Red Hat JBoss Enterprise Application Platform 8.0

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

Red Hat JBoss Enterprise Application Platform を使用すると、すぐに起動して実行できます。このガイドでは、基本的なインストール、管理、設定などの管理タスクについて説明します。JBoss EAP クイックスタートを使用して、Java EE アプリケーションの作成を開始します。

Last Updated: 2024-03-28

Red Hat JBoss Enterprise Application Platform 8.0 Red Hat JBoss Enterprise Application Platform のスタートガイド

Red Hat JBoss Enterprise Application Platform を使用すると、すぐに起動して実行できます。このガイドでは、基本的なインストール、管理、設定などの管理タスクについて説明します。JBoss EAP クイックスタートを使用して、Java EE アプリケーションの作成を開始します。

法律上の通知

概要

Red Hat JBoss Enterprise Application Platform を使用すると、すぐに起動して実行できます。このガイドでは、基本的なインストール、管理、設定などの管理タスクについて説明します。JBoss EAP クイックスタートを使用して、Java EE アプリケーションの作成を開始します。

目次

JBOSS EAP ドキュメントへのフィードバック (英語のみ) 3

多様性を受け入れるオープンソースの強化 4

第1章 JBOSS EAP の管理 5
1.1. JBOSS EAP のダウンロードおよびインストール 5
1.2. JBOSS EAP の開始および停止 6
1.3. JBOSS EAP の管理 8
1.4. JBOSS EAP のネットワークとポートの設定 21
1.5. JBOSS EAP サーバー設定の最適化 30

第2章 JBOSS EAP を使用したアプリケーションの開発 31
2.1. 概要 31
2.2. 開発環境の設定 31
2.3. クイックスタートサンプルの使用 32
2.4. クイックスタートのダウンロードおよび実行 32
2.5. クイックスタートサンプルの検証 41

付録A JBOSS EAP の使用を開始するための参考情報 51
A.1. サーバーランタイムの引数とスイッチ 51
A.2. ADD-USER の引数 54
A.3. インターフェイス属性 56
A.4. ソケットバインディング属性 57
A.5. デフォルトのソケットバインディング 59

第3章 JBOSS EAP 8.0 のパッケージ名前空間の変更 64
3.1. JAVAX から JAKARTA への名前空間の変更 64

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. **Summary** に課題の簡単な説明を入力します。
3. **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。
4. **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。多様性を受け入れる用語に変更する取り組みの詳細は、Red Hat CTO である Chris Wright のメッセージ を参照してください。

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

.zip ファイルオプションは、プラットフォームに依存しない、JBoss EAP をダウンロードしてインストールする簡単な方法です。

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 **.zip** 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: **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 8.0 を選択します。
5. リストで **Red Hat JBoss Enterprise Application Platform 8.0**を見つけ、**Download** リンクをクリックします。
   .zip 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 カスタマーポータル

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抽出することで、JBoss EAP **.zip** 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。
- オプション: **JAVA_HOME** および **PATH** 環境変数を設定します。

手順

1. JBoss EAP をインストールするサーバーと場所に **.zip** ファイルを移動します。
2. **.zip** 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-8.0.0.zip
```

   b. 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は **EAP_HOME** と呼ばれます。

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、マネージド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。

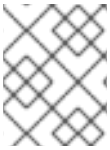
JBoss EAP を停止する方法は、JBoss EAP のインタラクティブインスタンスとバックグラウンドインスタンスのどちらを実行しているかによって異なります。

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。

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

- スタンドアロンサーバー設定ファイル
- サーバーランタイム引数とスイッチ

1.2.2. マネージドドメインでのサーバー用の JBoss EAP の起動

マネージド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。

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
```

   - 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。

マネージド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。

注記

使用可能なすべての起動スクリプト引数とその目的の完全なリストについては、**--help** 引数を使用してください。

関連情報

- 管理対象ドメイン設定ファイル
- サーバーランタイム引数とスイッチ

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 の実行中のインスタンスを用意しておく。

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マネージド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
```

マネージド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

コマンドライン管理 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 JBoss EAP を設定できます。これらの管理インターフェイスを使用して行った変更は自動的に保持され、管理 API は XML 設定ファイルを上書きします。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ます。

- 管理対象ドメイン内のサーバーのデフォルト設定は、**EAP_HOME/domain/configuration/domain.xml** ファイルに保管されます。
- 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。

1.3.1. 管理ユーザー

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。

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。

グラフィカルインストーラーを使用して JBoss EAP をインストールする場合、グラフィカルインストーラーはインストールプロセス中に管理ユーザーを作成します。

1.3.2. 管理ユーザーの追加

add-user スクリプトを使用して JBoss EAP の管理ユーザーを追加できます。このスクリプトは、新しいユーザーをプロパティーファイルに追加して即時認証を行うためのユーティリティーです。

前提条件

- JBoss EAP がインストールされている。

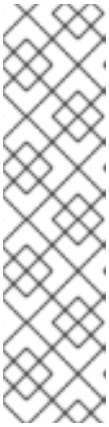
手順

1. 管理 CLI を起動します。
2. **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
```

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

3. **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。これにより、ユーザーが **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ユーザーに付与されます。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。
4. ユーザー名とパスワードを入力します。プロンプトが表示されたら、パスワードを確認する必要があります。



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 (=)

デフォルトでは、JBoss EAP は脆弱なパスワードを許可しますが、警告が表示されます。

5. 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。
6. 情報を確認し、正しければ **yes** を入力します。
7. 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。
ドメインコントローラーに接続する必要があるホストコントローラーを表すユーザーを **ManagementRealm** に追加する場合は、このプロンプトにで **yes** と入力します。エンコードされた機密な値が渡されます。これは、ホストコントローラーの **host*.xml** ファイルに追加する必要のあるユーザーのパスワードを表します。

パラメーターを **add-user** スクリプトに渡すことにより、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。

関連情報

- Add-User ユーティリティーの非対話的な実行

1.3.3. Add-User ユーティリティーの非対話的な実行

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**

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。

注記

使用可能なすべての **add-user** 引数とその目的の完全なリストについては、**--help** 引数を使用してください。

関連情報

- add-user 引数

1.3.4. 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削除、システム設定の設定、他の管理タスクの実行を行います。バッチモードで操作を実行できるため、複数のタスクをグループとして実行できます。

ls (リスト)、**cd** (ディレクトリーの移動)、**pwd** (作業ディレクトリーの出力) など、多くの一般的な端末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。

管理 **CLI** を起動します。

```
$ 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
```

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** の使用に関する情報を表示するには、次のコマンドを使用します。

```
deploy --help
```

