FUJITSU

FUJITSU Software
Enterprise Application Platform V1.1.0

Launcherユーザーズガイド

Windows(64) / Linux(64)

B1FW-6081-02Z0(00)
2022年10月

まえがき

本書の目的

本書は、Launcherを利用してアプリケーションの開発や運用をするために必要な、機能、開発方法、運用方法などを説明しています。

本書の読者

本書は、以下の読者を対象に書かれています。

- Launcherを利用してアプリケーションの開発や運用をする方

本書の構成

本書は、以下の構成になっています。

第1章 Launcherの概要

第2章 提供機能

第3章 開発

第4章 運用

第5章 セキュリティ

第6章 資源見積り

第7章 定義項目一覧

第8章 コマンドリファレンス

本書の表記について

本書では、説明するうえで、次に示す名称、略称および記号を使用しています。

- マニュアル中で、自身を参照する場合は「本書」と記述します。
  本製品のマニュアル名称を記述する場合、マニュアル名の先頭の「FUJITSU Software Enterprise Application Platform VXX.XX.XX(Xにはバージョンを表す数値)」を省略します。

商標について

- Eclipse、GlassFish、JakartaおよびMicroProfileは、Eclipse Foundation, Inc. の商標です。
- Linux(R)は米国およびその他の国におけるLinus Torvaldsの登録商標です。
- Red Hat、Red Hat Enterprise Linux、OpenShiftは、Red Hat, Inc.の米国およびその他の国における商標または登録商標です。
- Microsoft、Windows、およびWindows Serverは、米国 Microsoft Corporation の、米国およびその他の国における登録商標または商標です。
- OpenJDKは、Oracle America, Inc. の商標です。
- OracleおよびJavaは、オラクルおよびその関連会社の登録商標です。
- Dockerは、Docker, Inc. の米国およびその他の国における商標または登録商標です。
- そのほか、本マニュアルに記載されている会社名および製品名は、それぞれ各社の商標または登録商標です。

輸出管理規制について

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。

出版年月および版数

2022年10月 第2版

2020年10月 初版

著作権表示

Copyright 2020-2022 FUJITSU LIMITED

目　次

第1章 Launcherの概要....1
1.1 Launcherについて....1
1.1.1 Launcherの概要と特徴....1
1.1.2 バージョン情報....1
1.1.3 関連ドキュメント....1
1.2 ソフトウェア条件....2
1.2.1 JRE/JDKバージョン....2
1.2.2 コンテナ環境....2
1.3 既知の問題....2
1.3.1 アプリケーション運用時の制限事項....2
1.4 制限事項....3
1.5 注意事項....3
1.5.1 配備時の注意事項....3
1.5.2 Webアプリケーションの注意事項....4
1.5.3 ログファイルのローテーションの注意事項....7
1.5.4 domain.xmlの注意事項....7
1.5.5 サブコマンド実行時の注意事項....7
1.5.6 MicroProfile機能における注意事項....7
1.5.7 Contexts and Dependency Injection(CDI)における注意事項....7
1.5.8 一時ファイルに関する注意事項....8
1.5.9 日付/時刻に関する注意事項....8
1.5.10 セキュリティーマネージャーを使用する場合の注意事項....8
1.5.11 Java VM起動時に出力される警告メッセージ....8
1.5.12 個人情報を含む重要なデータの出力先について....8

第2章 提供機能....9
2.1 全体構成....9
2.2 MicroProfileアプリケーションの実行....9
2.3 Servlet/JSPアプリケーションの実行....9
2.4 Uber-JARファイルの作成....10
2.5 Webコンテナ....10
2.5.1 通信方式....10
2.5.2 JSP事前コンパイル....10
2.6 クラスローダー....10
2.6.1 Webクラスローダーの委譲モデルの変更....11

第3章 開発....12
3.1 アプリケーションの開発....12
3.2 JWT認証を使用するアプリケーションの開発....12
3.3 Rest Clientアプリケーションの開発....12
3.4 コンテナイメージの作成....13
3.5 deployment descriptor....14

第4章 運用....16
4.1 アプリケーションの起動....16
4.2 アプリケーションの停止....16
4.3 Uber-JARファイルの作成....16
4.4 インスタンスルートディレクトリー....17
4.5 ネットワークリスナーの設定....17
4.5.1 ポート番号の設定....17
4.5.2 ポートの閉塞....18
4.6 運用状況の監視....18
4.7 分散トレース....18
4.8 ログ....19
4.8.1 サーバーログ....19
4.8.2 HTTPアクセスログ....19

4.9 コンテナ動作環境での運用 21

第5章 セキュリティ 22
5.1 Launcherに関するセキュリティ対策 22
5.2 アプリケーションのセキュリティ機能 24
5.2.1 認証 24
5.2.2 認可 25
5.2.3 SSL/TLS 25
5.2.3.1 Launcherの設定 25
5.2.3.2 証明書の設定 25
5.2.3.3 キーストア/トラストストアの設定 26
5.2.4 ユーザー/グループ/ロール 27
5.2.5 レルム 27
5.3 セキュリティ機能を利用した運用方法 27
5.3.1 ロールとロールベースのアクセス制限の設定 28
5.3.2 ユーザー/グループをロールにマッピング 28
5.3.3 レルムの作成 28
5.3.4 デフォルトレルムの設定 28
5.3.5 fileレルムのユーザー・グループ管理 28

第6章 資源見積り 30
6.1 運用時に必要なディスク容量 30
6.2 運用時に必要なメモリー容量 30

第7章 定義項目一覧 31
7.1 domain.xml 31
7.1.1 domain.xmlの作成 31
7.1.2 記法 32
7.1.3 定義時に利用できるプロパティ 33
7.1.4 HTTPサービスの定義項目 34
7.1.5 ネットワーク設定の定義項目 35
7.1.6 Webコンテナの定義項目 39
7.1.7 セキュリティの定義項目 39
7.1.8 Java VMの定義項目 41
7.1.9 スレッドプールの定義項目 41
7.2 システムプロパティ 42
7.2.1 glassfish.embedded.tmpdirシステムプロパティ 42

第8章 コマンドリファレンス 43
8.1 コマンドライン仕様 43
8.2 サブコマンド 44
8.2.1 サブコマンド一覧 44
8.2.2 create-file-user 44
8.2.3 delete-file-user 45
8.2.4 update-file-user 46
8.2.5 list-file-users 47
8.2.6 list-file-groups 47

第1章 Launcherの概要

本章では、Launcherの概要およびソフトウェア条件、既知の問題、制限事項、注意事項について説明します。

1.1 Launcherについて

Launcherの概要について説明します。

1.1.1 Launcherの概要と特徴

Launcherは、エンタープライズ向けJavaアプリケーションの軽量な実行環境を提供します。提供される機能はマイクロサービスの構築に最適化されており、高信頼で運用性の高いクラウドネイティブなアプリケーション実行環境を容易に構築できます。

Launcherの特徴

MicroProfile準拠

Launcherは、マイクロサービスアーキテクチャー向けの規格のMicroProfileに準拠しています。MicroProfileにはマイクロサービスに必要となるAPIが定義されており、利用者は高信頼で運用性の高いマイクロサービスを構築できます。これにより、マイクロサービスアーキテクチャーがもたらす俊敏性や柔軟性といった恩恵を享受できます。また、特定ベンダーに依存しないプログラミングモデルを採用しているため、複数ベンダーのランタイムで動作する移植性の高いアプリケーションを作成できます。

軽量な実行環境

コンテナ環境での利用に適した軽量なアプリケーション実行環境を提供します。Launcherは1つのJARファイルで構成されており、1回のコマンド操作で容易にアプリケーションを実行できます。

既存資産の活用

エンタープライズ向けJavaアプリケーションの既存資産の活用により、開発生産性を向上できます。Launcherでは、エンタープライズ向けJavaアプリケーションで広く使用されているServletなどのJakarta EEのAPIを使用したアプリケーションの実行に対応しています（Launcherで使用可能なJakarta EEのAPIについては「2.2 MicroProfileアプリケーションの実行」および「2.3 Servlet/JSPアプリケーションの実行」を参照してください）。また、Jakarta EEアプリケーション実行環境で標準的に使われているWAR形式のアプリケーションを実行できます。

1.1.2 バージョン情報

本製品にはLauncher V4.0が搭載されています。

参照

Launcher GitHubリポジトリー

https://github.com/fujitsu/launcher

LauncherはEclipse GlassFish Embedded Server (Web Profile) V6.2を動作基盤として内包しています。

参照

Eclipse GlassFish

https://glassfish.org/

1.1.3 関連ドキュメント

Launcherの関連ドキュメントについては以下を参照してください。

参照

Launcherドキュメント（README）

https://github.com/fujitsu/launcher

Eclipse GlassFishドキュメント

https://glassfish.org/docs/

1.2 ソフトウェア条件

Launcherを使用する場合に必要となるソフトウェアについて説明します。

1.2.1 JRE/JDKバージョン

Launcherは、以下のJDKバージョンをサポートします。

- JDK 11

1.2.2 コンテナ環境

Launcherは以下のコンテナ動作環境をサポートしています。以下以外のコンテナ動作環境を利用したい場合、担当の営業やSEにご相談ください。

サポートするコンテナ環境	バージョン・レベル
Red Hat OpenShift Container Platform	4.8以降

サポートするコンテナベースイメージ	バージョン・レベル
Red Hat Universal Base Image	8 (ubi8-standard)

1.3 既知の問題

マニュアルに記載された機能に関する、既知の問題を記載しています。

1.3.1 アプリケーション運用時の制限事項

項番	制限事項	対処方法	解除予定
1	HTTPリスナー/HTTPSリスナーを無効にしてLauncherを起動すると、以下のエラーメッセージが出力されます。 Error adding HttpProbes. NetworkListener <リスナー名> GrizzlyProxy is NULL	メッセージが出力される以外に動作上の問題はないため、対処は必要ありません。	未定
2	h2(HTTP/2 over TLS)を使用しWebコンテナにアクセスする場合は、日本語などのマルチバイト文字を含むURIにアクセスできません。	HTTP/2通信を使用してWebコンテナにアクセスする場合は、以下の対処を行ってください。 ・h2c(HTTP/2 over cleartext TCP)を使用してください。 ・URIに日本語などのマルチバイト文字を使用しないでください。	未定

項番	制限事項	対処方法	解除予定
3	Launcherの起動処理中にJSPアプリケーションにリクエストを送信すると、サーバーログに以下のエラーメッセージが出力されます。 ・ PWC6117: File "null" not found ・ No context found:	起動処理が完了するまでの一過性の現象であり、メッセージが出力される以外に動作上の問題はないため、対処は必要ありません。 なお、起動処理が完了するまでアプリケーションへのHTTPリクエストの振り分けを防ぐ必要がある場合は、振り分け制御の設定(コンテナ環境ではヘルスチェックの設定)でアプリケーションのリソースからの応答を監視し、正常なHTTPステータスの応答が得られてからHTTPリクエストを振り分ける設定としてください。	未定
4	存在しないJSPリソースへのリクエストを送信した場合に、ファイル名がnullとなっている以下のエラーメッセージがサーバーログに出力されます。 PWC6117: File "null" not found	要求されたJSPリソースが存在しないことを表すメッセージです。メッセージの原因となっているリソースの特定が必要な場合は、HTTPアクセスログを参照してください。リソースの特定が不要な場合は、このメッセージを無視してください。	未定
5	以下の条件を満たした場合、アプリケーションがクライアントに返却するHTTPレスポンスのBODYについて、不正な文字コードでエンコードされることがあります。 1. アプリケーションで下記いずれかのメソッドを用いてHTTPレスポンスの文字コードを設定している。かつ、 ー jakarta.servlet.ServletResponse#setContentType(String) ー jakarta.servlet.http.HttpServletResponse#setHeader(String, String) ー jakarta.servlet.http.HttpServletResponse#addHeader(String, String) 2. 1.の変更後、アプリケーションでjakarta.servlet.ServletResponse#reset()メソッドを実行している。かつ、 3. 2.のメソッドを実行後、HTTPレスポンスの文字コードを再度設定していない場合。	以下の方法でアプリケーションを修正してください。 2.のメソッドを実行後、下記いずれかのメソッドを用いてHTTPレスポンスの文字コードを再度設定してください。 ・ jakarta.servlet.ServletResponse#setContentType(String) ・ jakarta.servlet.ServletResponse#setCharacterEncoding(String) ・ jakarta.servlet.http.HttpServletResponse#setHeader(String, String) ・ jakarta.servlet.http.HttpServletResponse#addHeader(String, String)	未定

1.4 制限事項

制限事項は特にありません。

1.5 注意事項

Launcherを使用する場合の注意事項について説明します。

1.5.1 配備時の注意事項

ライブラリー(--librariesオプション)指定文字列の注意事項

以下の指定は行わないでください。

- 2Byte文字
- 最後尾に空白文字(スペース、タブ)

1.5.2 Webアプリケーションの注意事項

以下について説明します。

- セッション使用時の注意
- 文字コードについて
- ServletContext/ServletRequestの属性について
- FORM認証について
- 日本語などのASCIIではないファイル名のファイルをダウンロードする場合
- レスポンスヘッダーに指定する文字コード
- HTTPレスポンスのデータ形式とヘッダーについて
- HTTP/2使用時の注意

