



Red Hat JBoss Enterprise Application Platform 7.0

Getting Started Guide

For Use with Red Hat JBoss Enterprise Application Platform 7.0

Red Hat JBoss Enterprise Application Platform 7.0 Getting Started Guide

For Use with Red Hat JBoss Enterprise Application Platform 7.0

Legal Notice

Copyright © 2018 Red Hat, Inc.

The text of and illustrations in this document are licensed by Red Hat under a Creative Commons Attribution-Share Alike 3.0 Unported license ("CC-BY-SA"). An explanation of CC-BY-SA is available at

<http://creativecommons.org/licenses/by-sa/3.0/>

. In accordance with CC-BY-SA, if you distribute this document or an adaptation of it, you must provide the URL for the original version.

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

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

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

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

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

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

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

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

All other trademarks are the property of their respective owners.

Abstract

This guide provides the basic information to help users get started with Red Hat JBoss Enterprise Application Platform 7.0.

Table of Contents

CHAPTER 1. INTRODUCTION	4
1.1. ABOUT RED HAT JBOSS ENTERPRISE APPLICATION PLATFORM 7	4
1.2. ABOUT THE GETTING STARTED GUIDE	4
CHAPTER 2. ADMINISTERING JBOSS EAP	5
2.1. DOWNLOADING AND INSTALLING JBOSS EAP	5
2.1.1. Installation Prerequisites	5
2.1.2. Download JBoss EAP	5
2.1.3. Install JBoss EAP	5
2.2. STARTING AND STOPPING JBOSS EAP	6
2.2.1. Starting JBoss EAP	6
Start JBoss EAP as a Standalone Server	6
Start JBoss EAP in a Managed Domain	6
2.2.2. Stopping JBoss EAP	7
Stop an Interactive Instance of JBoss EAP	7
Stop a Background Instance of JBoss EAP	7
2.3. JBOSS EAP MANAGEMENT	7
2.3.1. Management Users	8
2.3.1.1. Adding a Management User	8
2.3.1.2. Running the Add-User Utility Non-Interactively	9
Create a User Belonging to Multiple Groups	9
Specify an Alternative Properties File	9
2.3.2. Management Interfaces	10
2.3.2.1. Management CLI	10
Launch the Management CLI	10
Connect to a Running Server	10
Display Help	10
Quit the Management CLI	10
View System Settings	10
Update System Settings	11
Start Servers	11
2.3.2.2. Management Console	11
2.3.3. Configuration Files	12
2.3.3.1. Standalone Server Configuration Files	12
2.3.3.2. Managed Domain Configuration Files	13
2.3.3.3. Backing Up Configuration Data	13
2.3.3.4. Configuration File Snapshots	14
Take a Snapshot	14
List Snapshots	14
Delete a Snapshot	14
Start the Server with a Snapshot	14
2.3.3.5. Property Replacement	15
Nested Expressions	15
Descriptor-Based Property Replacement	16
2.4. NETWORK AND PORT CONFIGURATION	17
2.4.1. Interfaces	17
2.4.1.1. Default Interface Configurations	17
2.4.1.2. Configuring Interfaces	18
Add an Interface with a NIC Value	18
Add an Interface with Several Conditional Values	18
Update an Interface Attribute	19

Add an Interface to a Server in a Managed Domain	19
2.4.2. Socket Bindings	19
2.4.2.1. Management Ports	19
2.4.2.2. Default Socket Bindings	20
Standalone Server	20
Managed Domain	20
2.4.2.3. Configuring Socket Bindings	22
2.4.2.4. Port Offsets	22
2.4.3. IPv6 Addresses	23
Configure the JVM Stack for IPv6 Addresses	23
Update Interface Declarations for IPv6 Addresses	23
CHAPTER 3. DEVELOPING APPLICATIONS USING JBOSS EAP	25
3.1. OVERVIEW	25
3.2. SETTING UP THE DEVELOPMENT ENVIRONMENT	25
3.2.1. Download JBoss Developer Studio	25
3.2.2. Install JBoss Developer Studio	25
3.2.3. Start JBoss Developer Studio	26
3.2.4. Add the JBoss EAP Server to JBoss Developer Studio	26
3.3. USING THE QUICKSTART EXAMPLES	31
3.3.1. About Maven	32
3.3.2. Using Maven with the Quickstarts	32
3.3.3. Download and Run the Quickstarts	32
3.3.3.1. Download the Quickstarts	32
3.3.3.2. Run the Quickstarts in JBoss Developer Studio	33
3.3.3.3. Run the Quickstarts from the Command Line	39
APPENDIX A. REFERENCE MATERIAL	40
A.1. SERVER RUNTIME ARGUMENTS	40
A.2. ADD-USER UTILITY ARGUMENTS	42
A.3. INTERFACE ATTRIBUTES	43
A.4. SOCKET BINDING ATTRIBUTES	45
A.5. DEFAULT SOCKET BINDINGS	46

CHAPTER 1. INTRODUCTION

1.1. ABOUT RED HAT JBOSS ENTERPRISE APPLICATION PLATFORM 7

Red Hat JBoss Enterprise Application Platform 7 (JBoss EAP) is a middleware platform built on open standards and compliant with the Java Enterprise Edition 7 specification. It integrates WildFly Application Server 10 with messaging, high-availability clustering, and other technologies.

JBoss EAP includes a modular structure that allows service enabling only when required, improving startup speed.

The management console and management command-line interface (CLI) make editing XML configuration files unnecessary and add the ability to script and automate tasks.

JBoss EAP provides two operating modes for JBoss EAP instances: standalone server or managed domain. The standalone server operating mode represents running JBoss EAP as a single server instance. The managed domain operating mode allows for the management of multiple JBoss EAP instances from a single control point.

In addition, JBoss EAP includes APIs and development frameworks for quickly developing secure and scalable Java EE applications.

1.2. ABOUT THE GETTING STARTED GUIDE

The purpose of this guide is to get you up and running with JBoss EAP quickly. It covers [administrative](#) tasks such as basic installation, management, and configuration for JBoss EAP. This guide also helps [developers](#) get started writing Java EE 7 applications by using the JBoss EAP quickstarts.

To learn more, see the entire [JBoss EAP documentation suite](#).

CHAPTER 2. ADMINISTERING JBOSS EAP

2.1. DOWNLOADING AND INSTALLING JBOSS EAP

This guide provides basic instructions for downloading and installing JBoss EAP using the ZIP installation, which is platform independent.

See the [Installation Guide](#) for additional details, including instructions for installing JBoss EAP using the graphical installer or RPM package installation methods.

2.1.1. Installation Prerequisites

Verify that the following prerequisites have been met before installing JBoss EAP.

Common Prerequisites

- Your system is supported according to the [JBoss EAP 7 supported configurations](#).
- Your system is up-to-date with Red Hat issued updates and errata.

ZIP Installation Prerequisites

- The user who will run JBoss EAP has read and write access for the installation directory.
- The desired Java development kit has been installed.
- For Hewlett-Packard HP-UX, an **unzip** utility has been installed.
- For Windows Server, the **JAVA_HOME** and **PATH** environment variables have been set.

2.1.2. Download JBoss EAP

The JBoss EAP ZIP installation file can be downloaded from the Red Hat Customer Portal.

1. Log in to the [Red Hat Customer Portal](#)
2. Click **Downloads**.
3. In the **Product Downloads** list, click **Red Hat JBoss Enterprise Application Platform**.
4. Select the desired version in the **Version** drop-down menu.
5. Find the **Red Hat JBoss Enterprise Application Platform 7.x.x** entry in the table and click **Download**.
6. Save the ZIP file to the desired directory.