管理 **CLI** の終了
以下のコマンドを使用して、管理 CLI を終了します。

```
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サーバーを実行している場合には、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)
```

サブシステム設定の表示
次のコマンドは、**read-resource-description** 操作を使用して、リソースが必要かどうか、プロパティーの置換が利用可能かどうかなど、特定のサブシステム設定の説明を表示します。

```
/subsystem=datasources:read-resource-description(recursive=true)
```

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
次のコマンドを使用して、管理対象ドメインで実行されているサーバーを起動および停止します。

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

1.3.5. 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削除、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、サーバーの再起動またはリロードが必要な変更をユーザーが行ったときに、ライブ通知を使用して管理タスクを実行することもできます。

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。管理コンソールへのアクセス権限を持つユーザーロール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マネージド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。マネージドドメインでは、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。マネージドドメインでは、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。マネージド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。

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。

1.3.5.1. 管理コンソールでのリソース属性の更新

必要な権限を持っている場合、管理コンソールでリソース属性を編集できます。

前提条件

- JBoss EAP が実行されている。
- 選択したリソースを変更するための適切な権限を持っている。
- ユーザーを作成している。

手順

1. 管理コンソールにログインします。デフォルトのポートで実行されているローカルサーバーの場合は、http://localhost:9990/console/index.html で管理コンソールにアクセスできます。
2. 管理コンソールの適切なセクションに移動し、変更するリソースを探します。

3. **Edit** をクリックします。

4. 必要な変更を行います。
   必須フィールドにはアスタリスク (*) が付いています。**Help** をクリックすると、属性の説明を表示できます。

注記

入力フィールドは、属性のタイプに応じて、テキストフィールド、ON/OFF フィールド、またはドロップダウンになります。一部のテキストフィールドでは、入力すると、設定内の他の場所の値が候補として表示されます。

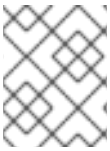
5. **Save** をクリックします。

6. 必要な場合は、サーバーをリロードして変更を反映します。
   有効にするためにリロードが必要な変更を加えると、ポップアップウィンドウが開きます。スタンドアロンサーバーをリロードするには、ポップアップウィンドウで **Reload** をクリックします。マネージドドメイン内のサーバーをリロードするには、**Topology** をクリックし、適切なサーバーを選択して、ドロップダウンリストから **Reload** を選択します。

最近実行した設定アクションの履歴を表示するには、通知アイコンをクリックします。

1.3.5.2. 管理コンソールの有効化または無効化

/core-service=management/management-interface=http-interface リソースの **console-enabled** ブール値属性を設定すると、管理コンソールを有効または無効にできます。ドメインモードマスターホストの場合は、**/host=master/core-service=management/management-interface=http-interface** を使用します。



注記

管理コンソールを有効または無効にした後、JBoss EAP インスタンスを再起動またはリロードする必要があります。

管理コンソールを有効にする場合の例

```
/core-service=management/management-interface=http-interface:write-attribute(name=console-enabled,value=true)
```

管理コンソールを無効にする場合の例

```
/core-service=management/management-interface=http-interface:write-attribute(name=console-enabled,value=false)
```

1.3.5.3. 管理コンソールの言語の変更

管理リソースの言語はデフォルトの英語に設定されています。以下の言語の 1 つを選択することもできます。

- ドイツ語 (de)
- 簡体中国語 (zh-Hans)

- ブラジルポルトガル語 (pt-BR)
- フランス語 (fr)
- スペイン語 (es)
- 日本語 (ja)

前提条件

- JBoss EAP が実行されている。
- ユーザーを作成している。

手順

1. 管理コンソールにログインします。デフォルトのポートで実行されているローカルサーバーの場合は、http://localhost:9990/console/index.html で管理コンソールにアクセスできます。
2. **Settings** をクリックします。
3. **Locale** リストから必要な言語を選択します。
4. **Save** をクリックします。確認ボックスに、アプリケーションのリロードが必要であると表示されます。
5. **Yes** をクリックします。システムによってブラウザーが自動的に更新され、選択したロケールが使用されます。

1.3.5.4. 管理コンソールのタイトルのカスタマイズ

各 JBoss EAP インスタンスを迅速かつ簡単に識別できるように、管理コンソールのタイトルをカスタマイズできます。

前提条件

- JBoss EAP が実行されている。
- ユーザーを作成している。

手順

1. 管理コンソールにログインします。デフォルトのポートで実行されているローカルサーバーの場合は、http://localhost:9990/console/index.html で管理コンソールにアクセスできます。
2. **Settings** をクリックし、**Title** フィールドでタイトルを変更します。
3. **Save** をクリックします。
   確認ボックスに、管理コンソールのリロードが必要であることが表示されます。
4. **Yes** をクリックします。
   システムは Web ブラウザーを自動的に更新し、新しいタイトルがタブヘッダーに表示されます。

1.3.6. 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。これらは、JBoss EAP の起動時に管理 CLI を使用して変更できる設定ファイルの例です。

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の起動時に JBoss EAP が使用するデフォルト設定です。この設定は Jakarta EE の Web Profile と Core Profile に対応するものです。サブシステム、ネットワーク、デプロイメント、ソケットバインディング、その他の設定可能な詳細を含む、サーバーに関するすべての情報を含んでいます。この設定では、メッセージングや高可用性のために必要なサブシステムは提供されません。
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。Jakarta EE フルプロファイルに対応し、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
```

1.3.7. 管理対象ドメイン設定ファイル

マネージド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。これらは、JBoss EAP の起動時に管理 CLI を使用して変更できる設定ファイルの例です。

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、マネージド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。

設定ファイル	目的
host.xml	このファイルには、マネージド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** (詳細は表を参照) 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、サーバーを管理対象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-master.xml** ファイルは、自身をドメインコントローラーとして定義し、サーバーインスタンスの定義はしません。
host-slave.xml	このファイルには、サーバーをマネージド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。これはドメインコントローラーを定義しないため、接続先の **host-slave.xml** のドメインコントローラーアドレスを設定する必要があります。この xml ファイルは、**host-slave.xml** がマシン上で実行され、リモートドメインコントローラーによって管理される設定の例を表しています。マシンは、サーバーインスタンスを定義および起動するためのホストコントローラーとして機能します。ドメインコントローラーは、これらのサーバーインスタンスを管理します。

デフォルトでは、JBoss EAP をマネージド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
```

