# Red Hat JBoss Enterprise Application Platform 7.4

## スタートガイド

Red Hat JBoss Enterprise Application Platform のダウンロード、インストール、起動、停止、および保守を行う手順

Last Updated: 2024-06-14

# Red Hat JBoss Enterprise Application Platform 7.4 スタートガイド

Red Hat JBoss Enterprise Application Platform のダウンロード、インストール、起動、停止、および保守を行う手順

## 法律上の通知

Copyright © 2024 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, the Red Hat 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 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。本書では、JBoss EAP の基本インストール、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 Jakarta 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、一連の JBoss EAP ドキュメント をすべて参照してください。

# 目次

# JBOSS EAP ドキュメントへのフィードバック (英語のみ)

エラーを報告したり、ドキュメントを改善したりするには、Red Hat Jira アカウントにログインし、課題を送信してください。Red Hat Jira アカウントをお持ちでない場合は、アカウントを作成するように求められます。

手順

1. このリンクをクリック してチケットを作成します。
2. ドキュメント **URL**、セクション番号、課題の説明 を記入してください。
3. **Summary** に課題の簡単な説明を入力します。
4. **Description** に課題や機能拡張の詳細な説明を入力します。問題があるドキュメントのセクションへの URL を含めてください。
5. **Submit** をクリックすると、課題が作成され、適切なドキュメントチームに転送されます。

# 多様性を受け入れるオープンソースの強化

Red Hat では、コード、ドキュメント、Web プロパティーにおける配慮に欠ける用語の置き換えに取り組んでいます。まずは、マスター (master)、スレーブ (slave)、ブラックリスト (blacklist)、ホワイトリスト (whitelist) の 4 つの用語の置き換えから始めます。この取り組みは膨大な作業を要するため、今後の複数のリリースで段階的に用語の置き換えを実施して参ります。詳細は、Red Hat CTO である Chris Wright のメッセージ をご覧ください。

# 第1章 JBOSS EAP の管理

## 1.1. JBOSS EAP のダウンロードおよびインストール

圧縮ファイルオプションは、プラットフォームに依存せずに、素早く JBoss EAP をダウンロードしてインストールするための方法です。

### 1.1.1. JBoss EAP のダウンロード

JBoss EAP をインストールする前に、JBoss EAP 圧縮ファイルをダウンロードする必要があります。

前提条件

- システムが JBoss EAP でサポートされる設定 を満たしていることを確認します。
- 最新の更新とエラータパッチをインストールします。
- インストールディレクトリーの読み取りおよび書き込みアクセスを設定します。
- 目的の Java Development Kit (JDK) をインストールします。
- オプション: Windows Server の場合には、 **JAVA_HOME** および **PATH** 環境変数を設定します。

手順

1. Red Hat カスタマーポータルにログインする。
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 **7.4** を選択します。
5. リストで **Red Hat JBoss Enterprise Application Platform 7.4.0** を見つけ、 **Download** リンクをクリックします。
   圧縮ファイルがシステムにダウンロードされます。

関連情報

- Red Hat 製品のダウンロードには、 Red Hat カスタマーポータル からアクセスしてください。

### 1.1.2. JBoss EAP のインストール

パッケージの内容を目的のファイルの場所に展開することで、JBoss EAP 圧縮ファイルをインストールできます。

前提条件

- JBoss EAP のダウンロード
- システムが JBoss EAP でサポートされる設定 を満たしていることを確認します。

- 最新の更新とエラータパッチをインストールします。
- インストールディレクトリーの読み取りおよび書き込みアクセスを設定します。
- 目的の Java Development Kit (JDK) をインストールします。
- Windows Server の場合には、 **JAVA_HOME** および **PATH** 環境変数を設定します。

手順

1. 圧縮ファイルを JBoss EAP をインストールするサーバーと場所に移動します。
2. 圧縮ファイルを展開します。
   a. Linux の場合は、以下のコマンドを使用します。

```
$ unzip jboss-eap-7.4.0.zip
```

   b. Windows Server では、圧縮ファイルを右クリックして、すべて展開 を選択します。
   圧縮ファイル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は **EAP_HOME** と呼ばれます。

関連情報

- グラフィカルインストーラーまたは RPM パッケージのインストール方法を使用した JBoss EAP のインストールの詳細は、インストールガイド を参照してください。

## 1.2. JBOSS EAP の開始および停止

JBoss EAP を開始する方法は、JBoss EAP をスタンドアロンサーバーとして実行しているか、管理対象ドメイン内のサーバーで実行しているかによって異なります。

JBoss EAP を停止する方法は、JBoss EAP のインタラクティブインスタンスとバックグラウンドインスタンスのどちらを実行しているかによって異なります。

### 1.2.1. JBoss EAP のスタンドアロンサーバーとしての起動

JBoss EAP をスタンドアロンサーバーとして実行して、JBoss EAP の単一インスタンスを管理できます。

JBoss EAP は、以下のプラットフォームでサポートされています。

- Red Hat Enterprise Linux
- Windows Server
- Oracle Solaris

サーバーは一時停止状態で起動し、必要なすべてのサービスが開始されるまで要求を受け入れません。必要なサービスが開始されると、サーバーは通常の実行状態に移行し、要求の受け入れを開始できます。

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を指定します。このファイルで設定をカスタマイズできます。

**注記**

ターミナルで起動スクリプトの引数のリストを表示するには、 **-help** 引数を使用します。

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。

前提条件

- JBoss EAP のインストール

手順

1. 端末を開きます。
2. 次のスクリプトを使用して、JBoss EAP をスタンドアロンサーバーとして起動します。

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

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

関連情報

- 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、サーバーランタイム引数 のセクションを参照してください。

### 1.2.2. 管理対象ドメインでのサーバー用の JBoss EAP の起動

管理対象ドメインオペレーティングモードで JBoss EAP を実行し、単一のドメインコントローラーを使用して複数の JBoss EAP インスタンスを管理できます。

JBoss EAP は、以下のプラットフォームでサポートされています。

- Red Hat Enterprise Linux
- Windows Server
- Oracle Solaris

サーバーは一時停止状態で起動し、必要なすべてのサービスが開始されるまで要求を受け入れません。必要なサービスが開始されると、サーバーは通常の実行状態に移行して、要求の受け入れを開始できます。

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。

前提条件

- JBoss EAP のインストール

手順

1. 端末を開きます。
2. 最初にドメインコントローラーを起動してから、次のスクリプトを使用して、関連付けられている各ホストコントローラーを起動します。

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

   a. 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を指定します。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開始できます。

管理対象ドメインの設定時には、起動スクリプトに追加の引数を渡す必要があります。

関連情報

- 管理対象ドメイン設定ファイルの詳細は、管理対象ドメイン設定ファイル セクション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。

### 1.2.3. JBoss EAP の対話的なインスタンスの停止

スタンドアロンサーバーまたはドメインコントローラーのインタラクティブインスタンスは、起動した端末から停止できます。

前提条件

- JBoss EAP のインスタンスを開始しました。

手順

- JBoss EAP を開始したターミナルで **Ctrl + C** を押します。

### 1.2.4. JBoss EAP のバックグラウンドインスタンスの停止

管理 CLI に接続して、スタンドアロンサーバーの実行中のインスタンスまたは管理対象ドメイン内のサーバーをシャットダウンできます。

前提条件

- ターミナルで実行されている JBoss EAP のインスタンスがある。

手順

1. 次のスクリプトを使用して、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
```

管理対象ドメインのサーバーで JBoss EAP のインスタンスを実行する場合は、 **shutdown** コマンドで **--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。

## 1.3. 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イ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。方法としては、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。

JBoss EAP は、YAML ファイルを使用したスタンドアロンサーバーの XML 設定の変更をサポートします。詳細は、YAML ファイルを使用したスタンドアロンサーバー設定の更新 を参照してください。

**注記**

YAML 設定は、管理対象ドメインのサーバーではサポートされ ません。

### 1.3.1. 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**Security Architecture**の Core Management Authentication を参照してください。

#### 1.3.1.1. 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。

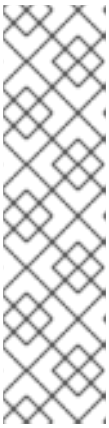
```
$ EAP_HOME/bin/add-user.sh
```

**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 ではデフォルトで、脆弱なパスワードは許可されますが、警告が表示されます。

デフォルト動作の変更に関する詳細は、JBoss EAP**Configuration Guide**の Setting Add-User Utility Password Restrictions 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。

#### 1.3.1.2. 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** のすべての引数の完全リストとそれら引数の目的については、**--help** 引数を指定するか、Add-User 引数 の項を参照してください。

## 1.3.2. 管理インターフェイス

### 1.3.2.1. 管理 CLI

管理コマンドラインインターフェイス (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。管理 CLI は、管理対象ドメインのドメインコントローラーに接続し、ドメインで管理操作を実行することもできます。

**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 EAPManagement 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
```