2.1.3. Install JBoss EAP

Once the JBoss EAP ZIP installation file has been downloaded, it can be installed by extracting the package contents.

1. If necessary, move the ZIP file to the server and location where JBoss EAP should be

installed.

- The user who will be running JBoss EAP must have read and write access to this directory.

2. Extract the ZIP archive.

```
$ unzip jboss-eap-7.x.x.zip
```



NOTE

For Windows Server, right-click the ZIP file and select **Extract All**.

The directory created by extracting the ZIP archive is the top-level directory for the JBoss EAP installation. This is referred to as **EAP_HOME**.

2.2. STARTING AND STOPPING JBOSS EAP

2.2.1. Starting JBoss EAP

JBoss EAP runs in one of two operating modes: as a standalone server or in a managed domain, and is supported on several platforms: Red Hat Enterprise Linux, Windows Server, Oracle Solaris, and Hewlett-Packard HP-UX.

The specific command to start JBoss EAP depends on the underlying platform and the desired operating mode.

Start JBoss EAP as a Standalone Server

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



NOTE

For Windows Server, use the **EAP_HOME\bin\standalone.bat** script.

This startup script uses the **EAP_HOME/bin/standalone.conf** file (or **standalone.conf.bat** for Windows Server) to set some default preferences, such as JVM options. You can customize the settings in this file.

JBoss EAP uses the **standalone.xml** configuration file by default, but can be started using a different one. For details on the available standalone configuration files and how to use them, see the [Standalone Server Configuration Files](#) section.

For a complete listing of all available startup script arguments and their purposes, use the **-help** argument or see the [Server Runtime Arguments](#) section.

Start JBoss EAP in a Managed Domain

The domain controller must be started before the servers in any of the server groups in the domain. Use this script to first start the domain controller, and then for each associated host controller.

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

**NOTE**

For Windows Server, use the **EAP_HOME\bin\domain.bat** script.

This startup script uses the **EAP_HOME/bin/domain.conf** file (or **domain.conf.bat** for Windows Server) to set some default preferences, such as JVM options. You can customize the settings in this file.

JBoss EAP uses the **host.xml** host configuration file by default, but can be started using a different one. For details on the available managed domain configuration files and how to use them, see the [Managed Domain Configuration Files](#) section.

When setting up a managed domain, additional arguments will need to be passed into the startup script. For a complete listing of all available startup script arguments and their purposes, use the **--help** argument or see the [Server Runtime Arguments](#) section.

2.2.2. Stopping JBoss EAP

The way that you stop JBoss EAP depends on how it was started.

Stop an Interactive Instance of JBoss EAP

Press **Ctrl+C** in the terminal where JBoss EAP was started.

Stop a Background Instance of JBoss EAP

Use the management CLI to connect to the running instance and shut down the server.

1. Launch the management CLI.

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

2. Issue the **shutdown** command.

```
shutdown
```

**NOTE**

When running in a managed domain, you must specify the host name to shut down by using the **--host** argument with the **shutdown** command.

2.3. JBOSS EAP MANAGEMENT

JBoss EAP uses a simplified configuration, with one configuration file per standalone server or managed domain. Default configuration for a standalone server is stored in the **EAP_HOME/standalone/configuration/standalone.xml** file and default configuration for a managed domain is stored in the **EAP_HOME/domain/configuration/domain.xml** file. Additionally, the default configuration for a host controller is stored in the **EAP_HOME/domain/configuration/host.xml** file.

JBoss EAP can be configured using the command-line management CLI, web-based management console, Java API, or HTTP API. Changes made using these management interfaces persist automatically and the XML configuration files are overwritten by the Management API. The management CLI and management console are the preferred methods, and it is not recommended to edit the XML configuration files manually.

2.3.1. Management Users

The default JBoss EAP configuration provides local authentication so that a user can access the management CLI on the local host without requiring authentication.

However, you must add a management user if you want to access the management CLI remotely or use the management console, which is considered remote access even if the traffic originates on the local host. If you attempt to access the management console before adding a management user, you will receive an error message.

If JBoss EAP is installed using the graphical installer, then a management user is created during the installation process.

This guide covers simple user management for JBoss EAP using the **add-user** script, which is a utility for adding new users to the properties files for out-of-the-box authentication. For more advanced authentication and authorization options, such as LDAP or Role-Based Access Control (RBAC), see the *Core Management Authentication* section of the JBoss EAP [Security Architecture](#) guide.

2.3.1.1. Adding a Management User

1. Run the **add-user** utility script and follow the prompts.

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



NOTE

For Windows Server, use the **EAP_HOME\bin\add-user.bat** script.

2. Press **ENTER** to select the default option **a** to add a management user. This user will be added to the *ManagementRealm* and will be authorized to perform management operations using the management console or management CLI. The other choice (**b**) adds a user to the *ApplicationRealm*, which is used for applications and provides no particular permissions.
3. Enter the desired username and password. You will be prompted to confirm the password. By default, JBoss EAP allows weak passwords but will issue a warning. See the [Setting Add-User Utility Password Restrictions](#) section of the JBoss EAP *Configuration Guide* for details on changing this default behavior.
4. Enter a comma-separated list of groups to which the user belongs. If you do not want the user to belong to any groups, press **ENTER** to leave it blank.
5. Review the information and enter **yes** to confirm.
6. Determine whether this user represents a remote JBoss EAP server instance. For a basic management user, enter **no**. One type of user that may need to be added to the *ManagementRealm* is a user representing another instance of JBoss EAP, which must be able to authenticate to join as a member of a cluster. If this is the case, then answer **yes** to this prompt and you will be given a hashed secret value representing the user's password, which will need to be added to a different configuration file.

Users can also be created non-interactively by passing parameters to the **add-user** script.

This approach is not recommended on shared systems, because the passwords will be visible in log and history files. For more information, see [Running the Add-User Utility Non-Interactively](#).

2.3.1.2. Running the Add-User Utility Non-Interactively

You can run the **add-user** script non-interactively by passing in arguments on the command line. At a minimum, the username and password must be provided.



WARNING

This approach is not recommended on shared systems, because the passwords will be visible in log and history files.

Create a User Belonging to Multiple Groups

The following command adds a management user (**mgmtuser1**) with the **guest** and **mgmtgroup** groups.

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

Specify an Alternative Properties File

By default, user and group information created using the **add-user** script are stored in properties files located in the server configuration directory.

User information is stored in the following properties files:

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

Group information is stored in the following properties files:

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

These default directories and properties file names can be overridden. The following command adds a new user, specifying a different name and location for the user properties files.

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

The new user was added to the user properties files now located at **/path/to/standaloneconfig/newname.properties** and **/path/to/domainconfig/newname.properties**. Note that these files must already exist or you will see an error.

For a complete listing of all available **add-user** arguments and their purposes, use the **--help** argument or see the [Add-User Utility Arguments](#) section.

2.3.2. Management Interfaces

2.3.2.1. Management CLI

The management command-line interface (CLI) is a command-line administration tool for JBoss EAP.

Use the management CLI to start and stop servers, deploy and undeploy applications, configure system settings, and perform other administrative tasks. Operations can be performed in batch mode, allowing multiple tasks to be run as a group.

Many common terminal commands are available, such as **ls**, **cd**, and **pwd**. The management CLI also supports tab completion.

For detailed information on using the management CLI, including commands and operations, syntax, and running in batch mode, see the JBoss EAP [Management CLI Guide](#).

Launch the Management CLI

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



NOTE

For Windows Server, use the **EAP_HOME\bin\jboss-cli.bat** script.

