Red Hat JBoss Enterprise Application Platform 7.1

スタートガイド

Red Hat JBoss Enterprise Application Platform 7.1 向け

Last Updated: 2018-08-29

Red Hat JBoss Enterprise Application Platform 7.1 スタートガイド

Red Hat JBoss Enterprise Application Platform 7.1 向け

法律上の通知

Copyright © 2018 Red Hat, Inc.

The text of and illustrations in this document are licensed by Red Hat under a Creative Commons Attribution–Share Alike 3.0 Unported license ("CC-BY-SA"). An explanation of CC-BY-SA is available at
http://creativecommons.org/licenses/by-sa/3.0/
. In accordance with CC-BY-SA, if you distribute this document or an adaptation of it, you must provide the URL for the original version.

Red Hat, as the licensor of this document, waives the right to enforce, and agrees not to assert, Section 4d of CC-BY-SA to the fullest extent permitted by applicable law.

Red Hat, Red Hat Enterprise Linux, the Shadowman logo, JBoss, OpenShift, Fedora, the Infinity logo, and RHCE are trademarks of Red Hat, Inc., registered in the United States and other countries.

Linux ® is the registered trademark of Linus Torvalds in the United States and other countries.

Java ® is a registered trademark of Oracle and/or its affiliates.

XFS ® is a trademark of Silicon Graphics International Corp. or its subsidiaries in the United States and/or other countries.

MySQL ® is a registered trademark of MySQL AB in the United States, the European Union and other countries.

Node.js ® is an official trademark of Joyent. Red Hat Software Collections is not formally related to or endorsed by the official Joyent Node.js open source or commercial project.

The OpenStack ® Word Mark and OpenStack logo are either registered trademarks/service marks or trademarks/service marks of the OpenStack Foundation, in the United States and other countries and are used with the OpenStack Foundation's permission. We are not affiliated with, endorsed or sponsored by the OpenStack Foundation, or the OpenStack community.

All other trademarks are the property of their respective owners.

概要

本ガイドは、Red Hat JBoss Enterprise Application Platform 7.1 を初めて使用するユーザーに基本的な情報を提供するためのガイドです。

目次

第1章 はじめに 4
1.1. RED HAT JBOSS ENTERPRISE APPLICATION PLATFORM 7 4
1.2. スタートガイド 4

第2章 JBOSS EAP の管理 5
2.1. JBOSS EAP のダウンロードおよびインストール 5
2.1.1. インストールの要件 5
2.1.2. JBoss EAP のダウンロード 5
2.1.3. JBoss EAP のインストール 5
2.2. JBOSS EAP の開始および停止 6
2.2.1. JBoss EAP の開始 6
JBoss EAP をスタンドアロンサーバーとして起動 6
管理対象ドメインでの JBoss EAP の起動 7
2.2.2. JBoss EAP の停止 7
JBoss EAP の対話的なインスタンスの停止 7
JBoss EAP のバックグラウンドインスタンスの停止 7
2.3. JBOSS EAP の管理 7
2.3.1. 管理ユーザー 8
2.3.1.1. 管理ユーザーの追加 8
2.3.1.2. Add-User ユーティリティーを非対話的に実行 9
複数のグループに属するユーザーの作成 9
代替プロパティーファイルの指定 9
2.3.2. 管理インターフェース 10
2.3.2.1. 管理 CLI 10
管理 CLI の起動 10
稼働中のサーバーへの接続 10
ヘルプの表示 11
管理 CLI の終了 11
システム設定の表示 11
システム設定の更新 11
サーバーの起動 11
2.3.2.2. 管理コンソール 11
2.3.3. 設定ファイル 12
2.3.3.1. スタンドアロンサーバー設定ファイル 12
2.3.3.2. 管理対象ドメイン設定ファイル 13
2.3.3.3. 設定データのバックアップ 14
2.3.3.4. 設定ファイルのスナップショット 14
スナップショットの取得 14
スナップショットのリスト 15
スナップショットの削除 15
スナップショットを用いたサーバーの起動 15
2.3.3.5. プロパティーの置き換え 15
ネストされた式 16
記述子ベースのプロパティー置換 16
2.4. ネットワークおよびポート設定 17
2.4.1. インターフェース 17
2.4.1.1. デフォルトインターフェース設定 18
2.4.1.2. インターフェースの設定 18
NIC 値があるインターフェースの追加 18
複数の条件値があるインターフェースの追加 19
インターフェース属性の更新 19

管理対象ドメインでインターフェースをサーバーに追加 19
2.4.2. ソケットバインディング 19
2.4.2.1. 管理ポート 20
2.4.2.2. デフォルトのソケットバインディング 20
スタンドアロンサーバー 21
管理対象ドメイン 21
2.4.2.3. ソケットバインディングの設定 22
2.4.2.4. ポートオフセット 23
2.4.3. IPv6 アドレス 24
IPv6 アドレスの JVM スタックの設定 24
IPv6 アドレスのインターフェース宣言の更新 24
2.5. OPTIMIZING THE JBOSS EAP SERVER CONFIGURATION 25

第3章 JBOSS EAP を使用したアプリケーションの開発 26
3.1. 概要 26
3.2. 開発環境の設定 26
3.3. クイックスタートサンプルの使用 26
3.3.1. Maven 26
3.3.2. クイックスタートでの Maven の使用 26
3.3.3. クイックスタートのダウンロードおよび実行 27
3.3.3.1. クイックスタートのダウンロード 27
3.3.3.2. JBoss Developer Studio でのクイックスタートの実行 27
3.3.3.3. コマンドラインでのクイックスタートの実行 35
3.4. クイックスタートサンプルの検証 35
3.4.1. helloworld クイックスタート 35
前提条件 35
ディレクトリー構造の確認 36
コードの確認 36
3.4.2. numberguess クイックスタート 37
前提条件 38
設定ファイルの確認 38
3.4.2.1. JSF コードの確認 39
3.4.2.2. クラスファイルの確認 40

付録A リファレンス資料 45
A.1. サーバーランタイム引数 45
A.2. ADD-USER ユーティリティー引数 48
A.3. インターフェース属性 49
A.4. ソケットバインディング属性 51
A.5. デフォルトのソケットバインディング 53

第1章 はじめに

1.1. RED HAT JBOSS ENTERPRISE APPLICATION PLATFORM 7

Red Hat JBoss Enterprise Application Platform 7 (JBoss EAP) は、オープンな標準に基いて構築され、Java Enterprise Edition 7 の仕様に準拠するミドルウェアプラットフォームです。

JBoss EAP には、必要な場合にだけサービスを有効にできるモジュール構造が含まれ、サービスの起動時間が短縮されます。

管理コンソールと管理コマンドラインインターフェース (CLI) により、XML 設定ファイルの編集が不要になり、タスクをスクリプト化および自動化する機能が追加されました。

JBoss EAP は、JBoss EAP インスタンスに対してスタンドアロンサーバーと管理対象ドメインの 2 つの操作モードを提供します。スタンドアロンサーバー操作モードでは、実行している JBoss EAP を 1 つのサーバーインスタンスとして表します。管理対象ドメイン操作モードでは、1 つの制御ポイントから複数の JBoss EAP インスタンスを管理できます。

また、JBoss EAP には、セキュアでスケーラブルな Java EE アプリケーションの迅速な開発を可能にする API と開発フレームワークが含まれます。

1.2. スタートガイド

本ガイドの目的は、ユーザーが JBoss EAP を短時間で使用できるようにすることです。JBoss EAP の基本のインストール、管理、設定などの管理タスクについて取り上げます。また、本ガイドは開発者が初めて JBoss EAP のクイックスタートを使用して Java EE 7 アプリケーションを書く場合にも役立ちます。

詳細は、JBoss EAP ドキュメントを参照してください。

第2章 JBOSS EAP の管理

2.1. JBOSS EAP のダウンロードおよびインストール

本ガイドでは、プラットフォームに依存しない ZIP インストールを使用した基本的な JBoss EAP のダウンロードおよびインストール手順を説明します。

グラフィカルインストーラーや RPM パッケージを使用して JBoss EAP をインストールする手順など、その他の詳細は『インストールガイド』を参照してください。

2.1.1. インストールの要件

JBoss EAP をインストールする前に、以下の要件を満たしていることを確認してください。

一般的な要件

- システムは「JBoss Enterprise Application Platform (EAP) 7 でサポートされる構成」に従ってサポートされます。
- Red Hat がリリースした更新とエラータを適用し、システムが最新の状態であるようにしてください。

ZIP インストールの要件

- JBoss EAP を実行するユーザーは、インストールディレクトリーへの読み書きアクセスが必要になります。
- 適切な Java Development Kit がインストールされている必要があります。
- Hewlett-Packard HP-UX では、**unzip** ユーティリティーがインストールされている必要があります。
- Windows Server では、**JAVA_HOME** と **PATH** 環境変数が設定されている必要があります。

2.1.2. JBoss EAP のダウンロード

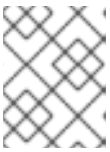
JBoss EAP の ZIP ファイルは Red Hat カスタマーポータルから入手できます。ZIP ファイルのインストールはプラットフォームに依存します。

1. Red Hat カスタマーポータルにログインします。
2. **ダウンロード**をクリックします。
3. **製品のダウンロード**リストの **Red Hat JBoss Enterprise Application Platform** をクリックします。
4. **Version** ドロップダウンメニューで **7.1** を選択します。
5. リストで **Red Hat JBoss Enterprise Application Platform 7.1.0** を見つけ、**Download** リンクをクリックします。

2.1.3. JBoss EAP のインストール

JBoss EAP の ZIP インストールファイルをダウンロードしたら、パッケージの内容を展開するとインストールできます。

1. 必要な場合は、JBoss EAP をインストールするサーバーおよび場所に ZIP ファイルを移動します。

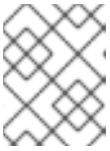


注記

JBoss EAP を実行するユーザーは、このディレクトリーへの読み書きアクセスが必要になります。

2. ZIP アーカイブを展開します。

```
$ unzip jboss-eap-7.1.0.zip
```



注記

Windows Server の場合は ZIP ファイルを右クリックし、**すべて展開** を選択します。

ZIP アーカイブを展開して作成したディレクトリーは、JBoss EAP インストールの最上位ディレクトリーとなります。このディレクトリーを **EAP_HOME** と呼びます。

2.2. JBOSS EAP の開始および停止

2.2.1. JBoss EAP の開始

JBoss EAP は、Red Hat Enterprise Linux、Windows Server、Oracle Solaris、および Hewlett-Packard HP-UX でサポートされ、スタンドアロンサーバーまたは管理対象ドメイン操作モードで実行されます。JBoss EAP を起動するコマンドは、基盤のプラットフォームと選択する操作モードによって異なります。

サーバーは最初に停止状態で起動され、必要なサービスがすべて起動するまでリクエストを受け入れません。必要なサービスがすべて起動すると、サーバーは通常の稼働状態となり、リクエストの受け入れを開始します。

JBoss EAP をスタンドアロンサーバーとして起動

```
$ EAP_HOME/bin/standalone.sh
```

注記

Windows Server の場合は、**EAP_HOME\bin\standalone.bat** スクリプトを使用します。

この起動スクリプトは、**EAP_HOME/bin/standalone.conf** ファイル (Windows Server の場合は **standalone.conf.bat**) を使用して、JVM オプションなどのデフォルト設定の一部を設定します。このファイルで設定をカスタマイズできます。

JBoss EAP はデフォルトで **standalone.xml** 設定ファイルを使用しますが、別の設定ファイルを使用して起動することもできます。利用できるスタンドアロン設定ファイルとそれらの使用方法については、「スタンドアロンサーバー設定ファイル」の項を参照してください。

使用できる起動スクリプトの引数の完全リストとそれら引数の目的については、**--help** 引数を使用するか、「サーバーランタイム引数」を参照してください。

管理対象ドメインでの JBoss EAP の起動
ドメイン内のサーバーグループのサーバーを起動する前にドメインコントローラーを起動する必要があります。このスクリプトを使用して最初にドメインコントローラーを起動した後、関連するホストコントローラーに対して使用します。

```
$ EAP_HOME/bin/domain.sh
```

注記

Windows Server の場合は **EAP_HOME\bin\domain.bat** スクリプトを使用します。

この起動スクリプトは、**EAP_HOME/bin/domain.conf** ファイル (Windows Server の場合は **standalone.conf.bat**) を使用して、JVM オプションなどのデフォルト設定の一部を設定します。このファイルで設定をカスタマイズできます。

JBoss EAP はデフォルトで **host.xml** ホスト設定ファイルを使用しますが、別の設定ファイルを使用して起動することもできます。利用できる管理対象ドメイン設定ファイルとそれらの使用方法については、「管理対象ドメイン設定ファイル」の項を参照してください。

管理対象ドメインを設定するとき、追加の引数を起動スクリプトに渡す必要があります。使用できる起動スクリプトの引数の完全リストとそれら引数の目的については、**--help** 引数を使用するか、「サーバーランタイム引数」の項を参照してください。

2.2.2. JBoss EAP の停止

JBoss EAP の停止方法は、開始した方法によって異なります。

JBoss EAP の対話的なインスタンスの停止
JBoss EAP を起動したターミナルで **Ctrl+C** を押します。

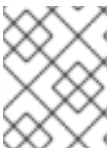
JBoss EAP のバックグラウンドインスタンスの停止
管理 CLI を使用して、稼働中のインスタンスへ接続し、サーバーをシャットダウンします。

1. 管理 CLI を起動します。

```
$ EAP_HOME/bin/jboss-cli.sh --connect
```

2. **shutdown** コマンドを実行します。

```
shutdown
```



注記

管理対象ドメインで実行している場合、**shutdown** コマンドに **--host** 引数を使用してシャットダウンする、ホスト名を指定する必要があります。

2.3. JBOSS EAP の管理

JBoss EAP は簡単な設定を使用し、スタンドアロンサーバーまたは管理対象ドメインごとに 1 つの設定ファイルを使用します。スタンドアロンサーバーのデフォルト設定は

EAP_HOME/standalone/configuration/standalone.xml ファイルに保存され、管理対象ドメインのデフォルト設定は **EAP_HOME/domain/configuration/domain.xml** ファイルに保存されます。また、ホストコントローラーのデフォルト設定は **EAP_HOME/domain/configuration/host.xml** ファイルに保存されます。

JBoss EAP はコマンドラインの管理 CLI、Web ベースの管理コンソール、Java API、または HTTP API を使用して設定できます。これらの管理インターフェースを使用して加えられた変更は自動的に永続化され、XML 設定ファイルは管理 API によって上書きされます。管理 CLI と管理コンソールの使用が推奨され、XML 設定ファイルの手作業による編集は推奨されません。

2.3.1. 管理ユーザー

デフォルトの JBoss EAP 設定はローカル認証を提供するため、ユーザーは認証の必要なくローカルホスト上で管理 CLI にアクセスできます。

しかし、リモートで管理 CLI にアクセスする場合や管理コンソールを使用する場合 (トラフィックの送信元がローカルホストであってもリモートアクセスとして見なされます) は、管理ユーザーを追加する必要があります。管理ユーザーを追加せずに管理コンソールへアクセスしようとすると、エラーメッセージが出力されます。