1.3.8. 設定データのバックアップ

JBoss EAP サーバー設定を復元するには、以下の場所にデータをバックアップする必要があります。

- **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/standalone/data**
  - data/content ディレクトリーに限定されている管理されたデプロイメントのデータをバックアップします。
- **EAP_HOME/standalone/deployments**
  - スタンドアロンサーバーのデプロイメントをバックアップします。
- **EAP_HOME/domain/configuration/**
  - ディレクトリー全体をバックアップして、マネージド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/domain/data**
  - データ/コンテンツディレクトリーに限定されている管理対象ドメインおよび管理対象ドメ

イン内のデプロイメントのデータをバックアップします。

- **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。

1.3.9. 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 を使用してすべてのスナップショットを一覧表示します。

```
: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)
```

1.3.10. スナップショットを使用したサーバーの起動

スナップショットまたは設定の自動保存バージョンを使用してサーバーを起動できます。

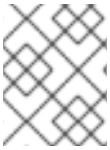
前提条件

- JBoss EAP がインストールされている。
- 設定ファイルのスナップショットを取得している。

手順

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サーバーを実行している場合は、代わりに **--host-config** および **--domain-config=<config>** 引数を使用して設定ファイルを指定します。

1.3.11. プロパティーの置き換え

JBoss EAP で式を使用して、設定のリテラル値の代わりに置き換え可能なプロパティーを定義できます。

standalone*.xml または **domain.xml** 設定ファイルでプロパティー置換を使用すると、プロパティーがシステムプロパティーで見つかった値に置き換えられます。システムプロパティーは、EAP プロファイル xml ファイルで定義するか、コマンドラインターミナルから **-D** コマンドを入力して定義します。

特定のサブシステムでプロパティーの置換が許可されているかどうかを判断するには、次のコマンドを使用して、サブシステム設定の説明を表示します。

```
/subsystem=datasources:read-resource-description(recursive=true)
```

Expressions-allowed 属性が **true** に設定されている場合には、プロパティーを置換できます。

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。

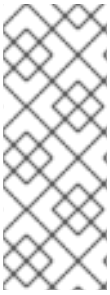
式の解決でサポートされているソースは、システムプロパティーと環境変数です。環境変数を使用して式を解決する場合は、**${env.LANG}** の形式を使用します。

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>
```

次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
```



注記

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。

1.3.12. ネストされた式

式はネストできるため、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}}
```

JBoss EAP はネストされた式を再帰的に評価するため、最初に 内部の式を、次に 外部の式を評価します。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)。

たとえば、データソース定義のパスワードがマスクされている場合は、ネストされた式を使用できます。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>
```

システムプロパティー (**datasource_name**) は、ネストされた式を使用して **ds_ExampleDS** の値を置き換えます。次の行は、データソースの設定例です。

```
<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。

1.3.13. デプロイメント記述子ベースのプロパティー置換

デプロイメント記述子ベースのプロパティー置換は、記述子に基づいてプロパティーを置換するため、アプリケーションと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。

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。

spec-descriptor-property-replacement フラグは Jakarta EE 記述子の置換を制御し、JBoss EAP はデフォルトでこれを 無効 にします。有効にすると、次のデプロイメント記述子のプロパティーを置き換えることができます。

- **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)
```

jboss-descriptor-property-replacement フラグは JBoss 固有の記述子の置換を制御し、JBoss EAP はデフォルトでこれを有効にします。有効にすると、次のデプロイメント記述子のプロパティーを置き換えることができます。

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

以下の管理 CLI コマンドを使用して、JBoss EAP 固有の記述子でプロパティー置換を有効または無効にします。

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

Annotation-property-replacement フラグは、アノテーション内のプロパティーの置換を制御します。デフォルトでは有効になっていません。有効にすると、アプリケーションクラス内のアノテーション属性のプロパティーを置き換えることができます。

以下の管理 CLI コマンドを使用して、アノテーションのプロパティー置換を有効または無効にします。

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

1.4. JBOSS EAP のネットワークとポートの設定

JBoss EAP でさまざまなサービスのネットワークアクセスを設定し、ポートオフセットを使用することで、同じインターフェイスを使用して同じマシン上の複数の JBoss EAP インスタンスを簡単に実行できます。ネットワーク設定は、インターフェイスとソケットバインディングの観点から編成されています。

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マネージド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ります。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

- サーバーランタイム引数とスイッチ

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。

管理コンソールまたは管理 CLI を使用してインターフェイスを設定できます。このセクションの後半には、インターフェイスの追加と更新の例がいくつか含まれています。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

- デフォルトインターフェイス設定

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. マネージド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、マネージド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、マネージドドメインには複数のグループを含むことができます。マネージド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。

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。

関連情報

- ソケットバインディング属性

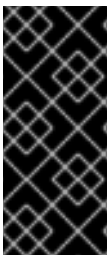
1.4.2.1. 管理ポート

JBoss EAP 8.0 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ります。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

- デフォルトのソケットバインディングのリファレンス

1.4.2.2.1. スタンドアロンサーバーのソケットバインドグループ

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>
```

1.4.2.2.2. 管理対象ドメイン内のソケットバインディンググループ

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。

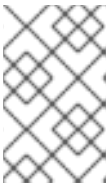
1.4.2.3. ソケットバインディングの設定

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。

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

JBoss EAP のインスタンスをスタンドアロンサーバーとして実行している場合、追加のソケットバインディングを追加することはできません。既存のソケットバインディングを削除、追加、または変更できます。

```
/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>
```

関連情報

- ソケットバインディング属性

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、同じホストおよびインターフェイス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。

このセクションの後の情報には、管理 CLI を使用して管理対象ドメイン内のサーバーに **250** のポートオフセットを設定する例が含まれます。

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

ポートオフセットは、マネージド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
```

ポートオフセットは、システムプロパティー名を使用して JBoss Profiles で定義されます。システムプロパティー名を変更するか削除して、ポートオフセット設定をハードコードすることができます。

```
<socket-binding-group name="standard-sockets" default-interface="public" port-offset
="${jboss.socket.binding.port-offset:0}">
```