セッション使用時の注意

セッションIDについて

セッションIDの一意性の範囲、桁数、使用文字、フォーマットは、本製品のバージョンアップや修正適用に伴い変更される可能性があります。
そのため、これらに依存したアプリケーションや、Webアプリケーション範囲以外へのセッションIDの利用はおすすめいたしません。
例えば、セッションIDをデータベースに登録した場合は、本製品のバージョンアップによってカラムの桁数不足が発生する可能性があります。
また、リクエストやユーザーの正当性や一意性につきましては、セッションIDのみに依存しない方法により適切にチェック/保証するようにしてください。

セッションCookieについて

Cookieによるセッション管理は、Webコンテナによって自動的に行われます。
セッションCookieの追加などの操作は、アプリケーションで行わないでください。
セッション継続やセキュリティ上の問題が発生する場合があります。

SSLアクセラレーターを使用する場合

Webサーバーが SSL環境で動作している場合には、セッションCookieに自動的にSecure属性を付加します。
SSLアクセラレーターを使用しておりWebサーバーが非SSL(HTTP)で動作している場合には、Secure属性は自動的に付加しません。
Secure属性を付加する方法については、Eclipse GlassFishドキュメントのdeployment descriptorの説明を参照してください。

セッションIDを含むURLを生成する場合

WebアプリケーションでセッションIDを含むURLを生成する場合、以下のどちらかのAPIを使用してください。これらのAPIを使用しない場合、Webサーバーがセッションのあるlauncherに対して正しくリクエストを振り分けない可能性があります。

- jakarta.servlet.http.HttpServletResponse#encodeURL(String)
- jakarta.servlet.http.HttpServletResponse#encodeRedirectURL(String)

文字コードについて

Webブラウザーからのリクエストを読み込む時の文字コードについて

Webブラウザーからのリクエストを正しい文字コードで処理するための方法には、以下の2つがあります。

Servlet APIでの指定

以下のServlet APIを使用して、文字コードを指定します。

- jakarta.servlet.ServletRequest.setCharacterEncoding

詳細は、Servletの仕様を参照してください。

例

```
request.setCharacterEncoding("SJIS");
```

注意

このメソッドは、リクエストのパラメーターを読み込む前、またはgetReaderメソッドを呼び出す前に実行されなければなりません。

glassfish-web.xmlでの指定

glassfish-web.xmlの<parameter-encoding>タグで文字コードを指定します。
定義方法の詳細は、Eclipse GlassFishドキュメントを参照してください。

例

```
<parameter-encoding default-charset="SJIS"/>
```

注意

jakarta.servlet.ServletRequest.setCharacterEncodingが指定されている場合は、setCharacterEncodingで指定された値が優先されます。

レスポンスヘッダーに指定する文字コード

HTTPレスポンスヘッダーには、印字可能なASCIIコード範囲の文字を指定してください。それ以外の文字は、半角空白に置き換えられてクライアントに返却されます。

日本語などのASCIIではないファイル名のファイルをダウンロードする場合

WebアプリケーションでHTTPレスポンスヘッダーにContent-Dispositionヘッダーを設定することでファイルのダウンロードが行われる場合、Content-Dispositionヘッダーのfilename属性に日本語やISO-8859-1などの印字可能なASCIIコード範囲でない文字を指定すると、ダウンロードしたファイルのファイル名が文字化けすることがあります。

そのため、RFC6266に則りfilename*属性にURLエンコードしたファイル名を指定するようにしてください。このとき、java.net.URLEncoderのencodeメソッドを使用すると半角空白は+に変換されるため、encodeメソッドを実行して得られた文字列の「+」を「%20」に置換してください。

例

ファイル名「日本語ファイル名 (1).txt」をダウンロードするサーブレットのコード例

```
String fileName="日本語ファイル名 (1).txt";
String encodedFileName = URLEncoder.encode(fileName, "UTF-8");
encodedFileName = encodedFileName.replace("+", "%20");
response.setContentType("text/plain");
response.setHeader("Content-Disposition", "attachment; filename*=UTF-8¥'¥'" + encodedFileName);
```

ServletContext/ServletRequestの属性について

以下のプレフィックスで始まる属性は、予約されています。
値は、変更しないでください。

- java.
- javax.

- jakarta.
- sun.
- com.sun.
- com.fujitsu.appsrv.

また、ServletContextAttributeListener/ServletRequestAttributeListenerが設定されている場合、Webコンテナが上記プレフィックスで始まる属性を追加、更新、削除するタイミングでリスナーのメソッドが呼び出されます。

FORM認証について

POSTメソッドに対して認証を実施する場合の注意事項

以下の条件を満たした場合、Webブラウザーに「HTTP Status 405」が表示され、リクエストの処理に失敗します。そのため、<http-method>には、GETも指定してください。

1. WebアプリケーションでJakarta EEセキュリティ機能を使用、かつ
2. FORM認証である場合、かつ
3. deployment descriptor(web.xml)で<http-method>にPOSTのみ指定している場合、かつ
4. 1.のWebアプリケーションに対して、3.の設定が有効となるコンテンツに対してHTTP POSTリクエストを送信、かつ
5. FORM認証の画面から目的のコンテンツに遷移した場合

認証後にリクエストのボディーを取得する場合の注意事項

以下の条件を満たした場合、ボディーを含むHTTP POSTリクエストを送信しても、Webアプリケーション内ではリクエストのボディーを正しく取得できません。HTTP POSTリクエストを送信するときには、FORM認証の画面が表示されない状態(認証が 完了した状態)になるようにアプリケーションを作成してください。

1. WebアプリケーションでJakarta EEセキュリティ機能を使用、かつ
2. FORM認証である場合、かつ
3. ServletまたはJSPでリクエストのボディーを取得、かつ
4. Webアプリケーションに対して、HTTP POSTリクエストを送信、かつ
5. FORM認証の画面から目的のコンテンツに遷移した場合

HTTPレスポンスのデータ形式とヘッダーについて

HTTP/1.1の場合に送信するデータが一定サイズを超えた場合、送信データはチャンク化されて“Content-Length”ではなく“Transfer-Encoding: chunked”ヘッダーが使用されます。HTTPクライアントを設計する場合は、送信データがチャンク化されることを想定した設計にしてください。

送信データのチャンク化を行わない設計にするためには、レスポンスのバッファーサイズに十分大きなサイズを指定し、レスポンスのflushは行わないでください。

HTTP/2使用時の注意

X-Powered-Byヘッダーフィールドについて

HTTP/2接続時には、HTTPの設定のxpowered-by属性の値にかかわらず、Webコンテナが生成するHTTPレスポンスにX-Powered-Byヘッダーフィールドは含まれません。

本属性の詳細については、「7.1.5 ネットワーク設定の定義項目」の「HTTP」を参照してください。

HTTPレスポンスのデータ形式とヘッダーについて

HTTP/2接続時には、サーブレットやJSPに対するHTTPレスポンスに“content-length”ヘッダーがデフォルトで含まれません。また、“transfer-encoding: chunked”ヘッダーが含まれることはありません。

HTTPクライアントを設計する場合は、HTTPレスポンスにこれらのヘッダーが含まれない場合があることを想定した設計にしてください。

ただし、以下のいずれかの条件を満たす場合には、HTTPレスポンスに“content-length”ヘッダーは含まれます。

- jakarta.servlet.ServletResponseインターフェイスのsetContentLengthメソッドまたはsetContentLengthLongメソッドにより、明示的に本文の長さを指定している。または、
- 静的リソース（HTMLページや画像など）に対するリクエストへのレスポンスである。または、
- サーブレットに対するHTTP HEADメソッドのリクエストへのレスポンスである場合。

1.5.3 ログファイルのローテーションの注意事項

Launcherの各種ログファイルは、製品機能が管理しています。そのため、製品以外の機能によるローテーションは行わないでください。

1.5.4 domain.xmlの注意事項

マルチバイト文字の利用について

定義項目の値にマルチバイト文字を含めることはできません。

属性/プロパティの省略に関する注意

domain.xmlで属性/プロパティを省略した場合、本製品の既定値で動作します。ただし、既定値のない属性/プロパティは省略できません。詳細については、「7.1.1 domain.xmlの作成」の「属性の変更」、「プロパティの変更」を参照してください。

1.5.5 サブコマンド実行時の注意事項

サブコマンドを実行する際は、本製品のマニュアルに記載されているコマンド仕様を参照して実行してください。無効なコマンドラインでサブコマンドを実行した場合にUsageが出力されますが、このUsageは参照しないでください。

1.5.6 MicroProfile機能における注意事項

以下の機能はContexts and Dependency Injection (CDI)を使用するアプリケーションを配備せずに単独で利用することはできません。必ずCDIを使用するアプリケーションを配備した上で利用してください。

- MicroProfile Metrics
- MicroProfile Health
- MicroProfile JWT Authentication

CDIを使用するアプリケーションを配備せずに上記機能を利用した場合、以下の現象となります。

- MicroProfile Metrics機能のRESTエンドポイントにアクセスすると、ボディーが空の応答が返却されます。
- MicroProfile Health機能のRESTエンドポイントにアクセスすると、HTTPステータス503の応答が返却されます。
- MicroProfile JWT Authentication機能を使用するアプリケーションを配備すると、以下の行を含むメッセージが出力されてLauncherが異常終了します。
  org.glassfish.hk2.api.UnsatisfiedDependencyException: There was no object available in __HK2_Generated_1 for injection at SystemInjecteeImpl(<詳細メッセージ>)

既存のアプリケーションを、CDIを使用するアプリケーションに変更する場合は、以下のどちらかを実施してください。

- CDIの対象となるアノテーション（例：@jakarta.enterprise.context.ApplicationScoped）を指定したクラスをアプリケーションに追加します。
- アプリケーション（WARファイル）のWEB-INFディレクトリーにbeans.xmlファイルを追加します。

1.5.7 Contexts and Dependency Injection(CDI)における注意事項

CDIを使用する場合、以下に注意してください。

jakarta.enterprise.context.Conversation#setTimeout(java.lang.Long)を使用する場合

引数に指定するタイムアウト時間には2147483647より大きな値を指定しないでください。

1.5.8 一時ファイルに関する注意事項

アプリケーション実行時には、一時的にインスタンスルートディレクトリーが作成され、アプリケーションの動作に必要なファイルがその下に作成されます。インスタンスルートディレクトリーの詳細は「4.4 インスタンスルートディレクトリー」を参照してください。

通常、停止時にはインスタンスルートディレクトリーは削除されますが、プロセスを強制終了した場合などに削除されず残る場合があります。もし不要なディレクトリー、ファイルが残っている場合は削除してください。

1.5.9 日付/時刻に関する注意事項

ログファイルのローテーションについて

ログファイルは、ローテーション実行時に時刻が付加されたファイル名で保存されます。保管するログファイルの数が、指定した世代数を超えた場合は、ファイル名に含まれる時刻が一番古いログファイルが削除されます。

システム日付/時刻を調整すると、現在時刻よりも未来の時刻が付加されたログファイルが存在する可能性があります。この状態で、ローテーションが行われると、未来の時刻が付加されたログファイルが残ってしまい、現在時刻のログファイルがバックアップされない可能性があります。

ログファイルのローテーションが意図通りに行われない事態を避けるため、システム日付/時刻を調整した場合には、保管されているログファイルのファイル名を確認し、必要に応じてファイル名を変更またはファイルを削除してください。

ログの出力先やファイル名については、「4.8 ログ」を参照してください。

1.5.10 セキュリティーマネージャーを使用する場合の注意事項

Launcherではセキュリティーマネージャーは使用できません。

1.5.11 Java VM起動時に出力される警告メッセージ

Launcher起動の際、以下のような警告メッセージが出力されることがあります。

```
WARNING: An illegal reflective access operation has occurred
WARNING: Illegal reflective access by <略>
WARNING: Please consider reporting this to the maintainers of <略>
WARNING: Use --illegal-access=warn to enable warnings of further illegal reflective access operations
WARNING: All illegal access operations will be denied in a future release
```

これはリフレクションを使用して特定パッケージに対してアクセスした際に、Java VMにより出力される警告です。現状では運用に影響しないため、無視して問題ありません。

1.5.12 個人情報を含む重要なデータの出力先について

アプリケーションが標準出力・標準エラー出力に出力した情報は、Launcherのサーバーログとともに標準出力・標準エラー出力に出力されます。

標準出力・標準エラー出力に出力されるサーバーログなどの情報は、トラブル発生時には調査資料として技術員に提出することがあります。

そのため、それらに個人情報を含む重要なデータが出力されないようにしてください。

ログ情報については、「4.8 ログ」を参照してください。

第2章 提供機能

本章では、Launcherの提供機能について説明します。

2.1 全体構成

Launcherは以下の機能を提供します。

- マイクロサービスアプリケーションの簡単かつ素早い実行
- 取り扱いや実行が容易なUber-JARファイルの作成

ファイル構成

Launcherは1つのJARファイル「launcher.jar」から構成されます。Launcherのすべての機能はこのファイルによって提供されます。

Windows64

LauncherのJARファイルは以下のフォルダーにインストールされます(本製品のインストール先がC:¥eapfの場合)。

```
C:¥eapf¥launcher¥lib
```