グラフィカルインストーラーを使用して JBoss EAP がインストールされた場合は、インストールプロセス中に管理ユーザーが作成されます。

本ガイドでは、**add-user** スクリプトを使用した JBoss EAP の簡単なユーザー管理を取り上げます。このスクリプトは既定の認証のプロパティーファイルに新しいユーザーを追加するためのユーティリティーです。

LDAP やロールベースアクセス制御 (RBAC) などの高度な認証および承認のオプションについては、JBoss EAP『**セキュリティーアーキテクチャー**』の「コア管理認証」を参照してください。

2.3.1.1. 管理ユーザーの追加

1. **add-user** ユーティリティースクリプトを実行し、プロンプトに従います。

```
$ EAP_HOME/bin/add-user.sh
```

注記

Windows Server の場合は、**EAP_HOME\bin\add-user.bat** スクリプトを使用します。

2. **ENTER** を押して、デフォルトのオプション **a** を選択し、管理ユーザーを追加します。このユーザーは **ManagementRealm** に追加され、管理コンソールまたは管理 CLI を使用して管理操作を実行する権限が与えられます。代わりに **b** を選択すると、アプリケーションに使用される **ApplicationRealm** にユーザーが追加され、特定のパーミッションは提供されません。

3. ユーザー名とパスワードを入力します。入力後、パスワードを確認するよう指示されます。

注記

ユーザー名には、以下の文字のみを使用できます。文字の数と順番は自由です。

- 英数字 (a-z、A-Z、0-9)
- ダッシュ (-)、ピリオド (.)、コンマ (,)、アットマーク(@)
- バックスラッシュ (\)
- 等号 (=)

JBoss EAP はデフォルトで弱いパスワードを許可しますが、警告が出力されます。

デフォルト動作の変更に関する詳細は、JBoss EAP『**設定ガイド**』の「Add-User ユーティリティーのパスワード制限の設定」を参照してください。

4. ユーザーが属するグループのコンマ区切りリストを入力します。ユーザーがグループに属さないようにする場合は **ENTER** を押して空白のままにします。
5. 情報を確認し、正しければ **yes** を入力します。
6. このユーザーがリモート JBoss EAP サーバーインスタンスを表すかどうかを決定します。基本的な管理ユーザーの場合は **no** を入力します。
ManagementRealm への追加が必要になることがあるユーザーのタイプの 1 つが、JBoss EAP の別のインスタンスを表すユーザーで、メンバーとしてクラスターに参加することを承認できる必要があります。この場合は、プロンプトで **yes** を選択すると、異なる設定ファイルに追加する必要がある、ユーザーのパスワードを表すハッシュ化された秘密の値が提供されます。

パラメーターを **add-user** スクリプトに渡すと、非対話的にユーザーを作成できます。ログや履歴ファイルにパスワードが表示されるため、この方法は共有システムでは推奨されません。詳細は「Add-User ユーティリティーを非対話的に実行」を参照してください。

2.3.1.2. Add-User ユーティリティーを非対話的に実行

コマンドラインで引数を渡すと **add-user** スクリプトを非対話的に実行することができます。最低でも、ユーザー名とパスワードを提供する必要があります。

警告

ログや履歴ファイルにパスワードが表示されるため、この方法は共有システムでは推奨されません。

複数のグループに属するユーザーの作成
以下のコマンドは、**guest** および **mgmtgroup** グループの管理ユーザー **mgmtuser1** を追加します。

```
$ EAP_HOME/bin/add-user.sh -u 'mgmtuser1' -p 'password1!' -g
'guest,mgmtgroup'
```

代替プロパティーファイルの指定

デフォルトでは、**add-user** スクリプトを使用して作成されたユーザーおよびグループ情報は、サーバー設定ディレクトリーにあるプロパティーファイルに保存されます。

ユーザー情報は以下のプロパティーファイルに保存されます。

- **EAP_HOME/standalone/configuration/mgmt-users.properties**
- **EAP_HOME/domain/configuration/mgmt-users.properties**

グループ情報は以下のプロパティーファイルに保存されます。

- **EAP_HOME/standalone/configuration/mgmt-groups.properties**
- **EAP_HOME/domain/configuration/mgmt-groups.properties**

これらのデフォルトディレクトリーとプロパティーファイル名は上書きできます。以下のコマンドは、ユーザープロパティーファイルの名前と場所を指定して、新しいユーザーを追加します。

```
$ EAP_HOME/bin/add-user.sh -u 'mgmtuser2' -p 'password1!' -sc
'/path/to/standaloneconfig/' -dc '/path/to/domainconfig/' -up
'newname.properties'
```

新しいユーザーは **/path/to/standaloneconfig/newname.properties** および **/path/to/domainconfig/newname.properties** にあるユーザープロパティーファイルに追加されます。これらのファイルは存在している必要があり、存在しない場合はエラーが出力されます。

使用できる **add-user** の引数の完全リストとそれら引数の目的については、**--help** 引数を使用するか、「Add-User ユーティリティー引数」の項を参照してください。

2.3.2. 管理インターフェース

2.3.2.1. 管理 CLI

管理コマンドラインインターフェース (CLI) は、JBoss EAP のコマンドライン管理ツールです。

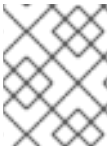
管理 CLI を使用して、サーバーの起動および停止、アプリケーションのデプロイおよびアンデプロイ、システムの設定、他の管理タスクの実行を行います。操作はバッチモードで実行でき、複数のタスクをグループとして実行できます。

ls、**cd**、**pwd** など、多くの共通するターミナルコマンドを使用できます。管理 CLI はタブ補完をサポートします。

コマンドと操作、構文、およびバッチモードでの実行を含む、管理 CLI の使用に関する詳細は、JBoss EAP『**Management CLI Guide**』を参照してください。

管理 CLI の起動

```
$ EAP_HOME/bin/jboss-cli.sh
```



注記

Windows Server の場合は、**EAP_HOME\bin\jboss-cli.bat** スクリプトを使用します。

稼働中のサーバーへの接続

```
connect
```

上記の代わりに、管理 CLI を起動し、**EAP_HOME/bin/jboss-cli.sh --connect** コマンドを使用すると 1 度に接続できます。

ヘルプの表示
以下のコマンドを実行してヘルプを表示します。

```
help
```

コマンドで **--help** フラグを使用すると、そのコマンドの使用に関する説明が表示されます。たとえば、**deploy** コマンドの使用に関する情報を表示するには、以下のコマンドを実行します。

```
deploy --help
```

管理 CLI の終了

```
quit
```

システム設定の表示
以下のコマンドは **read-attribute** 操作を使用して、データソースの例が有効になっているかどうかを表示します。

```
/subsystem=datasources/data-source=ExampleDS:read-attribute(name=enabled)
{
    "outcome" => "success",
    "result" => true
}
```

管理対象ドメインで実行している場合、コマンドの前に **/profile=PROFILE_NAME** を付けて更新するプロファイルを指定する必要があります。

```
/profile=default/subsystem=datasources/data-source=ExampleDS:read-
attribute(name=enabled)
```

システム設定の更新
以下のコマンドは **write-attribute** 操作を使用して、データソースの例を無効にします。

```
/subsystem=datasources/data-source=ExampleDS:write-
attribute(name=enabled,value=false)
```

サーバーの起動
管理対象ドメインで実行している場合、管理 CLI を使用してサーバーを起動および停止することもできます。

```
/host=HOST_NAME/server-config=server-one:start
```

2.3.2.2. 管理コンソール

管理コンソールは JBoss EAP の Web ベースの管理ツールです。

管理コンソールを使用して、サーバーの開始および停止、アプリケーションのデプロイおよびアンデプ

ロイ、システム設定の調整、サーバー設定の変更の永続化を行います。管理コンソールは管理タスクも実行でき、現在のユーザーが変更を行った後にサーバーインスタンスの再起動またはリロードが必要な場合はライブ通知も行います。

管理対象ドメインでは、同じドメイン内のサーバーインスタンスやサーバーグループをドメインコントローラーの管理コンソールから一元的に管理できます。

デフォルトの管理ポートを使用してローカルホストで稼働している JBoss EAP インスタンスの場合、Web ブラウザーを使用して http://localhost:9990/console/App.html で管理コンソールにアクセスできます。管理コンソールにアクセスできるパーミッションを持つユーザーで認証する必要があります。

管理コンソールでは、JBoss EAP スタンドアロンサーバーまたは管理対象ドメインを操作および管理するために以下のタブが提供されます。

Home (ホーム)

一般的な設定および管理タスクを行う方法を学ぶことができます。ツアーに参加して JBoss EAP 管理コンソールについてよく理解してください。

Deployments (デプロイメント)

デプロイメントを追加、削除、および有効化します。管理対象ドメインでは、デプロイメントをサーバーグループに割り当てます。

Configuration (設定)

Web サービス、メッセージング、高可用性などの機能を提供する利用可能なサブシステムを設定します。管理対象ドメインでは、異なるサブシステム設定が含まれるプロファイルを管理します。

Runtime (ランタイム)

サーバーの状態、JVM 使用率、サーバーログなどのランタイム情報を表示します。管理対象ドメインではホスト、サーバーグループ、およびサーバーを管理します。

Access Control (アクセス制御)

ロールベースアクセス制御を使用するときにユーザーとグループにロールを割り当てます。

Patching (パッチ)

JBoss EAP インスタンスにパッチを適用します。

注記

管理コンソールのツアーに参加するには、管理コンソールのホームページにある **ツアーをご覧ください** リンクをクリックします。

2.3.3. 設定ファイル

2.3.3.1. スタンドアロンサーバー設定ファイル

スタンドアロン設定ファイルは **EAP_HOME/standalone/configuration/** ディレクトリーにあります。事前定義された 5 つのプロファイル (**default**、**ha**、**full**、**full-ha**、**load-balancer**) ごとに個別のファイルがあります。

表2.1 スタンドアロン設定ファイル

設定ファイル	目的

設定ファイル	目的
standalone.xml	このスタンドアロン設定ファイルは、スタンドアロンサーバーを起動したときに使用されるデフォルト設定です。このファイルには、サブシステム、ネットワーキング、デプロイメント、ソケットバインディング、およびその他の設定詳細など、サーバーに関するすべての情報が含まれます。メッセージングや高可用性に必要なサブシステムは提供しません。
standalone-ha.xml	このスタンドアロン設定ファイルには、デフォルトのサブシステムすべてが含まれ、高可用性の **modcluster** および **jgroups** サブシステムを追加します。メッセージングに必要なサブシステムは提供しません。
standalone-full.xml	このスタンドアロン設定ファイルには、デフォルトのサブシステムすべてが含まれ、**messaging-activemq** および **iiop-openjdk** サブシステムを追加します。高可用性に必要なサブシステムは提供しません。
standalone-full-ha.xml	このスタンドアロン設定ファイルには、メッセージングおよび高可用性を含むすべてのサブシステムのサポートが含まれます。
standalone-load-balancer.xml	このスタンドアロン設定ファイルには、内蔵の mod_cluster フロントエンドロードバランサーを使用して他の JBoss EAP インスタンスの負荷を分散するために必要な最低限のサブシステムが含まれます。

デフォルトでは、スタンドアロンサーバーとして JBoss EAP を起動すると **standalone.xml** ファイルが使用されます。他の設定で JBoss EAP を起動するには **--server-config** 引数を使用します。以下に例を示します。

```
$ EAP_HOME/bin/standalone.sh --server-config=standalone-full.xml
```

2.3.3.2. 管理対象ドメイン設定ファイル

管理対象ドメインの設定ファイルは **EAP_HOME/domain/configuration/** ディレクトリーにあります。

表2.2 管理対象ドメイン設定ファイル

設定ファイル	目的
domain.xml	これは管理対象ドメインのメインの設定ファイルです。ドメインマスターのみがこのファイルを読み取ります。このファイルには、すべてのプロファイルの設定が含まれます (**default**、**ha**、**full**、**full-ha**、**load-balancer**)。
host.xml	このファイルには、管理対象ドメインの物理ホスト固有の設定情報が含まれています (ネットワークインターフェース、ソケットバインディング、ホスト名、その他のホスト固有の詳細など)。**host.xml** ファイルには、**host-master.xml** および **host-slave.xml** (詳細は下記参照) の両方の機能がすべて含まれています。
host-master.xml	このファイルには、サーバーをマスタードメインコントローラーとして実行するために必要な設定情報のみが含まれています。

設定ファイル	目的
host-slave.xml	このファイルには、サーバーを管理対象ドメインのホストコントローラーとして実行するために必要な設定情報のみが含まれています。

デフォルトでは、JBoss EAP を管理対象ドメインで起動すると **host.xml** ファイルが使用されます。他の設定で JBoss EAP を起動するには **--host-config** 引数を使用します。以下に例を示します。

```
$ EAP_HOME/bin/domain.sh --host-config=host-master.xml
```

2.3.3.3. 設定データのバックアップ

JBoss EAP のサーバー設定を後で復元するため、以下の場所にあるものはバックアップしておく必要があります。

- **EAP_HOME/standalone/configuration/**
  - ディレクトリー全体をバックアップして、スタンドアロンサーバーのユーザーデータ、サーバー設定、およびロギング設定を保存します。
- **EAP_HOME/domain/configuration/**
  - ディレクトリー全体をバックアップして、管理対象ドメインのユーザーおよびプロファイルデータ、ドメインおよびホスト設定、およびロギング設定を保存します。
- **EAP_HOME/modules/**
  - カスタムモジュールをバックアップします。
- **EAP_HOME/welcome-content/**
  - カスタムのウェルカムコンテンツをバックアップします。
- **EAP_HOME/bin/**
  - カスタムスクリプトまたは起動設定ファイルをバックアップします。

2.3.3.4. 設定ファイルのスナップショット

サーバーの保守や管理をしやすくするため、JBoss EAP は起動時に元の設定ファイルにタイムスタンプを付けたものを作成します。管理操作によってその他の設定変更が行われると、元のファイルが自動的にバックアップされ、インスタンスの作業用コピーが参照およびロールバック用に保持されます。さらに、現在のサーバー設定の現時点のコピーである設定スナップショットを撮ることができます。これらのスナップショットは管理者によって保存およびロードされます。

以下の例では、**standalone.xml** ファイルが使用されますが、同じプロセスが**domain.xml** および **host.xml** にも適用されます。

スナップショットの取得
管理 CLI を使用して、現在の設定のスナップショットを取得します。

```
:take-snapshot
{
    "outcome" => "success",
```

```
    "result" =>
"EAP_HOME/standalone/configuration/standalone_xml_history/snapshot/2015102
2-133109702standalone.xml"
}
```

スナップショットのリスト
管理 CLI を使用して、取得したすべてのスナップショットをリストします。

```
:list-snapshots
{
    "outcome" => "success",
    "result" => {
        "directory" =>
"EAP_HOME/standalone/configuration/standalone_xml_history/snapshot",
        "names" => [
            "20151022-133109702standalone.xml",
            "20151022-132715958standalone.xml"
        ]
    }
}
```

