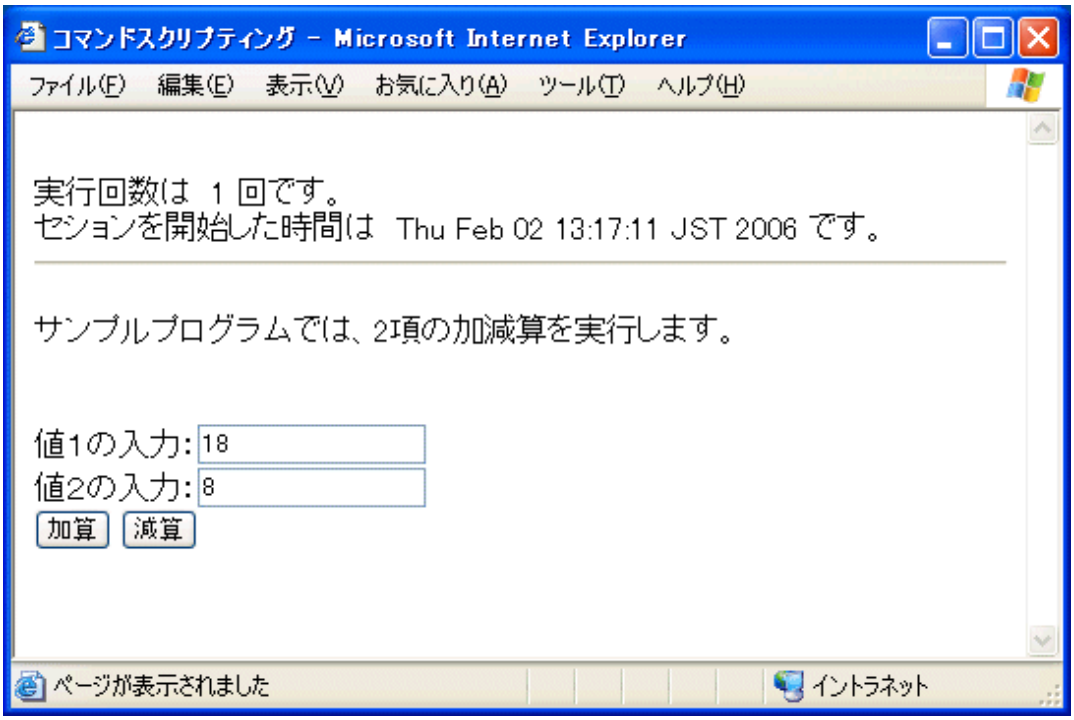


commandMapプロジェクト

入力された二つの値の加算/減算を行います。

使用機能は以下のとおりです。

- コマンドスクリプティングの利用

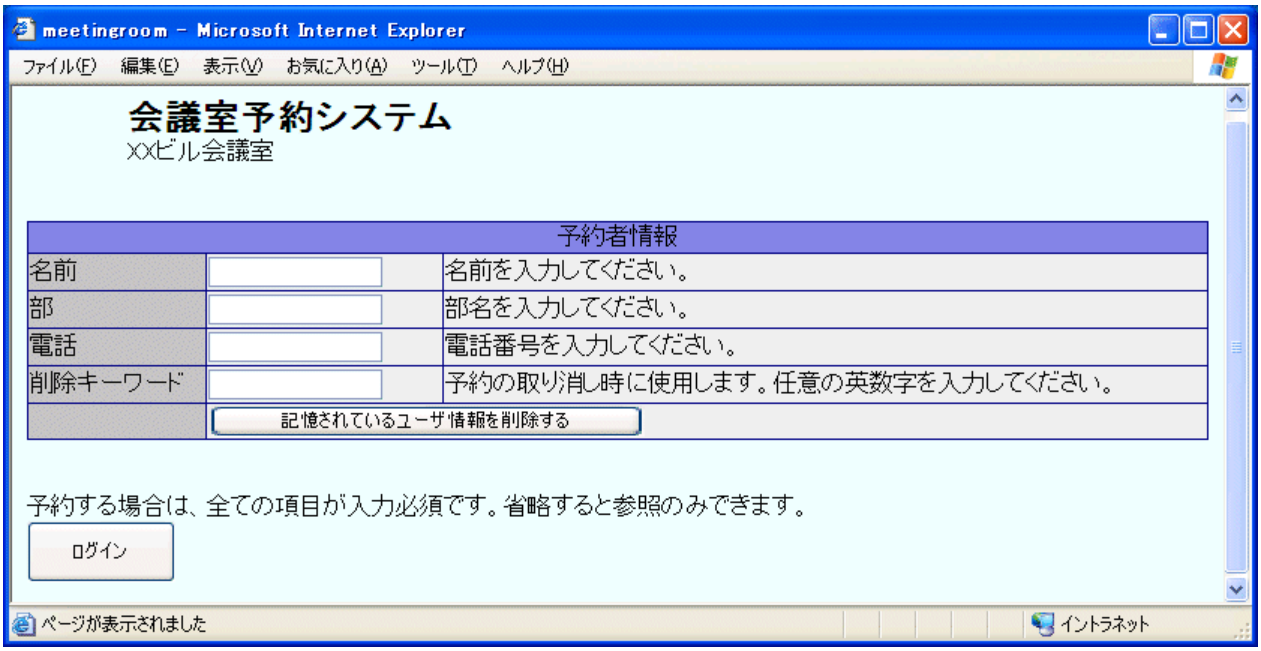


meetingroomプロジェクト

会議室の予約を行うサンプルです。

使用機能は以下のとおりです。

- リソースファイルの利用



A.4 応用的なサンプル

応用的な構成のサンプルとして、以下のものがあります。

使用機能	プロジェクト名	エディション		
		(1)	(2)	(3)
セション切断の検出	timeout	○	○	○