Connect to a Running Server

```
connect
```

Or you can launch the management CLI and connect in one step by using the **EAP_HOME/bin/jboss-cli.sh --connect** command.

Display Help

Use the following command for general help.

```
help
```

Use the following command for help on a specific command.

```
deploy --help
```

Quit the Management CLI

```
quit
```

View System Settings

The following command uses the **read-attribute** operation to display whether the example datasource is enabled.

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

```
{
  "outcome" => "success",
  "result" => true
}
```

When running in a managed domain, you must specify which profile to update by preceding the command with `/profile=PROFILE_NAME`.

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

Update System Settings

The following command uses the `write-attribute` operation to disable the example datasource.

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

Start Servers

The management CLI can also be used to start and stop servers when running in a managed domain.

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

2.3.2.2. Management Console

The management console is a web-based administration tool for JBoss EAP.

Use the management console to start and stop servers, deploy and undeploy applications, tune system settings, and make persistent modifications to the server configuration. The management console also has the ability to perform administrative tasks, with live notifications when any changes performed by the current user require the server instance to be restarted or reloaded.

In a managed domain, server instances and server groups in the same domain can be centrally managed from the management console of the domain controller.

For a JBoss EAP instance running on the local host using the default management port, the management console can be accessed through a web browser at <http://localhost:9990/console/App.html>. You will need to authenticate with a user that has permissions to access the management console.

The management console provides the following tabs for navigating and managing your JBoss EAP standalone server or managed domain.

Home

Learn how to accomplish several common configuration and management tasks. Take a tour to become familiar with the JBoss EAP management console.

Deployments

Add, remove, and enable deployments. In a managed domain, assign deployments to server groups.

Configuration

Configure available subsystems, which provide capabilities such as web services, messaging, or high availability. In a managed domain, manage the profiles that contain different subsystem configurations.

Runtime

View runtime information, such as server status, JVM usage, and server logs. In a managed domain, manage your hosts, server groups, and servers.

Access Control

Assign roles to users and groups when using Role-Based Access Control.

Patching

Apply patches to your JBoss EAP instances.



NOTE

To take a tour of the updated management console, click the **Take a Tour** link on the management console home page.

To view details about the form fields, click the **Need Help?** link.

To view the message history of configuration actions you have performed, click the **Messages** link in the top-right of the management console.

2.3.3. Configuration Files

2.3.3.1. Standalone Server Configuration Files

The standalone configuration files are located in the **EAP_HOME/standalone/configuration/** directory. A separate file exists for each of the four predefined profiles (*default*, *ha*, *full*, *full-ha*).

Table 2.1. Standalone Configuration Files

Configuration File	Purpose
standalone.xml	This standalone configuration file is the default configuration that is used when you start your standalone server. It contains all information about the server, including subsystems, networking, deployments, socket bindings, and other configurable details. It does not provide the subsystems necessary for messaging or high availability.
standalone-ha.xml	This standalone configuration file includes all of the default subsystems and adds the modcluster and jgroups subsystems for high availability. It does not provide the subsystems necessary for messaging.
standalone-full.xml	This standalone configuration file includes all of the default subsystems and adds the messaging-activemq and iiop-openjdk subsystems. It does not provide the subsystems necessary for high availability.
standalone-full-ha.xml	This standalone configuration file includes support for every possible subsystem, including those for messaging and high availability.

By default, starting JBoss EAP as a standalone server uses the **standalone.xml** file. To start JBoss EAP with a different configuration, use the **--server-config** argument. For example,

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

2.3.3.2. Managed Domain Configuration Files

The managed domain configuration files are located in the **EAP_HOME/domain/configuration/** directory.

Table 2.2. Managed Domain Configuration Files

Configuration File	Purpose
domain.xml	This is the main configuration file for a managed domain. Only the domain master reads this file. This file contains the configurations for all of the profiles (<i>default, ha, full, full-ha</i>).
host.xml	This file includes configuration details specific to a physical host in a managed domain, such as network interfaces, socket bindings, the name of the host, and other host-specific details. The host.xml file includes all of the features of both host-master.xml and host-slave.xml , which are described below.
host-master.xml	This file includes only the configuration details necessary to run a server as the master domain controller.
host-slave.xml	This file includes only the configuration details necessary to run a server as a managed domain host controller.

By default, starting JBoss EAP in a managed domain uses the **host.xml** file. To start JBoss EAP with a different configuration, use the **--host-config** argument. For example,

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

2.3.3.3. Backing Up Configuration Data

In order to later restore the JBoss EAP server configuration, items in the following locations should be backed up:

- **EAP_HOME/standalone/configuration/**
 - Back up the entire directory to save user data, server configuration, and logging settings for standalone servers.
- **EAP_HOME/domain/configuration/**
 - Back up the entire directory to save user and profile data, domain and host configuration, and logging settings for managed domains.
- **EAP_HOME/modules/**
 - Back up any custom modules.

- **EAP_HOME/welcome-content/**
 - Back up any custom welcome content.
- **EAP_HOME/bin/**
 - Back up any custom scripts or startup configuration files.

2.3.3.4. Configuration File Snapshots

To assist in the maintenance and management of the server, JBoss EAP creates a timestamped version of the original configuration file at the time of startup. Any additional configuration changes made by management operations will result in the original file being automatically backed up, and a working copy of the instance being preserved for reference and rollback. Additionally, configuration snapshots can be taken, which are point-in-time copies of the current server configuration. These snapshots can be saved and loaded by an administrator.

The following examples use the **standalone.xml** file, but the same process applies to the **domain.xml** and **host.xml** files.

Take a Snapshot

Use the management CLI to take a snapshot of the current configurations.

```
:take-snapshot
{
  "outcome" => "success",
  "result" =>
"EAP_HOME/standalone/configuration/standalone_xml_history/snapshot/2015102
2-133109702standalone.xml"
}
```

List Snapshots

Use the management CLI to list all snapshots that have been taken.

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

Delete a Snapshot

Use the management CLI to delete a snapshot.

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

Start the Server with a Snapshot

The server can be started using a snapshot or an automatically-saved version of the configuration.

1. Navigate to the **EAP_HOME/standalone/configuration/standalone_xml_history** directory and identify the snapshot or saved configuration file to be loaded.
2. Start the server and point to the selected configuration file. Pass in the file path relative to the configuration directory, **EAP_HOME/standalone/configuration/**.

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



NOTE

When running in a managed domain, use the **--host-config** argument instead to specify the configuration file.

2.3.3.5. Property Replacement

JBoss EAP allows you to use expressions to define replaceable properties in place of literal values in the configuration. Expressions use the format **\${PARAMETER:DEFAULT_VALUE}**. If the specified parameter is set, then the parameter's value will be used. Otherwise, the default value provided will be used.

The supported sources for resolving expressions are system properties, environment variables, and the vault. For deployments only, the source can be properties listed in a **META-INF/jboss.properties** file in the deployment archive. For deployment types that support subdeployments, the resolution is scoped to all subdeployments if the properties file is in the outer deployment, for example the EAR. If the properties file is in the subdeployment, then the resolution is scoped just to that subdeployment.

The example below from the **standalone.xml** configuration file sets the **inet-address** for the **public** interface to **127.0.0.1** unless the **jboss.bind.address** parameter is set.

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

The **jboss.bind.address** parameter can be set when starting EAP as a standalone server with the following command:

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

Nested Expressions

Expressions can be nested, which allows for more advanced use of expressions in place of fixed values. The format of a nested expression is like that of a normal expression, but one expression is embedded in the other, for example:

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

Nested expressions are evaluated recursively, so the *inner* expression is first evaluated, then the *outer* expression is evaluated. Expressions may also be recursive, where an expression resolves to another expression, which is then resolved. Nested expressions are

permitted anywhere that expressions are permitted, with the exception of management CLI commands.