スナップショットの削除
管理 CLI を使用して、スナップショットを削除します。

```
:delete-snapshot(name=20151022-133109702standalone.xml)
```

スナップショットを用いたサーバーの起動
スナップショットまたは自動保存された設定を使用してサーバーを起動できます。

1. **EAP_HOME/standalone/configuration/standalone_xml_history** ディレクトリーへ移動し、ロードするスナップショットまたは保存された設定ファイルを確認します。
2. サーバーを起動し、選択した設定ファイルを示します。設定ディレクトリー **EAP_HOME/standalone/configuration/** からの相対パスを渡します。

```
$ EAP_HOME/bin/standalone.sh --server-
config=standalone_xml_history/snapshot/20151022-
133109702standalone.xml
```

注記

管理対象ドメインで実行している場合は、代わりに **--host-config** 引数を使用し、設定ファイルを指定します。

2.3.3.5. プロパティーの置き換え

JBoss EAP では、設定のリテラル値の代わりに式を使用して置換可能なプロパティーを定義できます。式の形式は **${PARAMETER:DEFAULT_VALUE}** になります。指定のパラメーターが設定されると、パラメーターの値が使用されます。設定されない場合は、デフォルト値が使用されます。

式の解決でサポートされるリソースはシステムプロパティー、環境変数、および vault になります。デプロイメントの場合のみ、デプロイメントアーカイブの **META-INF/jboss.properties** ファイルにリストされたプロパティーをソースとすることができます。サブデプロイメントをサポートするデプロ

イメントタイプでは、プロパティーファイルが EAR などの外部のデプロイメントにある場合は解決がすべてのサブデプロイメントに対してスコープ指定されます。プロパティーファイルがサブデプロイメントにある場合は、解決はそのサブデプロイメントのみに対してスコープ指定されます。

以下の例では、**jboss.bind.address** パラメーターが設定されていなければ、**standalone.xml** 設定ファイルによって **public** インターフェースの **inet-address** が **127.0.0.1** に設定されます。

```
<interface name="public">
    <inet-address value="${jboss.bind.address:127.0.0.1}"/>
</interface>
```

以下のコマンドを使用して、EAP をスタンドアロンサーバーとして起動するときに **jboss.bind.address** パラメータを設定できます。

```
$ EAP_HOME/bin/standalone.sh -Djboss.bind.address=IP_ADDRESS
```

ネストされた式
式はネストすることができるため、固定値の代わりにさらに高度な式を使用できます。ネストされた式の書式は、通常の式の場合と同様ですが、ある式が別の式に組み込まれます。例を以下に示します。

```
${SYSTEM_VALUE_1${SYSTEM_VALUE_2}}
```

ネストされた式は、再帰的に評価されるため、最初に **内部**の式が評価され、次に **外部**の式が評価されます。式が別の式へ解決する場合は式も再帰的になることがあり、その後解決されます。ネストされた式は式が許可された場所ならどこでも許可されます (ただし、管理 CLI コマンドを除く)。

ネストされた式が使用される例としては、データソース定義で使用されるパスワードがマスクされている場合などがあります。データソースの設定には以下のような行がある場合があります。

```
<password>${VAULT::ds_ExampleDS::password::1}</password>
```

この場合、ネストされた式を使用すると、**ds_ExampleDS** の値をシステムプロパティー (**datasource_name**) に置き換えることができます。上記の行の代わりに以下の行をデータソースの設定に使用できます。

```
<password>${VAULT::${datasource_name}::password::1}</password>
```

JBoss EAP は、最初に式 **${datasource_name}** を評価し、次にこれを外側の大きい式に入力して、結果となる式を評価します。この設定の利点は、データソースの名前が固定された設定から抽象化されることです。

記述子ベースのプロパティー置換
データソース接続パラメーターなどのアプリケーションの設定は、通常は開発デプロイメント、テストデプロイメント、および本番環境によって異なります。Java EE 仕様にはこれらの設定を外部化するメソッドが含まれていないため、このような違いはビルドシステムスクリプトで対応することがあります。JBoss EAP では、記述子ベースのプロパティー置換を使用して設定を外部的に管理できます。

記述子ベースのプロパティー置換は、記述子を基にプロパティーを置き換えるため、アプリケーションやビルドチェーンから環境に関する仮定を除外できます。環境固有の設定は、アノテーションやビルドシステムスクリプトでなく、デプロイメント記述子に指定できます。設定はファイルに指定したり、パラメーターとしてコマンドラインで提供したりできます。

ee サブシステムには、プロパティー置換が適用されたかどうかを制御する複数のフラグがあります。

JBoss 固有の記述子置換は **jboss-descriptor-property-replacement** フラグによって制御さ

れ、デフォルトで**有効**になっています。有効になっていると、以下のデプロイメント記述子でプロパティーを置換できます。

- **jboss-ejb3.xml**
- **jboss-app.xml**
- **jboss-web.xml**
- ***-jms.xml**
- ***-ds.xml**

以下の管理 CLI コマンドを使用すると、JBoss 固有の記述子でプロパティー置換を有効または無効にできます。

```
/subsystem=ee:write-attribute(name="jboss-descriptor-property-
replacement",value=VALUE)
```

Java EE の記述子置換は **spec-descriptor-property-replacement** フラグによって制御され、デフォルトで**無効**になっています。有効にすると、以下のデプロイメント記述子でプロパティーを置換できます。

- **ejb-jar.xml**
- **persistence.xml**
- **application.xml**
- **web.xml**

以下の管理 CLI コマンドを使用すると、Java EE の記述子でプロパティー置換を有効または無効にできます。

```
/subsystem=ee:write-attribute(name="spec-descriptor-property-
replacement",value=VALUE)
```

2.4. ネットワークおよびポート設定

2.4.1. インターフェース

JBoss EAP は設定全体で名前付きインターフェースを参照します。これにより、使用ごとにインターフェースの完全な詳細を必要とせず、論理名を使用して個々のインターフェース宣言を参照できます。

また、複数のマシンでネットワークインターフェースの詳細が異なる場合に管理対象ドメインの設定が容易になります。各サーバーインスタンスは、論理名グループに対応できます。

standalone.xml、**domain.xml**、および **host.xml** ファイルにはインターフェース宣言が含まれます。使用されるデフォルトの設定に応じて、複数の事前設定されたインターフェース名があります。**management** インターフェースは、HTTP 管理エンドポイントを含む、管理レイヤーが必要なすべてのコンポーネントおよびサービスに使用できます。**public** インターフェースは、アプリケーション関連のネットワーク通信すべてに使用できます。**unsecure** インターフェースは、標準設定の IIOP ソケットに使用されます。**private** インターフェースは、標準設定の JGroups ソケットに使用されます。

2.4.1.1. デフォルトインターフェース設定

```
<interfaces>
  <interface name="management">
    <inet-address value="${jboss.bind.address.management:127.0.0.1}"/>
  </interface>
  <interface name="public">
    <inet-address value="${jboss.bind.address:127.0.0.1}"/>
  </interface>
  <interface name="private">
    <inet-address value="${jboss.bind.address.private:127.0.0.1}"/>
  </interface>
  <interface name="unsecure">
    <inet-address value="${jboss.bind.address.unsecure:127.0.0.1}"/>
  </interface>
</interfaces>
```

デフォルトでは、JBoss EAP はこれらのインターフェースを **127.0.0.1** にバインドしますが、適切なプロパティーを設定すると起動時に値を上書きできます。たとえば、以下のコマンドで JBoss EAP をスタンドアロンサーバーとして起動するときに **public** インターフェースの **inet-address** を設定できます。

```
$ EAP_HOME/bin/standalone.sh -Djboss.bind.address=IP_ADDRESS
```

この代わりに、サーバー起動のコマンドラインで **-b** スイッチを使用することができます。サーバー起動オプションの詳細は、「サーバーランタイム引数」を参照してください。

重要

JBoss EAP が使用するデフォルトのネットワークインターフェースまたはポートを変更する場合、変更したインターフェースまたはポートを使用するスクリプトを変更する必要があることに注意してください。これには JBoss EAP サービススクリプトが含まれます。また、管理コンソールまたは CLI にアクセスするときに適切なインターフェースとポートを指定するようにしてください。

2.4.1.2. インターフェースの設定

ネットワークインターフェースは、物理インターフェースの論理名および選択基準を指定して宣言されます。選択基準はワイルドカードアドレスを参照したり、一致が有効となるためにインターフェースまたはアドレスで必要となる 1 つ以上の特徴のセットを指定したりできます。使用できるすべてのインターフェース選択基準は「インターフェース属性」を参照してください。

インターフェースは管理コンソールまたは管理 CLI を使用して設定できます。以下にインターフェースの追加および更新の例をいくつか示します。最初に管理 CLI コマンドを示し、その後に対応する設定 XML を示します。

NIC 値があるインターフェースの追加
NIC 値が **eth0** であるインターフェースを新たに追加します。

```
/interface=external:add(nic=eth0)
```

```
<interface name="external">
   <nic name="eth0"/>
</interface>
```

複数の条件値があるインターフェースの追加
稼働時に適切なサブネットのすべてのインターフェースまたはアドレスと一致し、マルチキャストをサポートする、ポイントツーポイントでないインターフェースを新たに追加します。

```
/interface=default:add(subnet-
match=192.168.0.0/16,up=true,multicast=true,not={point-to-point=true})
```

```
<interface name="default">
   <subnet-match value="192.168.0.0/16"/>
   <up/>
   <multicast/>
   <not>
      <point-to-point/>
   </not>
</interface>
```

インターフェース属性の更新
public インターフェースのデフォルトの**inet-address** 値を更新し、**jboss.bind.address** プロパティーによってこの値が起動時に設定されるようにします。

```
/interface=public:write-attribute(name=inet-
address,value="${jboss.bind.address:192.168.0.0}")
```

```
<interface name="public">
    <inet-address value="${jboss.bind.address:192.168.0.0}"/>
</interface>
```

管理対象ドメインでインターフェースをサーバーに追加

```
/host=HOST_NAME/server-
config=SERVER_NAME/interface=INTERFACE_NAME:add(inet-address=127.0.0.1)
```

```
<servers>
   <server name="SERVER_NAME" group="main-server-group">
      <interfaces>
         <interface name="INTERFACE_NAME">
            <inet-address value="127.0.0.1"/>
         </interface>
      </interfaces>
   </server>
</servers>
```

2.4.2. ソケットバインディング

ソケットバインディングとソケットバインディンググループを使用することにより、ネットワークポートと、JBoss EAP の設定で必要なネットワーキングインターフェースとの関係を定義できます。ソケットバインディングはソケットの名前付き設定です。ソケットバインディンググループは、ある論理名でグループ化されたソケットバインディング宣言のコレクションです。

これにより、使用ごとにソケット設定の完全な詳細を必要とせずに、設定の他のセクションが論理名でソケットバインディングを参照できるようになります。

これらの名前付き設定の宣言は **standalone.xml** および **domain.xml** 設定ファイルにあります。ス

タンドアロンサーバーにはソケットバインディンググループが 1 つのみ含まれますが、管理対象ドメインには複数のグループを含むことができます。管理対象ドメインで各サーバーグループのソケットバインディンググループを作成するか、複数のサーバーグループ間でソケットバインディンググループを共有することができます。

デフォルトで JBoss EAP によって使用されるポートは、使用されるソケットバインディンググループと、個々のデプロイメントの要件に応じて異なります。

JBoss EAP 設定のソケットバインディンググループで定義できるソケットバインディングには 3 つの種類があります。

インバウンドソケットバインディング

socket-binding 要素は、JBoss EAP サーバーのインバウンドソケットバインディングを設定するために使用されます。デフォルトの JBoss EAP 設定には、HTTP や HTTPS トラフィック用などの、事前設定された **socket-binding** 要素が複数提供されます。JBoss EAP『**Configuring Messaging**』の「Broadcast Groups」には他の例も記載されています。
この要素の属性については、インバウンドソケットバインディングの属性の表を参照してください。

リモートアウトバウンドソケットバインディング

remote-destination-outbound-socket-binding 要素は、JBoss EAP サーバーのリモートとなる宛先のアウトバウンドソケットバインディングを設定するために使用されます。デフォルトの JBoss EAP 設定には、メールサーバーに使用できるリモート宛先のソケットバインディングの例が含まれています。JBoss EAP『**Configuring Messaging**』の「Using the Integrated Artemis Resource Adapter for Remote Connections」には、他の例も記載されています。
この要素の属性については、リモートアウトバウンドソケットバインディングの属性の表を参照してください。

ローカルアウトバウンドソケットバインディング

local-destination-outbound-socket-binding 要素は、JBoss EAP サーバーのローカルとなる宛先のアウトバウンドソケットバインディングを設定するために使用されます。通常、このソケットバインディングはあまり使用されません。
この要素の属性については、ローカルアウトバウンドソケットバインディングの属性の表を参照してください。

2.4.2.1. 管理ポート

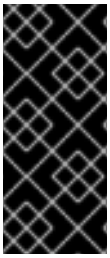
JBoss EAP 7 では、管理ポートが集約されました。JBoss EAP 7 は、管理 CLI によって使用されるネイティブ管理と、Web ベース管理コンソールによって使用される HTTP 管理の両方に **9990** ポートを使用します。JBoss EAP 6 でネイティブ管理ポートとして使用されていた **9999** ポートは使用されなくなりましたが、必要な場合は有効にできます。

管理コンソールに対して HTTPS を有効にすると、デフォルトではポート **9993** が使用されます。

2.4.2.2. デフォルトのソケットバインディング

JBoss EAP には、事前設定された 5 つのプロファイル (**default**、**ha**、**full**、**full-ha**、**load-balancer**) のソケットバインディンググループが含まれています。

デフォルトのポートや説明などのデフォルトのソケットバインディングに関する詳細情報は、「デフォルトのソケットバインディング」を参照してください。



重要

JBoss EAP が使用するデフォルトのネットワークインターフェースまたはポートを変更する場合、変更したインターフェースまたはポートを使用するスクリプトを変更する必要があることに注意してください。これには JBoss EAP サービススクリプトが含まれます。また、管理コンソールまたは CLI にアクセスするときに適切なインターフェースとポートを指定するようにしてください。

スタンドアロンサーバー
スタンドアロンサーバーとして実行されている場合、設定ファイルごとに 1 つのソケットバインディンググループのみが定義されます。各スタンドアロン設定ファイル (**standalone.xml**、**standalone-ha.xml**、**standalone-full.xml**、**standalone-full-ha.xml**、**standalone-load-balancer.xml**) は、対応するプロファイルによって使用される技術のソケットバインディングを定義します。

たとえば、デフォルトのスタンドアロン設定ファイル (**standalone.xml**) は以下のソケットバインディングを指定します。

```
<socket-binding-group name="standard-sockets" default-interface="public"
port-offset="${jboss.socket.binding.port-offset:0}">
    <socket-binding name="management-http" interface="management"
port="${jboss.management.http.port:9990}"/>
    <socket-binding name="management-https" interface="management"
port="${jboss.management.https.port:9993}"/>
    <socket-binding name="ajp" port="${jboss.ajp.port:8009}"/>
    <socket-binding name="http" port="${jboss.http.port:8080}"/>
    <socket-binding name="https" port="${jboss.https.port:8443}"/>
    <socket-binding name="txn-recovery-environment" port="4712"/>
    <socket-binding name="txn-status-manager" port="4713"/>
    <outbound-socket-binding name="mail-smtp">
        <remote-destination host="localhost" port="25"/>
    </outbound-socket-binding>
</socket-binding-group>
```