Linux64

LauncherのJARファイルは以下のディレクトリーにインストールされます。

```
/opt/FJSVeapf/launcher/lib
```

2.2 MicroProfileアプリケーションの実行

本製品ではMicroProfile 5.0に準拠したアプリケーションを実行できます。以下のMicroProfile実装を含んでおり、これらのAPIを使用したアプリケーションを実行できます。

API	包含するOSS実装
Jakarta Contexts and Dependency Injection 3.0	Eclipse GlassFish Embedded Server (Web Profile)
Jakarta Annotations 2.0	
Jakarta RESTful Web Services 3.0	
Jakarta JSON Binding 2.0	
Jakarta JSON Processing 2.0	
MicroProfile Config 3.0	SmallRye Config
MicroProfile Fault Tolerance 4.0	SmallRye Fault Tolerance
MicroProfile Health Check 4.0	SmallRye Health
MicroProfile JWT Authentication 2.0	SmallRye JWT
MicroProfile Metrics 4.0	SmallRye Metrics
MicroProfile OpenAPI 3.0	SmallRye OpenAPI
MicroProfile OpenTracing 3.0	SmallRye OpenTracing
MicroProfile Rest Client 3.0	Eclipse Jersey

2.3 Servlet/JSPアプリケーションの実行

本製品では、Jakarta EE 9.1規約に定められているJakarta Servlet、Jakarta Server Pages (JSP)などのAPIを使用したアプリケーションを実行できます。

Launcherで使用可能なJakarta EEのAPIを以下に示します。

API	包含するOSS実装
Jakarta Servlet 5.0	Eclipse GlassFish Embedded Server (Web Profile)
Jakarta Server Pages 3.0	
Jakarta Expression Language 4.0	
Jakarta RESTful Web Services 3.0	
Jakarta JSON Processing 2.0	
Jakarta JSON Binding 2.0	
Jakarta Standard Tag Library 2.0	
Jakarta Annotations 2.0	
Jakarta Contexts and Dependency Injection 3.0	

2.4 Uber-JARファイルの作成

指定したWARファイルをもとに、アプリケーションの実行に必要なファイルを組み込んだ単一の実行可能JARファイル（Uber-JARファイル）を生成できます。Uber-JARファイルの作成方法の詳細については、「4.3 Uber-JARファイルの作成」を参照してください。

2.5 Webコンテナ

本製品のWebコンテナの仕様について説明します。

2.5.1 通信方式

Webコンテナへの通信方式として、以下の方式が利用可能です。

- HTTP/1.1
- HTTP/1.1 over TLS
- HTTP/2 over TCP
- HTTP/2 over TLS

2.5.2 JSP事前コンパイル

JSP事前コンパイルは、初回リクエストよりも前にJSPコンパイルを行う、アプリケーションの配備方式です。

JSPは、通常初回リクエスト時にコンパイルを行いますが、この方式には以下の問題があります。

- 初回リクエストは、2回目以降のリクエストよりもレスポンスが返されるまでに時間がかかる。
- JSPに構文エラーがあった場合、初回リクエスト処理時にコンパイルエラーを検出するため、運用に支障を来すおそれがある。

これらの問題は、JSP事前コンパイル機能を使用することで回避できます。

本製品では、Webアプリケーション配備時にJSPコンパイルを行う方式である、配備時コンパイルを提供します。配備時に--precompilejspオプションを指定することで、配備時コンパイルが有効になります。--precompilejspオプションについては、「8.1 コマンドライン仕様」を参照してください。

なお、配備時コンパイルを有効にした場合、配備時にJSPコンパイルが行われるため、配備が完了するまでの時間が長くなります。

2.6 クラスローダー

本製品におけるクラスローダーの構成は、Eclipse GlassFishと同様です。クラスローダーの構成に関する詳細については、Eclipse GlassFishドキュメント「Application Development Guide」の「Class Loaders」を参照してください。

2.6.1 Webクラスローダーの委譲モデルの変更

本製品のデフォルト設定では、Webクラスローダーは先に親のクラスローダーにリソースの検索を委譲しますが、Webクラスローダーで検索した後に委譲する設定もできます。

設定方法

glassfish-web.xmlのclass-loader要素のdelegate属性で設定します。

設定値	意味
true(デフォルト値)	親クラスローダーへの委譲を先に行います。
false	親クラスローダーへの委譲を後に行います。

glassfish-web.xmlについては、「GlassFish Serverのdeployment descriptor」を参照してください。

委譲の例外

Webクラスローダーの委譲については、以下の2つの例外があります。

JDKのクラス

JDKのクラスは常に最初にロードされますので、このクラスはユーザーが置き換えることはできません。

特定パッケージ名で始まるクラス

以下のパッケージ名で始まるクラスは常に最初に親クラスローダーにロードが委譲されます。
親クラスローダーに以下のパッケージ名で始まるクラスが存在する場合は、子クラスローダーでクラスを置き換えることはできません。

パッケージ名	種類
jakarta	Jakarta EE
javax	Licensor Name Space
sun	JDKの内部実装
org.xml.sax	SAX
org.w3c.dom	DOM
org.apache.taglibs.standard	JSP Standard Tag Libraryの実装
com.sun.faces	Jakarta Server Facesの実装
org.apache.commons.logging	Apache Commons Logging

第3章 開発

本章では、アプリケーションの開発方法について説明します。

3.1 アプリケーションの開発

Launcherでは、アプリケーションの形式としてWARをサポートしています。アプリケーションで使用可能なAPIについては、「第2章 提供機能」を参照してください。

3.2 JWT認証を使用するアプリケーションの開発

アプリケーションのJWT認証を有効化するには、jakarta.ws.rs.core.Applicationクラスの派生クラスに以下のアノテーションを付与します。

```
@org.eclipse.microprofile.auth.LoginConfig(authMethod = "MP-JWT")
```

例

JWT認証を有効化したアプリケーションの実装例を示します。

```
import jakarta.ws.rs.ApplicationPath;
import jakarta.ws.rs.core.Application;
import org.eclipse.microprofile.auth.LoginConfig;

@LoginConfig(authMethod = "MP-JWT")
@ApplicationPath("/")
public class SampleApplication extends Application {
}
```

注意

web.xmlのlogin-config要素によるJWT認証の有効化はできません。

3.3 Rest Clientアプリケーションの開発

Rest Client機能では、RESTfulサービスへのアクセスを型安全な方法で記述できます。詳細については、MicroProfile Rest Client規約を参照してください。

ここでは、Rest ClientでHTTPプロキシを使用する場合のアプリケーション作成方法を説明します。

HTTPプロキシの使用

Rest ClientからRESTfulサービスへの接続にHTTPプロキシを使用するには、HTTPプロキシサーバーとコネクタープロバイダーの指定が必要です。

HTTPプロキシサーバーの指定

HTTPプロキシサーバーのホスト名とポート番号を以下のどちらかの方法で指定します。

- RestClientBuilderによりクライアントをインスタンス化する場合、RestClientBuilderインターフェイスのproxyAddressメソッドに指定します。
- CDIによりクライアントをインスタンス化する場合、MicroProfile Configプロパティ「<完全修飾クラス名またはconfigKey名>/mp-rest/proxyAddress」に指定します。

コネクタープロバイダーの指定

コネクタープロバイダー「org.glassfish.jersey.apache.connector.ApacheConnectorProvider」を以下のプロパティに指定して登録します。

－ MicroProfile Configプロパティ「<完全修飾クラス名またはconfigKey名>/mp-rest/providers」

例

HTTPプロキシのホスト名がmyproxy、ポート番号が8080の場合、RestClientBuilderを使用して以下のようにHTTPプロキシサーバーを指定します。

```
ProxiedClient client = RestClientBuilder.newBuilder()
        .proxyAddress("myproxy", 8080)
        ...
        .build(ProxiedClient.class);
```

また、以下の内容のmicroprofile-config.propertiesファイルをアプリケーションに含めることで、コネクタープロバイダーを指定します。

```
com.example.ProxiedClient/mp-rest/providers=org.glassfish.jersey.apache.connector.ApacheConnectorProvider
```

3.4 コンテナイメージの作成

コンテナ動作環境でLauncherを利用する場合、Launcherのコンテナイメージを作成します。

以下にdockerコマンドを利用してコンテナイメージを作成する例を説明します。

以降の操作はすべてrootユーザーで行うものとします。

1. インストール資材の準備

   本製品の製品メディア（DVD-ROM）をディレクトリーにマウントしてください。

   以下は、ディレクトリー"/docker/work/eap"を作成してマウントする例です。

```
# mkdir -p /docker/work/eap
# mount -t iso9660 -r /dev/cdrom /docker/work/eap
```

2. インストールパラメーターファイルの作成

   Launcherのみ必要になるため、インストールパラメーターファイルを作成する必要があります。

   以下のようにLauncherをインストールすることを指定したインストールパラメーターファイル(/docker/work/param.csv)を用意します。

   <param.csv>

```
FN_Launcher,Y
```

3. アプリケーション実行ファイルの配備

   Launcher上で起動させるアプリケーション実行ファイルを作業ディレクトリー(/docker/work/)に配備します。

   今回はアプリケーション実行ファイルを"application.war"とします。

4. Dockerfileの作成

   作業ディレクトリー(/docker/work/)にDockerfileを作成します。

   例

   Dockerfile記述例

```
# 本例ではベースイメージにRed Hat Universal Base Image 8 (ubi8-standard)を使用します。
FROM registry.access.redhat.com/ubi8/ubi:latest
```

```
# 必須ソフトウェアをインストールします。
# 本製品の「インストールガイド」に記載されている基本ソフトウェアのパッケージを参照しています。
RUN /bin/yum -y install alsa-lib.x86_64 bzip2-libs.x86_64 freetype.x86_64 glibc.x86_64 ¥
libX11.x86_64 libXau.x86_64 libXext.x86_64 ¥
libXi.x86_64 libXrender.x86_64 ¥
libXtst.x86_64 libpng.x86_64 libxcb.x86_64 tar.x86_64 unzip.x86_64 && ¥
/bin/yum clean all -y;

# 本製品のインストール作業
RUN mkdir work
COPY eap/ work/
COPY param.csv work/
RUN work/install.sh -s -f work/param.csv

# Launcher起動
COPY application.war work/
CMD ["/opt/FJSVeapf/openjdk/jdk11/bin/java", "-jar", "/opt/FJSVeapf/launcher/lib/launcher.jar", "--deploy", "work/
application.war"]
```

注意

- ビルド時のインストール段階のエラーについては、本製品の「インストールガイド」をご参照ください。
- 修正の適用が必要な場合は、本製品のインストール作業からLauncher起動の間にUpdateAdvisor(ミドルウェア)による修正適用が実施されるよう、Dockerfileに以下の処理を記述してください。UpdateAdvisor(ミドルウェア)の使用方法については、UpdateAdvisor(ミドルウェア)のヘルプをご参照ください。
  - UpdateAdvisor(ミドルウェア)インストールモジュール、修正適用管理簿設定ファイル、修正ファイルをイメージにコピーする処理
  - UpdateAdvisor(ミドルウェア)をサイレントモードでインストールする処理
  - 修正をサイレントモードで適用する処理

5. コンテナイメージのビルド

   上記のDockerfileを使用して、コンテナイメージを作成します。

   ここでは、イメージ名を"demo"として登録します。

```
# docker build . -t <リポジトリー名>/demo
# docker push <リポジトリー名>/demo
```

3.5 deployment descriptor

ここでは、deployment descriptorについて説明します。

Jakarta EE規約で規定されたdeployment descriptor

Jakarta EE規約で規定されたdeployment descriptorについては、Jakarta EE規約、およびそのサブ規約を参照してください。

GlassFish Serverのdeployment descriptor

Launcherの動作を定義するdeployment descriptorとして、GlassFish Serverのdeployment descriptorを使用できます。

Launcherで使用できるGlassFish Serverのdeployment descriptorを以下に示します。

GlassFish Serverのdeployment descriptor	配備モジュール
GlassFish Web application deployment descriptor (glassfish-web.xml)	WARファイル

glassfish-web.xmlの仕様と各タグの詳細については、Eclipse GlassFishドキュメント「Application Deployment Guide」の「The glassfish-web.xml File」を参照してください。

注意

glassfish-web.xmlのcontext-root要素を指定してコンテキストルートを変更することはできません。コンテキストルートを変更するには、--contextrootオプションを使用してください。--contextrootオプションについては、「8.1 コマンドライン仕様」を参照してください。

第4章 運用

本章では、Launcherの運用方法について説明します。

4.1 アプリケーションの起動

アプリケーションの起動方法には、WARファイルを使用する方法とUber-JARファイルを使用する方法があります。

WARファイルを使用する方法

コマンドラインオプション--deployにWARファイルを指定してアプリケーションを起動します。コマンドラインオプションの詳細については「8.1 コマンドライン仕様」を参照してください。

WARファイルを使用する場合のコマンドラインは以下のとおりです（*war-file*はWARファイルへのパスを表します）。

```
java -jar launcher.jar --deploy war-file
```