#### 1.3.2.2. 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、同じドメインのサーバーインスタンスと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集中管理できます。

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/index.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。

**Patching (パッチ)**
: JBoss EAP インスタンスにパッチを適用します。

**Access Control (アクセス制御)**
: ロールベースのアクセス制御を使用するときのユーザーとグループにロールを割り当てます。

### 1.3.3. 設定ファイル

#### 1.3.3.1. 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。事前定義されたプロファイルは 5 つあり (**default**、**ha**、**full**、**full-ha**、および **load-balancer**)、それぞれに個別のファイルが存在します。

表1.1 スタンドアロン設定ファイル

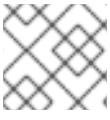
| 設定ファイル | 目的 |
|---|---|
| **standalone.xml** | このスタンドアロン設定ファイルは、スタンドアロンサーバーを起動したときに使用されるデフォルト設定です。このファイルには、サブシステム、ネットワーキング、デプロイメント、ソケットバインディング、およびその他の設定詳細など、サーバーに関するすべての情報が含まれます。メッセージングや高可用性に必要なサブシステムは提供しません。 |
| **standalone-ha.xml** | このスタンドアロン設定ファイルには、デフォルトのサブシステムすべてが含まれ、高可用性の **modcluster** および **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、ビルトインの 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
```

#### 1.3.3.1.1. YAML ファイルを使用してスタンドアロンサーバー設定を更新します。

YAML ファイルを使用してスタンドアロンサーバーを設定すると、カスタマイズプロセスを外部化し、サーバーアップグレードの速度が向上します。この機能を使用する場合、サーバーは読み取り専用モードで起動します。これは、サーバーの再起動後に設定の変更が維持されないことを意味します。



**注記**

YAML 設定は、管理対象ドメインのサーバーではサポートされ ません。

ユーザーは YAML ファイル内のさまざまなリソースを変更できます。以下のリソースは YAML ファイルでサポートされています。

- **core-service**
- **interface**

- **socket-binding-group**
- **subsystem**
- **system-property**

以下のリソースは、YAML ファイルではサポートされてい ません。

- **Extension:** 拡張 をサーバーに追加します。この要素は、欠落しているモジュールが必要となる可能性があるため、サポートされていません。
- **Deployment**: デプロイメントをサーバーに追加します。この要素は設定に加えてより詳細な変更が必要であるため、サポート対象外です。
- **deployment-overlay**: deployment-overlays をサーバーに追加します。この要素は設定に加えてより詳細な変更が必要であるため、サポート対象外です。
- **path**: YAML ファイルが解析される際に定義されます。

YAML ルートノードは **wildfly-configuration** です。モデルツリーに従って、リソースを変更できます。すでにリソースが存在する場合(XML 設定ファイルまたは以前の YAML ファイルによって作成された)、モデルツリーを使用してリソースを更新できます。リソースが存在しない場合は、モデルツリーを使用してリソースを作成できます。

### 新しい PostGresql データソースを定義する YAML 設定ファイルの例

```
wildfly-configuration:
  subsystem:
    datasources:
      jdbc-driver:
        postgresql:
          driver-name: postgresql
          driver-xa-datasource-class-name: org.postgresql.xa.PGXADataSource
          driver-module-name: org.postgresql.jdbc
      data-source:
        PostgreSQLDS:
          enabled: true
          exception-sorter-class-name:
org.jboss.jca.adapters.jdbc.extensions.postgres.PostgreSQLExceptionSorter
          jndi-name: java:jboss/datasources/PostgreSQLDS
          jta: true
          max-pool-size: 20
          min-pool-size: 0
          connection-url: "jdbc:postgresql://localhost:5432}/demo"
          driver-name: postgresql
          user-name: postgres
          password: postgres
          validate-on-match: true
          background-validation: false
          background-validation-millis: 10000
          flush-strategy: FailingConnectionOnly
          statistics-enable: false
          stale-connection-checker-class-name:
org.jboss.jca.adapters.jdbc.extensions.novendor.NullStaleConnectionChecker
```

```
        valid-connection-checker-class-name:
org.jboss.jca.adapters.jdbc.extensions.postgres.PostgreSQLValidConnectionChecker
        transaction-isolation: TRANSACTION_READ_COMMITTED
```

上記の例では、**postgresql** という **jdbc-driver** と、**PostgreSQLDS** と呼ばれる データソース を定義します。

**注記**

YAML 設定ファイルを使用してモジュールを管理することはできません。代わりに、**org.postgresql.jdbc** モジュールを手動で作成またはプロビジョニングするか、管理 CLI を使用する必要があります。

#### 1.3.3.1.2. タグを使用した YAML ファイル操作

タグを使用して、YAML 設定ファイルに対して複数の操作を実行できます。

- **!undefine**: 属性の定義を解除します

  **定義解除 CONSOLE ロガーレベルの YAML 設定ファイルの例**

```
wildfly-configuration:
  subsystem:
    logging:
      console-handler:
        CONSOLE:
          level: !undefine
```

- **!remove** : リソースを削除します。

  **埋め込み Artemis ブローカーを削除し、リモートブローカーの YAML 設定ファイルの例に接続します。**

```
wildfly-configuration:
 socket-binding-group:
  standard-sockets:
   remote-destination-outbound-socket-binding:
    remote-artemis:
     host: localhost
     port: 61616
 subsystem:
  messaging-activemq:
   server:
    default: !remove
   remote-connector:
    artemis:
     socket-binding: remote-artemis
   pooled-connection-factory:
    RemoteConnectionFactory:
     connectors:
      - artemis
     entries:
      - "java:jboss/RemoteConnectionFactory"
      - "java:jboss/exported/jms/RemoteConnectionFactory"
```

```
      enable-amq1-prefix: false
      user: admin
      password: admin
  ejb3:
    default-resource-adapter-name: RemoteConnectionFactory
  ee:
    service:
      default-bindings:
        jms-connection-factory: "java:jboss/RemoteConnectionFactory"
```

- **!list-add** : 要素をリストに追加します（任意のインデックスを含む）

  **RemoteTransactionPermission をパーミッション一覧の YAML 設定ファイルの例に追加します。**

```
wildfly-configuration:
    subsystem:
        elytron:
          permission-set:
           default-permissions:
             permissions: !list-add
              - class-name: org.wildfly.transaction.client.RemoteTransactionPermission
                module: org.wildfly.transaction.client
                target-name: "*"
                index: 0
```

**注記**

**index** 属性が定義されていない場合は、エントリーがリストの最後に追加されます。

#### 1.3.3.1.3. YAML ファイルを使用したスタンドアロンサーバーの起動

YAML 設定ファイルを使用してスタンドアロンサーバーを起動できます。

手順

1. ターミナルを開きます。
2. 以下のコマンドを使用して、YAML ファイルでスタンドアロンサーバーを起動します。