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。
   - マネージド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$JAVA_OPTS" = "x" ]; then
   JAVA_OPTS="-Xms1303m -Xmx1303m -XX:MaxPermSize=256m -
Djava.net.preferIPv4Stack=true"
   JAVA_OPTS="$JAVA_OPTS -
Djboss.modules.system.pkgs=$JBOSS_MODULES_SYSTEM_PKGS -Djava.awt.headless=true"
   JAVA_OPTS="$JAVA_OPTS -Djboss.modules.policy-permissions=true"
else
   echo "JAVA_OPTS already set in environment; overriding default settings with values:
$JAVA_OPTS"
fi
```

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

最新の更新を JBoss EAP に適用して、セキュリティー CVE およびその他の顧客から報告されたバグ修正を最新の状態に保ちます。

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します。

一般的な最適化には次のようなものがあります。

- **ulimit** を設定して、オペレーティングシステムが Web 接続に必要な十分なファイル記述子を提供できるようにする
- スレッドプールサイズを調整する

第2章 JBOSS EAP を使用したアプリケーションの開発

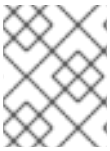
JBoss Tools と Eclipse 統合開発環境 (IDE)、および JBoss EAP 8.0 クイックスタートサンプルを使用して、アプリケーションの開発を開始できます。

注記

JBoss Tools は JBoss EAP 8.0 で非推奨になりました。今後この機能は拡張されず、将来のリリースで削除される可能性があります。

2.1. 概要

JBoss Tools は、JBoss EAP テクノロジーのサポートを強化する Eclipse ベースのプラグインのセットです。JBoss Tools は Eclipse 統合開発環境 (IDE) で動作します。



注記

JBoss Tools は JBoss EAP 8.0 で非推奨になりました。今後この機能は拡張されず、将来のリリースで削除される可能性があります。

重要

JBoss Tools はコミュニティープロジェクトであり、Red Hat ではサポートされていません。JBoss Tools のインスタンスのセットアップと実行については、コミュニティー Web サイト を参照してください。JBoss Tools をダウンロードするには、 JBoss Tools Downloads を参照してください。

JBoss EAP 8.0 には、さまざまな Jakarta EE テクノロジーを使用してアプリケーションの作成を開始できるように、多くのクイックスタートコードサンプルが用意されています。JBoss Tools を使用して、クイックスタートの例を実行できます。

関連情報

- テスト済みの JBoss Tools バージョンの詳細は、Red Hat JBoss Enterprise Application Platform (EAP) と JBoss Tools を参照してください。

2.2. 開発環境の設定