An example of where a nested expression might be used is if the password used in a datasource definition is masked. The configuration for the datasource might have the following line:

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

The value of `ds_ExampleDS` could be replaced with a system property (`datasource_name`) using a nested expression. The configuration for the datasource could instead have the following line:

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

JBoss EAP would first evaluate the expression `${datasource_name}`, then input this to the larger expression and evaluate the resulting expression. The advantage of this configuration is that the name of the datasource is abstracted from the fixed configuration.

Descriptor-Based Property Replacement

Application configuration, such as datasource connection parameters, typically varies between development, testing, and production environments. This variance is sometimes accommodated by build system scripts, as the Java EE specification does not contain a method to externalize these configurations. With JBoss EAP, you can use descriptor-based property replacement to manage configuration externally.

Descriptor-based property replacement substitutes properties based on descriptors, allowing you to remove assumptions about the environment from the application and the build chain. Environment-specific configurations can be specified in deployment descriptors rather than annotations or build system scripts. You can provide configuration in files or as parameters at the command line.

There are several flags in the `ee` subsystem that control whether property replacement is applied.

JBoss-specific descriptor replacement is controlled by the `jboss-descriptor-property-replacement` flag and is *enabled* by default. When enabled, properties can be replaced in the following deployment descriptors:

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

The following management CLI command can be used to enable or disable property replacement in JBoss-specific descriptors:

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

Java EE descriptor replacement controlled by the **spec-descriptor-property-replacement** flag and is *disabled* by default. When enabled, properties can be replaced in the following deployment descriptors:

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

The following management CLI command can be used to enable or disable property replacement in Java EE descriptors:

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

2.4. NETWORK AND PORT CONFIGURATION

2.4.1. Interfaces

JBoss EAP references named interfaces throughout the configuration. This allows the configuration to reference individual interface declarations with logical names, rather than requiring the full details of the interface at each use.

This also allows for easier configuration in a managed domain, where network interface details can vary across multiple machines. Each server instance can correspond to a logical name group.

The **standalone.xml**, **domain.xml**, and **host.xml** files all include interface declarations. There are several preconfigured interface names, depending on which default configuration is used. The **management** interface can be used for all components and services that require the management layer, including the HTTP management endpoint. The **public** interface can be used for all application-related network communications. The **unsecure** interface is used for IIOP sockets in the standard configuration. The **private** interface is used for JGroups sockets in the standard configuration.

2.4.1.1. Default Interface Configurations

```
<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>
```

By default, JBoss EAP binds these interfaces to **127.0.0.1**, but these values can be overridden at runtime by setting the appropriate property. For example, the **inet-address** of the **public** interface can be set when starting JBoss EAP as a standalone server with the following command.

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

Alternatively, you can use the **-b** switch on the server start command line. For more information about server start options, see [Server Runtime Arguments](#).



IMPORTANT

If you modify the default network interfaces or ports that JBoss EAP uses, you must also remember to change any scripts that use the modified interfaces or ports. These include JBoss EAP service scripts, as well as remembering to specify the correct interface and port when accessing the management console or management CLI.

2.4.1.2. Configuring Interfaces

Network interfaces are declared by specifying a logical name and selection criteria for the physical interface. The selection criteria can reference a wildcard address or specify a set of one or more characteristics that an interface or address must have in order to be a valid match. For a listing of all available interface selection criteria, see the [Interface Attributes](#) section.

Interfaces can be configured using the management console or the management CLI. Below are several examples of adding and updating interfaces. The management CLI command is shown first, followed by the corresponding configuration XML.

Add an Interface with a NIC Value

Add a new interface with a NIC value of **eth0**.

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

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

Add an Interface with Several Conditional Values

Add a new interface that matches any interface/address on the correct subnet if it is up, supports multicast, and is not point-to-point.

```
/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>
```

Update an Interface Attribute

Update the **public** interface's default **inet-address** value, keeping the **jboss.bind.address** property to allow for this value to be set at runtime.

```
/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>
```

Add an Interface to a Server in a Managed Domain

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

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

2.4.2. Socket Bindings

Socket bindings and socket binding groups allow you to define network ports and their relationship to the networking interfaces required for your JBoss EAP configuration. A socket binding is a named configuration for a socket. A socket binding group is a collection of socket binding declarations that are grouped under a logical name.

This allows other sections of the configuration to reference socket bindings by their logical name, rather than requiring the full details of the socket configuration at each use.

The declarations for these named configurations can be found in the **standalone.xml** and **domain.xml** configuration files. A standalone server contains only one socket binding group, while a managed domain can contain multiple groups. You can create a socket binding group for each server group in the managed domain, or share a socket binding group between multiple server groups.

The ports used by JBoss EAP by default depend on which socket binding groups are used and the requirements of your individual deployments.

2.4.2.1. Management Ports

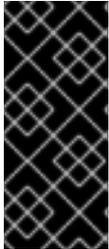
Management ports were consolidated in JBoss EAP 7. By default, JBoss EAP 7 uses port **9990** for both native management, used by the management CLI, and HTTP management, used by the web-based management console. Port **9999**, which was used as the native management port in JBoss EAP 6, is no longer used but can still be enabled if desired.

If HTTPS is enabled for the management console, then port **9993** is used by default.

2.4.2.2. Default Socket Bindings

JBoss EAP ships with a socket binding group for each of the four predefined profiles (*default*, *ha*, *full*, *full-ha*).

For detailed information about the default socket bindings, such as default ports and descriptions, see the [Default Socket Bindings](#) section.



IMPORTANT

If you modify the default network interfaces or ports that JBoss EAP uses, you must also remember to change any scripts that use the modified interfaces or ports. These include JBoss EAP service scripts, as well as remembering to specify the correct interface and port when accessing the management console or management CLI.

Standalone Server

When running as a standalone server, only one socket binding group is defined per configuration file. Each standalone configuration file (**standalone.xml**, **standalone-ha.xml**, **standalone-full.xml**, **standalone-full-ha.xml**) defines socket bindings for the technologies used by its corresponding profile.

For example, the default standalone configuration file (**standalone.xml**) specifies the below socket bindings.

```
<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>
```

Managed Domain

When running in a managed domain, all socket binding groups are defined in the **domain.xml** file. There are four predefined socket binding groups:

- **standard-sockets**
- **ha-sockets**
- **full-sockets**
- **full-ha-sockets**

Each socket binding group specifies socket bindings for the technologies used by its corresponding profile. For example, the **full-ha-sockets** socket binding group defines several **jgroups** socket bindings, which are used by the *full-ha* profile for high availability.

```

<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-tcp-fd" interface="private"
port="57600"/>
    <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="jgroups-udp-fd" interface="private"
port="54200"/>
    <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-groups>

```

**NOTE**

The socket configuration for the management interfaces is defined in the domain controller's `host.xml` file.

2.4.2.3. Configuring Socket Bindings

When defining a socket binding, you can configure the `port` and `interface` attributes, as well as multicast settings such as `multicast-address` and `multicast-port`. For details on all available socket bindings attributes, see the [Socket Binding Attributes](#) section.

Socket bindings can be configured using the management console or the management CLI. The following steps go through adding a socket binding group, adding a socket binding, and configuring socket binding settings using the management CLI.

1. Add a new socket binding group. Note that this step cannot be performed when running as a standalone server.

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

2. Add a socket binding.

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

3. Change the socket binding to use an interface other than the default, which is set by the socket binding group.

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

The following example shows how the XML configuration may look after the above steps have been completed.