```
./standalone.sh -y=/home/ehsavoie/dev/wildfly/config2.yml:config.yml -c standalone-full.xml
```

**- yaml** 引数または**y** 引数を使用すると、YAML ファイルのリストを渡すことができます。Windows Server の場合はセミコロン(;)を使用し、Mac および Unix ベースのオペレーティングシステムの場合はコロン(:)を使用して各 YAML ファイルパスを区切る必要があります。絶対パス、現在の実行ディレクトリーへの相対パス、またはスタンドアロン設定ディレクトリーからの相対パスを使用できます。

操作は、ファイルが定義された順序で適用され、初期操作は XML 設定によって定義されます。

### 1.3.3.2. 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1.2 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主要設定ファイルです。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 (**default**、**ha**、**full**、**full-ha**、および **load-balancer**) の設定が含まれています。 |
| **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イス、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。 |
| **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
```

#### 1.3.3.3. 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。

#### 1.3.3.4. 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作成
管理 CLI を使用して、現在の設定のスナップショットを作成します。

```
:take-snapshot
{
    "outcome" => "success",
    "result" => "EAP_HOME/standalone/configuration/standalone_xml_history/snapshot/20151022-
133109702standalone.xml"
}
```

スナップショットのリスト
管理 CLI を使用して、作成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。

#### 1.3.3.5. 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** インターフェイスの **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** パラメーターを設定できます。

```
$ 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。Jakarta 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。

- **jboss-ejb3.xml**
- **jboss-app.xml**
- **jboss-web.xml**
- **jboss-permissions.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)
```

Jakarta 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**
- **permissions.xml**
- **persistence.xml**
- **application.xml**
- **web.xml**

以下の管理 CLI コマンドを使用すると、Jakarta 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)
```

## 1.4. ネットワークとポートの設定 JBOSS EAP

JBoss EAP には、設定を簡素化するインターフェイス、ソケットバインディング、および IPv6 アドレスが付属しています。これらの各ネットワークおよびポート設定に関する以下の詳細情報を使用して、JBoss EAP を正常に実行します。

### 1.4.1. インターフェイス

JBoss EAP は設定全体で名前付きインターフェイスを参照します。JBoss EAP を設定して、使用ごとにインターフェイ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イス宣言を参照できます。

複数のマシンでネットワークインターフェイ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イ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イス名があります。**management** インターフェイスは、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** インターフェイスは、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** インターフェイスは、標準設定の IIOP ソケットに使用されます。**private** インターフェイスは、標準設定の JGroups ソケットに使用されます。

#### 1.4.1.1. デフォルトインターフェイス設定

JBoss EAP には、以下のようにデフォルトのインターフェイスが 4 つ含まれています。

```
<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イスを **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** インターフェイスの **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。

**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イスまたはポートを変更する場合は、変更したインターフェイ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イスとポートを指定するようにしてください。

関連情報

- 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。

#### 1.4.1.2. オプションのインターフェイス設定

ネットワークインターフェイスは、物理インターフェイ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イ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。

インターフェイ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イ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。

関連情報

- 使用できるすべてのインターフェイス選択基準は インターフェイス属性 を参照してください。

##### 1.4.1.2.1. NIC 値とのインターフェイス

次の例を使用して、NIC 値が **eth0** の新しいインターフェイスを追加できます。

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

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

##### 1.4.1.2.2. 複数の条件値があるインターフェイス

以下の例を使用して、稼働時に適切なサブネットのインターフェイスまたはアドレスと一致して、マルチキャストをサポートし、ポイとツーポイントではないインターフェイスを新たに追加できます。

```
/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>
```

##### 1.4.1.2.3. インターフェイス属性の更新

この例では、実行時にこの値を設定できるように、 **jboss.bind.address** プロパティーを保持したまま、パブリック のデフォルトインターフェイスの **inet-address** 値を更新できます。

```
/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>
```

#### 1.4.1.2.4. 管理対象ドメインのサーバーへの追加インターフェイス

次のコードを使用して、管理対象ドメインのサーバーにインターフェイスを追加できます。

```
/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>
```

### 1.4.2. 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イ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。

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。

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。

関連情報

- インバウンドソケットバインディングの属性を表示するには、インバウンドソケットバインディング属性 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の属性を表示するには、リモートアウトバウンドソケットバインディング属性 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の他の例は、JBoss EAP のメッセージングの設定の リモート接続用の統合 Artemis リソースアダプターの使用 セクション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の属性を表示するには、ローカルアウトバウンドソケットバインディング属性 の表を参照してください。

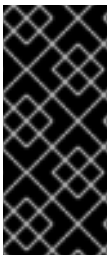
#### 1.4.2.1. 管理ポート

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。

#### 1.4.2.2. 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。



**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イスまたはポートを変更する場合は、変更したインターフェイ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イスとポートを指定するようにしてください。

関連情報

- 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。

##### 1.4.2.2.1. 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="${jboss.socket.binding.port-offset:0}">
        <socket-binding name="ajp" port="${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="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="txn-recovery-environment" port="4712"/>
        <socket-binding name="txn-status-manager" port="4713"/>
        <outbound-socket-binding name="mail-smtp">
            <remote-destination host="${jboss.mail.server.host:localhost}"
port="${jboss.mail.server.port:25}"/>
        </outbound-socket-binding>
</socket-binding-group>
```

#### 1.4.2.2.2. 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**

管理インターフェイ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。

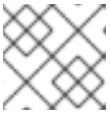
#### 1.4.2.3. ソケットバインディングの設定

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の属性 を参照してください。

**手順**

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。



**注記**

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ません。

```
/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イスを使用するよう、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>
```

#### 1.4.2.4. 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
```

### 1.4.3. 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に JBoss EAP を設定する方法について説明します。

#### 1.4.3.1. IPv6 アドレスの JVM スタックの設定

IPv6 を使用して実行するように JBoss EAP を設定できます。

**手順**

IPv6 アドレスで実行するように起動設定を更新するには、次の手順を実行します。

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
```

#### 1.4.3.2. デフォルトのインターフェイス値の IPv6 アドレスへの更新

設定のデフォルトのインターフェイス値は、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** インターフェイスを 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]}")
```