管理対象ドメイン
管理対象ドメインで実行されている場合、すべてのソケットバインディンググループは **domain.xml** ファイルで定義されます。事前定義されたソケットバインディンググループは 5 つあります。

- **standard-sockets**
- **ha-sockets**
- **full-sockets**
- **full-ha-sockets**
- **load-balancer-sockets**

各ソケットバインディンググループは、対応するプロファイルによって使用される技術のソケットバインディングを指定します。たとえば、**full-ha-sockets** ソケットバインディンググループは、高可用性のために **full-ha** プロファイルによって使用される複数の **jgroups** ソケットバインディングを定義します。

```
<socket-binding-groups>
  <socket-binding-group name="standard-sockets" default-
interface="public">
```

```
    <!-- Needed for server groups using the 'default' profile  -->
    <socket-binding name="ajp" port="${jboss.ajp.port:8009}"/>
    <socket-binding name="http" port="${jboss.http.port:8080}"/>
    <socket-binding name="https" port="${jboss.https.port:8443}"/>
    <socket-binding name="txn-recovery-environment" port="4712"/>
    <socket-binding name="txn-status-manager" port="4713"/>
    <outbound-socket-binding name="mail-smtp">
      <remote-destination host="localhost" port="25"/>
    </outbound-socket-binding>
  </socket-binding-group>
  <socket-binding-group name="ha-sockets" default-interface="public">
    <!-- Needed for server groups using the 'ha' profile  -->
    ...
  </socket-binding-group>
  <socket-binding-group name="full-sockets" default-interface="public">
    <!-- Needed for server groups using the 'full' profile  -->
    ...
  </socket-binding-group>
  <socket-binding-group name="full-ha-sockets" default-interface="public">
    <!-- Needed for server groups using the 'full-ha' profile  -->
    <socket-binding name="ajp" port="${jboss.ajp.port:8009}"/>
    <socket-binding name="http" port="${jboss.http.port:8080}"/>
    <socket-binding name="https" port="${jboss.https.port:8443}"/>
    <socket-binding name="iiop" interface="unsecure" port="3528"/>
    <socket-binding name="iiop-ssl" interface="unsecure" port="3529"/>
    <socket-binding name="jgroups-mping" interface="private" port="0"
multicast-address="${jboss.default.multicast.address:230.0.0.4}"
multicast-port="45700"/>
    <socket-binding name="jgroups-tcp" interface="private" port="7600"/>
    <socket-binding name="jgroups-udp" interface="private" port="55200"
multicast-address="${jboss.default.multicast.address:230.0.0.4}"
multicast-port="45688"/>
    <socket-binding name="modcluster" port="0" multicast-
address="224.0.1.105" multicast-port="23364"/>
    <socket-binding name="txn-recovery-environment" port="4712"/>
    <socket-binding name="txn-status-manager" port="4713"/>
    <outbound-socket-binding name="mail-smtp">
      <remote-destination host="localhost" port="25"/>
    </outbound-socket-binding>
  </socket-binding-group>
  <socket-binding-group name="load-balancer-sockets" default-
interface="public">
    <!-- Needed for server groups using the 'load-balancer' profile  -->
    ...
  </socket-binding-group>
</socket-binding-groups>
```

注記

管理インターフェースのソケット設定は、ドメインコントローラーの **host.xml** ファイルに定義されます。

2.4.2.3. ソケットバインディングの設定

ソケットバインディングを設定するとき、**port** および **interface** 属性や、**multicast-address** および **multicast-port** などのマルチキャスト設定を設定できます。使用できるソケットバインディング属性すべての詳細は、「ソケットバインディング属性」を参照してください。

ソケットバインディングは管理コンソールまたは管理 CLI を使用して設定できます。以下の手順では、ソケットバインディンググループの追加、ソケットバインディングの追加、および管理 CLI を使用したソケットバインディングの設定を行います。

1. 新しいソケットバインディンググループを追加します。これは、スタンドアロンサーバーとして実行している場合は追加できないことに注意してください。

```
/socket-binding-group=new-sockets:add(default-interface=public)
```

2. ソケットバインディングを追加します。

```
/socket-binding-group=new-sockets/socket-binding=new-socket-
binding:add(port=1234)
```

3. ソケットバインディンググループによって設定されるデフォルト以外のインターフェースを使用するよう、ソケットバインディングを変更します。

```
/socket-binding-group=new-sockets/socket-binding=new-socket-
binding:write-attribute(name=interface,value=unsecure)
```

以下の例は、上記の手順の完了後に XML 設定がどのようになるかを示しています。

```
<socket-binding-groups>
    ...
    <socket-binding-group name="new-sockets" default-interface="public">
        <socket-binding name="new-socket-binding" interface="unsecure"
port="1234"/>
    </socket-binding-group>
</socket-binding-groups>
```

2.4.2.4. ポートオフセット

ポートオフセットとは、該当するサーバーのソケットバインディンググループに指定されたすべてのポート値に追加される数値のオフセットのことです。これにより、同じホストの別のサーバーとの競合を防ぐため、サーバーはソケットバインディンググループに定義されたポート値とオフセットを継承できるようになります。たとえば、ソケットバインディンググループの HTTP ポートが **8080** で、サーバーが **100** をポートオフセットとして使用する場合、HTTP ポートは**8180** になります。

管理 CLI を使用して管理対象ドメインのサーバーにポートオフセットとして **250** を設定する例を以下に示します。

```
/host=master/server-config=server-two/:write-attribute(name=socket-
binding-port-offset,value=250)
```

ポートオフセットは、管理対象ドメインのサーバーと、同じホストで複数のスタンドアロンサーバーを実行する場合に使用できます。

jboss.socket.binding.port-offset プロパティーを使用してスタンドアロンサーバーを起動するときにポートオフセットを渡すことができます。

```
$ EAP_HOME/bin/standalone.sh -Djboss.socket.binding.port-offset=100
```

2.4.3. IPv6 アドレス

デフォルトでは、JBoss EAP は IPv4 アドレスを使用して実行するように設定されます。以下の手順では、IPv6 アドレスを使用して実行するよう JBoss EAP を設定する方法を示します。

IPv6 アドレスの JVM スタックの設定
IPv6 アドレスを優先するように、起動設定を更新します。

1. 起動設定ファイルを開きます。
   - スタンドアロンサーバーとして実行している場合は、**EAP_HOME/bin/standalone.conf** ファイル (Windows Server の場合は **standalone.conf.bat**) を編集します。
   - 管理対象ドメインで実行している場合は、**EAP_HOME/bin/domain.conf** ファイル (Windows Server の場合は **domain.conf.bat**) を編集します。
2. **java.net.preferIPv4Stack** プロパティーを **false** に設定します。

   ```
   -Djava.net.preferIPv4Stack=false
   ```
3. **java.net.preferIPv6Addresses** プロパティーを追加し、**true** に設定します。

   ```
   -Djava.net.preferIPv6Addresses=true
   ```

以下の例は、上記の変更を行った後に起動設定ファイルの JVM オプションがどのようになるかを示しています。

```
# Specify options to pass to the Java VM.
#
if [ "x$JAVA_OPTS" = "x" ]; then
   JAVA_OPTS="-Xms1303m -Xmx1303m -Djava.net.preferIPv4Stack=false"
   JAVA_OPTS="$JAVA_OPTS -
Djboss.modules.system.pkgs=$JBOSS_MODULES_SYSTEM_PKGS -
Djava.awt.headless=true"
   JAVA_OPTS="$JAVA_OPTS -Djava.net.preferIPv6Addresses=true"
else
```

IPv6 アドレスのインターフェース宣言の更新
設定のデフォルトのインターフェース値は、IPv6 アドレスに変更できます。たとえば、以下の管理 CLI コマンドは **management** インターフェースを IPv6 ループバックアドレス (**::1**) に設定します。

```
/interface=management:write-attribute(name=inet-
address,value="${jboss.bind.address.management:[::1]}")
```

以下の例は、上記のコマンド実行後に XML 設定がどのようになるかを示しています。

```
<interfaces>
    <interface name="management">
        <inet-address value="${jboss.bind.address.management:[::1]}"/>
    </interface>
    ....
</interfaces>
```

2.5. OPTIMIZING THE JBOSS EAP SERVER CONFIGURATION

Once you have installed the JBoss EAP server, and you have created a management user, Red Hat recommends that you optimize your server configuration.

Make sure you review information in the **Performance Tuning Guide** for information about how to optimize the server configuration to avoid common problems when deploying applications in a production environment. Common optimizations include setting ulimits, enabling garbage collection, creating Java heap dumps, and adjusting the thread pool size.

It is also a good idea to apply any existing patches for your release of the product. Each patch for EAP contains numerous bug fixes. For more information, see Patching JBoss EAP in the **Patching and Upgrading Guide** for JBoss EAP.

第3章 JBOSS EAP を使用したアプリケーションの開発

3.1. 概要

本ガイドは、Red Hat JBoss Developer Studio と JBoss EAP 7 クイックスタートを使用してアプリケーションの開発を始めるための情報を提供します。

Red Hat JBoss Developer Studio は、JBoss アプリケーション開発のプラグインを統合する Eclipse ベースの統合開発環境 (IDE) です。JBoss Developer Studio では、JBoss 固有のウィザードやアプリケーションを JBoss EAP サーバーへデプロイする機能を使用してアプリケーション開発を補助します。異なる Java EE 7 技術を使用してアプリケーションの開発を始められるように、JBoss EAP 7 には多くのクイックスタートコードサンプルが含まれています。

3.2. 開発環境の設定

JBoss EAP 7.1 では、JBoss Developer Studio 11.0 以上の使用が推奨されます。

1. JBoss Developer Studio をダウンロードし、インストールします。
   手順については、JBoss Developer Studio『**Installation Guide**』の「Installing JBoss Developer Studio Stand-alone Using the Installer」を参照してください。

2. JBoss Developer Studio で JBoss EAP サーバーを設定します。
   手順については、『**Getting Started with JBoss Developer Studio Tools**』の「Using Runtime Detection to Set Up JBoss EAP from within the IDE」を参照してください。

3.3. クイックスタートサンプルの使用

JBoss EAP で提供されるクイックスタートサンプルは Maven プロジェクトです。

3.3.1. Maven

Apache Maven は、ソフトウェアプロジェクトの作成、管理、および構築を行う Java アプリケーションの開発で使用される分散型ビルド自動化ツールです。Maven は Project Object Model (POM) と呼ばれる標準の設定ファイルを利用して、プロジェクトの定義や構築プロセスの管理を行います。POM はモジュールやコンポーネントの依存関係、ビルドの順番、結果となるプロジェクトパッケージングのターゲットを記述し、XML ファイルを使用して出力します。こうすることで、プロジェクトが正しく統一された状態で構築されるようにします。

Maven は、リポジトリーを使用してアーカイブを行います。Maven リポジトリーには Java ライブラリー、プラグイン、およびその他のビルドアーティファクトが格納されています。デフォルトのパブリックリポジトリーは Maven 2 Central Repository ですが、複数の開発チームの間で共通のアーティファクトを共有する目的で、社内のプライベートおよび内部リポジトリーとすることが可能です。また、サードパーティーのリポジトリーも利用できます。詳細は Apache Maven プロジェクトおよび『Introduction to Repositories』ガイドを参照してください。

JBoss EAP には、Java EE 開発者が JBoss EAP 6 でアプリケーションを構築する際に使用する要件の多くが含まれる Maven リポジトリーが含まれます。

JBoss EAP で Maven を使用する方法の詳細は、JBoss EAP『**開発ガイド**』の「JBoss EAP で Maven を使用」を参照してください。

3.3.2. クイックスタートでの Maven の使用

アプリケーションをビルドし、JBoss EAP 7 にデプロイするのに必要なアーティファクトと依存関係はパブリックリポジトリーでホストされます。JBoss EAP 7 のクイックスタートでは、Maven **settings.xml** ファイルを設定して、クイックスタートをビルドするときにこれらのリポジトリーを使用する必要がなくなりました。Maven リポジトリーはクイックスタートプロジェクト POM ファイルに設定されるようになりました。この設定方法は、クイックスタートを容易に使えるようにするために提供されますが、ビルドが遅くなる可能性があるため、通常は本番プロジェクトでの使用は推奨されません。

Red Hat JBoss Developer Studio には Maven が含まれるため、個別にダウンロードおよびインストールする必要はありません。JBoss Developer Studio のバージョン 11.0 以上を使用することが推奨されます。

Maven コマンドラインを使用してアプリケーションをビルドおよびデプロイする場合は、最初に Apache Maven プロジェクトから Maven をダウンロードし、Maven のドキュメントに記載されている手順に従ってインストールします。

3.3.3. クイックスタートのダウンロードおよび実行

3.3.3.1. クイックスタートのダウンロード

JBoss EAP には、さまざまな Java EE 7 の技術を使用してアプリケーションを作成するのに役立つ包括的なクイックスタートコードサンプルが含まれています。クイックスタートは Red Hat カスタマーポータルからダウンロードできます。

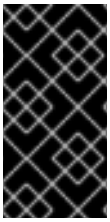
1. Red Hat カスタマーポータルにログインします。
2. **ダウンロード**をクリックします。
3. **製品のダウンロード**リストで**Red Hat JBoss Enterprise Application Platform** をクリックします。
4. **Version** ドロップダウンメニューで**7.1** を選択します。
5. 表で **Red Hat JBoss Enterprise Application Platform 7.1.0 Quickstarts** を見つけ、**Download** をクリックします。
6. ZIP ファイルを希望の場所に保存します。
7. Zip ファイルを展開します。

3.3.3.2. JBoss Developer Studio でのクイックスタートの実行

クイックスタートがダウンロードされたら、JBoss Developer Studio にインポートし、JBoss EAP にデプロイできます。

クイックスタートの JBoss Developer Studio へのインポート

各クイックスタートには、プロジェクトおよび設定情報が含まれる POM ファイルが同梱されています。この POM ファイルを使用すると、簡単にクイックスタートを JBoss Developer Studio にインポートできます。