Eclipse IDE で使用する開発環境をセットアップする必要があります。

注記

JBoss Tools は JBoss EAP 8.0 で非推奨になりました。今後この機能は拡張されず、将来のリリースで削除される可能性があります。

手順

1. JBoss ツール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、JBoss Tools インストールガイドの インストール方法 を参照してください。
2. JBoss Tools で JBoss EAP サーバーを設定します。

手順については、JBoss Tools ガイドの How To: Configure the IDE for use with JBoss EAP and JBoss Web Framework Kit を参照してください。

関連情報

- テスト済みの JBoss Tools バージョンの詳細は、Red Hat JBoss Enterprise Application Platform (EAP) と JBoss Tools 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。

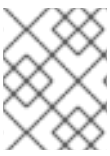
Maven は、リポジトリーを使用してアーカイブを行います。Maven リポジトリーには Java ライブラリー、プラグイン、およびその他のビルドアーティファクトが格納されています。デフォルトのパブリックリポジトリーは Maven 2 Central Repository ですが、複数の開発チームの間で共通のアーティファクトを共有する目的で、社内のプライベートおよび内部リポジトリーとすることが可能です。また、サードパーティーのリポジトリーも利用できます。詳細は Apache Maven プロジェクトおよび Introduction to Repositories ガイドを参照してください。Jakarta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

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て JBoss EAP 8.0 にデプロイするため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。

JBoss Tools には Maven が含まれているため、個別にダウンロードして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。



注記

JBoss Tools は JBoss EAP 8.0 で非推奨になりました。今後この機能は拡張されず、将来のリリースで削除される可能性があります。

2.4. クイックスタートのダウンロードおよび実行

2.4.1. クイックスタートのダウンロード

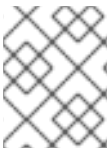
JBoss EAP には、さまざまな Jakarta EE の技術を使用してアプリケーションを作成するのに役立つ包括的なクイックスタートコードサンプルが含まれています。クイックスタートは Red Hat カスタマーポータルからダウンロードできます。

手順

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 **8.0** を選択します。
3. リストで **Red Hat JBoss Enterprise Application Platform 8.0.0 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** ファイルを任意の場所に保存します。