以下の例では、前のコマンドの実行後に、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>
```

## 1.5. JBOSS EAP サーバー設定の最適化

JBoss EAP サーバーをインストール し、管理ユーザーを作成 したら、サーバー設定を最適化することが推奨されます。

**Performance Tuning Guide** で、本番環境にアプリケーションをデプロイするときに一般的な問題が発生しないようサーバー設定を最適化する方法を必ず確認してください。通常の最適化には、ulimit の設定、ガベッジコレクションの有効化、Java ヒープダンプの作成、スレッドプールサイズの調整 などが含まれます。

また、製品のリリースに既存のパッチを適用するとよいでしょう。EAP の各パッチには、多くのバグ修正が含まれています。詳細は、JBoss EAP**Patching and Upgrading Guide**の Patching JBoss EAP を参照してください。

# 第2章 JBOSS EAP を使用したアプリケーションの開発

## 2.1. 概要

本ガイドは、Red Hat CodeReady 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 CodeReady 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。Red Hat CodeReady 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 Jakarta EE 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。

## 2.2. 開発環境の設定

1. Red Hat CodeReady Studio をダウンロードしてインストールします。
   手順については、Red Hat CodeReady Studio**Installation Guide**の Installing CodeReady Studio stand-alone using the Installer を参照してください。

2. Red Hat CodeReady Studio で JBoss EAP サーバーを設定します。
   手順は、Getting Started with CodeReady Studio Tools の**Downloading, Installing, and Setting Up JBoss EAP from within the IDE**を参照してください。

## 2.3. 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。

### 2.3.1. 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、Jakarta 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**Development Guide**の Using Maven with JBoss EAP を参照してください。

### 2.3.2. 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利用できますが、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 CodeReady 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。

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。

## 2.3.3. クイックスタートのダウンロードおよび実行

### 2.3.3.1. クイックスタートのダウンロード

JBoss EAP には、さまざまな Jakarta EE 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の JBoss EAP ダウンロードページ にログインします。
2. **Version** ドロップダウンメニューで **7.4** を選択します。
3. 一覧で **Red Hat JBoss Enterprise Application Platform 7.4.0 Quickstarts**を見つけ、**Download** をクリックしてクイックスタートが含まれる ZIP ファイルをダウンロードします。
4. ZIP ファイルを希望の場所に保存します。
5. ZIP ファイルを展開します。