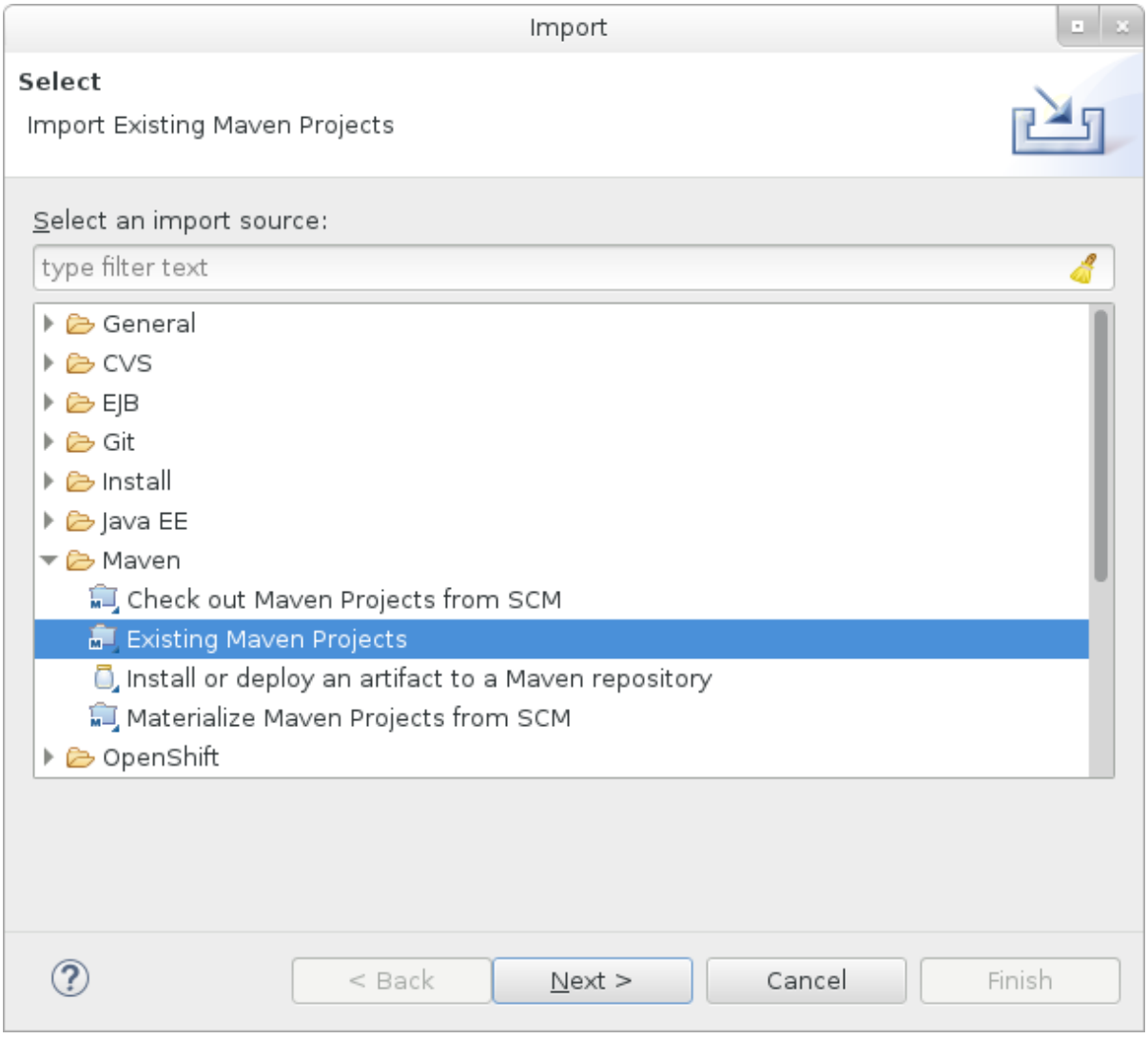


重要

JBoss Developer Studio へのインポート時にクイックスタートプロジェクトフォルダーが IDE ワークスペース内にある場合、IDE は無効なプロジェクト名と WAR アーカイブ名を生成します。作業を開始する前に、クイックスタートプロジェクトフォルダーが IDE ワークスペースの外部にあることを確認してください。

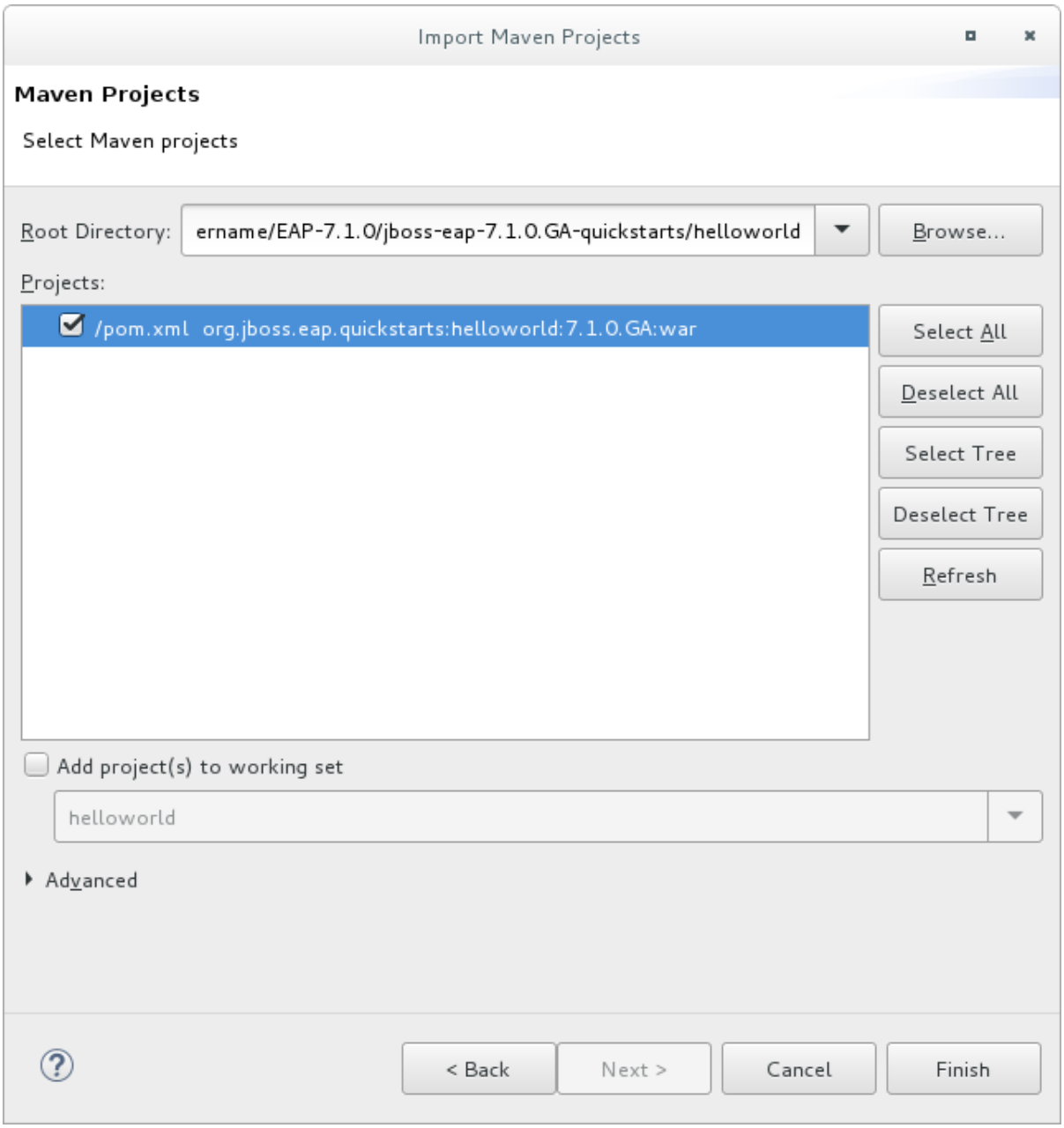
1. JBoss Developer Studio を起動します。
2. **File** → **Import** と選択します。
3. **Maven** → **Existing Maven Projects** と選択し、**Next** をクリックします。

図3.1 既存の Maven プロジェクトのインポート



4. 希望のクイックスタートのディレクトリー (**helloworld** など) を参照し、**OK** をクリックします。**Projects** リストボックスに、選択したクイックスタートプロジェクトの**pom.xml** ファイルが示されます。

図3.2 Maven プロジェクトの選択



5. **Finish** をクリックします。

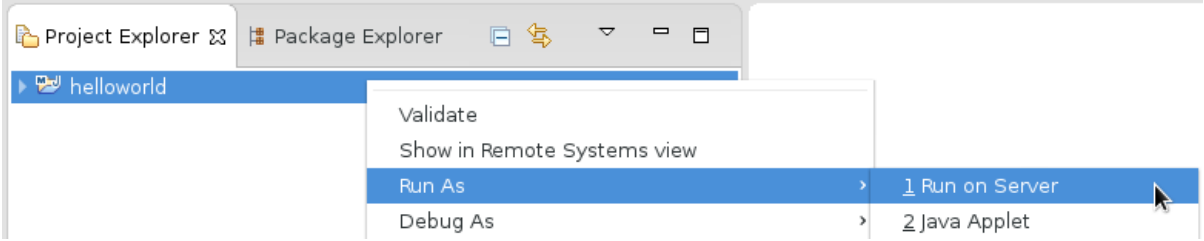
helloworld クイックスタートの実行

helloworld クイックスタートを実行すると、JBoss EAP サーバーが適切に設定および実行されたことを簡単に検証できます。

1. サーバーを定義していない場合は、JBoss EAP サーバーを JBoss Developer Studio に追加します。『**Getting Started with JBoss Developer Studio Tools**』の「Using Runtime Detection to Set Up JBoss EAP from within the IDE」を参照してください。

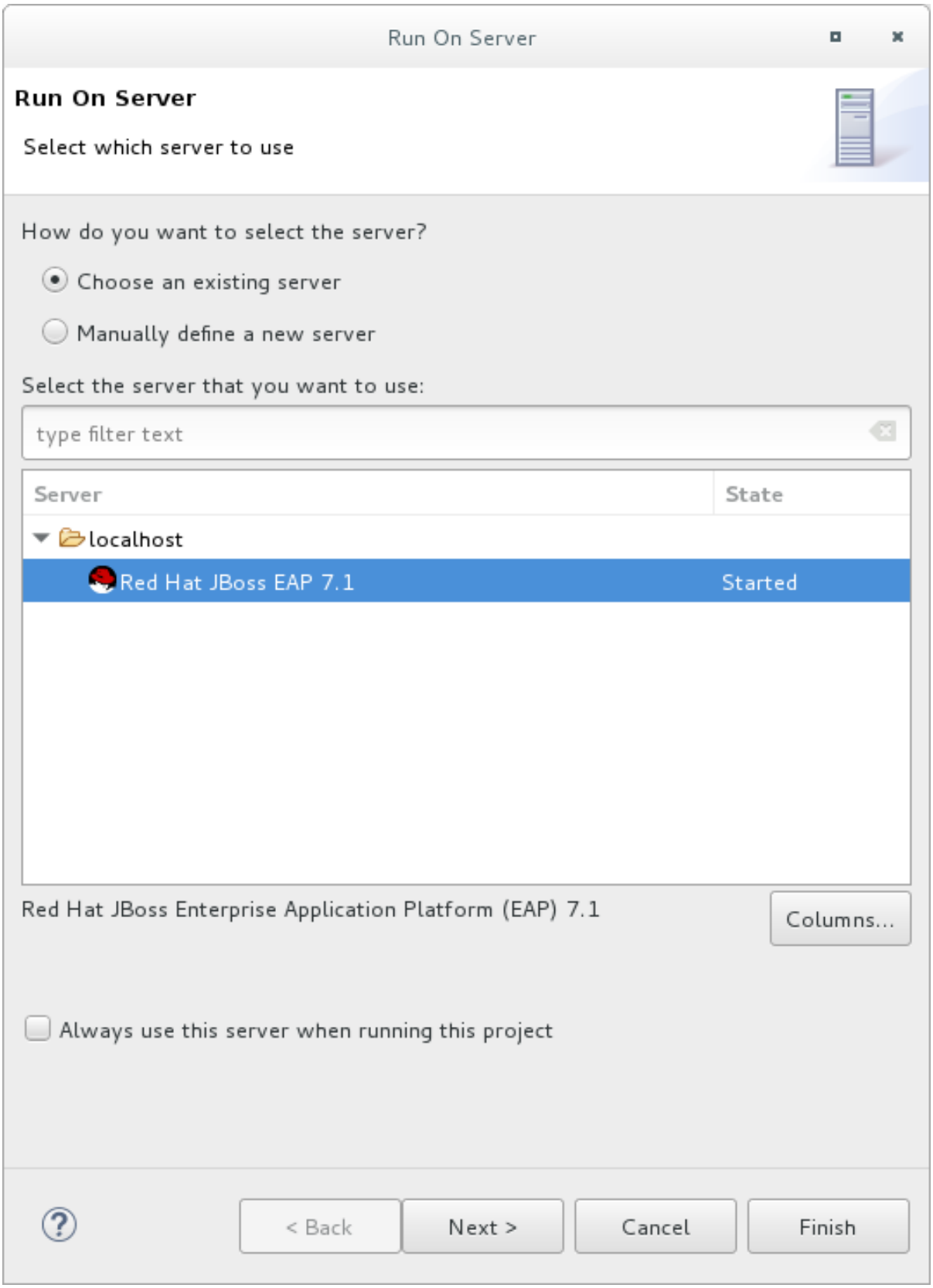
2. **Project Explorer** タブの **helloworld** プロジェクトを右クリックし、**Run As → Run on Server** と選択します。

図3.3 Run As - Run on Server



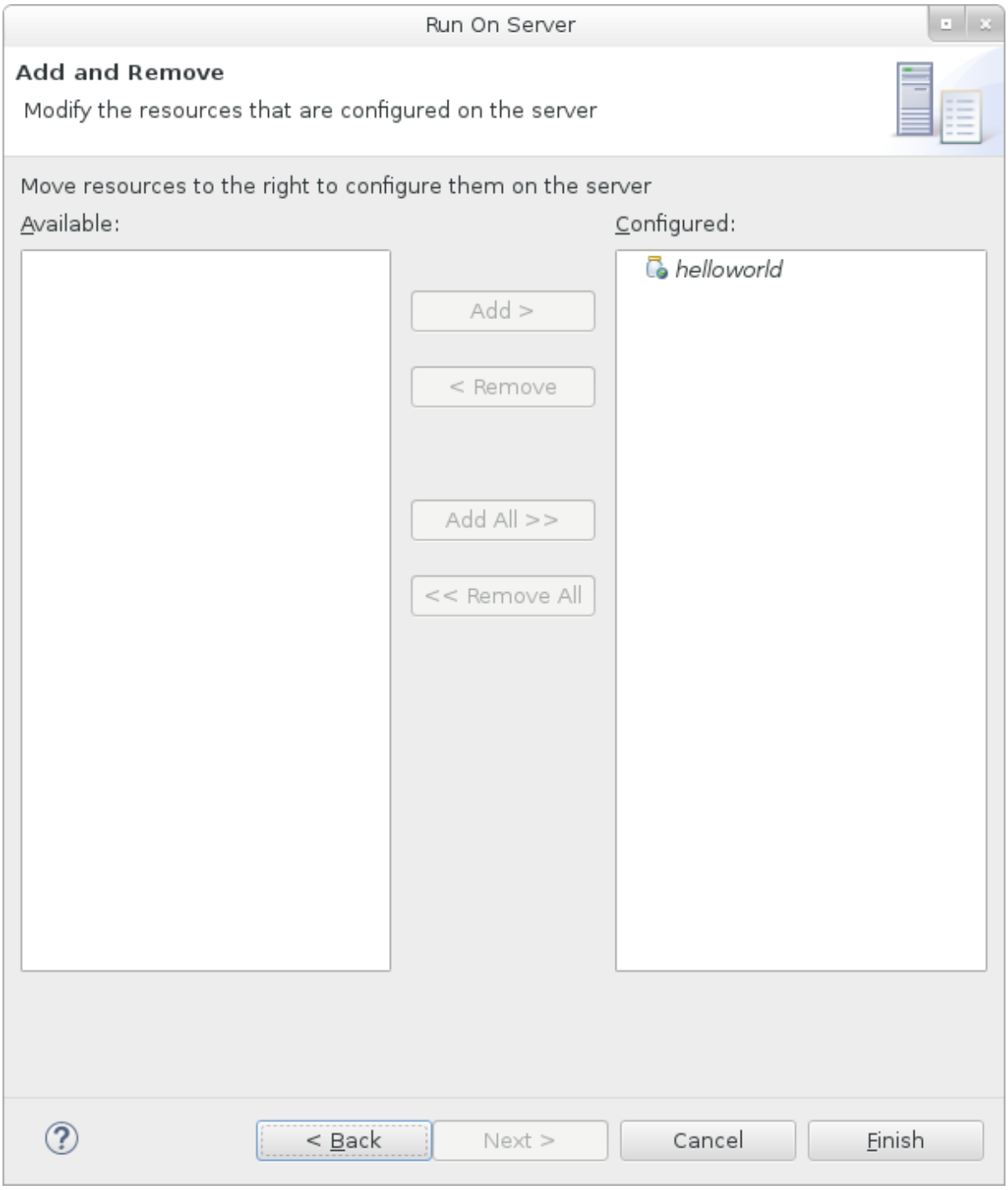
3. サーバーリストから JBoss EAP 7.1 サーバーを選択し、**Next** をクリックします。