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

2.4.2.4. Port Offsets

A port offset is a numeric offset value added to all port values specified in the socket binding group for that server. This allows the server to inherit the port values defined in its socket binding group, with an offset to ensure that it does not conflict with any other servers on the same host. For instance, if the HTTP port of the socket binding group is **8080**, and a server uses a port offset of **100**, then its HTTP port is **8180**.

Below is an example of setting a port offset of **250** for a server in a managed domain using the management CLI.

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

Port offsets can be used for servers in a managed domain and for running multiple standalone servers on the same host.

You can pass in a port offset when starting a standalone server using the **jboss.socket.binding.port-offset** property.

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

2.4.3. IPv6 Addresses

By default, JBoss EAP is configured to run using IPv4 addresses. The steps below show how to configure JBoss EAP to run using IPv6 addresses.

Configure the JVM Stack for IPv6 Addresses

Update the startup configuration to prefer IPv6 addresses.

1. Open the startup configuration file.
 - When running as a standalone server, edit the **EAP_HOME/bin/standalone.conf** file (or **standalone.conf.bat** for Windows Server).
 - When running in a managed domain, edit the **EAP_HOME/bin/domain.conf** file (or **domain.conf.bat** for Windows Server).
2. Set the **java.net.preferIPv4Stack** property to **false**.

```
-Djava.net.preferIPv4Stack=false
```

3. Append the **java.net.preferIPv6Addresses** property and set it to **true**.

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

The following example shows how the JVM options in the startup configuration file may look after making the above changes.

```
# 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
```

Update Interface Declarations for IPv6 Addresses

The default interface values in the configuration can be changed to IPv6 addresses. For example, the below management CLI command sets the **management** interface to the IPv6 loopback address (**::1**).

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

The following example shows how the XML configuration may look after running the above command.

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

CHAPTER 3. DEVELOPING APPLICATIONS USING JBOSS EAP

3.1. OVERVIEW

This guide provides information on getting started developing applications by using Red Hat JBoss Developer Studio and the JBoss EAP 7 quickstart examples.

Red Hat JBoss Developer Studio is an Eclipse-based integrated development environment (IDE) that integrates JBoss application development plug-ins. JBoss Developer Studio can assist with your application development with the availability of JBoss-specific wizards and the ability to deploy applications to JBoss EAP servers. Many quickstart code examples are provided with JBoss EAP 7 to help users get started writing applications using different Java EE 7 technologies.

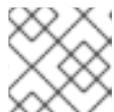
This guide provides basic installation instructions for JBoss Developer Studio. See the [Install Red Hat JBoss Developer Studio](#) guide for complete installation instructions.

3.2. SETTING UP THE DEVELOPMENT ENVIRONMENT

3.2.1. Download JBoss Developer Studio

JBoss Developer Studio can be downloaded from the Red Hat Customer Portal.

1. Log in to the [Red Hat Customer Portal](#)
2. Click **Downloads**.
3. In the **Product Downloads** list, click **Red Hat JBoss Developer Studio**.
4. Select the desired version in the **Version** drop-down menu.



NOTE

It is recommended to use JBoss Developer Studio version 9.1 or later.

5. Find the **Red Hat JBoss Developer Studio 9.x.x Stand-alone Installer** entry in the table and click **Download**.
6. Save the JAR file to the desired directory.

3.2.2. Install JBoss Developer Studio

1. Open a terminal and navigate to the directory containing the downloaded JAR file.
2. Run the following command to launch the GUI installation program:

```
$ java -jar jboss-devstudio-BUILD_VERSION-installer-standalone.jar
```

**NOTE**

Alternatively, you may be able to double-click the JAR file to launch the installation program.

3. Click **Next** to start the installation process.
4. Select **I accept the terms of this license agreement** and click **Next**.
5. Adjust the installation path and click **Next**.

**NOTE**

If the installation path folder does not exist, a prompt will appear. Click **OK** to create the folder.

6. Choose a JVM, or leave the default JVM selected, and click **Next**.
7. Click **Next** when asked to select platforms and servers.
8. Review the installation details, and click **Next**.
9. Click **Next** when the installation process is complete.
10. Configure the desktop shortcuts for JBoss Developer Studio, and click **Next**.
11. Click **Done**.

3.2.3. Start JBoss Developer Studio

To start JBoss Developer Studio, you can double-click on the desktop shortcut created during the installation, or you can start it from a command line. Follow the below steps to start JBoss Developer Studio using the command line.

1. Open a terminal and navigate to the JBoss Developer Studio installation directory.
2. Run the following command to start JBoss Developer Studio:

```
└─$ ./jbdevstudio
```

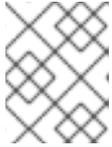
**NOTE**

For Windows Server, use the **jbdevstudio.bat** file.

3.2.4. Add the JBoss EAP Server to JBoss Developer Studio

These instructions assume that you have not yet added any JBoss EAP servers to JBoss Developer Studio. Use the following steps to add your JBoss EAP server using the **Define New Server** wizard.

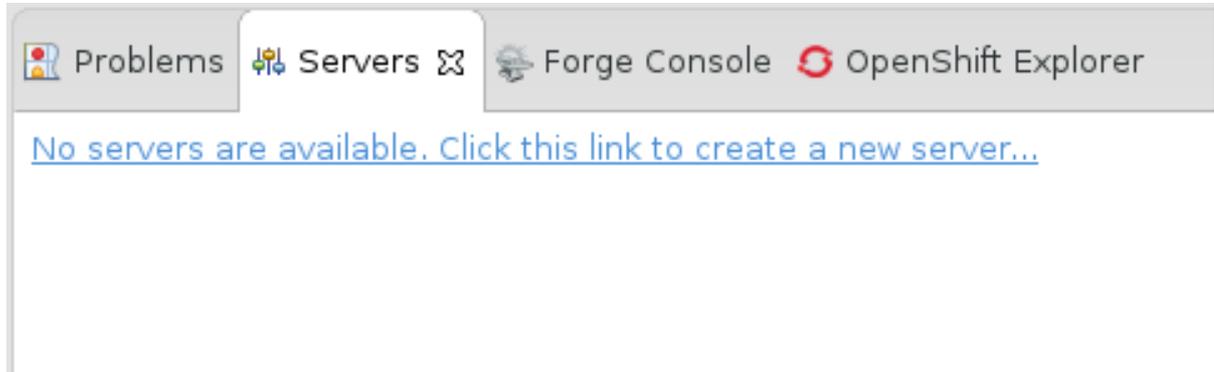
1. Open the **Servers** tab.