### 2.3.3.2. Red Hat CodeReady Studio でのクイックスタートの実行

クイックスタートのダウンロード完了後、Red Hat 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 Red Hat CodeReady 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 Red Hat CodeReady Studio にインポートできます。

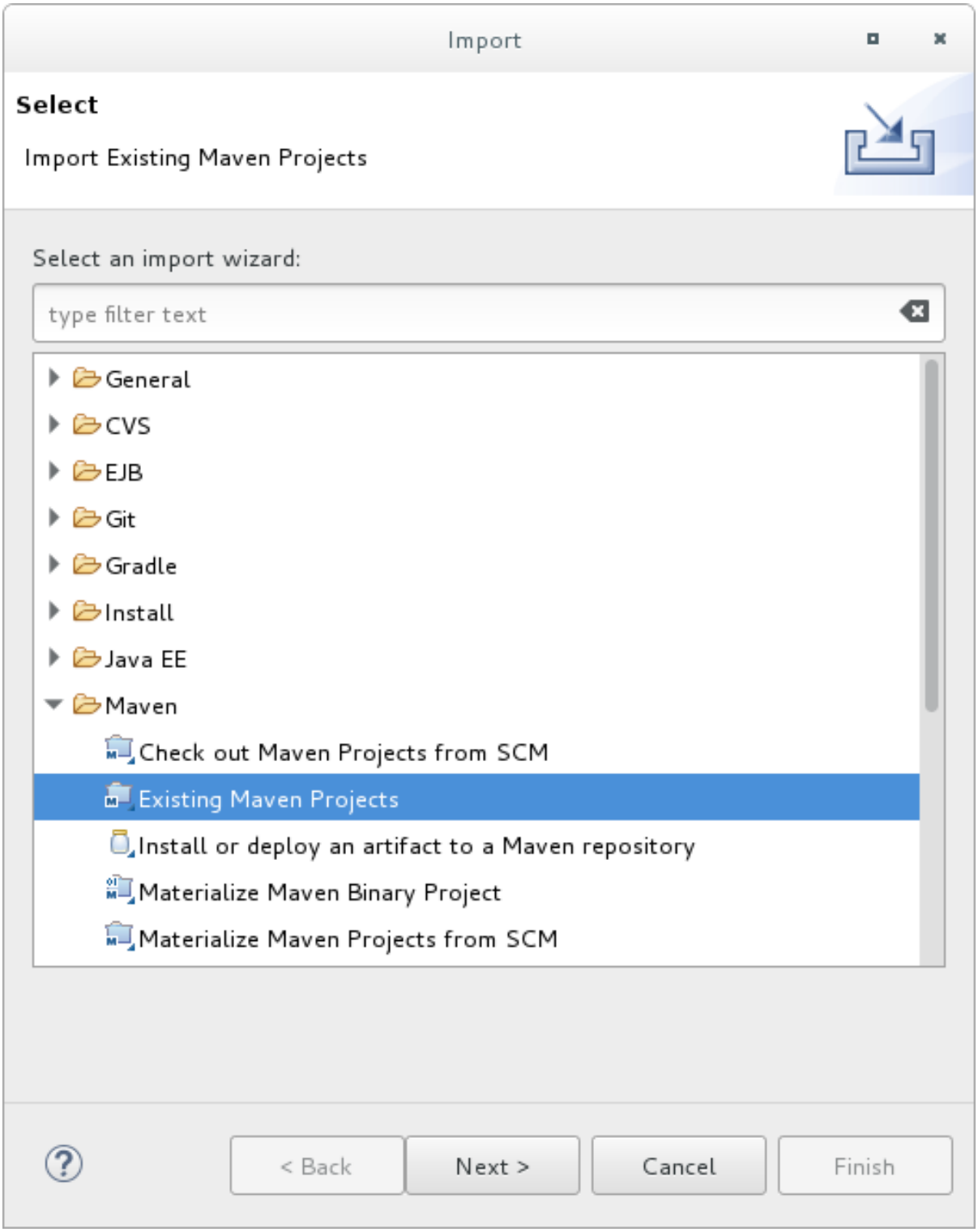
**重要**

Red Hat CodeReady 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. Red Hat CodeReady Studio を起動します。
2. **File → Import** の順に選択します。

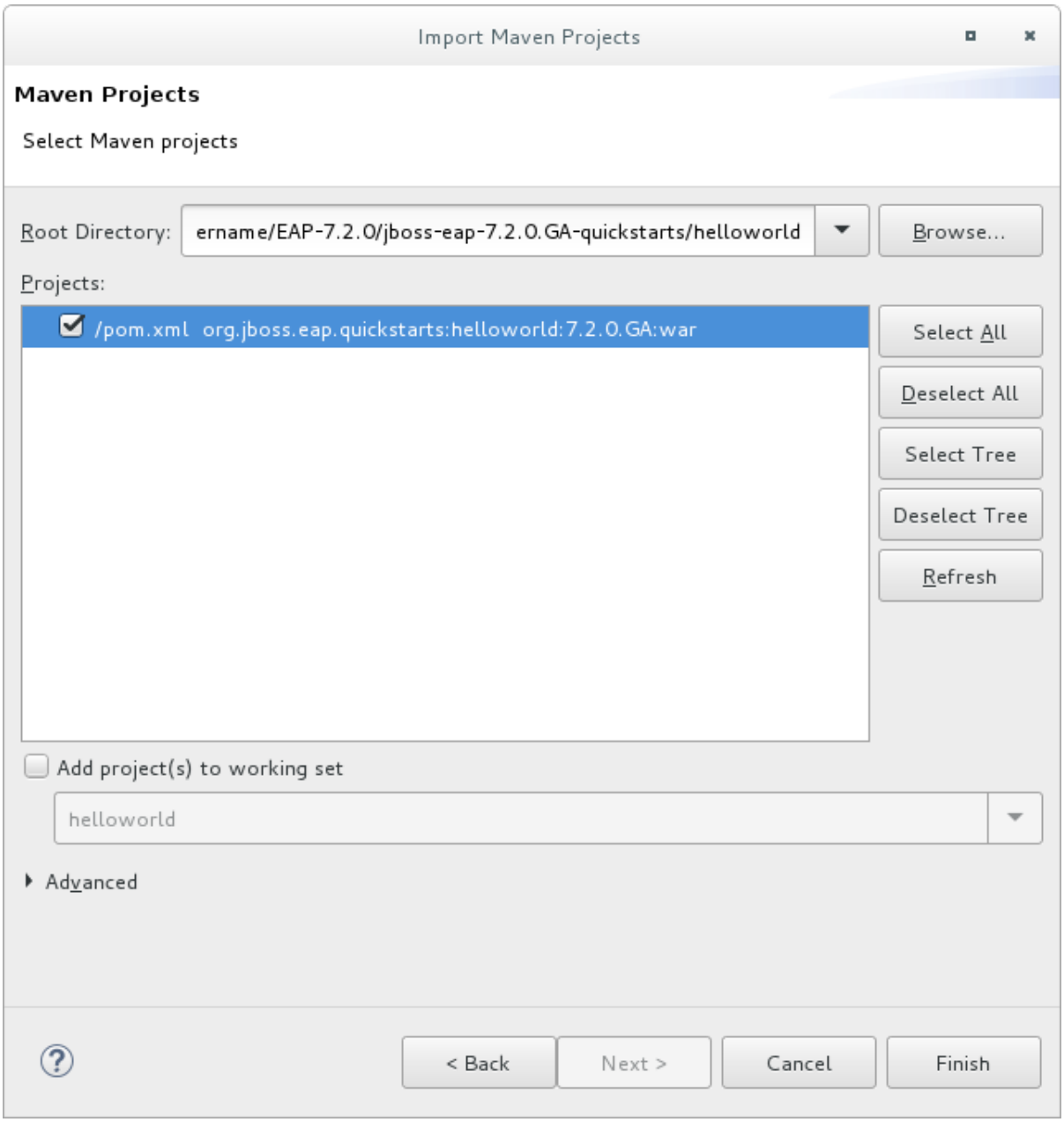
3. **Maven → Existing Maven Projects**の順に選択し、**Next** をクリックします。

図2.1 既存の Maven プロジェクトのインポート




4. 対象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2.2 Maven プロジェクトの選択




5. **Finish** をクリックします。

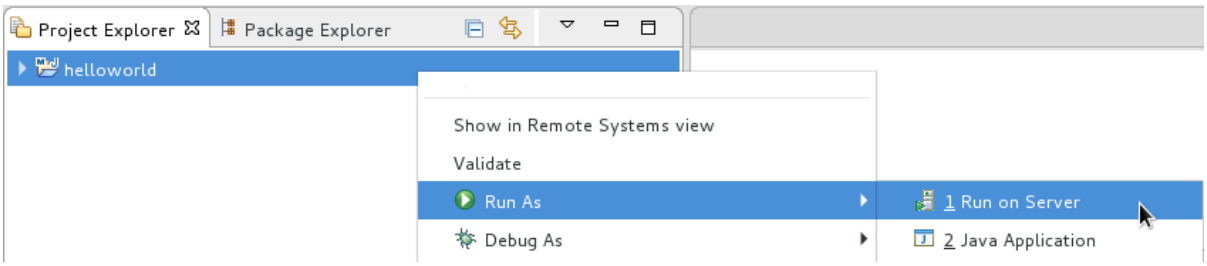
### helloworld クイックスタートの実行

**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 Red Hat CodeReady Studio に追加します。**Getting Started with CodeReady Tools**の Downloading, Installing, and Setting Up JBoss EAP from within the IDE を参照してください。

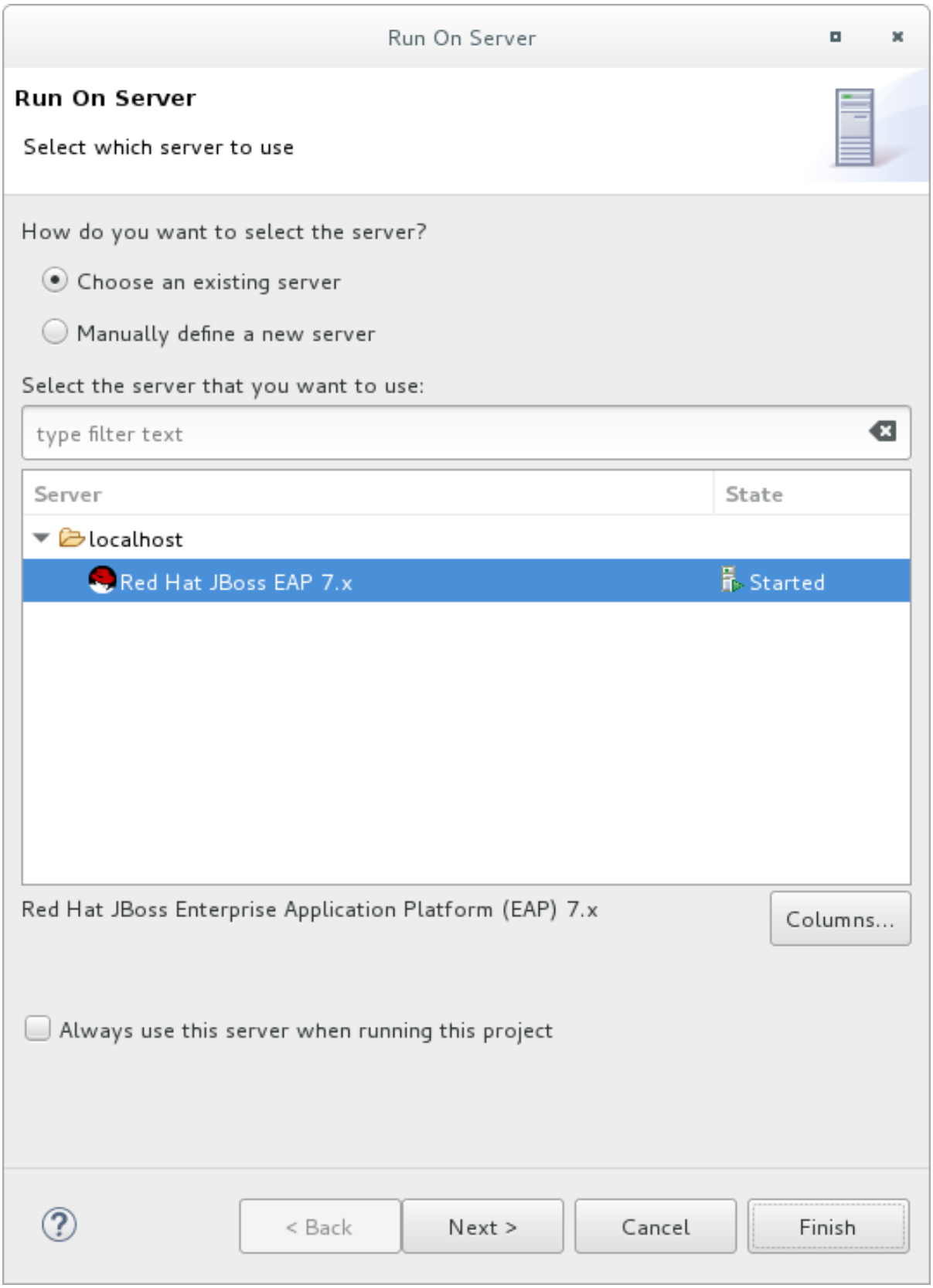
2. **Project Explorer** タブの **helloworld** プロジェクトを右クリックし、**Run As → Run on Server** の順に選択します。

図2.3 Run As - Run on Server



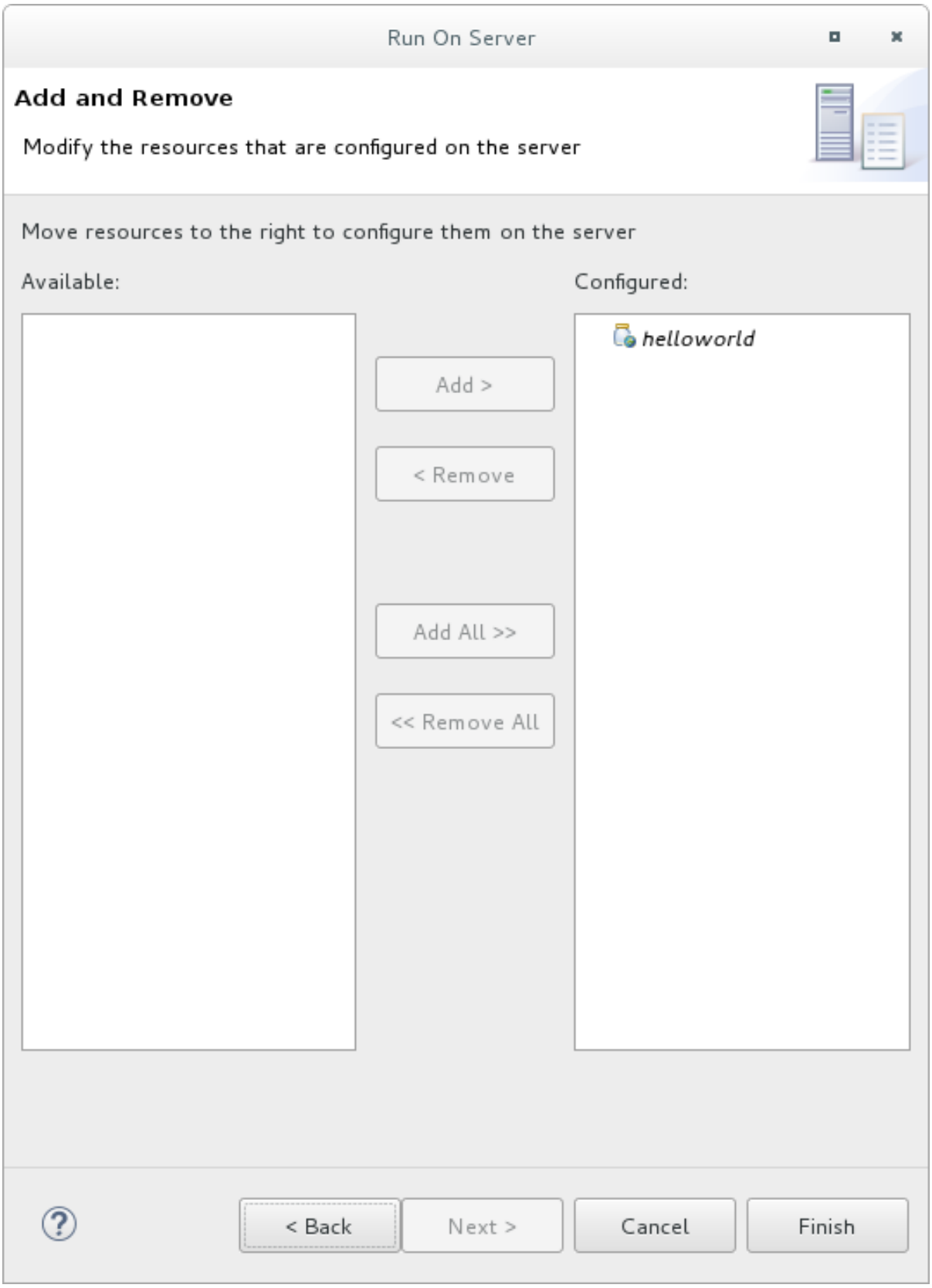

3. サーバーリストから JBoss EAP 7.4 サーバーを選択し、**Next** をクリックします。

図2.4 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2.5 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 7.4 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 で利用でき、**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。

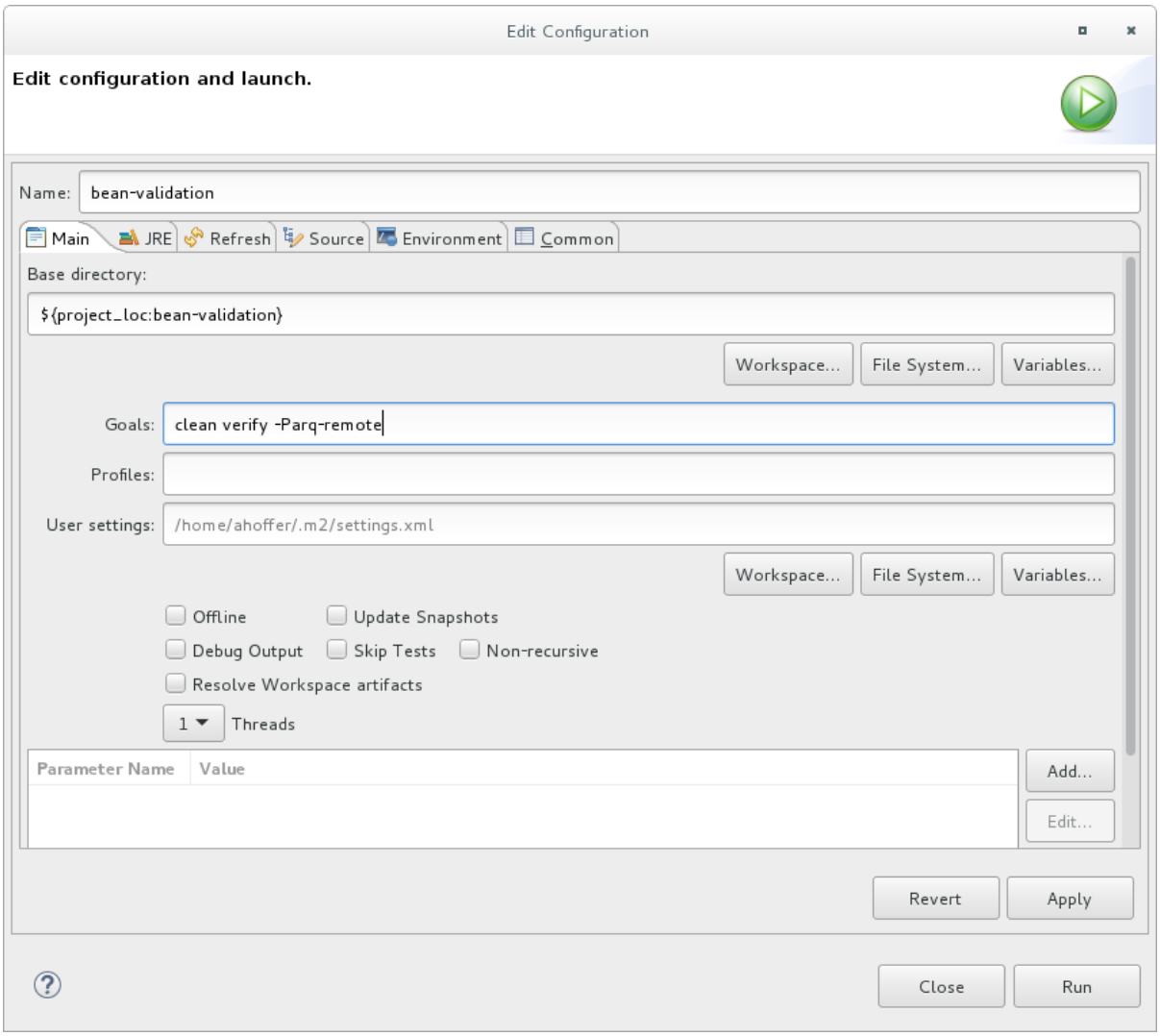
### bean-validation クイックスタートの実行

**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イ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 Red Hat CodeReady 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 Red Hat CodeReady Studio に追加します。**Getting Started with CodeReady Tools**の Downloading, Installing, and Setting 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** の順に選択します。
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2.6 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] ------------------------------------------------------------------------
```