図3.4 Run on Server



4. **helloworld** クイックスタートはすでにリストされ、サーバーで設定できる状態です。**Finish** をクリックしてクイックスタートをデプロイします。

図3.5 サーバーで設定されたリソースの変更



5. 結果を検証します。

- **Server** タブで、JBoss EAP 7.1 サーバーの状態が**Started** に変わります。
- **Console** タブに、JBoss EAP サーバーの起動と**helloworld** クイックスタートのデプロイメントに関するメッセージが表示されます。

```
WFLYUT0021: Registered web context: /helloworld
WFLYSRV0010: Deployed "helloworld.war" (runtime-name :
"helloworld.war")
```

- **helloworld** アプリケーションは http://localhost:8080/helloworld で使用でき、**Hello World!** というテキストが表示されます。

helloworld クイックスタートの詳細は、「helloworld クイックスタート」を参照してください。

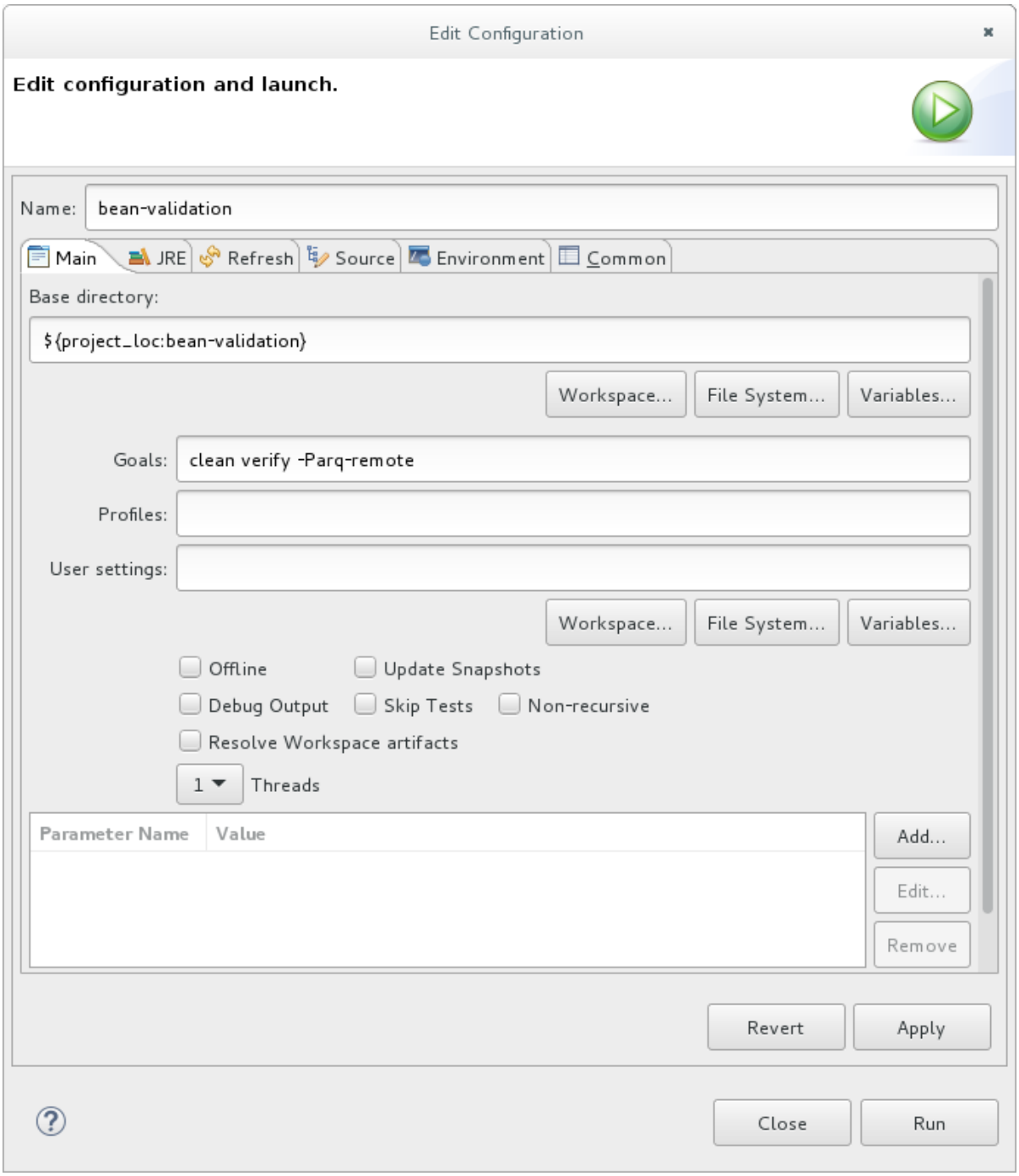
bean-validation クイックスタートの実行

bean-validation などの一部のクイックスタートは、ユーザーインターフェースレイヤーの代わりに Arquillian テストを提供して機能を示します。

1. **bean-validation** クイックスタートを JBoss Developer Studio にインポートします。
2. **Servers** タブでサーバーを右クリックし、**Start** を選択して JBoss EAP サーバーを起動します。**Servers** タブが表示されない場合またはサーバーが未定義の場合は、JBoss EAP サーバーを JBoss Developer Studio に追加します。『**Getting Started with JBoss Developer Studio Tools**』の「Using Runtime Detection to Set Up JBoss EAP from within the IDE」を参照してください。
3. **Project Explorer** タブの **bean-validation** プロジェクトを右クリックし、**Run As** → **Maven Build** と選択します。
4. 以下を **Goals** 入力フィールドに入力し、**Run** を実行します。

```
clean verify -Parq-remote
```

図3.6 設定の編集



5. 結果を検証します。
Console タブに **bean-validation** Arquillian テストの結果が表示されます。

```
-------------------------------------------------------
 T E S T S
-------------------------------------------------------
Running
org.jboss.as.quickstarts.bean_validation.test.MemberValidationTest
Tests run: 5, Failures: 0, Errors: 0, Skipped: 0, Time elapsed:
2.189 sec

Results :

Tests run: 5, Failures: 0, Errors: 0, Skipped: 0
```

```
[INFO] ------------------------------------------------------------
-------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------
-------------
```

3.3.3.3. コマンドラインでのクイックスタートの実行

Maven を使用すると、コマンドラインから簡単にクイックスタートをビルドおよびデプロイできます。Maven がインストールされていない場合は Apache Maven プロジェクトを参照し、ダウンロードとインストールを行ってください。

README.md ファイルは、システム要件、Maven の設定、ユーザーの追加、およびクイックスタートの実行に関する一般的な情報が含まれるクイックスタートのルートディレクトリーにあります。

各クイックスタートには、クリックスタートを実行するための特定の手順と Maven コマンドが含まれる独自の **README.md** ファイルも含まれます。

コマンドラインでの helloworld クイックスタートの実行

1. **helloworld** クイックスタートのルートディレクトリーにある**README.md** ファイルを確認します。
2. JBoss EAP サーバーを起動します。

   ```
   $ EAP_HOME/bin/standalone.sh
   ```

3. **helloworld** クイックスタートディレクトリーへ移動します。
4. クイックスタートの **README.md** ファイルにある Maven コマンドを使用して、クイックスタートをビルドおよびデプロイします。

   ```
   $ mvn clean install wildfly:deploy
   ```

5. **helloworld** アプリケーションは http://localhost:8080/helloworld で使用でき、**Hello World!** というテキストが表示されます。

3.4. クイックスタートサンプルの検証

3.4.1. helloworld クイックスタート

helloworld クイックスタートは JBoss EAP に単純なサーブレットをデプロイする方法を示します。ビジネスロジックは CDI (Contexts and Dependency Injection: コンテキストと依存関係の挿入) Bean として提供されるサービスにカプセル化され、サーブレットに挿入されます。このクイックスタートに基づいて、サーバーを適切に設定および起動することができます。

コマンドラインを使用してこのクイックスタートをビルドしデプロイする手順の詳細については、**helloworld** クイックスタートディレクトリーのルートにある**README.html** ファイルを参照してください。このトピックでは、Red Hat JBoss Developer Studio を使用してクイックスタートを実行する方法を説明します (Red Hat JBoss Developer Studio がインストールされ、Maven が設定された状態で **helloworld** クイックスタートがインポートされ、正常に実行されたことを前提とします)。

前提条件

- JBoss Developer Studio をインストールします。手順については、JBoss Developer Studio『**Installation Guide**』の「Installing JBoss Developer Studio Stand-alone Using the Installer」を参照してください。
- **helloworld** クイックスタートを実行します。手順については「JBoss Developer Studio でのクイックスタートの実行」を参照してください。
- Web ブラウザーを開いて、http://localhost:8080/helloworld でアプリケーションにアクセスし、**helloworld** クイックスタートが正常に JBoss EAP にデプロイされたことを確認します。

ディレクトリー構造の確認
helloworld クイックスタートのコードは**QUICKSTART_HOME/helloworld/** ディレクトリーにあります。**helloworld** クイックスタートはサーブレットと CDI Bean によって構成されます。また、バージョン番号が 1.1 であり、**bean-discovery-mode** が **all** であるアプリケーションの**WEB-INF** ディレクトリーに **beans.xml** ファイルが含まれます。このマーカーファイルにより、WAR が Bean アーカイブとして識別され、JBoss EAP がこのアプリケーションで Bean を検索し、CDI をアクティベートするよう指示されます。

src/main/webapp/ ディレクトリーにクイックスタートのファイルが含まれます。このサンプルのすべての設定ファイルは、**src/main/webapp/** 内の **WEB-INF/** ディレクトリーにあり、**beans.xml** ファイルが含まれます。**src/main/webapp/** ディレクトリーには **index.html** ファイルも含まれています。このファイルは簡単なメタリフレッシュ (meta refresh) を使用して、ユーザーのブラウザーを http://localhost:8080/helloworld/HelloWorld にあるサーブレットにリダイレクトします。このクイックスタートには **web.xml** ファイルは必要ありません。

コードの確認
パッケージの宣言とインポートはこれらのリストには含まれていません。完全なリストはクイックスタートのソースコードにあります。

1. **HelloWorldServlet** コードを確認します。
   HelloWorldServlet.java ファイルは **src/main/java/org/jboss/as/quickstarts/helloworld/** ディレクトリーにあります。このサーブレットが情報をブラウザーに送ります。

   例: HelloWorldServlet クラスコード

```
42 @SuppressWarnings("serial")
43 @WebServlet("/HelloWorld")
44 public class HelloWorldServlet extends HttpServlet {
45
46     static String PAGE_HEADER = "<html><head>
<title>helloworld</title></head><body>";
47
48     static String PAGE_FOOTER = "</body></html>";
49
50     @Inject
51    HelloService helloService;
52
53     @Override
54     protected void doGet(HttpServletRequest req,
HttpServletResponse resp) throws ServletException, IOException {
55         resp.setContentType("text/html");
56         PrintWriter writer = resp.getWriter();
57         writer.println(PAGE_HEADER);
58         writer.println("<h1>" +
```

```
helloService.createHelloMessage("World") + "</h1>");
59          writer.println(PAGE_FOOTER);
60          writer.close();
61      }
62
63 }
```

表3.1 HelloWorldServlet の詳細

行	注記
43	必要な作業は **@WebServlet** アノテーションを追加し、サーブレットにアクセスするために使用する URL にマッピングを提供するだけです。
46〜48	各 Web ページには適切な形式の HTML が必要になります。本クイックスタートは静的な文字列を使用して最低限のヘッダーとフッターの出力を書き出します。
50〜51	これらの行は、実際のメッセージを生成する HelloService CDI Bean を挿入します。HelloService の API を変更しない限り、ビューレイヤーを変更せずに HelloService の実装を後で変更することが可能です。
58	この行はサービスを呼び出し、「Hello World」というメッセージを生成して HTTP 要求へ書き出します。

2. **HelloService** コードを確認します。
HelloService.java ファイルは **src/main/java/org/jboss/as/quickstarts/helloworld/** ディレクトリーにあります。このサービスは単にメッセージを返します。XML やアノテーションの登録は必要ありません。

例: HelloService クラスコード

```
public class HelloService {

    String createHelloMessage(String name) {
        return "Hello " + name + "!";
    }
}
```

3.4.2. numberguess クイックスタート

numberguess クイックスタートは単純な非永続アプリケーションを作成し、JBoss EAP にデプロイする方法を示します。情報は JSF ビューを使用して表示され、ビジネスロジックは 2 つの CDI Bean にカプセル化されます。**numberguess** クイックスタートでは 1 から 100 までの数字を当てるチャンスが 10 回与えられます。数字を選択した後、その数字が正解の数字よりも大きいかまたは小さいかが表示されます。

numberguess クイックスタートのコードは**QUICKSTART_HOME/numberguess/** ディレクトリーにあります。**QUICKSTART_HOME** は JBoss EAP のクイックスタートをダウンロードし、展開したディレクトリーです。**numberguess** クイックスタートは複数の Bean、設定ファイル、および Facelets (JSF) ビューによって構成され、 WAR モジュールとしてパッケージ化されています。

コマンドラインを使用してこのクイックスタートをビルドしデプロイする手順の詳細については、**numberguess** クイックスタートディレクトリーのルートにある**README.html** ファイルを参照してください。以下の例では、Red Hat JBoss Developer Studio を使用してクイックスタートを実行します。