**NOTE**

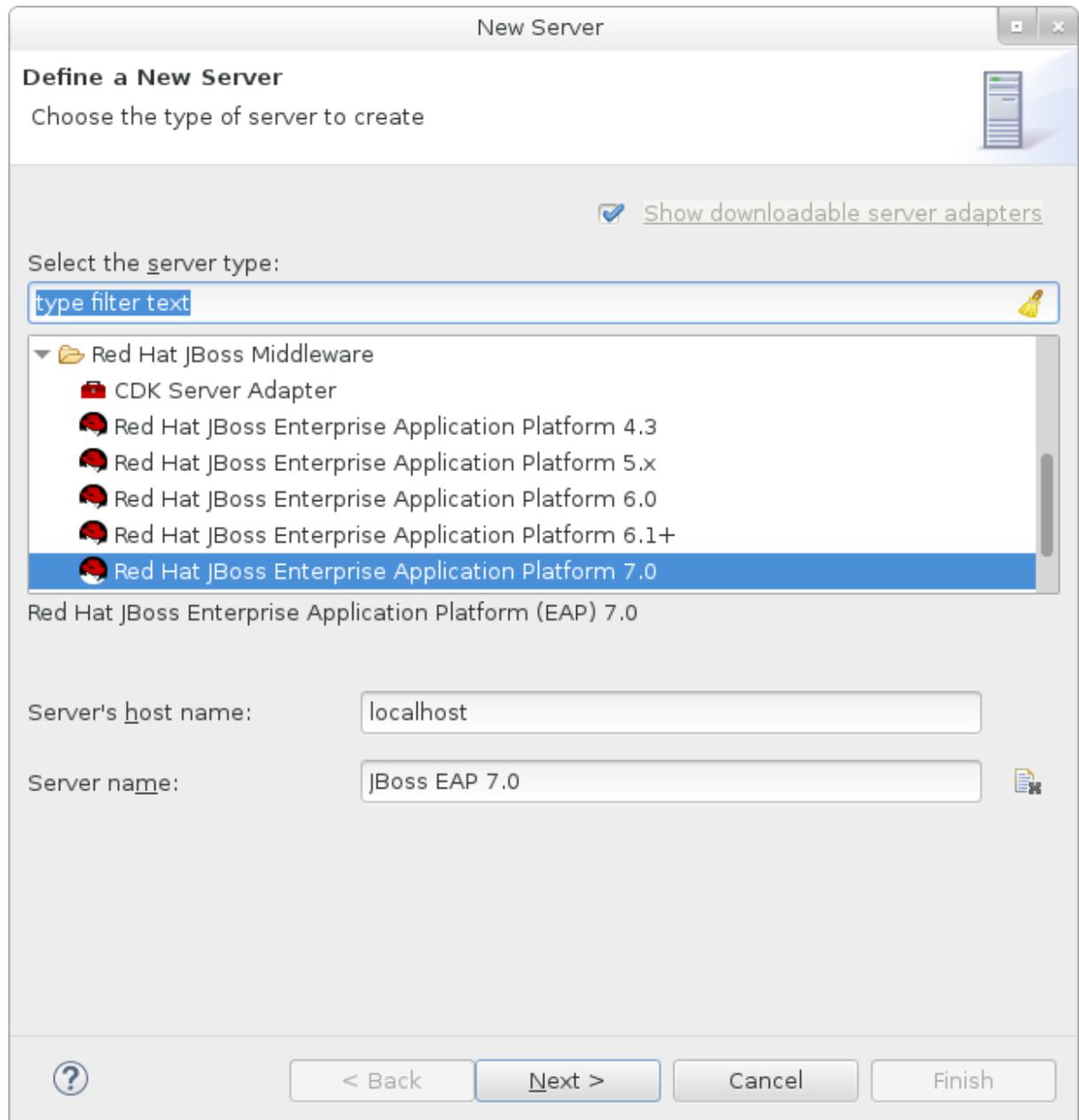
If the **Servers** tab is not shown, add it to the panel by selecting **Window** → **Show View** → **Servers**.

2. Click on the **No servers are available. Click this link to create a new server** link.

Figure 3.1. Add a New Server



3. Expand **Red Hat JBoss Middleware** and choose **JBoss Enterprise Application Platform 7.0**. Enter a server name, for example, **JBoss EAP 7.0**, then click **Next**.

Figure 3.2. Define a New Server

4. Create a server adapter to manage starting and stopping the server. Keep the defaults and click **Next**.

Figure 3.3. Create a New Server Adapter

New Server

Create a new Server Adapter
Red Hat JBoss Enterprise Application Platform (EAP) 7.0

RED HAT JBOSS MIDDLEWARE

A Server Adapter manages starting and stopping instances of your server. It manages command line arguments and keeps track of which modules have been deployed.

The server is:

Local
 Remote

Controlled by:

Filesystem and shell operations
 Management Operations

Server lifecycle is externally managed.

The selected profile requires a runtime.

Assign a runtime to this server

Create new runtime (next page) ▾

Runtime Details

JRE:
Home Directory:
Base Directory:
Configuration File:

? < Back Next > Cancel Finish

5. Enter a name, for example **JBoss EAP 7.0 Runtime**. Click **Browse** next to **Home Directory** and navigate to your JBoss EAP installation directory. Then click **Next**.

Figure 3.4. Add New Server Runtime Environment

JBoss Runtime
Red Hat JBoss Enterprise Application Platform (EAP) 7.0

A JBoss Server runtime references a JBoss installation directory. It can be used to set up classpaths for projects which depend on this runtime, as well as by a "server" which will be able to start and stop instances of JBoss.

Name
JBoss EAP 7.0 Runtime

Home Directory [Download and install runtime...](#)
/home/username/tools/jboss-eap-7.0