Uber-JARファイルを使用する方法

javaコマンドの-jarコマンドラインオプションにUber-JARファイルを指定してアプリケーションを起動します。Uber-JARの作成方法については「4.3 Uber-JARファイルの作成」を参照してください。

Uber-JARファイルを使用する場合のコマンドラインは以下の通りです（*uber-jar-file*はUber-JARファイルへのパスを表します）。

```
java -jar uber-jar-file
```

参考

アプリケーション実行時には、一時的にインスタンスルートディレクトリーが作成され、アプリケーションの動作に必要なファイルがその下に作成されます。インスタンスルートディレクトリーの詳細は「4.4 インスタンスルートディレクトリー」を参照してください。

4.2 アプリケーションの停止

Windows64

Ctrl+Cを入力してアプリケーションを停止してください。

Linux64

Ctrl+Cを入力するか、シグナル（SIGINT）によりアプリケーションを停止してください。

注意

停止時にはインスタンスルートディレクトリーは削除されますが、プロセスを強制終了した場合などに削除されず残る場合があります。もし不要なディレクトリー、ファイルが残っている場合は削除してください。

4.3 Uber-JARファイルの作成

--generateオプションを使用することで、--deployオプションで指定したアプリケーションと必要なランタイムを含んだ単体実行可能なJARファイル（Uber-JARファイル）を作成できます。

Uber-JARファイルの作成は以下のコマンドラインで実施します（*war-file*はWARファイルへのパス、*uber-jar-file*は作成するUber-JARファイルへのパスを表します）。

```
java -jar launcher.jar --deploy war-file --generate uber-jar-file
```

Uber-JARファイルを使用したアプリケーションの実行方法については「4.1 アプリケーションの起動」を参照してください。

注意

Uber-JARファイル作成時に以下のコマンドラインオプションを指定した場合、これらのコマンドラインオプションの情報はUber-JARファイルに保存され、Uber-JARファイルの実行時に有効になります。

- --config-file
- --contextroot
- --http-listener
- --https-listener
- --libraries
- --precompilejsp

--config-fileオプションに指定したファイルは、Uber-JARファイルに保存されます。--librariesオプションに指定したファイルは、Uber-JARファイルに格納されません。

4.4 インスタンスルートディレクトリー

Launcherの実行時には、作業ディレクトリーとしてインスタンスルートディレクトリーが作成されます。アプリケーションの動作に必要なファイルやディレクトリーは、インスタンスルートディレクトリーの下に作成されます。

インスタンスルートディレクトリーは、デフォルトではjava.io.tmpdirシステムプロパティで指定されるディレクトリーの下に、「gfembed*digits*tmp」（*ditits*は数字列を表します）というディレクトリー名で作成されます。

インスタンスルートディレクトリーの作成先となるディレクトリーを変更するには、glassfish.embedded.tmpdirシステムプロパティを設定します。システムプロパティの詳細については「7.2.1 glassfish.embedded.tmpdirシステムプロパティ」を参照してください。

例

この例では、/somewhere/tmp ディレクトリーの下にインスタンスルートディレクトリーが作成されます。

```
java -Dglassfish.embedded.tmpdir=/somewhere/tmp -jar launcher.jar --deploy my-app.war
```

4.5 ネットワークリスナーの設定

ネットワークリスナーに関する設定について説明します。

4.5.1 ポート番号の設定

ポート番号を変更するには、コマンドラインオプションまたはdomain.xmlによる設定を実施します。

コマンドラインオプションによるポート番号の設定方法については、「8.1 コマンドライン仕様」を参照してください。domain.xmlによるポート番号の設定方法については、「7.1.5 ネットワーク設定の定義項目」を参照してください。

例

domain.xmlを使用してHTTPリスナーのポート番号を9090に変更するには、port属性を以下のように変更します。

変更前

```
<network-listener port="8080" protocol="http-listener" transport="tcp" name="http-listener"
thread-pool="http-thread-pool" enabled="true"/>
```

変更後

```
<network-listener port="9090" protocol="http-listener" transport="tcp" name="http-listener"
thread-pool="http-thread-pool" enabled="true"/>
```

4.5.2 ポートの閉塞

ポートを閉塞するには、domain.xmlの以下の要素のenabled属性に「false」を指定します。詳細については「7.1.5 ネットワーク設定の定義項目」を参照してください。

```
/domain/configs/config/network-config/network-listeners/network-listener
```

例

HTTPリスナーのポートを閉塞するには、enabled属性を以下のように変更します。

変更前

```
<network-listener port="8080" protocol="http-listener" transport="tcp" name="http-listener"
thread-pool="http-thread-pool" enabled="true"/>
```

変更後

```
<network-listener port="8080" protocol="http-listener" transport="tcp" name="http-listener"
thread-pool="http-thread-pool" enabled="false"/>
```

4.6 運用状況の監視

運用状況の監視にはヘルスチェックおよびメトリクスを利用できます。

ヘルスチェックによる監視

MicroProfile Health Checkに準拠したヘルスチェック機能を利用できます。本機能は、アプリケーションの死活監視を可能とするRESTエンドポイントを提供します。本機能は、コンテナオーケストレーションシステムのヘルスチェック機能などからの利用を想定しています。

メトリクスによる監視

MicroProfile Metricsに準拠したメトリクス機能を利用できます。本機能は、メトリクス監視システムからの性能監視やリソース監視を可能とするRESTエンドポイントを提供します。

4.7 分散トレース

MicroProfile OpenTraicngに準拠した分散トレース機能を利用できます。

分散トレース機能を利用するには、アプリケーションにトレーサーとトレーサーリゾルバーを含める必要があります。トレーサーリゾルバーは、サービスローダーによってトレーサーを提供する仕組みです。以下の抽象クラスの実装がトレーサーリゾルバーとして使用されます。

```
io.opentracing.contrib.tracerresolver.TracerResolver
```

例

トレーサーとしてJaegerトレーサー(https://www.jaegertracing.io/)を使用する場合、トレーサーとトレーサーリゾルバーが含まれる以下のライブラリーをアプリケーションに追加します。version要素は必要に応じて設定してください。

```
<dependency>
    <groupId>io.jaegertracing</groupId>
    <artifactId>jaeger-client</artifactId>
```

```
    <version>${jaeger-client.version}</version>
</dependency>
```

4.8 ログ

本製品では、以下のログを提供しています。

これらログに出力された情報は、トラブル発生時の調査に役立てることができます。

- 4.8.1 サーバーログ
- 4.8.2 HTTPアクセスログ

4.8.1 サーバーログ

Launcherの各コンポーネントが出力する情報は、サーバーログとしてコンソールに出力されます。

ログの出力先

標準出力・標準エラー出力

仕様

Launcherの各コンポーネントから提供される情報が出力されます。

出力フォーマット

JavaのJUL(java.util.logging)の出力フォーマットに準じます。

出力例

```
5月 18, 2022 10:07:34 午前 org.glassfish.deployment.admin.DeployCommand execute
情報: launcher_application was successfully deployed in 1,201 milliseconds.
```

4.8.2 HTTPアクセスログ

HTTPアクセスログには、HTTPサービスの通信情報が出力されます。HTTPアクセスログは、設計通りにHTTPサービスで処理が実施されたかを確認するなどの用途のために、開発時およびテスト運用時の調査用として使用してください。

ファイル名

server_access_log.<サフィックス>.txt

ログファイルの出力先

domain.xmlに出力先ディレクトリーを指定します。設定方法については、「設定方法」を参照してください。

仕様

HTTPアクセスログは、Webコンテナへのリクエストに対する処理が完了したタイミングで出力されます。

HTTPアクセスログのディスクへの書き込みは、バッファーが満杯になるか、設定された時間に到達したタイミングで行われます。バッファーサイズ(buffer-size-bytes)とディスクに書き込むまでの時間(write-interval-seconds)の設定については、「7.1.4 HTTPサービスの定義項目」の「HTTPアクセスログ」を参照してください。

設定方法

HTTPアクセスログの設定は、domain.xmlにより変更可能です。

- アクセスログの有効化
  HTTPアクセスログの出力を有効にするかを指定します。詳細は、「7.1.4 HTTPサービスの定義項目」の「HTTPサービス」を参照してください。

- 形式
  HTTPアクセスログに出力する項目を指定します。詳細は「7.1.4 HTTPサービスの定義項目」の「HTTPアクセスログ」を参照してください。
- ローテーションに関する設定
  ローテーションの条件などを設定します。詳細は「7.1.4 HTTPサービスの定義項目」の「HTTPアクセスログ」を参照してください。
- ログ出力ディレクトリー
  ログファイルの出力先を変更できます。詳細は「7.1.4 HTTPサービスの定義項目」の「仮想サーバー」を参照してください。

例

HTTPアクセスログを有効化、ローテーションの世代数を1000、ログ出力ディレクトリーを「/var/log/launcher/access」に設定する場合のdomain.xmlの記載例です。

```
<http-service access-logging-enabled="true">
  <access-log max-history-files="1000"/>
  <virtual-server access-log="/var/log/launcher/access" id="server" network-listeners="http-listener,https-listener"/>
</http-service>
```

ローテーション

HTTPアクセスログのファイルはローテーションされます。

HTTPアクセスログのファイルは、一定の時間間隔(デフォルトでは1日間隔)でローテーションされます。起動時点または前回ローテーションした時点から、一定の時間間隔を経過したあとの最初のHTTPアクセスログ出力時にローテーションされます。時間間隔の設定については「7.1.4 HTTPサービスの定義項目」－「HTTPアクセスログ」のrotation-interval-in-minutes属性を参照してください。

HTTPアクセスログのファイル名は、ローテーションの実行時刻に基づいたサフィックスを付加した名前で保存されます。サフィックスの設定については「7.1.4 HTTPサービスの定義項目」－「HTTPアクセスログ」のrotation-suffix属性を参照してください。

```
server_access_log.<サフィックス>.txt
```

デフォルトの設定では、以下の形式の名前で保存されます。

```
server_access_log.yyyy-MM-dd.txt
```

出力フォーマット

HTTPアクセスログの出力フォーマットはカスタマイズできます。設定方法については、「設定方法」を参照してください。

出力フォーマットを指定するには、出力する項目を空白区切りで並べます。

出力フォーマットの初期値を以下に示します。

```
%client.name% %auth-user-name% %datetime% %request% %status% %response.length%
```

出力フォーマットに指定可能な項目を以下の表に示します。

項目名	説明	デフォルト
%auth-user-name%	Webコンテナで認証したユーザー名を出力します。 リクエストにユーザー名が付与されていない場合は、“NULL-AUTH-USER”を出力します。	○
%client.name%	クライアントまたは、プロキシサーバーなどのIPアドレス／ホスト名を出力します。	○
%cookies.any%	Cookieヘッダーの内容を出力します。anyに指定する値は、大文字、小文字を区別するのでHTTP/1.1で定義された値を指定してください。	
%cookies.JSESSIONID%	セッションIDを出力します。	
%datetime%	ログを出力した時点の日時を出力します。	○

項目名	説明	デフォルト
%headers.any%	リクエストHTTPヘッダーの値を出力します。anyには、HTTP/1.1で定義された任意の有効なリクエストHTTPヘッダーの値を小文字で指定します。	
%headers.host%	Hostヘッダーの内容を出力します。	
%headers.referer%	Refererヘッダーの内容を出力します。	
%headers.user-agent%	User-Agentヘッダーの内容を出力します。	
%http-method%	HTTPメソッドを出力します。	
%http-uri%	URIを出力します。	
%http-version%	プロトコルバージョンを出力します。	
%query-str%	クエリ文字列を出力します。	
%request%	リクエストラインの内容を出力します。	○
%response.headers.any%	レスポンスHTTPヘッダーの値を出力します。anyには、HTTP/1.1で定義された任意の有効なレスポンスHTTPヘッダーの値を小文字で指定します。	
%response.headers.set-cookie%	レスポンス時のSet-Cookieヘッダーの内容を出力します。	
%response.length%	レスポンスのデータ長を出力します。単位はbyteです。	○
%status%	ステータスコードを出力します。	○
%time-taken%	リクエストの処理開始から、Webアプリケーションの処理終了までの時間(ミリ秒)を出力します。 以下についても含みます。 ・ 初回アクセス時のJSPコンパイル、クラスのロード、Servletなどの初期化処理 HTTP/2のServer Pushのリクエストに対する%time-taken%の値は参照しないでください。	

出力例

出力フォーマットに指定した値が空白で区切られて出力されます。一部の値はダブルクォーテーション(")で囲まれて出力されます。

```
"192.0.2.237" "NULL-AUTH-USER" "31/Jan/2022:12:00:00 +0900" "GET /foo HTTP/1.1" 200 256
```

4.9 コンテナ動作環境での運用

「3.4 コンテナイメージの作成」を参照して作成したコンテナイメージをコンテナ動作環境にデプロイして運用してください。

コンテナ動作環境へのデプロイ方法については、利用するコンテナ動作環境のマニュアルを参照してください。

第5章 セキュリティ

本章では、Launcherのセキュリティについて説明します。

5.1 Launcherに関するセキュリティ対策

セキュリティ侵害に対する対策は、一般的にはサービスだけでは対処できません。運用でも対処を行うことによって、安全性が強化されます。

■サーバー資源への不正アクセス

悪意のあるユーザーが、正当なアクセス権限をもつユーザーになりすまし、運用資産にアクセスする脅威があります。サーバーマシンは、不正なアクセスをされないように、セキュリティ的に守られた施設内で運用してください。