高度なセション管理	sessionControl	○	○	○
セキュリティ機能の利用	security	○	○	○

表中の番号に対応する製品とエディションは以下のとおりです。

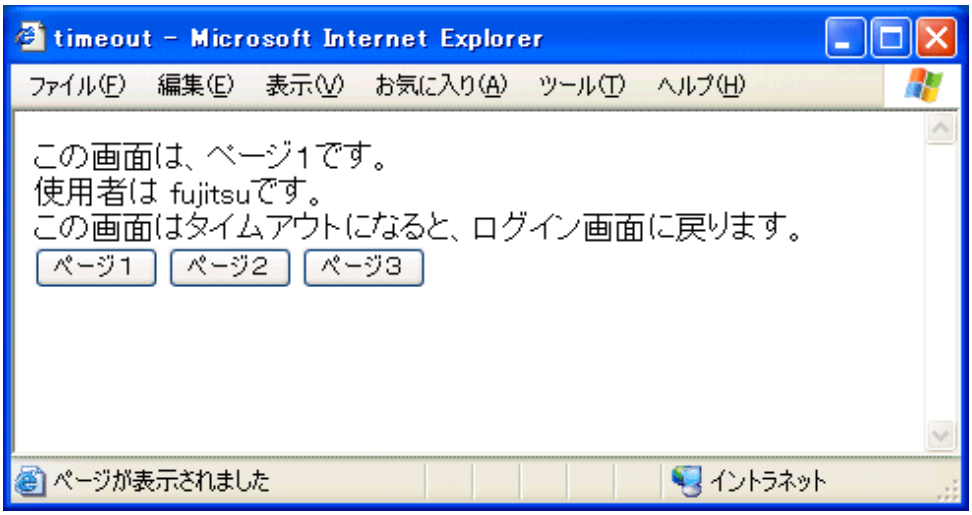
番号	製品名とエディション
(1)	Interstage Interaction Manager
(2)	Interstage Studio Standard-J Edition
(3)	Interstage Business Application Server Standard Edition Interstage Business Application Server Enterprise Edition