### 2.3.3.3. 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。

## 2.4. クイックスタートサンプルの検証

### 2.4.1. 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 Jakarta Contexts and Dependency Injection (Jakarta 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 CodeReady 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 CodeReady 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

- Red Hat CodeReady Studio をインストールします。手順については、Red Hat CodeReady Studio**Installation Guide**の Installing CodeReady 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。手順は Red Hat 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 Jakarta Contexts and Dependency Injection

Bean で設定されます。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、Jakarta Contexts and Dependency Injection 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. **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2.1 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。 |

| 行 | 注記 |
| --- | --- |
| 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 Jakarta Contexts and Dependency Injection 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 + "!";
    }
}
```

### 2.4.2. 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 Jakarta Server Faces ビューを使用して表示され、ビジネスロジックは 2 つの Jakarta Contexts and Dependency Injection 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 Jakarta Server Faces ビューによって設定され、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 CodeReady Studio を使用してクイックスタートを実行します。

前提条件

- Red Hat CodeReady Studio をインストールします。手順については、Red Hat CodeReady Studio**Installation Guide**の Installing CodeReady 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、Red Hat CodeReady 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 Jakarta Server Faces 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 Jakarta Server Faces 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設定され、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するマーカーファイルで、JBoss EAP がこのアプリケーションで Bean を検索し、Jakarta Contexts and Dependency Injection をアクティベートするよう指示されます。