また、Webブラウザーからアプリケーションにアクセスする際、ネットワーク上の悪意のあるユーザーが、正当なアクセス権限をもつユーザーになりすまし、サーブレットアプリケーションやHTMLファイルにアクセスする脅威があります。

サーバーの資源に対する不正なアクセスを防止するために、Launcherには各種のセキュリティ機能があります。詳細は、本章の以降の節を参照してください。

■通信データの盗聴・改ざん

ネットワーク上の悪意のあるユーザーが、正当にアクセスする権限をもつユーザーとサーバーとの間の通信データをのぞき見する脅威があります。さらにサーバーとの間の通信データを改ざんされ、それが正当なデータとしてやりとりされる脅威があります。

通信データの盗聴・改ざんを防ぐためには、暗号化によるセキュリティ保護が必要です。暗号化通信として、LauncherではSSL通信を使用できます。

SSL通信で使用する暗号は、鍵長が十分に長く、安全なアルゴリズムのみを有効にしてください。

詳細は、「5.2 アプリケーションのセキュリティ機能」を参照してください。

なお、性能を重視する場合やイントラネット内の通信に対する脅威を防止する場合はSSLアクセラレーターの使用を検討してください。

■サービス停止・サービス拒否

ネットワーク上の悪意のあるユーザーが、標的とするサーバーに対して、大量のアクセスを発生させてサーバーの負荷を上げます。これによりレスポンスが悪くなりサービスの質が劣化する、または正規の利用者がサービスを利用できなくなってしまう脅威があります。

サービスへの攻撃に対しては、ユーザー認証、IPアクセス制限、SSL通信、サイズ制限の機能の利用をお勧めします。

ユーザー認証

登録したユーザーからのアクセスだけを許可するように設定できます。詳細は、「5.2 アプリケーションのセキュリティ機能」を参照してください。

IPアクセス制限

特定のIPアドレスからのアクセスに対してのみ、リクエスト処理を実行させる設定をすることができます。設定の詳細は、「7.1.4 HTTPサービスの定義項目」－「仮想サーバー」のallowRemoteAddressプロパティを参照してください。

サイズ制限

バッファーオーバーフローに備えて、以下のような各種最大サイズを設定できます。設定の詳細は、「7.1.5 ネットワーク設定の定義項目」の「HTTP」を参照してください。

- リクエストラインやヘッダー：受信バッファーサイズ(header-buffer-length-bytes)
- POSTリクエストのBODY：POSTリクエストの最大サイズ(max-form-post-size-bytes)、リクエストBODYの最大サイズ(max-post-size-bytes)
- リクエストのパラメーター数：HTTPリクエストパラメーター数の最大値(max-request-parameters)
- ヘッダー数：リクエストヘッダー数の上限値(max-request-headers)、レスポンスヘッダー数の上限値(max-response-headers)

なお、アップロードするファイルの最大サイズは、アプリケーションで設定可能です。

■サーバー情報の漏洩

ネットワーク上の悪意のあるユーザーが、HTTPレスポンスヘッダーのX-Powered-Byヘッダーフィールドに存在するサーバー情報を入手し、製品情報から不正アクセスを試みることが可能となる脅威があります。

Launcherでは、X-Powered-Byヘッダーフィールドを付加しない設定で運用することを推奨します。設定の詳細は、「7.1.5 ネットワーク設定の定義項目」－「HTTP」のxpowered-by属性を参照してください。

■クロスサイト・スクリプティングによるセッションCookieの漏洩

クロスサイト・スクリプティングによりCookieが第三者に漏えいする脅威があります。

HttpOnly属性をCookieに付加すると、クライアントサイドスクリプトからCookieを取得できなくなります。このため、クロスサイト・スクリプティングの影響を軽減できます。

Launcherでは、CookieにHttpOnly属性を付加する運用を推奨します。なお、セッションCookieはデフォルトでHttpOnly属性を付加するようになっています。また、Servlet 3.0からセッションCookieにHttpOnly属性を付加できるようになっています。

■セッション情報の漏洩

セッション情報は、CookieまたはURLパラメーターに埋め込まれます。WebサーバーとWebブラウザーの間がインターネットの場合には、通信内容が傍受/改変されるおそれがあります。そのため、重要なセッション情報をやり取りする場合には、SSL通信を使用することをお勧めします。

なお、サーブレットAPIのHttpServletResponse#encodeURL(java.lang.String)、encodeRedirectURL(java.lang.String)を使用すると、URLにセッションIDが含まれる場合があります。URLにセッションIDが含まれると、Cookieを利用することができないクライアントがセッションを継続できます。ただし、URLにセッションIDが含まれる場合には、以下の脅威が存在します。

- Webブラウザーが表示中のWebページのURLを、Refererヘッダーに含めて他のWebサイトに送ることにより、セッションIDが漏洩する

そのため、セッションの追跡には、ブラウザー等のクライアントがCookieを利用できない場合などを除き、セッションCookieを使用することをお勧めします。

なお、セッションCookieを利用する場合でも、初回リクエストで上記APIを使用すると、セッションIDにURLが埋め込まれてしまいます。これらのAPIが呼び出された際に、URLにセッションIDを含めないようにするためには、Servlet 3.0で導入された以下の方法を利用してください。

- jakarta.servlet.ServletContext#setSessionTrackingModes(Set<SessionTrackingMode> sessionTrackingModes)で指定する。

例

セッションの追跡にセッションCookieのみを使用する場合のServletContextListener

```
public class MyServletContextListener implements ServletContextListener {

    @Override
    public void contextInitialized(ServletContextEvent event) {
        ServletContext sc = event.getServletContext();
        Set<SessionTrackingMode> set = new HashSet<SessionTrackingMode>();
        set.add(SessionTrackingMode.COOKIE);
        sc.setSessionTrackingModes(set);
    }

    @Override
    public void contextDestroyed(ServletContextEvent event) {
   }
}
```

- web.xmlの<tracking-mode>にCOOKIEを指定する

例

セッションの追跡にセッションCookieのみを使用する場合のweb.xml

```
<?xml version="1.0" encoding="UTF-8" ?>
<web-app xmlns="https://jakarta.ee/xml/ns/jakartaee"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="https://jakarta.ee/xml/ns/jakartaee
         web-app_5_0.xsd"
         version="5.0">
<session-config>
  <tracking-mode>COOKIE</tracking-mode>
</session-config>
</web-app>
```

注意

- アプリケーションでHttpServletResponse#encodeURL(java.lang.String)、encodeRedirectURL(java.lang.String)を使用していない場合でも、サーブレットベースのフレームワークが内部で使用する場合があります。そのため、web.xmlでtracking-modeを指定することをお勧めします。
- サーブレットAPIおよびweb.xmlの双方で指定した場合、サーブレットAPIで設定した値が優先されます。

■HTTP TRACEメソッド悪用による脅威

ネットワーク上の悪意のある人(またはマシン)が、HTTPリクエストデータ内に存在する個人情報を盗み見たり、任意のコードを実行したりする脅威があります。
このような脅威に備え、HTTP TRACEメソッドを無効にすることをお勧めします。

なお、TRACEメソッドは、ネットワーク上の診断を行うためなどに使用され、クライアント側から送出したデータをそのままレスポンスデータとして受信するHTTP/1.1のメソッドです。通常、使用されないため、無効にしても運用上の影響はありません。

5.2 アプリケーションのセキュリティ機能

アプリケーションの資源に対する不正なアクセスを防止するための機能について説明します。

5.2.1 認証

認証は、ユーザーIDとパスワードによって、正当なユーザーであるかをチェックする機能です。これにより、不当なユーザーからのアクセスを防止できます。

MicroProfileにおける認証

MicroProfileアプリケーションでは、JWTを使用してRESTエンドポイントを保護できます。JWTによる認証・認可を有効にする方法については、「3.2 JWT認証を使用するアプリケーションの開発」を参照してください。

Servletにおける認証

Servletでは以下の認証方法を使用してWebコンテナへ接続するクライアントを認証できます。

- BASIC認証
- FORM認証

認証に失敗した場合、BASIC認証は401、FORM認証は200のステータスコードが返されます。

5.2.2 認可

認証済みのユーザーが可能な操作を認可のレベルによって制限できます。認可のレベルはユーザーのロールに基づいています。例えば、会員管理アプリケーションでは、管理者にはすべての会員の個人情報を見ることを認可し、会員には自身の個人情報だけを見ることを認可します。

MicroProfileにおける認可

MicroProfileアプリケーションでは、JWTを使用したロールベースのアクセス制御をアプリケーション内に記述できます。JWTによる認証・認可を有効にする方法については、「3.2 JWT認証を使用するアプリケーションの開発」を参照してください。

Servletにおける認可

Servletで認可を行う方法には、プログラムによる制御とdeployment descriptorによる制御の2つの方法があります。

プログラムによる制御

ServletのAPIを使用して認証済みのユーザーやロールによる認可を行うコードをアプリケーション内に記述できます。
アプリケーションの開発後に変更することができないため、動作環境によってはアプリケーションの変更が必要となることがあります。

deployment descriptorによる制御

deployment descriptorに記述した内容に従って、コンテナが認可を行います。
アプリケーションの開発後に動作環境に合わせてdeployment descriptorを変更することができるため、通常はアプリケーションを変更する必要はありません。

認可に失敗した場合はステータスコード403が返されます。

5.2.3 SSL/TLS

SSL/TLSに関する設定について説明します。

5.2.3.1 Launcherの設定

証明書のニックネーム

Launcherで使用する証明書を対応するキーストアの別名で指定します。JDKのkeytoolを使用してキーストアにサーバー証明書を設定する場合、keytool -genkeypairの-aliasに指定した別名を証明書のニックネームとして設定してください。設定の詳細については「7.1.5 ネットワーク設定の定義項目」－「SSL/TLS」のcert-nickname属性を参照してください。

プロトコルバージョン

SSL/TLSのプロトコルバージョンの有効/無効を設定により変更可能です。設定の詳細については、「7.1.5 ネットワーク設定の定義項目」－「SSL/TLS」のtls12-enabled、tls13-enabled属性を参照してください。

暗号化方式

SSLのサーバー、クライアント間でコネクション確立時に選択される暗号化方式のリストを設定します。設定の詳細については、「7.1.5 ネットワーク設定の定義項目」－「SSL/TLS」のssl3-tls-ciphers属性を参照してください。

クライアント認証

クライアント認証では、SSLのサーバーがクライアントから送信された証明書を使って、クライアントの本人確認を行います。

動作の詳細については、「7.1.5 ネットワーク設定の定義項目」－「SSL/TLS」のclient-auth属性を参照してください。

5.2.3.2 証明書の設定

SSL/TLS通信に使用する証明書を変更するには、使用する証明書の別名をdomain.xmlの以下の要素のcert-nickname属性に指定します。詳細については「7.1.5 ネットワーク設定の定義項目」を参照してください。

```
/domain/configs/config/network-config/protocols/protocol/ssl
```

例

SSL/TLS通信に使用する証明書を「s1as」から「custom-cert」に変更する場合、cert-nickname属性を以下のように変更します。

変更前

```
<ssl classname="com.sun.enterprise.security.ssl.GlassfishSSLImpl" ssl3-enabled="false"
cert-nickname="s1as"></ssl>
```

変更後

```
<ssl classname="com.sun.enterprise.security.ssl.GlassfishSSLImpl" ssl3-enabled="false"
cert-nickname="custom-cert"></ssl>
```

5.2.3.3 キーストア/トラストストアの設定

キーストア/トラストストアを変更するには、以下のシステムプロパティを設定します。

プロパティ	説明
javax.net.ssl.keyStore	キーストアへのパス
javax.net.ssl.keystorePassword	キーストアのパスワード
javax.net.ssl.trustStore	トラストストアへのパス
javax.net.ssl.trustStorePassword	トラストストアのパスワード

キーストア/トラストストアへのパスを指定しなかった場合、デフォルトのキーストア/トラストストアが使用されます。キーストア/トラストストアのパスワードを指定しなかった場合、デフォルトのパスワード「changeit」が使用されます。

キーストア/トラストストアに関するこれらのシステムプロパティを設定するには、domain.xmlの以下の要素にJVMオプションを指定します。詳細については「7.1.8 Java VMの定義項目」を参照してください。

```
/domain/configs/config/java-config/jvm-options
```

例

SSL/TLS通信にユーザー定義のキーストア(${user.dir}/custom-keystore.jks)とトラストストア(${user.dir}/custom-cacerts.jks)を使用する場合、値を以下のように変更します。

変更前

```
<jvm-options>-Djavax.net.ssl.keyStore=${com.sun.aas.instanceRoot}/config/keystore.jks</jvm-options>
<jvm-options>-Djavax.net.ssl.trustStore=${com.sun.aas.instanceRoot}/config/cacerts.jks</jvm-options>
```