5. **.zip** ファイルを展開します。

2.4.2. クイックスタートの JBoss Tools へのインポート

クイックスタートをダウンロードしたら、JBoss Tools にインポートして JBoss EAP にデプロイできます。



注記

JBoss Tools は JBoss EAP 8.0 で非推奨になりました。今後この機能は拡張されず、将来のリリースで削除される可能性があります。

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 ファイルを使用して、クイックスタートを JBoss Tools に簡単にインポートできます。

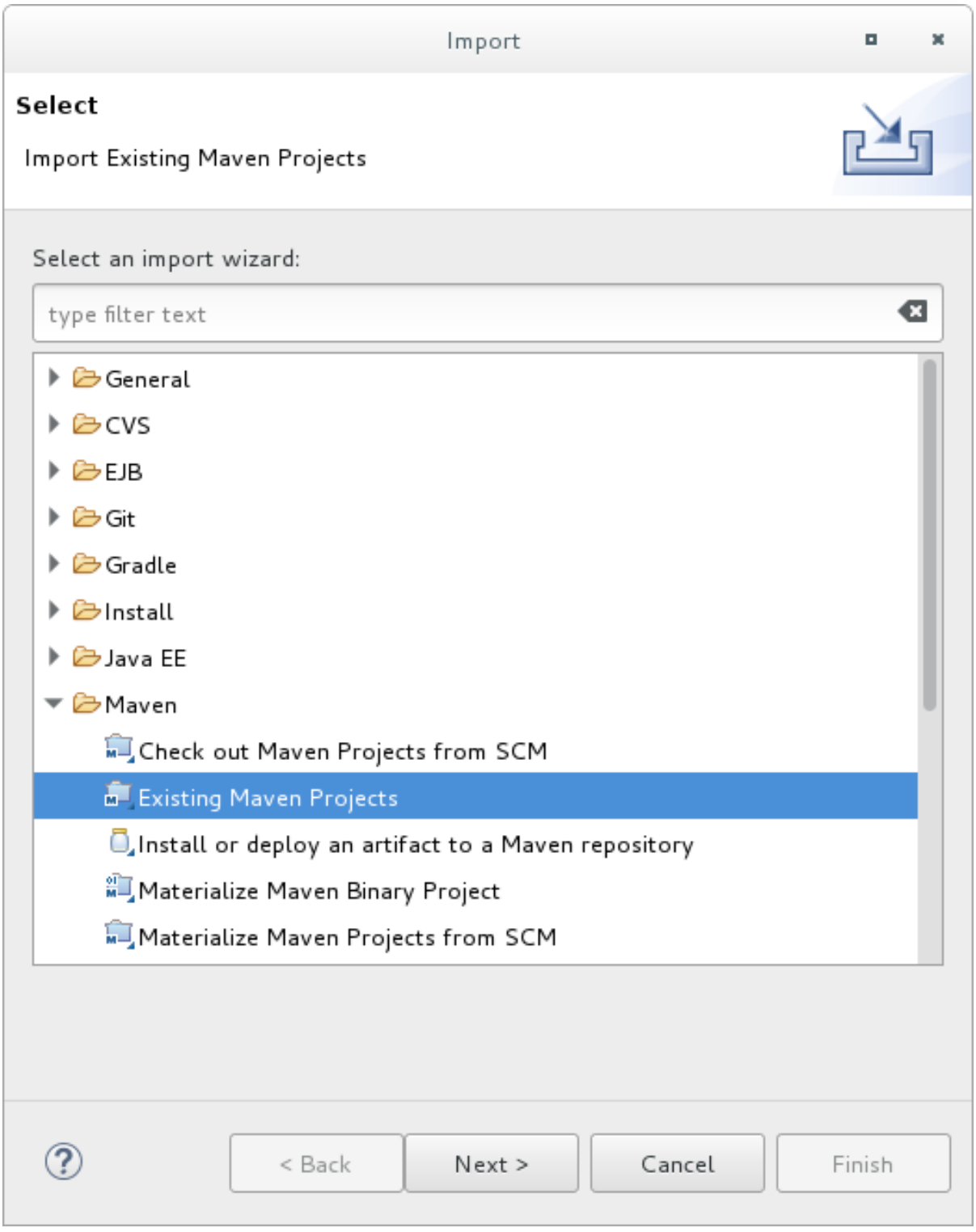
重要

JBoss Tools にインポートするとき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には、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。

手順

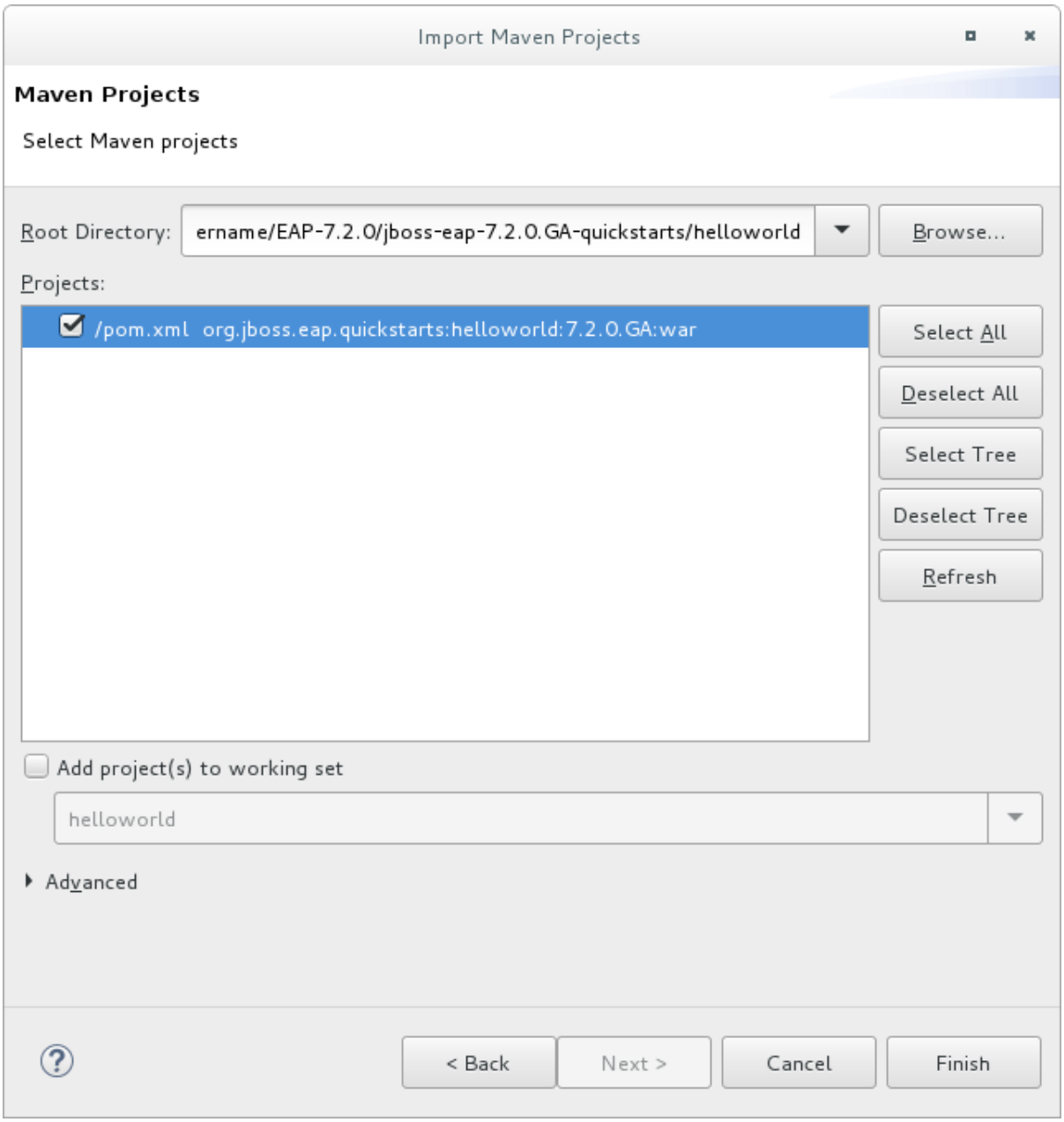
1. JBoss ツールを起動します。
2. **File → Import** の順に選択します。
3. **Maven → Existing Maven Projects**の順に選択し、**Next** をクリックします。

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。

関連情報

- テスト済みの JBoss Tools バージョンの詳細は、Red Hat JBoss Enterprise Application Platform (EAP) と JBoss Tools を参照してください。

2.4.3. 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。

注記

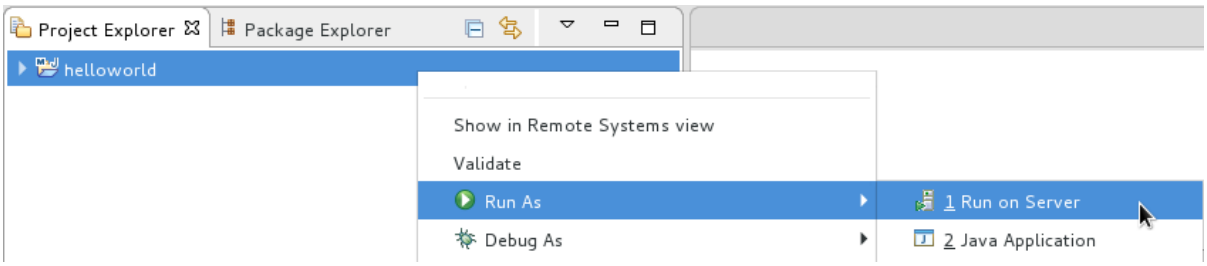
JBoss Tools は JBoss EAP 8.0 で非推奨になりました。今後この機能は拡張されず、将来のリリースで削除される可能性があります。

手順

1. サーバーをまだ定義していない場合は、JBoss EAP サーバーを JBoss Tools に追加します。JBoss Tools ガイドの How To: Configure the IDE for use with JBoss EAP and JBoss Web Framework Kit を参照してください。

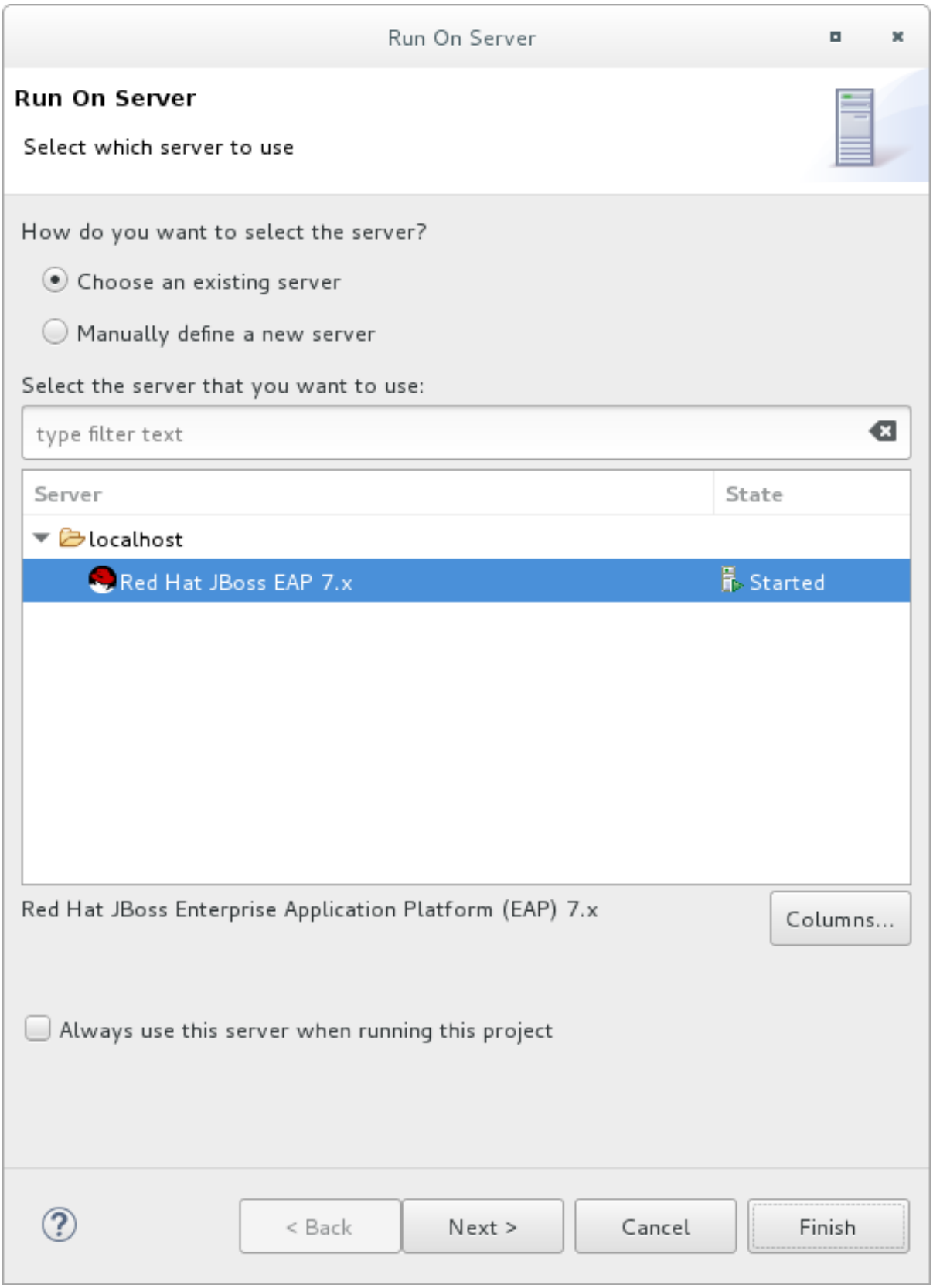
2. **Project Explorer** タブの **helloworld** プロジェクトを右クリックし、**Run As → Run on Server** の順に選択します。

   図2.3 Run As - Run on Server



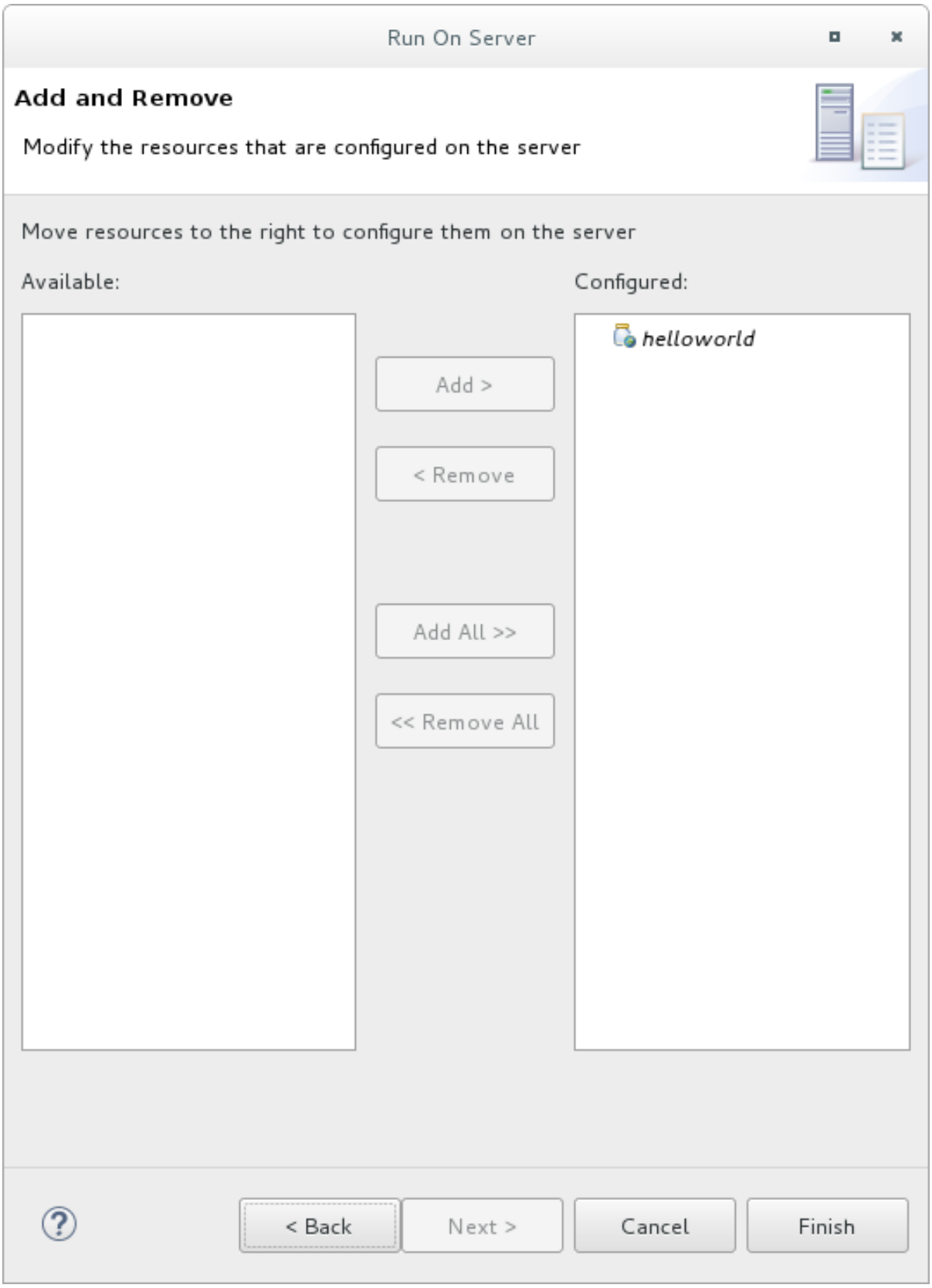
3. サーバーリストから JBoss EAP 8.0 サーバーを選択し、**Next** をクリックします。

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。

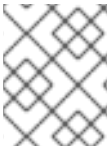
- **Server** タブで、JBoss EAP 8.0 サーバーの状態が **Started** に変わります。
- **Console** タブに、JBoss EAP サーバーの起動と **helloworld** クイックスタートのデプロイメントに関するメッセージが表示されます。