Runtime JRE

Execution Environment: javaSE-1.8

Alternate JRE: java-1.8.0-openjdk-1.8.0.65

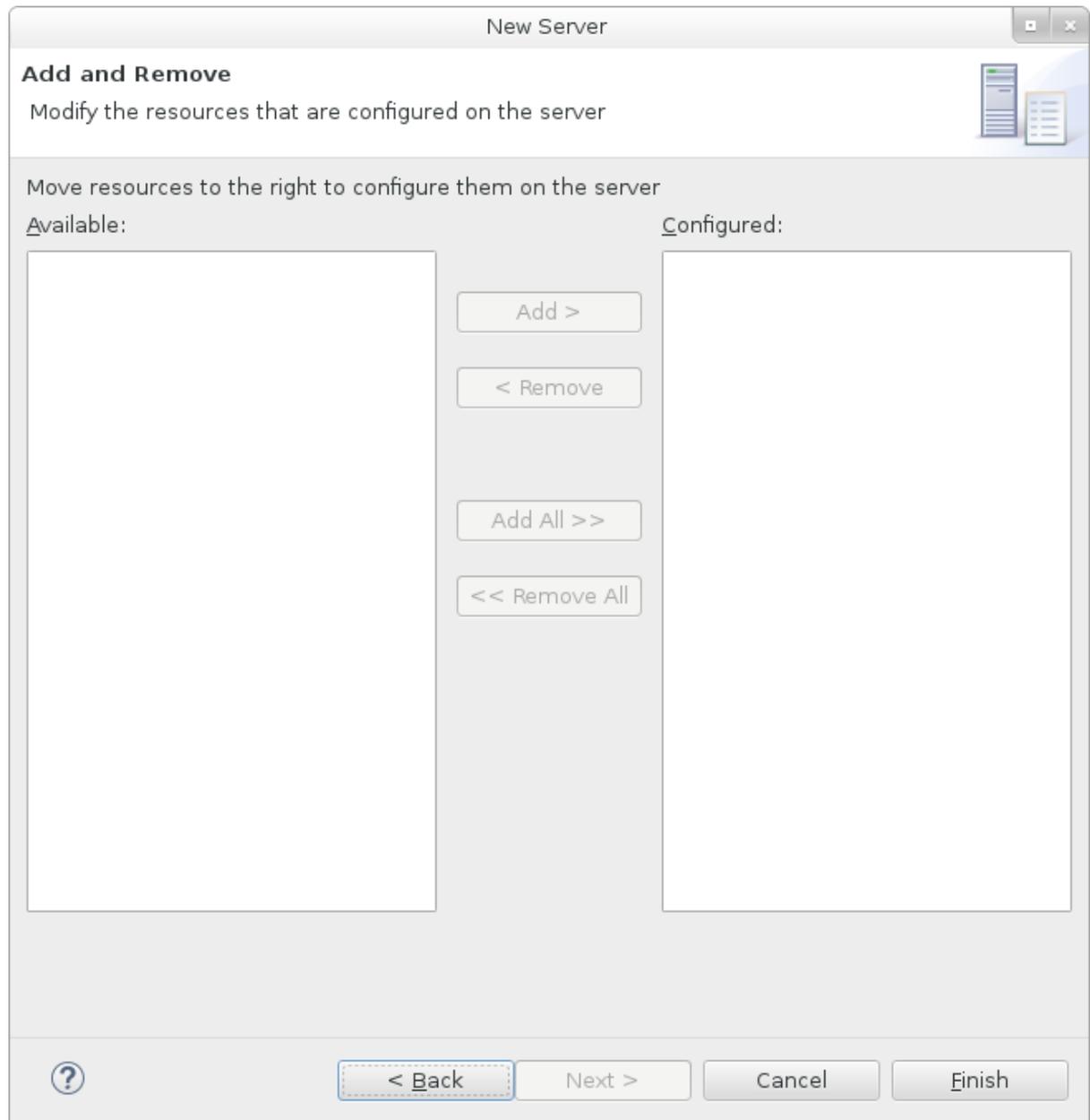
Server base directory: standalone

Configuration file: standalone.xml

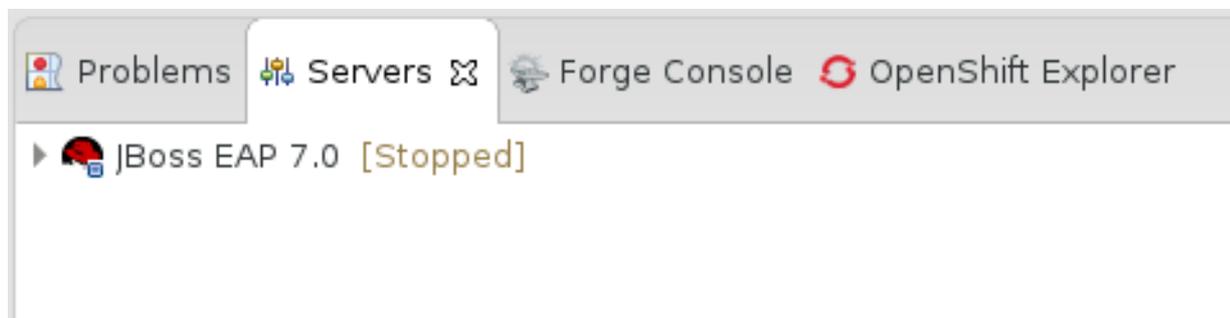
**NOTE**

Some quickstarts require that you run the server with a different profile or additional arguments. For example, to deploy a quickstart that requires the *full* profile, you must define a new server and specify **standalone-full.xml** in the **Configuration file** field. Be sure to give the new server a descriptive name.

- Configure existing projects for the new server. Because you do not have any projects at this point, click **Finish**.

Figure 3.5. Modify Resources for the New Server

The **JBoss EAP 7.0** server is now listed in the **Servers** tab.

Figure 3.6. Server List

3.3. USING THE QUICKSTART EXAMPLES

The quickstart examples provided with JBoss EAP are Maven projects.

3.3.1. About Maven

Apache Maven is a distributed build automation tool used in Java application development to create, manage, and build software projects. Maven uses standard configuration files called Project Object Model (POM) files to define projects and manage the build process. POMs describe the module and component dependencies, build order, and targets for the resulting project packaging and output using an XML file. This ensures that the project is built in a correct and uniform manner.

Maven achieves this by using a repository. A Maven repository stores Java libraries, plugins, and other build artifacts. The default public repository is the [Maven 2 Central Repository](#), but repositories can be private and internal within a company with a goal to share common artifacts among development teams. Repositories are also available from third-parties. For more information, see the [Apache Maven](#) project and the [Introduction to Repositories](#) guide.

JBoss EAP includes a Maven repository that contains many of the requirements that Java EE developers typically use to build applications on JBoss EAP.

For more information about how to use Maven with JBoss EAP, see [Using Maven with JBoss EAP](#) in the *JBoss EAP Development Guide*.

3.3.2. Using Maven with the Quickstarts

The artifacts and dependencies needed to build and deploy applications to JBoss EAP 7 are hosted on a public repository. Starting with the JBoss EAP 7 quickstarts, it is no longer necessary to configure your Maven `settings.xml` file to use these repositories when building the quickstarts. The Maven repositories are now configured in the quickstart project POM files. This method of configuration is provided to make it easier to get started with the quickstarts, however, is generally not recommended for production projects because it can slow down your build.

Red Hat JBoss Developer Studio includes Maven, so there is no need to download and install it separately. It is recommended to use JBoss Developer Studio version 9.1 or later.

If you plan to use the Maven command line to build and deploy your applications, then you must first download Maven from the [Apache Maven](#) project and install it using the instructions provided in the Maven documentation.

3.3.3. Download and Run the Quickstarts

3.3.3.1. Download the Quickstarts

JBoss EAP comes with a comprehensive set of quickstart code examples designed to help users begin writing applications using various Java EE 7 technologies. The quickstarts can be downloaded from the Red Hat Customer Portal.

1. Log in to the [Red Hat Customer Portal](#)
2. Click **Downloads**.
3. In the **Product Downloads** list, click **Red Hat JBoss Enterprise Application Platform**.
4. Select the desired version in the **Version** drop-down menu.

5. Find the **Red Hat JBoss Enterprise Application Platform 7.0.0 Quickstarts** entry in the table and click **Download**.
6. Save the ZIP file to the desired directory.
7. Extract the ZIP file.

3.3.3.2. Run the Quickstarts in JBoss Developer Studio

Once the quickstarts have been downloaded, they can be imported into JBoss Developer Studio and deployed to JBoss EAP.