前提条件

- JBoss Developer Studio をインストールします。手順については、JBoss Developer Studio『**Installation Guide**』の「Installing JBoss Developer Studio Stand-alone Using the Installer」を参照してください。
- **numberguess** クイックスタートを実行します。手順については、『JBoss Developer Studio でのクイックスタートの実行」を参照し、手順の **helloworld** を **numberguess** に置き換えてください。
- Web ブラウザーを開いて http://localhost:8080/numberguess でアプリケーションにアクセスし、**numberguess** クイックスタートが正常に JBoss EAP にデプロイされたことを確認します。

設定ファイルの確認
このサンプルのすべての設定ファイルは、クイックスタートの **QUICKSTART_HOME/numberguess/src/main/webapp/WEB-INF/** ディレクトリーにあります。

1. **faces-config.xml** ファイルを確認します。
   本クイックスタートは **faces-config.xml** ファイル名の JSF 2.2 バージョンを使用します。Facelets の標準的なバージョンが JSF 2.2 のデフォルトのビューハンドラーであるため、設定は必要ありません。このファイルはルート要素のみで構成され、JSF をアプリケーションで有効にする必要があることを示すマーカーファイルにすぎません。

```
<faces-config version="2.2"
   xmlns="http://xmlns.jcp.org/xml/ns/javaee"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xsi:schemaLocation="
      http://xmlns.jcp.org/xml/ns/javaee
      http://xmlns.jcp.org/xml/ns/javaee/web-facesconfig_2_2.xsd">

</faces-config>
```

2. **beans.xml** ファイルを確認します。
   beans.xml ファイルには、1.1 のバージョン番号と**all** の **bean-discovery-mode** が含まれます。このファイルは、WAR を Bean アーカイブとして識別し、JBoss EAP がこのアプリケーションで Bean を検索し、CDI をアクティベートするよう指示するマーカーファイルです。

```
<beans xmlns="http://xmlns.jcp.org/xml/ns/javaee"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="
      http://xmlns.jcp.org/xml/ns/javaee
      http://xmlns.jcp.org/xml/ns/javaee/beans_1_1.xsd"
    bean-discovery-mode="all">
</beans>
```

注記

このクイックスタートは **web.xml** ファイルを必要としません。

3.4.2.1. JSF コードの確認

JSF はソースファイルに **.xhtml** ファイル拡張子を使用しますが、レンダリングされたビューは**.jsf** 拡張子で提供されます。**home.xhtml** ファイルは **src/main/webapp/** ディレクトリーにあります。

例: JSF ソース コード

```
19<html xmlns="http://www.w3.org/1999/xhtml"
20 xmlns:ui="http://java.sun.com/jsf/facelets"
21 xmlns:h="http://java.sun.com/jsf/html"
22 xmlns:f="http://java.sun.com/jsf/core">
23
24 <head>
25 <meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1"
/>
26 <title>Numberguess</title>
27 </head>
28
29 <body>
30 <div id="content">
31  <h1>Guess a number...</h1>
32  <h:form id="numberGuess">
33
34  <!-- Feedback for the user on their guess -->
35 <div style="color: red">
36  <h:messages id="messages" globalOnly="false" />
37  <h:outputText id="Higher" value="Higher!"
38     rendered="#{game.number gt game.guess and game.guess ne 0}" />
39  <h:outputText id="Lower" value="Lower!"
40     rendered="#{game.number lt game.guess and game.guess ne 0}" />
41 </div>
42
43 <!-- Instructions for the user -->
44 <div>
45 I'm thinking of a number between <span
46 id="numberGuess:smallest">#{game.smallest}</span> and <span
47 id="numberGuess:biggest">#{game.biggest}</span>. You have
48 #{game.remainingGuesses} guesses remaining.
49 </div>
50
51 <!-- Input box for the users guess, plus a button to submit, and reset
-->
52 <!-- These are bound using EL to our CDI beans -->
53 <div>
54 Your guess:
55 <h:inputText id="inputGuess" value="#{game.guess}"
56  required="true" size="3"
57  disabled="#{game.number eq game.guess}"
58  validator="#{game.validateNumberRange}" />
59  <h:commandButton id="guessButton" value="Guess"
60   action="#{game.check}"
61   disabled="#{game.number eq game.guess}" />
62 </div>
63 <div>
64 <h:commandButton id="restartButton" value="Reset"
65 action="#{game.reset}" immediate="true" />
```

```
66 </div>
67 </h:form>
68
69 </div>
70
71 <br style="clear: both" />
72
73 </body>
74</html>
```

以下の行番号は、JBoss Developer Studio でファイルを表示するときに示されるものに対応します。

表3.2 JSF の詳細

行	注記
36〜40	これらはユーザーに送信できるメッセージ、「Higher」(より大きい) と「Lower」(より小さい) です。
45〜48	ユーザーが数を選択するごとに数字の範囲が狭まります。有効な数の範囲が分かるようにこの文章は変更されます。
55〜58	この入力フィールドは値式を使用して Bean プロパティーにバインドされます。
58	ユーザーが誤って範囲外の数字を入力しないようにバリデーターのバインディングが使用されます。バリデーターがないと、ユーザーが範囲外の数字を使用する可能性があります。
59〜61	ユーザーの選択した数字をサーバーに送る方法がなければなりません。ここでは、Bean 上のアクションメソッドをバインドします。

3.4.2.2. クラスファイルの確認

numberguess クイックスタートのソースファイルはすべて **QUICKSTART_HOME/numberguess/src/main/java/org/jboss/as/quickstarts/numberguess/** ディレクトリーにあります。パッケージの宣言とインポートはこれらのリストからは除外されています。完全リストはクイックスタートのソースコードで確認できます。

1. **Random.java** 修飾子コードの検証
   修飾子は、型を基にしたインジェクションの対象となる 2 つの bean 間のあいまいさを取り除くために使用されます。修飾子に関する情報は、 JBoss EAP『**開発ガイド**』の「修飾子を使用したあいまいなインジェクションの解決」を参照してください。**@Random** 修飾子は乱数のインジェクトに使用されます。

   ```
   @Target({ TYPE, METHOD, PARAMETER, FIELD })
   @Retention(RUNTIME)
   @Documented
   @Qualifier
   public @interface Random {

   }
   ```

2. **MaxNumber.java** 修飾子コードの検証

@MaxNumber qualifier は最大許可数の挿入に使用されます。

```
@Target({ TYPE, METHOD, PARAMETER, FIELD })
@Retention(RUNTIME)
@Documented
@Qualifier
public @interface MaxNumber {
}
```

3. **Generator.java** コードの検証
Generator クラスは、producer メソッドを介して乱数を作成し、producer メソッドを介して最大可能数を公開します。このクラスはアプリケーションスコープであるため、毎回異なる乱数になることはありません。

```
@SuppressWarnings("serial")
@ApplicationScoped
public class Generator implements Serializable {

    private java.util.Random random = new
java.util.Random(System.currentTimeMillis());

    private int maxNumber = 100;

    java.util.Random getRandom() {
        return random;
    }

    @Produces
    @Random
    int next() {
        // a number between 1 and 100
        return getRandom().nextInt(maxNumber - 1) + 1;
    }

    @Produces
    @MaxNumber
    int getMaxNumber() {
        return maxNumber;
    }
}
```

4. **Game.java** コードの検証
セッションスコープのクラス **Game** は、アプリケーションのプライマリーエントリーポイントであり、ゲームの設定や再設定、ユーザーが選択する数字のキャプチャーや検証、**FacesMessage** によるユーザーへのフィードバック提供を行います。コンストラクト後の lifecycle メソッドを使用し、**@Random Instance<Integer>** bean から乱数を取得することによりゲームを初期化します。

このクラスの **@Named** アノテーションを見てください。このアノテーションは式言語 (EL) を使用して Bean が JSF ビューにアクセスできるようにしたい場合のみ必要です。この場合 **#{game}** が EL になります。

```
@SuppressWarnings("serial")
@Named
@SessionScoped
```

```
public class Game implements Serializable {

    /**
     * The number that the user needs to guess
     */
    private int number;

    /**
     * The users latest guess
     */
    private int guess;

    /**
     * The smallest number guessed so far (so we can track the valid
guess range).
     */
    private int smallest;

    /**
     * The largest number guessed so far
     */
    private int biggest;

    /**
     * The number of guesses remaining
     */
    private int remainingGuesses;

    /**
     * The maximum number we should ask them to guess
     */
    @Inject
    @MaxNumber
    private int maxNumber;

    /**
     * The random number to guess
     */
    @Inject
    @Random
    Instance<Integer> randomNumber;

    public Game() {
    }

    public int getNumber() {
        return number;
    }

    public int getGuess() {
        return guess;
    }

    public void setGuess(int guess) {
        this.guess = guess;
    }
```

```
    public int getSmallest() {
        return smallest;
    }

    public int getBiggest() {
        return biggest;
    }

    public int getRemainingGuesses() {
        return remainingGuesses;
    }

    /**
     * Check whether the current guess is correct, and update the
biggest/smallest guesses as needed. Give feedback to the user
     * if they are correct.
     */
    public void check() {
        if (guess > number) {
            biggest = guess - 1;
        } else if (guess < number) {
            smallest = guess + 1;
        } else if (guess == number) {
            FacesContext.getCurrentInstance().addMessage(null, new
FacesMessage("Correct!"));
        }
        remainingGuesses--;
    }

    /**
     * Reset the game, by putting all values back to their defaults,
and getting a new random number. We also call this method
     * when the user starts playing for the first time using
{@linkplain PostConstruct @PostConstruct} to set the initial
     * values.
     */
    @PostConstruct
    public void reset() {
        this.smallest = 0;
        this.guess = 0;
        this.remainingGuesses = 10;
        this.biggest = maxNumber;
        this.number = randomNumber.get();
    }

    /**
     * A JSF validation method which checks whether the guess is
valid. It might not be valid because there are no guesses left,
     * or because the guess is not in range.
     *
     */
    public void validateNumberRange(FacesContext context,
UIComponent toValidate, Object value) {
        if (remainingGuesses <= 0) {
            FacesMessage message = new FacesMessage("No guesses
```

```
left!");
                context.addMessage(toValidate.getClientId(context),
message);
                ((UIInput) toValidate).setValid(false);
                return;
            }
            int input = (Integer) value;

            if (input < smallest || input > biggest) {
                ((UIInput) toValidate).setValid(false);

                FacesMessage message = new FacesMessage("Invalid
guess");
                context.addMessage(toValidate.getClientId(context),
message);
            }
        }
}
```

付録A リファレンス資料

A.1. サーバーランタイム引数

アプリケーションサーバーの起動スクリプトは実行時に引数とスイッチを受け入れます。そのため、**standalone.xml**、**domain.xml**、および **host.xml** 設定ファイルに定義されていない他の設定でサーバーを起動できます。

他の設定には、ソケットバインディングの代替セットを持つサーバーの起動や 2 次設定が含まれていることがあります。

help スイッチ **-h** または **--help** を起動時に渡すと、利用可能なパラメーターのリストを使用できます。

表A.1 ランタイムスイッチおよび引数

引数またはスイッチ	操作モード	説明
--admin-only	スタンドアロン	サーバーの実行タイプを **ADMIN_ONLY** に設定します。これにより管理インターフェースが開かれ、管理リクエストが許可されますが、他のランタイムサービスは起動されず、エンドユーザーのリクエストは許可されません。この代わりに **--start-mode=admin-only** を使用することが推奨されます。
--admin-only	ドメイン	ホストコントローラーの実行タイプを **ADMIN_ONLY** に設定します。これにより管理インターフェースが開かれ、管理リクエストが許可されますが、サーバーは起動しません。ホストコントローラーがドメインのマスターである場合はスレーブホストコントローラーからの受信接続が許可されます。
-b=<value>、-b <value>	スタンドアロン、ドメイン	パブリックインターフェースのバインドアドレスを設定するために使用される **jboss.bind.address** システムプロパティーを設定します。値の指定がない場合は、デフォルトで **127.0.0.1** が指定されます。他のインターフェースにバインドアドレスを設定するには **-b<interface>=<value>** エントリーを確認します。
-b<interface>=<value>	スタンドアロン、ドメイン	システムプロパティー **jboss.bind.address.<interface>** を指定の値に設定します (例: **-bmanagement=IP_ADDRESS**)。
--backup	ドメイン	このホストがドメインコントローラーではない場合でも永続ドメイン設定のコピーを保持します。
-c=<config>、-c <config>	スタンドアロン	使用するサーバー設定ファイルの名前。デフォルトは **standalone.xml** です。

引数またはスイッチ	操作モード	説明
-c=<config>、-c <config>	ドメイン	使用するサーバー設定ファイルの名前。デフォルトは **domain.xml** です。
--cached-dc	ドメイン	ホストがドメインコントローラーではなく、起動時にドメインコントローラーに接続できない場合、ローカルでキャッシュされたドメイン設定のコピーを使用してブートします。
--debug [<port>]	スタンドアロン	オプションの引数を用いてデバッグモードを有効にし、ポートを指定します。起動スクリプトがサポートする場合のみ動作します。
-D<name>[=<value>]	スタンドアロン、ドメイン	システムプロパティーを設定します。
--domain-config=<config>	ドメイン	使用するサーバー設定ファイルの名前。デフォルトは **domain.xml** です。
-h、--help	スタンドアロン、ドメイン	ヘルプメッセージを表示し、終了します。
--host-config=<config>	ドメイン	使用するホスト設定ファイルの名前。デフォルトは **host.xml** です。
--interprocess-hc-address=<address>	ドメイン	ホストコントローラーがプロセスコントローラーからの通信をリッスンしなければならないアドレス。
--interprocess-hc-port=<port>	ドメイン	ホストコントローラーがプロセスコントローラーからの通信をリッスンしなければならないポート。
--master-address=<address>	ドメイン	システムプロパティー **jboss.domain.master.address** を指定の値に設定します。デフォルトのスレーブホストコントローラー設定では、マスターホストコントローラーのアドレスを設定するために使用されます。
--master-port=<port>	ドメイン	システムプロパティー **jboss.domain.master.port** を指定の値に設定します。デフォルトのスレーブホストコントローラー設定では、マスターホストコントローラーによるネイティブ管理の通信で使用されるポートを設定するために使用されます。
--read-only-server-config=<config>	スタンドアロン	使用するサーバー設定ファイルの名前。元のファイルは上書きされないため、**--server-config** および **-c** とは異なります。