```
<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。

#### 2.4.2.1. Jakarta Server Faces コードの確認

Jakarta Server Faces 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: Jakarta Server Faces のソースコード**

```
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 Jakarta Contexts and Dependency Injection 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、Red Hat CodeReady 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2.2 Jakarta Server Faces の詳細

| 線 | 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。 |

#### 2.4.2.2. 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**Development Guide**の Use a Qualifier to Resolve an Ambiguous Injection 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す。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 Jakarta Expression Language を使用して Bean が Jakarta Server Faces 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 Jakarta Server Faces 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 JBOSS EAP の使用を開始するための参考情報

引数、属性、およびデフォルトのソケットバインディングを使用して、JBoss EAP の使用を開始するのに役立ちます。たとえば、引数を使用して、デフォルトの JBoss EAP スタンドアロンサーバーに別の設定を指定できます。これは、ニーズを合ったサーバー設定に役立ちます。

## A.1. サーバーランタイムの引数とスイッチ

スタンドアロンサーバーおよび管理対象ドメイン内のサーバーでは、アプリケーションの起動スクリプトで特定のサーバーランタイム引数を使用できます。スクリプトは、 **standalone.xml**、**domain.xml**、および **host.xml** 設定ファイルで定義されている設定とは別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。

サーバーを起動する前に、ターミナルでヘルプスイッチ **-h** または **--help** を実行して、使用可能なパラメーターのリストにアクセスできます。

表A.1 ランタイム引数とスイッチの説明:

| 引数またはスイッチ | サーバータイプ | 説明 |
|---|---|---|
| --admin-only | Standalone | サーバーの実行タイプを **ADMIN_ONLY** に設定します。引数は管理インターフェイスを開き、管理要求を受け入れますが、他のランタイムサービスを開始したり、ユーザー要求を受け入れたりすることはありません。パフォーマンスを最大限にするには、 **-start-mode = admin-only** 引数を使用します。 |
| --admin-only | Domain | ホストコントローラーの実行タイプを **ADMIN_ONLY** に設定すると、ホストコントローラーは管理インターフェイスを開いて管理要求を受け入れますが、サーバーを起動しません。ドメインのマスターホストコントローラーの場合には、スレーブホストコントローラーからの着信接続を受け入れます。 |
| -b=<value>、 -b <value> | Standalone、Domain | システムプロパティー **jboss.bind.address** を設定します。これを使用して、パブリックインターフェイスのバインドアドレスを設定できます。バインドアドレスのデフォルトは **127.0.0.1** です。他のインターフェイ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> | Standalone、Domain | 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 | Domain | 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。 |

| 引数またはスイッチ | サーバータイプ | 説明 |
| --- | --- | --- |
| -c=<config>、-c <config> | Standalone | 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。 |
| -c=<config>、-c <config> | Domain | 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 | Domain | 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する必要があります。 |
| --debug [<port>] | Standalone | 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。引数は、起動スクリプトが引数をサポートしている場合にのみ機能します。 |
| -D<name>[=<value>] | Standalone、Domain | システムプロパティーを設定します。 |
| --domain-config=<config> | Domain | 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。 |
| --git-repo | Standalone | サーバー設定データの管理および永続化に使用される Git リポジトリーの場所。これは、ローカルで保存する場合は **local** を指定し、リモートの場合はリモートリポジトリーへの URL を指定します。 |
| --git-branch | Standalone | 使用する Git リポジトリーのブランチまたはタグ名。ブランチまたはタグ名が存在**しない**と作成されないため、この引数には既存のブランチまたはタグ名を指定する必要があります。タグ名を使用する場合、リポジトリーを detached HEAD 状態にし、今後のコミットがブランチにアタッチされないようにします。タグ名は読み取り専用で、通常複数のノード全体で設定をレプリケートする必要があるときに使用されます。 |
| --git-auth | Standalone | サーバーがリモート Git リポジトリーへの接続時に使用する資格情報を含む Elytron 設定ファイルに対する URL。リモート Git リポジトリーで認証が必要な場合は、この引数を使用できます。Elytron は SSH をサポートしません。Elytron は、パスワードなしで秘密鍵を使用することによるデフォルトの SSH 認証のみをサポートします。ローカル リポジトリーで引数を使用することはできません。 |

| 引数またはスイッチ | サーバータイプ | 説明 |
| --- | --- | --- |
| -h, --help | Standalone、Domain | ヘルプメッセージを表示し、ヘルプインデックスを終了します。 |
| --host-config=<config> | Domain | 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> | Domain | 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できるアドレス。 |
| --interprocess-hc-port=<port> | Domain | 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できるポート。 |
| --master-address=<address> | Domain | 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、引数を使用してマスターホストコントローラーのアドレスを設定できます。 |
| --master-port=<port> | Domain | 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この引数が使用されます。 |
| --read-only-server-config=<config> | Standalone | 使用するサーバー設定ファイルの名前。引数は、元のファイルを上書きしないという点で **--server-config** および **-c** とは異なります。 |
| --read-only-domain-config=<config> | Domain | 使用するドメイン設定ファイルの名前。引数は、初期ファイルを上書きしないという点で **--domain-config** および **-c** とは異なります。 |
| --read-only-host-config=<config> | Domain | 使用するホスト設定ファイルの名前。引数は、初期ファイルを上書きしないという点で **--host-config** とは異なります。 |
| -P=<url>、-P <url>、--properties=<url> | Standalone、Domain | 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> | Domain | 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> | Domain | 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>] | Standalone | セキュリティープロパティーを設定します。 |

| 引数またはスイッチ | サーバータイプ | 説明 |
| --- | --- | --- |
| -secmgr | Standalone、Domain | 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> | Standalone | 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> | Standalone | サーバーの起動モードを設定します。**--admin-only** 引数では、この引数を使用できません。この引数は、以下のエントリーと合わせて使用できます。<br>● **normal**: サーバーは正常に起動します。<br>● **admin-only**: サーバーは管理インターフェイスのみを開き、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。<br>● **suspend**: サーバーは一時停止モードで起動しますが、サーバーが再開するまでサーバーはサービス要求を受信しません。 |
| -u=<value>、 -u <value> | Standalone、Domain | 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。デフォルトは **230.0.0.4** です。 |
| -v、-V、--version | Standalone、Domain | 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 は、追加された設定ファイルを指定して、スイッチの動作を処理します。たとえば、 **-b** と **-u** です。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に設定ファイルを変更した場合には、システムプロパティーを start コマンドに追加しても機能しません。

## A.2. ADD-USER の引数

**add-user.sh** スクリプトまたは **add-user.bat** スクリプトで引数を使用して、これらのスクリプトが認証目的でプロパティーファイルに新しいユーザーを追加する方法を設定できます。

表A.2 add-user 引数の説明

| コマンドライン引数 | 説明 |
| --- | --- |
| -a | アプリケーションレルムでのユーザーの作成。アプリケーションレルムにユーザーを作成しない場合には、スクリプトはデフォルトで管理レルムにユーザーを作成します。 |
| -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。引数を省略すると、スクリプトは **EAP_HOME/domain/configuration /** をデフォルトのディレクトリーとして設定します。 |
| -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。引数を省略すると、スクリプトは **EAP_HOME/Standalone/configuration /** をデフォルトのディレクトリーとして設定します。 |
| -up、--user-properties <value> | 代替のユーザープロパティーファイルの名前。**-sc** または **-dc** 引数を指定した引数を使用して、別の設定ディレクトリーを設定すると、ファイルの絶対パスを設定したり、ファイル名を指定したりできます。 |
| -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。**-sc** または **-dc** 引数を指定した引数を使用して、別の設定ディレクトリーを設定すると、ファイルの絶対パスを設定したり、ファイル名を指定したりできます。 |