変更後

```
<jvm-options>-Djavax.net.ssl.keyStore=${user.dir}/custom-keystore.jks</jvm-options>
<jvm-options>-Djavax.net.ssl.trustStore=${user.dir}/custom-cacerts.jks</jvm-options>
```

パスワード「password」のキーストアおよびトラストストアを使用する場合、加えて以下の要素を指定します。

```
<jvm-options>-Djavax.net.ssl.keyStorePassword=password</jvm-options>
<jvm-options>-Djavax.net.ssl.trustStorePassword=password</jvm-options>
```

注意

キーストアおよびトラストストアには、暗号化、復号化、デジタル署名に使用する鍵が含まれるため、パスワードをdomain.xmlに記載する場合はファイルの取扱いに注意する必要があります。

5.2.4 ユーザー/グループ/ロール

ユーザー/グループ/ロールについて説明します。

ユーザー

個人またはアプリケーションプログラムを識別するIDです。

セキュリティ機能を使用する場合、ユーザーは必須です。

グループ

ユーザーの集まりです。

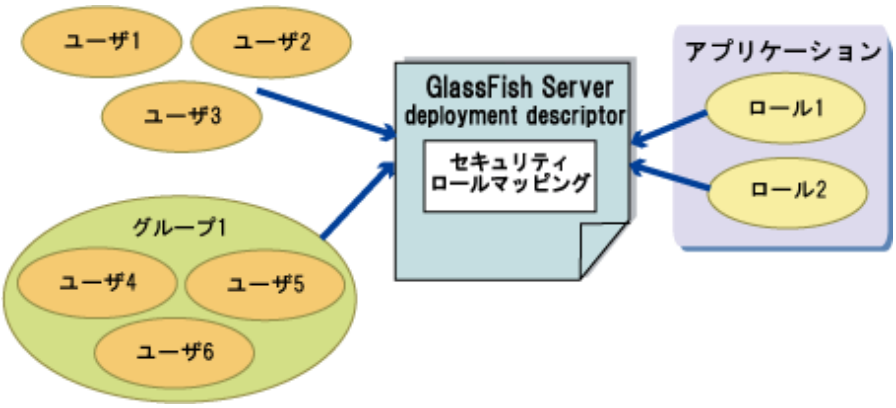
例えば、人事部門に所属するメンバーに対して同一の認可レベルを割り当てる場合に使用します。

ロール

アプリケーションによって定義される名前付きの認可レベルです。

アプリケーションはロールに基づいてアクセス制限を行います。

ユーザー/グループ/ロールは以下の関係を持ちます。ユーザー/グループはGlassFish Serverのdeployment descriptorにセキュリティロールマッピングを定義することでアプリケーションによって定義されるロールと関連付けます。



5.2.5 レルム

レルムとは、セキュリティポリシーが定義および適用されている範囲のことです。実際には、レルムとはサーバーがユーザーおよびグループの情報を格納するリポジトリーです。

本製品では、以下のレルムを使用できます。

fileレルム

レルムに登録されたユーザー資格をキーファイルとして管理します。

本製品では、あらかじめfileレルム「file」が定義されており、デフォルトレルムに設定されています。デフォルトレルムの設定については、「5.3.4 デフォルトレルムの設定」を参照してください。

5.3 セキュリティ機能を利用した運用方法

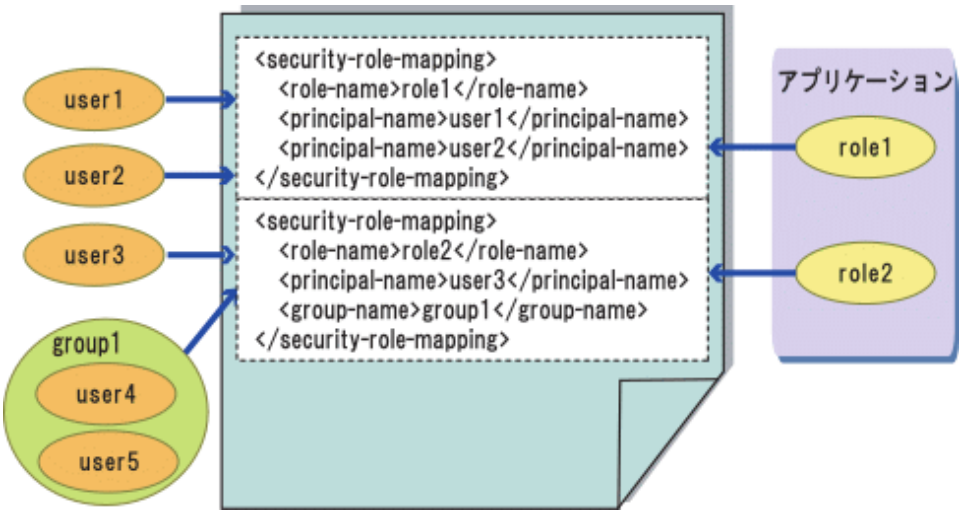
セキュリティ機能を利用した運用方法について説明します。

5.3.1 ロールとロールベースのアクセス制限の設定

Servletのセキュリティ機能を利用する場合、アプリケーション開発者はdeployment descriptorを編集し、ロールの設定およびロールによるアクセス制限の設定を行います。詳細は、Jakarta EE、Servletの仕様を参照してください。

5.3.2 ユーザー/グループをロールにマッピング

Servletのセキュリティ機能を利用する場合、配備担当者はGlassFish Serverのdeployment descriptorを編集し、ユーザーまたはグループをロールにマッピングします。ユーザー/グループをロールにマッピングすることで、deployment descriptorのsecurity-role-mapping要素で定義されたロールに対応するユーザー/グループに対して認可レベルを制御することができるようになります。詳細は、「GlassFish Serverのdeployment descriptor」の説明を参照してください。



5.3.3 レルムの作成

Launcherには、fileレルム「file」がデフォルトレルムとして事前に用意されています。

また、用途に合わせレルムを新たに作成できます。レルムを新たに作成するには、domain.xmlにauth-realm要素を追加します。

例

レルム名「realm1」のfileレルムを追加する場合のdomain.xmlの記載例です。

```
<auth-realm name="realm1" classname="com.sun.enterprise.security.auth.realm.file.FileRealm">
    <property name="jaas-context" value="fileRealm"></property>
    <property name="file" value="/path/to/keyfile"></property>
</auth-realm>
```

5.3.4 デフォルトレルムの設定

アプリケーションが使用するレルムはアプリケーションのdeployment descriptorに指定することが可能ですが、deployment descriptorに指定しない場合にデフォルトで使用するレルムを設定することができます。

このレルムをデフォルトレルムと呼びます。アプリケーションが使用するデフォルトレルムを設定する場合は、domain.xmlを使用します。

項目の詳細については、「7.1.7 セキュリティの定義項目」を参照してください。

5.3.5 fileレルムのユーザー・グループ管理

システム管理者は、Launcherのコマンドラインインターフェイスを使用してfileレルムのユーザー/グループを管理します。

fileレルムのユーザー/グループの追加・更新・削除・一覧表示などの操作は、以下のサブコマンドを利用します。

- create-file-userサブコマンド

- update-file-userサブコマンド
- delete-file-userサブコマンド
- list-file-usersサブコマンド
- list-file-groupsサブコマンド

fileレルムのユーザー/グループを管理するには、以下の手順を実施してください。

1. 初めてユーザー/グループを作成する場合、キーファイルとなる空のファイルを作成します。

例

Windows64

空のファイル「C:¥foo¥keyfile」を作成するコマンドラインの例

```
copy nul C:¥foo¥keyfile
```

Linux64

空のファイル「/foo/keyfile」を作成するコマンドラインの例

```
touch /foo/keyfile
```

2. domain.xmlのレルムの定義に、fileレルムのキーファイルのパスを指定します。fileプロパティに絶対パスで指定してください。

例

キーファイル「/foo/keyfile」を使用する場合のdomain.xmlの記載例

```
<auth-realm name="file" classname="com.sun.enterprise.security.auth.realm.file.FileRealm">
    <property name="jaas-context" value="fileRealm"></property>
    <property name="file" value="/foo/keyfile"></property>
</auth-realm>
```

3. fileレルムのユーザー/グループの管理コマンドを実行します。

例

fileレルムのユーザー「user1」を作成する例

```
java -jar launcher.jar --config-file domain.xml --execute create-file-user --passwordfile password.txt --groups
group1:group2 user1
```

fileレルムのユーザー/グループを追加・更新する際には、以下の点に注意してください。

- ユーザー名とグループ名は、1文字以上、255文字以内とし、以下の文字で構成してください。
  - 英数字
  - アンダースコア「_」
  - ハイフン「-」
  - ピリオド「.」
- パスワードには、マルチバイト文字の指定はできません。また、空白や制御コードは指定できません。

第6章 資源見積り

本章では、運用時に必要となる資源量の目安について説明します。

6.1 運用時に必要なディスク容量

最小限必要なディスク容量（JARファイルのサイズおよび、インスタンスルートディレクトリーの容量の和）は70Mバイトです。これにアプリケーションが使用する容量、安全係数などを加味し算出してください。

6.2 運用時に必要なメモリー容量

最小限必要なメモリー容量は350Mバイトです。これにアプリケーションが使用する容量、Java VMが使用するヒープサイズ、および安全係数などを加味し算出してください。

第7章 定義項目一覧

本章では、Launcherの定義項目について説明します。

7.1 domain.xml

構成ファイル(domain.xml)によって構成可能な定義項目について説明します。

7.1.1 domain.xmlの作成

domain.xmlを作成するには、サンプルを元にして必要な定義項目のみ設定を変更してください。

サンプル

サンプルは、LauncherのGitHubリポジトリーから入手できます。ファイルの入手先に関する詳細については、リポジトリー上のREADMEファイルを参照してください。

参照

Launcher GitHubリポジトリー

https://github.com/fujitsu/launcher

注意

サンプルには、運用するLauncherのバージョンに合ったものを使用してください。特定のバージョンのファイルは以下のURLから参照できます(versionはLauncherのバージョンを表します)。

```
https://github.com/fujitsu/launcher/tree/version
```

要素の変更

修正前のdomain.xmlに要素が記載されている場合は、その要素の値を変更してください。修正前のdomain.xmlに要素が記載されていない場合は、要素を追加してください。

例

java-config要素の子要素として2つのjvm-options要素を記載する例

```
<java-config>
  <jvm-options>-Djavax.net.ssl.keyStore=${com.sun.aas.instanceRoot}/config/keystore.jks</jvm-options>
  <jvm-options>-Djavax.net.ssl.trustStore=${com.sun.aas.instanceRoot}/config/cacerts.jks</jvm-options>
</java-config>
```

属性の変更

修正前のdomain.xmlに属性が記載されている場合は、その属性の値を変更してください。修正前のdomain.xmlに属性が記載されていない場合は、属性を追加してください。

なお、domain.xmlに属性を記載しない場合、本章に記載されている既定値が使用されます。

注意

- 本章に表示されている属性の既定値は、domain.xmlで属性を省略した場合に採用される値を表します。ただし、domain.xmlにあらかじめ値が設定されている属性については、その値を既定値として表示している場合があります。詳細については、各属性の注意を参照してください。
- 本章で既定値が「なし」と表示されている属性については、属性を指定しないことが特別な意味を持つ場合を除き、domain.xmlで属性を省略できません。

例

thread-pool要素のmax-thread-pool-size属性に値64を指定する例

```
<thread-pool name="http-thread-pool" max-thread-pool-size="64"/>
```

プロパティの変更

domain.xmlの要素に関連づけられたプロパティを設定するには、設定対象の要素の子要素としてproperty要素を記載します。property要素のname属性にプロパティ名、value属性にプロパティの値を指定します。

修正前のdomain.xmlにプロパティが記載されている場合は、そのプロパティの値を変更してください。修正前のdomain.xmlにプロパティが記載されていない場合は、property要素を追加してください。

なお、domain.xmlにプロパティを記載しない場合、本章に記載されている既定値が使用されます。

注意

- 本章に表示されているプロパティの既定値は、domain.xmlでプロパティを省略した場合に採用される値を表します。ただし、domain.xmlにあらかじめ値が設定されているプロパティについては、その値を既定値として表示している場合があります。詳細については、各プロパティの注意を参照してください。
- 本章で既定値が「なし」と表示されているプロパティについては、プロパティを指定しないことが特別な意味を持つ場合を除き、domain.xmlでプロパティを省略できません。

例

web-container要素のdispatcher-max-depthプロパティに値10を指定する例

```
<web-container>
  <property name="dispatcher-max-depth" value="10"/>
</web-container>
```

7.1.2 記法

本書では、domain.xmlの要素を指定するためにXPath式を使用します。

例

XPath式「/domain/configs/config/java-config/jvm-options」は、以下のdomain.xmlにおける2つのjvm-options要素を示します([...]は省略を表します)。

```
<domain [...]>
  [...]
  <configs>
    <config name="server-config">
```

```
      [...]
      <java-config>
        <jvm-options>-Djavax.net.ssl.keyStore=${com.sun.aas.instanceRoot}/config/keystore.jks</jvm-options>
        <jvm-options>-Djavax.net.ssl.trustStore=${com.sun.aas.instanceRoot}/config/cacerts.jks</jvm-options>
      </java-config>
      [...]
    </config>
  </configs>
  [...]
</domain>
```