引数またはスイッチ	操作モード	説明
--read-only-domain-config=<config>	ドメイン	使用するドメイン設定ファイルの名前。最初のファイルは上書きされないため、**--domain-config** および **-c** とは異なります。
--read-only-host-config=<config>	ドメイン	使用するホスト設定ファイルの名前。最初のファイルは上書きされないため、**--host-config** とは異なります。
-P=<url>、-P <url>、--properties=<url>	スタンドアロン、ドメイン	該当する URL からシステムプロパティーをロードします。
--pc-address=<address>	ドメイン	プロセスコントローラーが制御するプロセスからの通信をリッスンするアドレス。
--pc-port=<port>	ドメイン	プロセスコントローラーが制御するプロセスからの通信をリッスンするポート。
-S<name>[=<value>]	スタンドアロン	セキュリティープロパティーを設定します。
-secmgr	スタンドアロン、ドメイン	セキュリティーマネージャーがインストールされた状態でサーバーを実行します。
--server-config=<config>	スタンドアロン	使用するサーバー設定ファイルの名前。デフォルトは **standalone.xml** です。
--start-mode=<mode>	スタンドアロン	サーバーの起動モードを設定します。このオプションは、 **--admin-only** と併用できません。有効な値は以下のとおりです。 • **normal**: サーバーは通常どおりに起動します。 • **admin-only**: サーバーは管理インターフェースのみを開き、管理リクエストを許可しますが、他のランタイムサービスは起動せず、エンドユーザーのリクエストを許可しません。 • **suspend**: サーバーは中断モードで起動され、再開するまでリクエストに対処しません。
-u=<value>、-u <value>	スタンドアロン、ドメイン	設定ファイルの socket-binding 要素のマルチキャストアドレスを設定するために使用される **jboss.default.multicast.address** システムプロパティーを設定します。値の指定がない場合はデフォルトで **230.0.0.4** が指定されます。

引数またはスイッチ	操作モード	説明
-v、-V、--version	スタンドアロン、ドメイン	アプリケーションサーバーのバージョンを表示し、終了します。

警告

JBoss EAP に同梱される設定ファイルは、スイッチ (**-b**、**-u** など) を処理するよう設定されます。スイッチによって制御されるシステムプロパティーを使用しないよう設定ファイルを変更した場合は、実行するコマンドにスイッチを追加しても効果はありません。

A.2. ADD-USER ユーティリティー引数

以下の表は、**add-user.sh** または **add-user.bat** スクリプトで使用できる引数を示しています。これらのスクリプトは既定の認証のプロパティーファイルに新しいユーザーを追加するためのユーティリティーです。

表A.2 add-user コマンド引数

コマンドライン引数	説明
-a	アプリケーションレルムでユーザーを作成します。省略した場合、デフォルトでは管理レルムでユーザーが作成されます。
-dc <value>	プロパティーファイルが含まれるドメイン設定ディレクトリー。省略した場合、デフォルトのディレクトリーは **EAP_HOME/domain/configuration/** になります。
-sc <value>	プロパティーファイルが含まれる代替のスタンドアロンサーバー設定ディレクトリー。省略した場合、デフォルトのディレクトリーは **EAP_HOME/standalone/configuration/** になります。
-up、--user-properties <value>	代替のユーザープロパティーファイルの名前。絶対パスを使用でき、代替の設定ディレクトリーを指定する **-sc** または **-dc** 引数と共に使用されるファイル名を使用することもできます。
-g、--group <value>	このユーザーに割り当てるグループのコンマ区切りリスト。
-gp、--group-properties <value>	代替のグループプロパティーファイルの名前。絶対パスを使用でき、代替の設定ディレクトリーを指定する **-sc** または **-dc** 引数と共に使用されるファイル名を使用することもできます。
-p、--password <value>	ユーザーのパスワード。

コマンドライン引数	説明
-u、--user <value>	ユーザーの名前。ユーザー名には、以下の文字のみを使用できます。文字の数と順番は自由です。 • 英数字 (a-z、A-Z、0-9) • ダッシュ (-)、ピリオド (.)、コンマ (,)、アットマーク(@) • バックスラッシュ (\) • 等号 (=)
-r、--realm <value>	管理インターフェースをセキュアにするために使用されるレルムの名前。省略した場合、デフォルト値は **ManagementRealm** です。
-s、--silent	コンソールへ出力せずに **add-user** スクリプトを実行します。
-e、--enable	ユーザーを有効にします。
-d、--disable	ユーザーを無効にします。
-cw、--confirm-warning	対話モードで自動的に警告を確認します。
-h、--help	**add-user** スクリプトの使用情報を表示します。
-ds、--display-secret	非対話モードで秘密の値を出力します。

A.3. インターフェース属性

注記

この表は、管理モデルで使用される属性名を示しています (管理 CLI を使用している場合など)。XML で使用される名前は管理モデルの名前と異なる場合があるため、XML で使用される要素を **EAP_HOME/docs/schema/wildfly-config_5_0.xsd** のスキーマ定義ファイルで確認してください。

表A.3 インターフェース属性と値

インターフェース要素	説明
any	インターフェースの選択基準の一部は、最低でも基準のネストされたセットの 1 つ (すべてとは限らない) を満たす必要があることを示す要素。

インターフェース要素	説明
any-address	このインターフェースを使用するソケットをワイルドカードアドレスにバインドする必要があることを示す空の要素。**java.net.preferIPv4Stack** システムプロパティーが true に設定されていない限り、IPv6 ワイルドカードアドレス (**::**) が使用されます。true に設定された場合は、IPv4 ワイルドカードアドレス (**0.0.0.0**) が使用されます。ソケットがデュアルスタックマシンの IPv6 anylocal アドレスにバインドされた場合は、IPv6 および IPv4 トラフィックを受け入れることができます。IPv4 (IPv4 マッピング) anylocal アドレスにバインドされた場合は、IPv4 トラフィックのみを受け入れることができます。
inet-address	IPv6 または IPv4 のドット区切り表記の IP アドレス、または IP アドレスに解決できるホスト名。
link-local-address	インターフェースの選択基準の一部として、関連付けられたアドレスがリンクローカルであるかどうかを示す空の要素。
loopback	インターフェースの選択基準の一部として、ループバックインターフェースであるかどうかを示す空の要素。
loopback-address	マシンのループバックインターフェースで実際には設定できないループバックアドレス。IP アドレスが関連付けられた NIC が見つからない場合であっても該当する値が使用されるため、inet-address タイプとは異なります。
multicast	インターフェースの選択基準の一部として、マルチキャストをサポートするかどうかを示す空の要素。
name	インターフェースの名前。
nic	ネットワークインターフェースの名前 (eth0、eth1、lo など)。
nic-match	使用できるインターフェースを見つけるために、マシンで利用可能なネットワークインターフェースの名前を検索する正規表現。
not	インターフェースの選択基準の一部は、基準のネストされたセットを満たしてはならないことを示す要素。
point-to-point	インターフェースの選択基準の一部として、ポイントツーポイントインターフェースであるかどうかを示す空の要素。
public-address	インターフェースの選択基準の一部として、公開されたルーティング可能なアドレスを持つかどうかを示す空の要素。
site-local-address	インターフェースの選択基準の一部として、関連付けられたアドレスがサイトローカルであるかどうかを示す空の要素。

インターフェース要素	説明
subnet-match	**スラッシュ表記法**で記述されたネットワーク IP アドレスとアドレスのネットワーク接頭辞のビット数 (例: 192.168.0.0/16)。
up	インターフェースの選択基準の一部として、現在稼動しているかどうかを示す空の要素。
virtual	インターフェースの選択基準の一部として、仮想インターフェースであるかどうかを示す空の要素。

A.4. ソケットバインディング属性

注記

これらの表は、管理モデルで使用される属性名を示しています (管理 CLI を使用している場合など)。XML で使用される名前は管理モデルの名前と異なる場合があるため、XML で使用される要素を **EAP_HOME/docs/schema/wildfly-config_5_0.xsd** のスキーマ定義ファイルで確認してください。

以下の表は、3 種類のソケットバインディングそれぞれに設定できる属性を表しています。

- socket-binding
- remote-destination-outbound-socket-binding
- local-destination-outbound-socket-binding

表A.4 インバウンドソケットバインディング (socket-binding) の属性

属性	説明
client-mappings	このソケットバインディングのクライアントマッピングを指定します。このソケットへ接続するクライアントは、希望のアウトバウンドインターフェースと一致するマッピングに指定された宛先アドレスを使用する必要があります。これにより、ネットワークアドレスの変換を使用する高度なネットワークトポロジーまたは複数のネットワークインターフェースにバインディングを持つ高度なネットワークトポロジーが機能します。各マッピングは宣言された順序で評価される必要があり、最初に一致したマッピングを使用して宛先が決定されます。
fixed-port	ソケットグループの他のソケットに数値のオフセットが適用された場合でもポートの値を固定したままにするかどうか。
interface	ソケットがバインドされる必要があるインターネットの名前、またはマルチキャストソケットの場合はリッスンするインターフェース。宣言されたインターフェースの 1 つである必要があります。定義されないと、エンクロージングソケットバインディンググループからの **default-interface** の値が使用されます。

属性	説明
multicast-address	ソケットがマルチキャストトラフィックを受信するマルチキャストアドレス。指定しないと、ソケットがマルチキャストを受信するよう設定されません。
multicast-port	ソケットがマルチキャストトラフィックを受信するポート。**multicast-address** が設定されている場合はこれも設定する必要があります。
name	ソケットの名前。ソケット設定情報にアクセスする必要があるサービスは、この名前を使用してソケット設定情報を探します。必須の属性です。
port	ソケットがバインドされる必要があるポートの番号。サーバーによってポートオフセットが適用され、ポートの値がすべて増加または減少される場合、この値は上書きされることに注意してください。

表A.5 リモートアウトバウンドソケットバインディング (remote-destination-outbound-socket-binding) の属性

属性	説明
fixed-source-port	数値のオフセットポートがソケットグループの別のアウトバウンドソケットに適用される場合、ポートの値を固定すべきかどうか。
host	このアウトバウンドソケットが接続するリモート宛先のホスト名または IP アドレス。
port	アウトバウンドソケットが接続すべきリモート宛先のポート番号。
source-interface	アウトバウンドソケットのソースアドレスに使用されるインターフェースの名前。
source-port	アウトバウンドソケットのソースポートとして使用されるポート番号。

表A.6 ローカルアウトバウンドソケットバインディング (local-destination-outbound-socket-binding) の属性

属性	説明
fixed-source-port	数値のオフセットポートがソケットグループの別のアウトバウンドソケットに適用される場合、ポートの値を固定すべきかどうか。
socket-binding-ref	このアウトバウンドソケットが接続するポートを決定するために使用されるローカルソケットバインディングの名前。
source-interface	アウトバウンドソケットのソースアドレスに使用されるインターフェースの名前。
source-port	アウトバウンドソケットのソースポートとして使用されるポート番号。

A.5. デフォルトのソケットバインディング

以下の表は、各ソケットバインディンググループのデフォルトのソケットバインディングを示しています。

- standard-sockets
- ha-sockets
- full-sockets
- full-ha-sockets
- load-balancer-sockets

表A.7 standard-sockets

ソケットバインディング	ポート	説明
ajp	8009	Apache JServ プロトコル。HTTP クラスタリングおよび負荷分散に使用されます。
http	8080	デプロイされた Web アプリケーションのデフォルトポート。
https	8443	デプロイされた Web アプリケーションとクライアントとの間の SSL で暗号化された接続。
management-http	9990	管理レイヤーを用いた HTTP 通信に使用されます。
management-https	9993	管理レイヤーを用いた HTTPS 通信に使用されます。
txn-recovery-environment	4712	JTA トランザクションリカバリーマネージャー。
txn-status-manager	4713	JTA / JTS トランザクションマネージャー。

表A.8 ha-sockets

ソケットバインディング	ポート	マルチキャストポート	説明
ajp	8009		Apache JServ プロトコル。HTTP クラスタリングおよび負荷分散に使用されます。
http	8080		デプロイされた Web アプリケーションのデフォルトポート。

ソケットバインディング	ポート	マルチキャストポート	説明
https	8443		デプロイされた Web アプリケーションとクライアントとの間の SSL で暗号化された接続。
jgroups-mping		45700	マルチキャスト。HA クラスターでの初期メンバーシップの検出に使用されます。
jgroups-tcp	7600		TCP を使用した、HA クラスター内でのユニキャストピア検出。
jgroups-udp	55200	45688	UDP を使用した、HA クラスター内でのマルチキャストピア検出。
management-http	9990		管理レイヤーを用いた HTTP 通信に使用されます。
management-https	9993		管理レイヤーを用いた HTTPS 通信に使用されます。
modcluster		23364	JBoss EAP と HTTP ロードバランサー間の通信に対するマルチキャストポート。
txn-recovery-environment	4712		JTA トランザクションリカバリーマネージャー。
txn-status-manager	4713		JTA / JTS トランザクションマネージャー。

表A.9 full-sockets

ソケットバインディング	ポート	説明
ajp	8009	Apache JServ プロトコル。HTTP クラスタリングおよび負荷分散に使用されます。
http	8080	デプロイされた Web アプリケーションのデフォルトポート。
https	8443	デプロイされた Web アプリケーションとクライアントとの間の SSL で暗号化された接続。
iiop	3528	JTS トランザクションおよび他の ORB 依存サービス用の CORBA サービス。
iiop-ssl	3529	SSL 暗号化 CORBA サービス。

ソケットバインディング	ポート	説明
management-http	9990	管理レイヤーを用いた HTTP 通信に使用されます。
management-https	9993	管理レイヤーを用いた HTTPS 通信に使用されます。
txn-recovery-environment	4712	JTA トランザクションリカバリーマネージャー。
txn-status-manager	4713	JTA / JTS トランザクションマネージャー。

表A.10 full-ha-sockets

名前	ポート	マルチキャストポート	説明
ajp	8009		Apache JServ プロトコル。HTTP クラスタリングおよび負荷分散に使用されます。
http	8080		デプロイされた Web アプリケーションのデフォルトポート。
https	8443		デプロイされた Web アプリケーションとクライアントとの間の SSL で暗号化された接続。
iiop	3528		JTS トランザクションおよび他の ORB 依存サービス用の CORBA サービス。
iiop-ssl	3529		SSL 暗号化 CORBA サービス。
jgroups-mping		45700	マルチキャスト。HA クラスターでの初期メンバーシップの検出に使用されます。
jgroups-tcp	7600		TCP を使用した、HA クラスター内でのユニキャストピア検出。
jgroups-udp	55200	45688	UDP を使用した、HA クラスター内でのマルチキャストピア検出。
management-http	9990		管理レイヤーを用いた HTTP 通信に使用されます。
management-https	9993		管理レイヤーを用いた HTTPS 通信に使用されます。

名前	ポート	マルチキャストポート	説明
modcluster		23364	JBoss EAP と HTTP ロードバランサー間の通信に対するマルチキャストポート。
txn-recovery-environment	4712		JTA トランザクションリカバリーマネージャー。
txn-status-manager	4713		JTA / JTS トランザクションマネージャー。

表A.11 load-balancer-sockets

名前	ポート	マルチキャストポート	説明
http	8080		デプロイされた Web アプリケーションのデフォルトポート。
https	8443		デプロイされた Web アプリケーションとクライアントとの間の SSL で暗号化された接続。
management-http	9990		管理レイヤーを用いた HTTP 通信に使用されます。
management-https	9993		管理レイヤーを用いた HTTPS 通信に使用されます。
mcmp-management	8090		ライフサイクルイベントを送信する Mod-Cluster Management Protocol (MCMP) 接続のポート。
modcluster		23364	JBoss EAP と HTTP ロードバランサー間の通信に対するマルチキャストポート。

Revised on 2018-08-29 12:43:52 EDT