| -p、--password <value> | ユーザーのパスワード。 |
| -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。<br>● 英数字 (a-z、A-Z、0-9)<br>● ダッシュ (-)、ピリオド (.)、コンマ (,)、アットマーク (@)<br>● バックスラッシュ (\)<br>● 等号 (=) |
| -r、--realm <value> | 管理インターフェイ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。 |

| コマンドライン引数 | 説明 |
| --- | --- |
| -ds、 --display-secret | 非対話モードで秘密の値を出力します。 |

## A.3. インターフェイス属性

インターフェイス属性を使用して、JBoss EAP インターフェイスを設定できます。

**注記**

テーブル内の属性名は、JBoss EAP が管理モデルにリストする順序で表示されます。XML に表示される要素を確認するには、**EAP_HOME/docs/schema/wildfly-config_5_0.xsd** にあるスキーマ定義ファイルを参照してください。XML 要素のリストは、管理モデルに表示される要素とは異なるものにする必要があります。

表A.3 インターフェイス属性の説明:

| インターフェイス属性 | 説明 |
| --- | --- |
| any | インターフェイスは、ネスト化された基準で選択したもののうち、最低でも 1 つ (必ずしもすべてでなくてよい) を満たす必要があります。 |
| any-address | インターフェイスを使用するソケットにワイルドカードアドレスをバインドする空の属性。この属性には、次の設定オプションがあります。<br>● 属性は、デフォルトとして IPv6 ワイルドカードアドレス (**::**) を使用します。**java.net.prefer IPv4Stack** システムプロパティーを **true** に設定すると、ソケットは IPv4 ワイルドカードアドレス ( **0.0.0.0**) を使用します。<br>● ソケットがデュアルスタックマシン上の IPv6 anylocal アドレスにバインドする場合には、ソケットは IPv6 と IPv4 の両方のトラフィックを受け入れます。<br>● ソケットが IPv4 (IPv4 マップ) anylocal アドレスにバインドする場合には、ソケットは IPv4 トラフィックのみを受け入れます。 |
| 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の指定。 |
| link-local-address | インターフェイスにリンクローカルアドレスが含まれるかどうかの基準を指定する空の属性。 |
| loopback | インターフェイスがループバックインターフェイスとして識別されるかどうかの基準を指定する空の属性。 |

| インターフェイス属性 | 説明 |
| --- | --- |
| loopback-address | マシンのループバックインターフェイスで設定されていない可能性のあるループバックアドレス。値に IP アドレスのない NIC が含まれている場合でも、インターフェイスは属性値を使用するため、属性は **inet-address** タイプとは異なります。 |
| multicast | インターフェイスがマルチキャストをサポートするかどうかの基準を指定する空の属性。 |
| name | インターフェイスの名前。 |
| nic | **eth0**、**eth1**、**lo** などのネットワークインターフェイスの名前。 |
| nic-match | マシンで使用可能なネットワークインターフェイスの名前を受け入れ可能なインターフェイスに一致させる正規表現。 |
| not | インターフェイスを該当させないようにする選択基準を示す属性。 |
| point-to-point | インターフェイスがポイントツーポイントインターフェイスとして識別されるかどうかの基準を指定する空の属性。 |
| public-address | インターフェイスにパブリックにルーティング可能なアドレスが含まれているかどうかの基準を指定する空の属性。 |
| site-local-address | インターフェイスにサイトローカルアドレスが含まれるかどうかの基準を指定する空の属性。 |
| subnet-match | 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を指定します。これは、 **192.168.0.0/16** などの**スラッシュ表記**で記述されます。 |
| up | インターフェイスがアップとして配置されるかどうかの基準を指定する空の属性。 |
| virtual | インターフェイスを仮想インターフェイスとして識別するかどうかの基準を指定する空の属性。 |

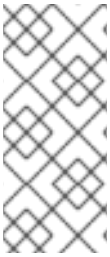
## A.4. ソケットバインディング属性

ソケットバインディング属性を使用して、JBoss EAP サーバーのソケットバインディングを設定できます。

次のタイプのソケットバインディングには、特定の属性があります。

- インバウンドソケットバインディング

- リモートアウトバウンドソケットバインディング
- ローカルアウトバウンドソケットバインディング



**注記**

テーブル内の属性名は、JBoss EAP が管理モデルにリストする順序で表示されます。XML に表示される要素を確認するには、**EAP_HOME/docs/schema/wildfly-config_5_0.xsd** にあるスキーマ定義ファイルを参照してください。XML 要素のリストは、管理モデルに表示される要素とは異なるものにする必要があります。

表A.4 インバウンドソケットバインディング、ソケットバインディング 属性の説明:

| 属性 | 説明 |
|---|---|
| client-mappings | インバウンド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。インバウンドソケットに接続するクライアントは、マッピングで指定された宛先アドレスを使用する必要があります。このアドレスは、アウトバウンド、インターフェイスと一致します。インバウンドソケットバインディングで client-mapping 属性を使用することで、ネットワークアドレス変換を使用するか、複数のネットワークインターフェイスにバインディングを含める高度なネットワークトポロジーを適用できます。各マッピングを宣言された順序で評価する必要があります。つまり、最初の一致をもとに、マッピングの宛先が決まります。 |
| fixed-port | この属性を使用して、ポートの値を固定したままにする必要があるかどうかを判断します。ソケットグループ内の他のソケットに数値オフセットを適用した場合でも、この属性を使用できます。 |
| interface | ソケットがバインドするインターフェイスの名前を設定する属性。この属性を使用して、マルチキャストソケットがリッスンする必要のあるインターフェイスを設定することもできます。宣言されたインターフェイスを定義しない場合、属性は、囲んでいるソケットバインディンググループの **default-interface** 値を使用します。 |
| 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。属性の値を指定しない場合は、マルチキャスト機能を受信するようにソケットは設定されません。 |
| 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。マルチキャストアドレス 属性を設定した場合は、属性を設定する必要があります。 |
| name | ソケットの名前を設定する必要があります。ソケット設定情報へのアクセス権が必要なサービスは、名前でのソケットの検索はできません。 |
| port | ソケットがバインドするポートの番号。ポートオフセットを適用してすべてのポート値をインクリメントまたはデクリメントするようにサーバーを設定した場合は、属性値をオーバーライドする必要があります。 |

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)** の属性の説明

| 属性 | 説明 |
| --- | --- |
| fixed-source-port | ソケットグループ内の他のアウトバウンドソケットに数値オフセットを適用した場合でも、ポート値を固定したままにする必要があるかどうかを決定します。 |
| 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るリモート宛先のポート番号。 |
| source-interface | JBoss EAP がアウトバウンドソケットの送信元アドレスに使用するインターフェイスの名前。 |
| source-port | JBoss EAP がアウトバウンドソケットの送信元ポートとして使用するポート番号。 |

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)** の属性の説明

| 属性 | 説明 |
| --- | --- |
| fixed-source-port | ソケットグループ内の他のアウトバウンドソケットに数値オフセットを適用した場合でも、ポート値を固定したままにする必要があるかどうかを決定します。 |
| socket-binding-ref | JBoss EAP が接続するアウトバウンドソケットのポートを決定するために使用するローカルソケットバインディングの名前。 |
| source-interface | JBoss EAP がアウトバウンドソケットの送信元アドレスに使用するインターフェイスの名前。 |
| source-port | JBoss EAP がアウトバウンドソケットの送信元ポートとして使用するポート番号。 |

## A.5. デフォルトのソケットバインディング

ソケットバインディンググループごとにデフォルトのソケットバインディングを設定できます。

JBoss EAP には、次の 5 種類のデフォルトのソケットバインディングがあります。

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

表A.7 デフォルトの **standard-sockets** ソケットバインディングの説明:

| 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 クラスターリングおよび負荷分散に使用します。 |
| 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間の SSL 暗号化接続。 |
| 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 デフォルトの **ha-sockets** ソケットバインディングの説明:

| 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 クラスターリングおよび負荷分散に使用します。 |
| 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間の SSL 暗号化接続。 |
| 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。 |

| 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 |
|---|---|---|---|
| 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 デフォルトの **full-sockets** ソケットバインディングの説明:

| 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 クラスターリングおよび負荷分散に使用します。 |
| 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間の SSL 暗号化接続。 |
| 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。 |
| 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 デフォルトの **full-ha-sockets** ソケットバインディングの説明:

| Name | 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 クラスターリングおよび負荷分散に使用します。 |

| Name | 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間の SSL 暗号化接続。 |
| 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。 |
| 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 デフォルト **load-balancer-sockets** ソケットバインディングの説明:

| Name | 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。 |

| Name | 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間の SSL 暗号化接続。 |
| 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 MCMP (Mod-Cluster Management Protocol) 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。 |

改訂日時: 2024-06-13