7.1.3 定義時に利用できるプロパティ

定義済みプロパティを定義値中の変数として利用できます。
以下のようにプロパティ名を「{」と「}」で囲い、先頭に「$」を付加することでプロパティを指定できます。

```
${com.sun.aas.instanceRoot}
```

定義済みプロパティ

以下の定義済みプロパティを利用できます。
なお、これらの項目は参照だけ可能であり、ユーザーがプロパティの値を変更することはできません。

プロパティ名	説明
com.sun.aas.javaRoot	JDK/JREのインストールディレクトリーです。
com.sun.aas.instanceRoot	インスタンスルートディレクトリーです。 値の例は以下のとおりです。 Windows64 C:¥Users¥Administrator¥AppData¥Local¥Temp¥gfembed0123456789012345678tmp Linux64 /tmp/gfembed0123456789012345678tmp
com.sun.aas.instanceRootURI	インスタンスルートディレクトリーのURI形式です。 値の例は以下のとおりです。 Windows64 file:/C:/Users/Administrator/AppData/Local/Temp/gfembed0123456789012345678tmp/ Linux64 file:/tmp/gfembed0123456789012345678tmp/
com.sun.aas.hostName	マシンのホスト名です。
file.separator	ファイルパスの区切り文字です。 Windows64 「¥」が使用されます。 Linux64 「/」が使用されます。
path.separator	環境変数の区切り文字です。 Windows64 「;」が使用されます。 Linux64 「:」が使用されます。
user.dir	カレントディレクトリーです。

7.1.4 HTTPサービスの定義項目

定義項目

項目名	要素
HTTPサービス	/domain/configs/config/http-service
HTTPアクセスログ	/domain/configs/config/http-service/access-log
仮想サーバー	/domain/configs/config/http-service/virtual-server

HTTPサービス

要素

/domain/configs/config/http-service

説明

HTTPサービスの設定を表します。

属性

属性名	既定値	説明
access-logging-enabled	false	HTTPアクセスログの出力を有効にする(true)/しない(false)を指定します。

HTTPアクセスログ

要素

/domain/configs/config/http-service/access-log

説明

HTTPアクセスログの設定を表します。

属性

属性名	既定値	説明
buffer-size-bytes	32768	ログ出力のバッファーサイズを指定します。指定できる値は、5120～2147483647(Byte)です。
format	%client.name% %auth-user-name% %datetime% %request% %status% %response.length%	HTTPアクセスログに出力する項目を、1～512文字の文字列で指定します。設定できる項目については、「4.8.2 HTTPアクセスログ」を参照してください。
max-history-files	-1	ローテーションしたHTTPアクセスログを保管する世代数(ファイル数)を指定します。指定できる値は、-1～999(世代)です。-1を指定した場合、世代数は制限されません。
rotation-interval-in-minutes	1440	ログのローテーション間隔を指定します。指定できる値は、1～35791394(分)です。
rotation-policy	time	HTTPアクセスログのローテーションの方法を指定します。指定できる値は「time」のみです。
rotation-suffix	yyyy-MM-dd	ログファイル名のサフィックスを指定します。指定できる値は、java.text.SimpleDateFormatクラスのパターン文字列です。

属性名	既定値	説明
write-interval-seconds	300	ログをディスクに書き込むまでの時間を指定します。指定できる値は、0～2147483647（秒）です。0を指定した場合、常にログが書き込まれます。

仮想サーバー

要素

/domain/configs/config/http-service/virtual-server

説明

仮想サーバーの設定を表します。

属性

属性名	既定値	説明
access-log	${com.sun.aas.instanceRoot}/logs/access	HTTPアクセスログを出力するディレクトリーのパスを絶対パスで指定します。
id	なし	仮想サーバーの名前です。常に「server」を指定します。
network-listeners	なし	仮想サーバーが参照するHTTPリスナーを指定します。常に「http-listener, https-listener」を指定します。

プロパティ

プロパティ名	既定値	説明
allowRemoteAddress	なし（要求を受け付けるIPアドレスを制限しない）	Webコンテナが要求を受け付けるIPアドレスを正規表現で指定します。複数のパターンを指定する場合には、カンマ（,）で区切ります。 本属性を指定した場合は、指定したIPアドレスからのリクエストのみを受け付け、それ以外のリクエストではステータスコード403（Forbidden）が返却されます。 **注意** 正規表現ではピリオド（.）は任意の一文字として扱われます。ピリオドとして扱うにはエスケープ文字（¥またはバックスラッシュ）を付加してください。正規表現の詳細については、java.util.regex.Patternクラスの仕様を参照してください。

7.1.5 ネットワーク設定の定義項目

定義項目

項目名	要素
プロトコル	/domain/configs/config/network-config/protocols/protocol
HTTP	/domain/configs/config/network-config/protocols/protocol/http
SSL/TLS	/domain/configs/config/network-config/protocols/protocol/ssl
ネットワークリスナー	/domain/configs/config/network-config/network-listeners/network-listener
トランスポート	/domain/configs/config/network-config/transports/transport

プロトコル

要素

/domain/configs/config/network-config/protocols/protocol

説明

プロトコルの設定を表します。

属性

属性名	既定値	説明
name	なし	プロトコルの名前を指定します。HTTPリスナーの場合は「http-listener」、HTTPSリスナーの場合は「https-listener」を指定します。
security-enabled	false	リスナーのSSL/TLS通信を有効にする(true)/しない(false)を指定します。HTTPリスナーの場合は「false」、HTTPSリスナーの場合は「true」を指定します。

HTTP

要素

/domain/configs/config/network-config/protocols/protocol/http

説明

HTTPの設定を表します。

属性

属性名	既定値	説明
dns-lookup-enabled	false	DNS逆引きを有効にする(true)/しない(false)を指定します。
header-buffer-length-bytes	8192	受信バッファーサイズを指定します。受信バッファーサイズは、リクエストライン、ヘッダーを解析するために使用するバッファーのサイズを指定する設定です。指定できる値は、1～2147483647(Byte)です。
http2-enabled	true	HTTP/2を有効にする(true)/しない(false)を指定します。 **注意** HTTP/2を有効にする場合の注意について、「1.5.2 Webアプリケーションの注意事項」の「HTTP/2使用時の注意」を参照してください。
max-connections	250(注1)	キープアライブの有効/無効および、1回のコネクションで処理可能なリクエスト数を指定します。指定できる値は以下のとおりです。 ・ 0:キープアライブ無効 ・ -1:キープアライブ有効、1回のコネクションで処理可能なリクエスト数は無制限 ・ 1～2147483647:キープアライブ有効、1回のコネクションで処理可能なリクエスト数に上限あり(設定値が上限) HTTP/2リクエストの場合、設定値にかかわらず、1回のコネクションで処理可能なリクエスト数は無制限です。
max-form-post-size-bytes	2097152	POSTで送られたBODYの最大サイズをバイト単位で指定します。指定できる値は、1～2147483647(Byte)です。
max-post-size-bytes	-1	リクエストBODYの最大サイズをバイト単位で指定します。指定できる値は、-1～2147483647(Byte)です。-1を指定した場合、リクエストBODYの最大サイズのチェックを実施しません。
max-request-headers	100	リクエストヘッダー数の上限値を指定します。指定できる値は、1～2147483647です。
max-request-parameters	10000	HTTPリクエストパラメーター数の最大値を指定します。指定できる値は、-1～2147483647です。-1を指定した場合、パラメーター数は制限されません。

属性名	既定値	説明
max-response-headers	100	レスポンスヘッダー数の上限値を指定します。指定できる値は、1～2147483647です。
timeout-seconds	30	HTTP接続のタイムアウトを指定します。HTTP接続のタイムアウトは、HTTP接続を維持する秒数です。本属性で設定した値が、以下のすべてのタイムアウト値として適用されます。 ・キープアライブ接続 Webコンテナがレスポンスを返却後、次のリクエストが来るまでの間キープアライブ接続を維持する時間 ・HTTPの接続 HTTP接続後、リクエストがスレッドプールのキューに格納されるまで接続を維持する時間 ・HTTP接続キュー スレッドプールのキューに格納後、リクエストがスレッドで処理を開始されるまで接続を維持する時間 指定できる値は、10～2147483647(秒)です。
uri-encoding	UTF-8	リクエストのURIの解析に使用するエンコーディングを指定します。指定できる値は、「SJIS」、「EUC_JP」など、Javaでサポートされているエンコーディングです。
xpowered-by	false(注2)	Webコンテナが生成するHTTPレスポンスヘッダーに、X-Powered-Byヘッダーフィールドを含める(true)/含めない(false)を指定します。 注意 HTTP/2接続時には、本属性の値にかかわらず、HTTPレスポンスにX-Powered-Byヘッダーフィールドは含まれません。

注1)サンプルのdomain.xmlに本属性と既定値があらかじめ記載されています。本属性を削除した場合、値256が指定されたものとして動作します。

注2)サンプルのdomain.xmlに本属性と既定値があらかじめ記載されています。本属性を削除した場合、値trueが指定されたものとして動作します。

SSL/TLS

要素

/domain/configs/config/network-config/protocols/protocol/ssl

説明

SSL/TLSの設定を表します。

属性

属性名	既定値	説明
cert-nickname	なし	キーストアに含まれる証明書の別名を指定します。「s1as」が初期値としてあらかじめ設定されています。
classname	なし	SSL/TLS実装のクラス名を指定します。常に「com.sun.enterprise.security.ssl.GlassfishSSLImpl」を指定します。
client-auth	空文字列	クライアント認証の動作を指定します。指定できる値は以下のとおりです。 ・空文字列:クライアント認証を行いません。 ・want:クライアント証明書が送信された場合はクライアント認証を行います。送信されなかった場合はクライアント認証を行わず、処理を継続します。

属性名	既定値	説明
		・ need：必ずクライアント認証を行います。
ssl3-enabled	false	SSL3の有効(true)/無効(false)を指定します。指定できる値は「false」のみです。
tls-enabled	false(注)	TLS1.0の有効(true)/無効(false)を指定します。常に「false」を指定します。
tls11-enabled	false(注)	TLS1.1の有効(true)/無効(false)を指定します。常に「false」を指定します。
tls12-enabled	true	TLS1.2の有効(true)/無効(false)を指定します。
tls13-enabled	true	TLS1.3の有効(true)/無効(false)を指定します。
ssl3-tls-ciphers	なし(すべての暗号化方式が有効)	SSL/TLSで利用可能な暗号化方式群を指定します。 指定できる値は、Javaでサポートする暗号化方式です。指定時は暗号化方式にプレフィックスとしてプラス("+")を付加します。複数指定する場合はカンマ(",")で区切ります。 本属性を設定した場合、指定した暗号化方式のみが利用可能になります。ただし、JDK/JREがデフォルトで無効としている暗号化方式は利用できません。 HTTP/2を利用する場合、HTTP/2規約のブラックリストに該当する暗号化方式は利用できません。

注)サンプルのdomain.xmlに本属性と既定値があらかじめ記載されています。本属性を削除すると異なる既定値で動作するため、domain.xmlから本属性を削除しないでください。

ネットワークリスナー

要素

/domain/configs/config/network-config/network-listeners/network-listener

説明

ネットワークリスナーの設定を表します。

属性

属性名	既定値	説明
address	0.0.0.0	HTTPリスナーがバインドするIPアドレスを指定します。
enabled	true	HTTPリスナーを有効にする(true)/しない(false)を指定します。「false」を指定した場合は、ポートが閉塞されます。
name	なし	リスナーの名前を指定します。HTTPリスナーの場合は「http-listener」、HTTPSリスナーの場合は「https-listener」を指定します。
port	なし	リスナーが待機するポート番号を指定します。指定できる値は、1～65535です。HTTPリスナーの場合は8080、HTTPSリスナーの場合は8181が初期値としてあらかじめ設定されています。
protocol	なし	リスナーが参照するプロトコル名を指定します。HTTPリスナーの場合は「http-listener」、HTTPSリスナーの場合は「https-listener」を指定します。
thread-pool	なし	リスナーが参照するスレッドプール名を指定します。常に「http-thread-pool」を指定します。
transport	なし	リスナーが参照するトランスポート名を指定します。常に「tcp」を指定します。

トランスポート

要素

/domain/configs/config/network-config/transports/transport

説明

トランスポートの設定を表します。

属性

属性名	既定値	説明
max-connections-count	4096	最大接続数を指定します。リスナーポートのソケットのバックログ数です。接続がアクセプトされるまでの間、接続要求を保持できます。指定できる値は、1～2147483647です。
name	なし	トランスポートの名前です。常に「tcp」を指定します。
read-timeout-millis	30000	受信タイムアウトを指定します。受信タイムアウトは、リクエストのヘッダーやPOSTリクエストのBODYの受信が、受信タイムアウトで指定した時間以上途絶えた場合に、データの受信を中止するまでのタイムアウトです。指定できる値は、1～2147483647（ミリ秒）です。
write-timeout-millis	30000	送信タイムアウトを指定します。送信タイムアウトは、レスポンスをクライアントに返却するときのタイムアウトです。指定できる値は、1～2147483647（ミリ秒）です。