```
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。

2.4.4. 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。



注記

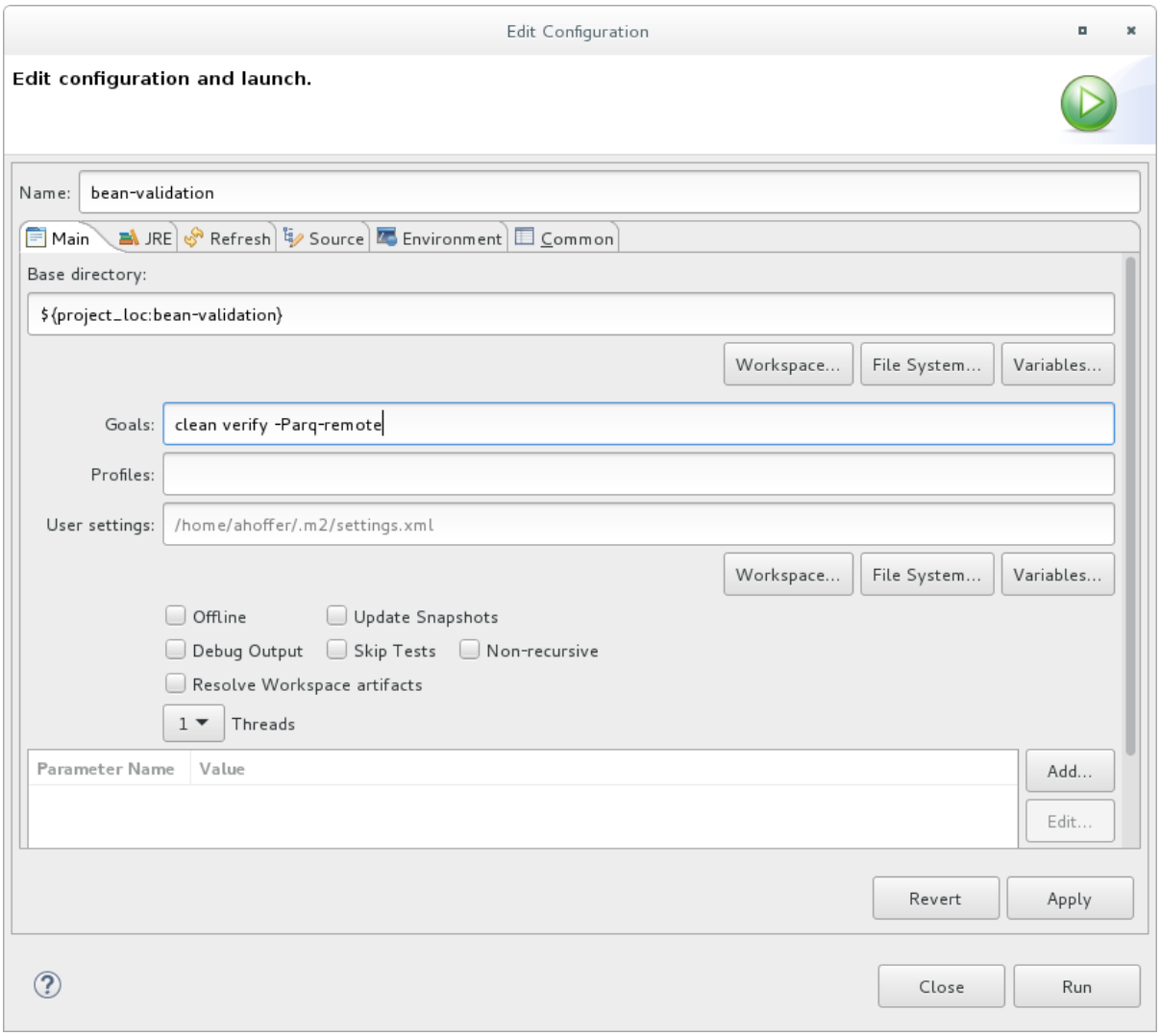
JBoss Tools は JBoss EAP 8.0 で非推奨になりました。今後この機能は拡張されず、将来のリリースで削除される可能性があります。