timeoutプロジェクト

タイムアウト処理のサンプルです。

使用機能は以下のとおりです。

- セション切断の検出

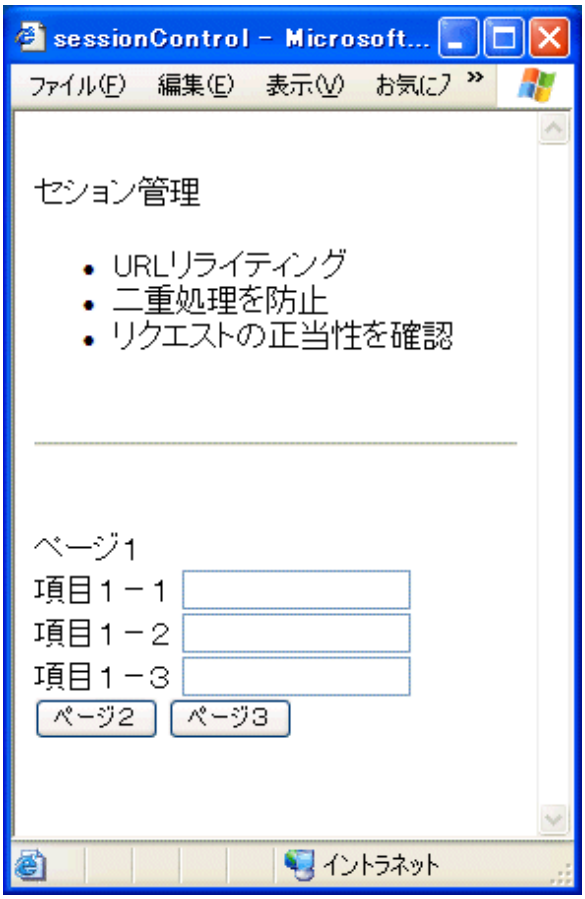


sessionControlプロジェクト

高度なセション管理機能を使用したサンプルです。

使用機能は以下のとおりです。

- 高度なセション管理

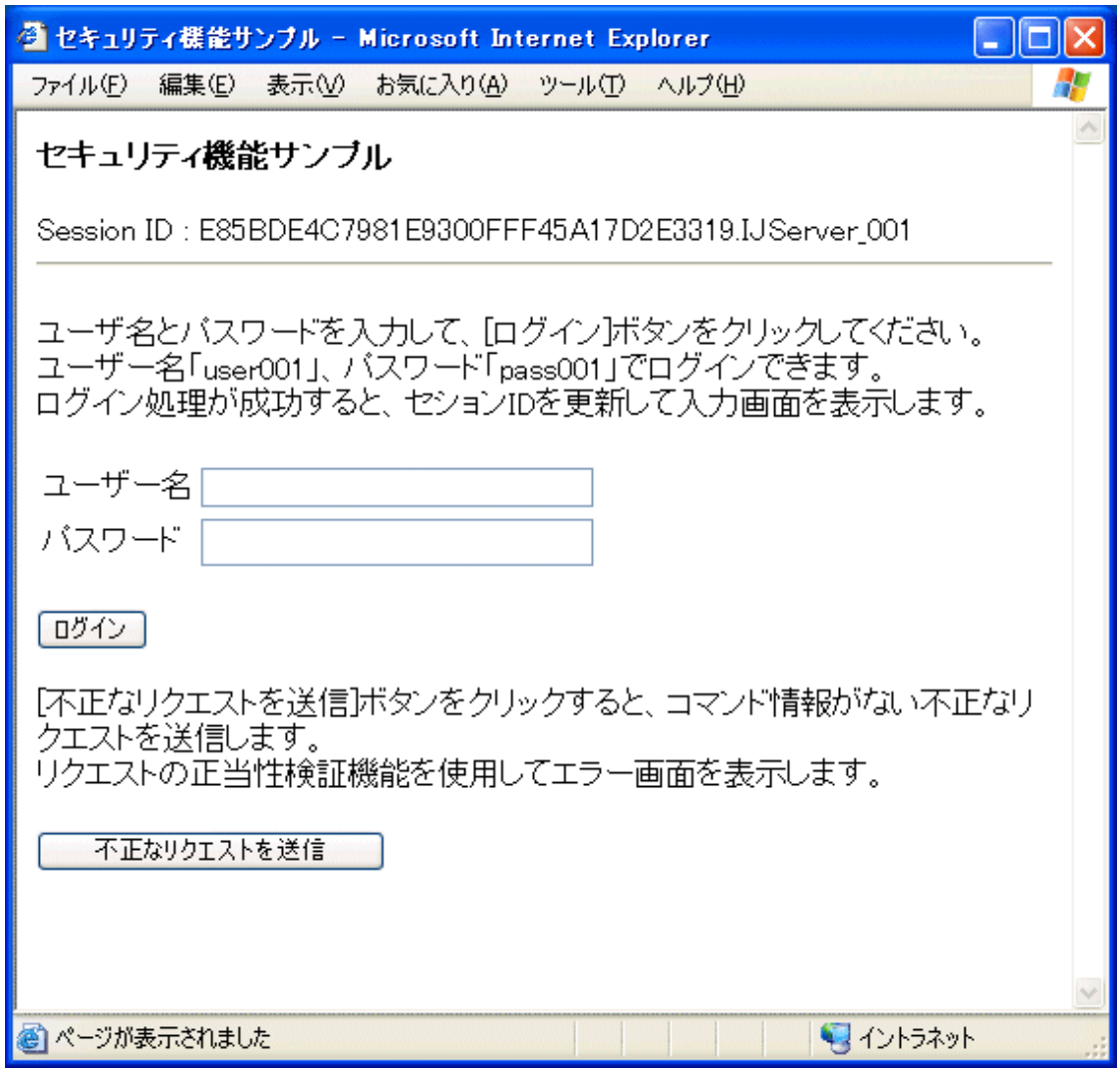


securityプロジェクト

セションの開始と破棄機能、セションIDの更新機能、リクエストの正当性検証機能およびエスケープ機能を使用したサンプルです。

使用機能は以下のとおりです。

- セキュリティ機能の利用


セキュリティ機能サンプル - Microsoft Internet Explorer
ファイル(F) 編集(E) 表示(V) お気に入り(A) ツール(T) ヘルプ(H)
セキュリティ機能サンプル
Session ID : E85BDE4C7981E9300FFF45A17D2E3319.IJServer_001
ユーザ名とパスワードを入力して、[ログイン]ボタンをクリックしてください。
ユーザー名「user001」、パスワード「pass001」でログインできます。
ログイン処理が成功すると、セションIDを更新して入力画面を表示します。
ユーザー名
パスワード
ログイン
[不正なリクエストを送信]ボタンをクリックすると、コマンド情報がない不正なリクエストを送信します。
リクエストの正当性検証機能を使用してエラー画面を表示します。
不正なリクエストを送信
ページが表示されました
イントラネット

索　引

[A]

Apcoordinator.....1

[W]

WARファイル.....2
Webページ.....1

[さ]

サンプルの格納場所.....73
サンプルの利用方法.....73

[た]

データBean.....2

[な]

入出力ページ.....2

[は]

配備.....2
ビジネスクラス.....2