7.1.6 Webコンテナの定義項目

定義項目

項目名	要素
Webコンテナ	/domain/configs/config/web-container

Webコンテナ

要素

/domain/configs/config/web-container

説明

Webコンテナの設定を表します。

プロパティ

プロパティ名	既定値	説明
dispatcher-max-depth	20	1リクエスト内でのディスパッチ回数の上限値を指定します。指定できる値は、1～2147483647（回）です。

7.1.7 セキュリティの定義項目

定義項目

項目名	要素
セキュリティサービス	/domain/configs/config/security-service
レルム	/domain/configs/config/security-service/auth-realm

セキュリティサービス

要素

/domain/configs/config/security-service

説明

セキュリティサービスの設定を表します。

属性

属性名	既定値	説明
default-realm	file	デフォルトレルムを指定します。デフォルトレルムは、deployment descriptorにレルムを指定していない場合に、認証に使用されるレルムです。定義済みのレルム名を指定してください。
jacc	simple（注）	設定されているJACCプロバイダーのクラス名を指定します。常に「simple」を指定します。

注）サンプルのdomain.xmlに本属性と既定値があらかじめ記載されています。本属性を削除すると異なる既定値で動作するため、domain.xmlから本属性を削除しないでください。

レルム

要素

/domain/configs/config/security-service/auth-realm

説明

レルムの設定を表します。

属性

属性名	既定値	説明
classname	なし	レルムクラスの名前を指定します。fileレルムの場合、常に「com.sun.enterprise.security.auth.realm.file.FileRealm」を指定します。
name	なし	レルムの名前を指定します。指定できる値は、以下の条件を満たす文字列です。 ・英数字または特殊文字（'-'、'_'、'.'、'/'、';'、'#'）から構成される、かつ ・先頭の文字は英数字または'_'である

プロパティ

プロパティ名	既定値	説明
assign-groups	なし	認証済みのユーザーすべてに対して割り当てるグループを指定します。 グループ名は、1文字以上、255文字以内とし、以下の文字で構成してください。 ・英数字 ・アンダースコア「_」 ・ハイフン「-」 ・ピリオド「.」 複数グループを指定する場合は、カンマ「,」で区切って指定してください。
file	なし	fileレルムのキーファイルの格納場所（ファイル名含む）を絶対パスで設定します。fileレルム「file」の場合には「${com.sun.aas.instanceRoot}/config/keyfile」が初期値としてあらかじめ設定されています。
jaas-context	なし	レルムが使用するログインモジュールのタイプを指定します。常に「fileRealm」を指定します。

注意

サンプルのdomain.xmlにはレルム「admin-realm」、「file」、「certificate」があらかじめ定義されています。これらのうち、「file」のみ定義の変更が可能です。なお、これら以外のレルムの定義を追加することも可能です。

7.1.8 Java VMの定義項目

定義項目

項目名	要素
JVMオプション	/domain/configs/config/java-config/jvm-options

JVMオプション

要素

/domain/configs/config/java-config/jvm-options

説明

JVMオプションの設定を表します。

値

Launcher上で有効にするJVMオプションを指定します。

注意

本定義項目はキーストア/トラストストアの設定のためだけに使用してください。それ以外の場合でJVMオプションを設定するには、Launcher起動時のjavaコマンドにJVMオプションを指定してください。キーストア/トラストストアの設定については、「5.2.3.3 キーストア/トラストストアの設定」を参照してください。

7.1.9 スレッドプールの定義項目

定義項目

項目名	要素
スレッドプール	/domain/configs/config/thread-pools/thread-pool

スレッドプール

要素

/domain/configs/config/thread-pools/thread-pool

説明

スレッドプールの設定を表します。

属性

属性名	既定値	説明
idle-thread-timeout-seconds	900	スレッドプールのアイドルタイムアウトを指定します。アイドルタイムアウトは、プール内のスレッドがアイドル状態のままでいられる最長時間(秒単位)です。指定できる値は、1～2147483647(秒)です。
max-queue-size	4096	スレッドプールのキューサイズを指定します。指定できる値は、1～2147483647です。
max-thread-pool-size	64(注)	スレッドプールの最大サイズを指定します。リクエストを同時に処理できる最大数を表します。指定できる値は、1～2147483647です。
min-thread-pool-size	5	スレッドプールの最小サイズを指定します。指定できる値は、1～2147483647です。max-thread-pool-size以下の値を指定してください。
name	なし	スレッドプールの名前を指定します。HTTPリスナー/HTTPSリスナー用スレッドプールには「http-thread-pool」を指定します。

注）http-thread-poolの場合の既定値です。サンプルのdomain.xmlに本属性と既定値があらかじめ記載されています。本属性を削除した場合、値5が指定されたものとして動作します。

注意

サンプルのdomain.xmlにはスレッドプール「http-thread-pool」と「thread-pool-1」があらかじめ定義されています。これらのうち、「http-thread-pool」のみ定義の変更が可能です。

7.2 システムプロパティ

システムプロパティによって構成可能な定義項目について説明します。

7.2.1 glassfish.embedded.tmpdirシステムプロパティ

glassfish.embedded.tmpdirシステムプロパティには、インスタンスルートディレクトリーの作成先となるディレクトリーを設定します。プロパティを設定しない場合、デフォルトの一時ディレクトリーにインスタンスルートディレクトリーが作成されます。

インスタンスルートディレクトリーの詳細は「4.4 インスタンスルートディレクトリー」を参照してください。

第8章 コマンドリファレンス

本章では、Launcherのコマンドについて説明します。

8.1 コマンドライン仕様

形式

JDKのjavaコマンドを使用して、以下の形式でLauncherを実行します。

(1) アプリケーションの配備・Uber-JARファイルの作成

```
java -jar launcher.jar
    [--config-file file-path]
    [--contextroot context-root]
    [--http-listener port-number]
    [--https-listener port-number]
    [--libraries jar-filepath[,jar-filepath]*]
    [--precompilejsp[=true|=false]]
    --deploy war-file
    [--generate uber-jar-filepath]
```

(2) サブコマンドの実行

```
java -jar launcher.jar
    [--config-file file-path]
    --execute subcommand
    subcommand-args...
```

機能説明

アプリケーションの配備、Uber-JARファイルの作成、またはサブコマンドで指定された処理を実行します。

オプション

オプション名	初期値	説明
--config-file（注1）（注2）	なし	構成ファイル(domain.xml)へのパスを指定します。
--contextroot	/	アプリケーションのコンテキストルートを指定します。なお、WARファイル内のglassfish-web.xml等を使用してコンテキストルートを指定することはできません。
--deploy	なし	配備するWARファイルを指定します。--generateオプションが指定されていなければ、引数に指定したアプリケーションが配備された後、実行されます。WAR形式以外のファイルは指定できません。
--execute	なし	実行するサブコマンドを指定します。このオプションを指定した場合、アプリケーションの配備・実行は行われません。
--generate	なし	生成するUber-JARファイルのパスを指定します。このオプションを指定した場合、アプリケーションの配備・実行は行われません。
--http-listener（注1）	8080	HTTPポート番号を指定します。
--https-listener（注2）	8181	HTTPSポート番号を指定します。
--libraries	なし	アプリケーションから参照されるJARファイルのパス、またはパスのリストを指定します。複数JARファイルを指定するためにパスのリストを使用する場合は、コンマ「,」で区切ってください。相対パスを指定した場合、カレントディレクトリーからの相対パスとなります。パスにコンマ「,」を含むファイルは指定できません。

オプション名	初期値	説明
		注意 --librariesオプションで指定したJARファイルは、--generateオプションで生成されるUber-JARファイルには格納されません。Uber-JARファイル作成の詳細については、「4.3 Uber-JARファイルの作成」を参照してください。
--precompilejsp	false	JSPアプリケーションの配備時コンパイルを有効にする(true)/しない(false)を指定します。

注1)--config-fileオプションと--http-listenerオプションは同時に使用しないでください。

注2)--config-fileオプションと--https-listenerオプションは同時に使用しないでください。

オペランド

オペランド名	説明
subcommand-args	サブコマンドのオプションとオペランドを指定します。

戻り値

0: 正常終了

0以外: 異常終了

使用例

アプリケーションを配備するには、以下のように実行します。

```
java -jar launcher.jar --config-file domain.xml --libraries lib1.jar,lib2.jar --deploy application.war
```

Uber-JARファイルを作成するには、以下のように実行します。

```
java -jar launcher.jar --deploy application.war --generate uber.jar
```

サブコマンドを実行するには、以下のように実行します。

```
java -jar launcher.jar --config-file domain.xml --execute create-file-user --passwordfile password.txt user1
```

8.2 サブコマンド

Launcherのサブコマンドについて説明します。

8.2.1 サブコマンド一覧

Launcherの--executeオプションに指定できるサブコマンドの一覧を示します。

サブコマンド名	説明
create-file-user	fileレルムのユーザーの作成
delete-file-user	fileレルムのユーザーの削除
update-file-user	fileレルムのユーザーの更新
list-file-users	fileレルムのユーザーの一覧の表示
list-file-groups	fileレルムのグループの一覧の表示

8.2.2 create-file-user

形式

create-file-user --passwordfile password_file [--groups user_groups[:user_groups]...] [--authrealmname authrealm_name] username

機能説明

fileレルムのユーザーを作成します。

オプション

オプション名	初期値	説明
--passwordfile	なし	パスワードエントリーを格納しているファイルの名前を指定します。 fileユーザーのパスワードを記載した以下のエントリーを含むファイルを指定してください。 AS_ADMIN_USERPASSWORD=パスワード
--groups	なし	ユーザーが属するグループのリストを指定します。 複数のグループを指定する場合、各グループを「:」で区切って指定します。省略した場合は設定されません。
--authrealmname	なし	fileレルムの名前を指定します。 fileレルムとして設定したレルム名を指定してください。省略した場合はデフォルトレルムが選択されます。

オペランド

オペランド名	説明
username	ユーザー名を指定します。

戻り値

0: 正常終了

0以外: 異常終了

使用例

以下のように実行します。

```
java -jar launcher.jar --config-file domain.xml --execute create-file-user --passwordfile password.txt --groups
group1:group2 user1
```

8.2.3 delete-file-user

形式

delete-file-user [--authrealmname authrealm_name] username

機能説明

fileレルムのユーザーを削除します。

オプション

オプション名	初期値	説明
--authrealmname	なし	fileレルムの名前を指定します。 fileレルムとして設定したレルム名を指定してください。省略した場合はデフォルトレルムが選択されます。

オペランド

オペランド名	説明
username	ユーザー名を指定します。

戻り値

0: 正常終了

0以外: 異常終了

使用例

以下のように実行します。

```
java -jar launcher.jar --config-file domain.xml --execute delete-file-user user1
```

8.2.4 update-file-user

形式

update-file-user [--passwordfile password_file] [--groups user_groups[:user_groups]...] [--authrealmname authrealm_name] username

機能説明

fileレルムのユーザーを更新します。

オプション

オプション名	初期値	説明
--passwordfile	なし	パスワードエントリーを格納しているファイルの名前を指定します。fileユーザーのパスワードを記載した以下のエントリーを含むファイルを指定してください。 `AS_ADMIN_USERPASSWORD=パスワード` 本オプションを省略した場合、パスワードは更新されません。
--groups	なし	ユーザーが属するグループのリストを指定します。 複数のグループを指定する場合、各グループを「:」で区切って指定します。本オプションを省略した場合、グループのリストは更新されません。
--authrealmname	なし	fileレルムの名前を指定します。 fileレルムとして設定したレルム名を指定してください。省略した場合はデフォルトレルムが選択されます。

オペランド

オペランド名	説明
username	ユーザー名を指定します。

戻り値

0: 正常終了

0以外: 異常終了

使用例

以下のように実行します。

```
java -jar launcher.jar --config-file domain.xml --execute update-file-user --passwordfile password.txt --groups
group1:group2 user1
```

8.2.5 list-file-users

形式

list-file-users [--authrealmname authrealm_name]

機能説明

fileレルムのユーザーの一覧を表示します。

オプション

オプション名	初期値	説明
--authrealmname	なし	fileレルムの名前を指定します。 fileレルムとして設定したレルム名を指定してください。省略した場合はデフォルトレルムが選択されます。

戻り値

0: 正常終了

0以外: 異常終了

使用例

以下のように実行します。

```
java -jar launcher.jar --config-file domain.xml --execute list-file-users
```

8.2.6 list-file-groups

形式

list-file-groups [--name username] [--authrealmname authrealm_name]

機能説明

fileレルムのグループの一覧を表示します。

オプション

オプション名	初期値	説明
--name	なし	fileレルムのユーザーを指定します。 本オプションを指定した場合、指定したユーザーが属するグループの一覧を表示します。 本オプションを省略した場合、--authrealmnameオプションで指定したfileレルム(省略した場合、デフォルトレルム)に存在するグループの一覧を表示します。
--authrealmname	なし	fileレルムの名前を指定します。 fileレルムとして設定したレルム名を指定してください。省略した場合はデフォルトレルムが選択されます。

戻り値

0: 正常終了

0以外: 異常終了

使用例

以下のように実行します。

```
java -jar launcher.jar --config-file domain.xml --execute list-file-groups
```