手順

1. **bean-validation** クイックスタートを JBoss Tools 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、またはサーバーをまだ定義していない場合は、JBoss EAP サーバーを JBoss Tools に追加します。JBoss Tools ガイドの How To: Configure the IDE for use with JBoss EAP and JBoss Web Framework Kit 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] ------------------------------------------------------------------------
```

2.4.5. 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。

手順

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。

2.5. クイックスタートサンプルの検証

2.5.1. 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、JBoss Tools を使用してクイックスタートを実行する方法を示します。JBoss Tools をインストールし、Maven を設定し、**helloworld** クイックスタートをインポートして正常に実行していることを前提としています。

注記

JBoss Tools は JBoss EAP 8.0 で非推奨になりました。今後この機能は拡張されず、将来のリリースで削除される可能性があります。

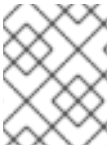
2.5.1.1. ディレクトリー構造の確認

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、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。

2.5.1.2. HelloWorldServlet.java 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。



注記

JBoss Tools は JBoss EAP 8.0 で非推奨になりました。今後この機能は拡張されず、将来のリリースで削除される可能性があります。

前提条件

- JBoss ツールをインストールしている。手順については、JBoss Tools インストールガイドの インストール方法 を参照してください。
- **helloworld** クイックスタートを実行します。
- 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。

手順

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 }
```

2.5.1.2.1. HelloWorldServlet の詳細

このサーブレットは、ブラウザーに情報を送信します。

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。
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。

1. **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 + "!";
    }
}
```

関連情報

- テスト済みの JBoss Tools バージョンの詳細は、Red Hat JBoss Enterprise Application Platform (EAP) と JBoss Tools を参照してください。

2.5.2. 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、デプロイメント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、JBoss Tools を使用してクイックスタートを実行します。

注記

JBoss Tools は JBoss EAP 8.0 で非推奨になりました。今後この機能は拡張されず、将来のリリースで削除される可能性があります。

2.5.2.1. numberguess 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。

前提条件

- JBoss ツールをインストールしている。手順については、JBoss Tools インストールガイドのインストール方法 を参照してください。
- **numberguess** クイックスタートを実行します。
- 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。

注記

JBoss Tools は JBoss EAP 8.0 で非推奨になりました。今後この機能は拡張されず、将来のリリースで削除される可能性があります。

手順

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。

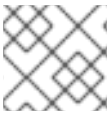
```
<faces-config version="2.2"
  xmlns="http://xmlns.jcp.org/xml/ns/javaee"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="
    http://xmlns.jcp.org/xml/ns/javaee
    http://xmlns.jcp.org/xml/ns/javaee/web-facesconfig_2_2.xsd">

</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。

関連情報

- テスト済みの JBoss Tools バージョンの詳細は、Red Hat JBoss Enterprise Application Platform (EAP) と JBoss Tools を参照してください。

2.5.2.2. 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>
```

次の行番号は、JBoss Tools でファイルを確認したときに表示される行番号に対応しています。

注記

JBoss Tools は JBoss EAP 8.0 で非推奨になりました。今後この機能は拡張されず、将来のリリースで削除される可能性があります。

表2.2 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。

行	注記
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。

2.5.2.3. numberguess 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。

手順

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。**@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 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 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マネージド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	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	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。

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>	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として設定します。
--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。
--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、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認証情報を含む 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>	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できるアドレス。
--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できるポート。
--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、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>	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>	ドメイン	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>	ドメイン	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>	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>]	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** 引数と一緒にこの引数を使用することはできません。この引数は、以下のエントリーと合わせて使用できます。 • **normal**: サーバーは正常に起動します。 • **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。 • **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。 ● 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。 ● 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。 ● 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。 ● 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-mapping	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。**port-offset** 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:

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します。

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:

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。

名前	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。

第3章 JBOSS EAP 8.0 のパッケージ名前空間の変更

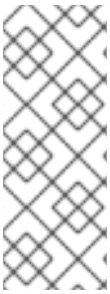
このセクションでは、JBoss EAP 8.0 のパッケージ名前空間の変更に関する追加情報を提供します。JBoss EAP 8.0 は、Jakarta EE 10 および Jakarta EE 10 API の他の多くの実装を完全にサポートします。JBoss EAP 8.0 の Jakarta EE 10 でサポートされる重要な変更点は、パッケージの名前空間の変更です。

3.1. JAVAX から JAKARTA への名前空間の変更

Jakarta EE 8 と EE 10 の主な違いは、EE API Java パッケージの名前が **javax.*** から **jakarta.*** に変更されたことです。これは、Java EE が Eclipse Foundation に移行し、Jakarta EE が確立されたことに続くものです。

アプリケーションを JBoss EAP 7 から JBoss EAP 8 に移行する際には、この名前空間変更への対応が最大のタスクとなります。アプリケーションを Java EE 10 に移行するには、次の手順を完了する必要があります。

- import ステートメントまたはその他のソースコードにおける EE API クラスの使用を **javax** パッケージから **jakarta** パッケージに更新します。
- **javax** で始まる EE 指定のシステムプロパティーまたはその他の設定プロパティーの名前を、**jakarta** で始まるものに更新します。
- **java.util.ServiceLoader** メカニズムを使用してブートストラップされる EE インターフェイスまたは抽象クラスのアプリケーション提供の実装がある場合は、実装クラスを識別するリソースの名前を **META-INF/services/javax.[rest_of_name]** から **META-INF/services/jakarta.[rest_of_name]** に変更します。



注記

Red Hat Migration Toolkit を使用すると、アプリケーションソースコード内の名前空間の更新が容易になります。詳細は、How to use Red Hat Migration Toolkit for Auto-Migration of an Application to the Jakarta EE 10 Namespace を参照してください。ソースコードを移行できない場合は、オープンソースの Eclipse Transformer プロジェクトで、既存の Java アーカイブを **javax** 名前空間から **jakarta** 名前空間に変換するバイトコード変換ツールが提供されています。

注記

この変更は、Java SE に含まれる **javax** パッケージには影響しません。

関連情報

- 詳細は、javax から jakarta パッケージ名前空間への変更 を参照してください。

改訂日時: 2024-02-08