Import a Quickstart into JBoss Developer Studio

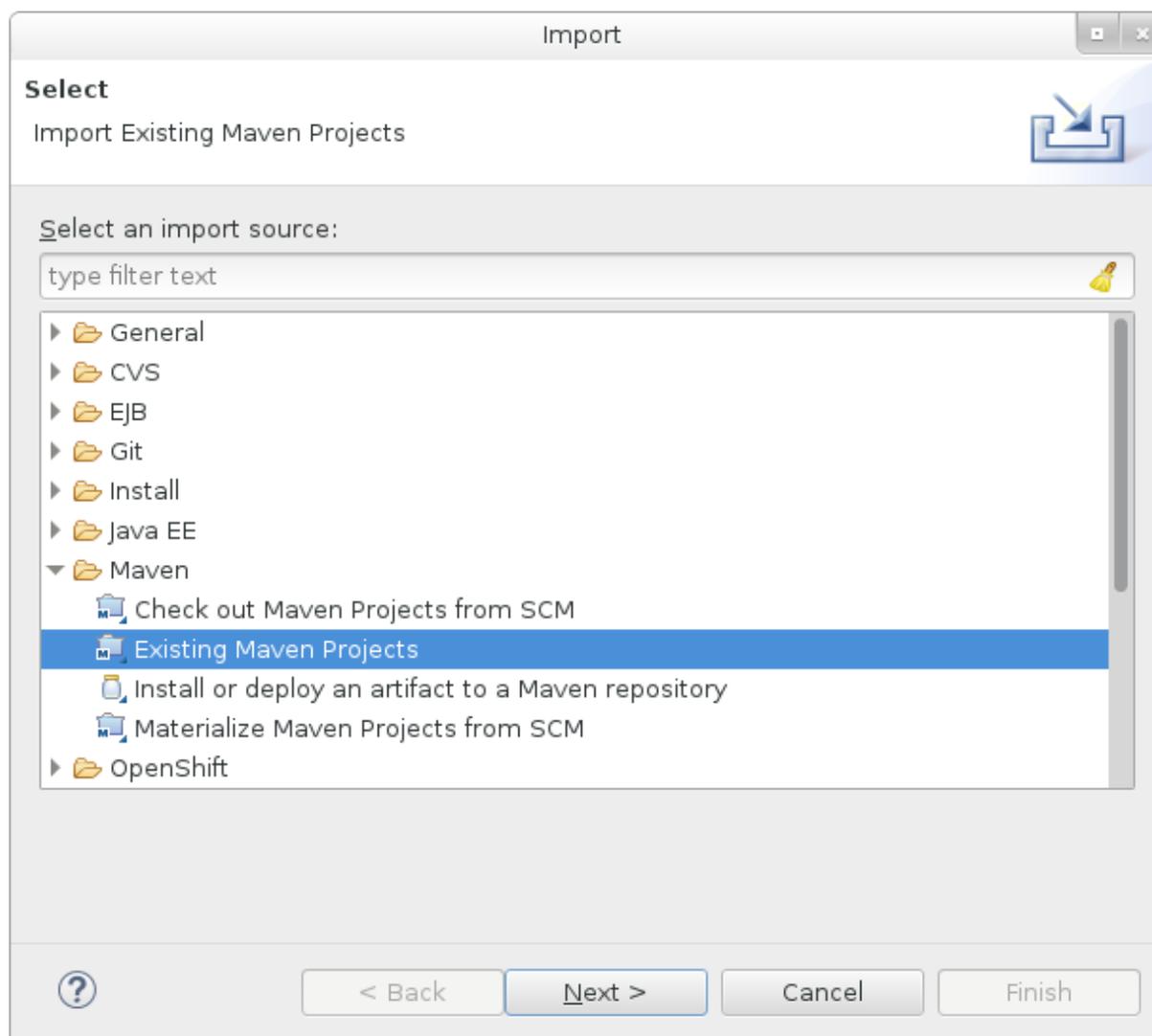
Each quickstart ships with a POM file that contains its project and configuration information. Use this POM file to easily import the quickstart into JBoss Developer Studio.



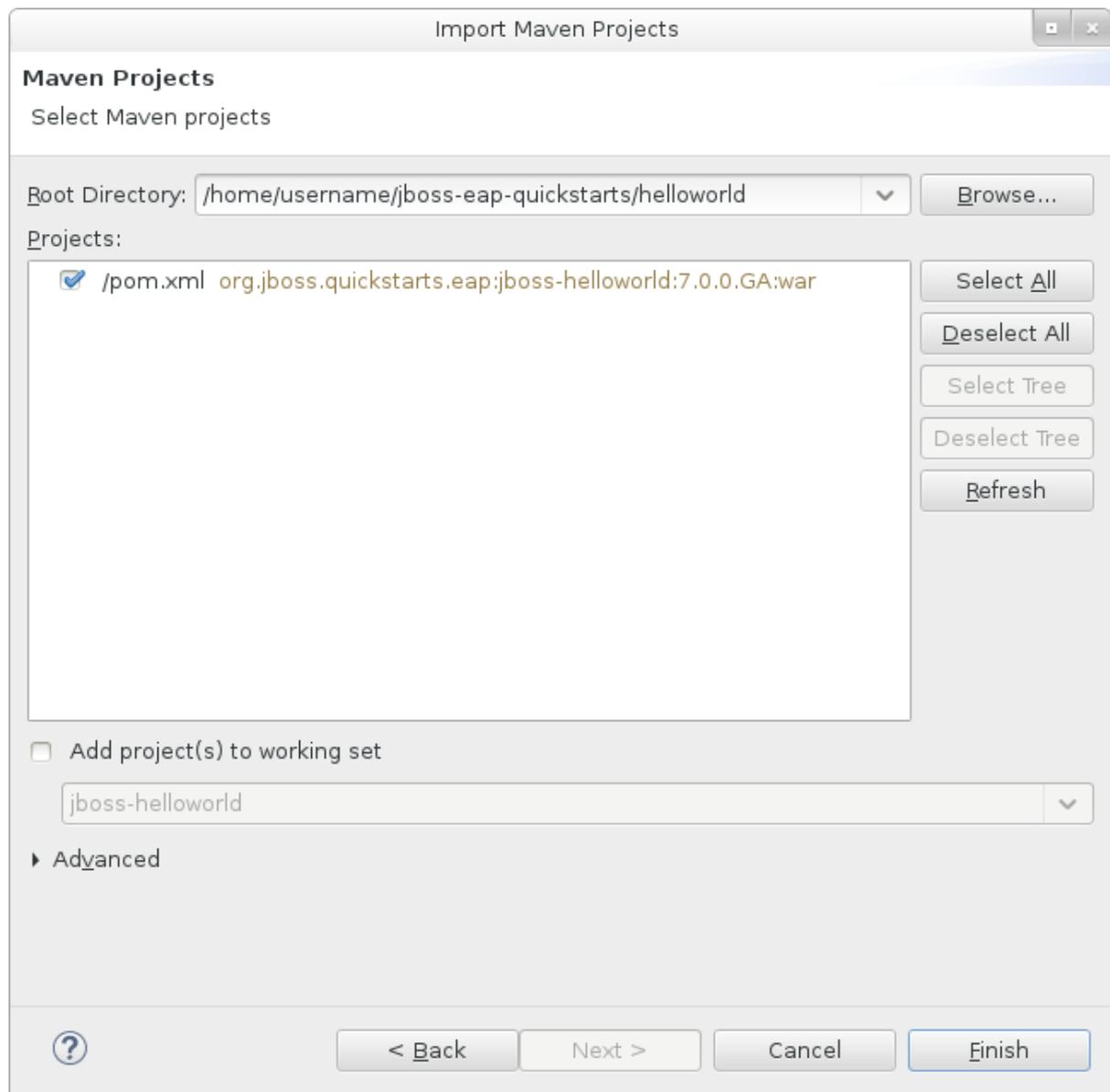
IMPORTANT

If your quickstart project folder is located within the IDE workspace when you import it into JBoss Developer Studio, the IDE generates an invalid project name and WAR archive name. Be sure your quickstart project folder is located outside the IDE workspace before you begin.

1. Start JBoss Developer Studio.
2. Select **File** → **Import**.
3. Choose **Maven** → **Existing Maven Projects**, then click **Next**.

Figure 3.7. Import Existing Maven Projects

4. Browse to the desired quickstart's directory (for example the **helloworld** quickstart), and click **OK**. The **Projects** list box is populated with the **pom.xml** file of the selected quickstart project.

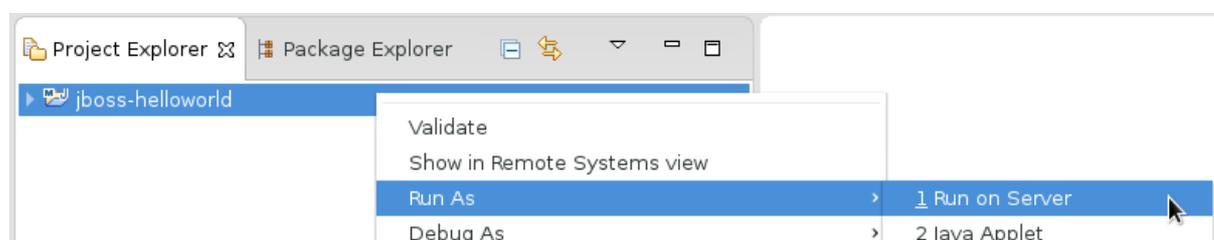
Figure 3.8. Select Maven Projects

5. Click **Finish**.

Run the *helloworld* Quickstart

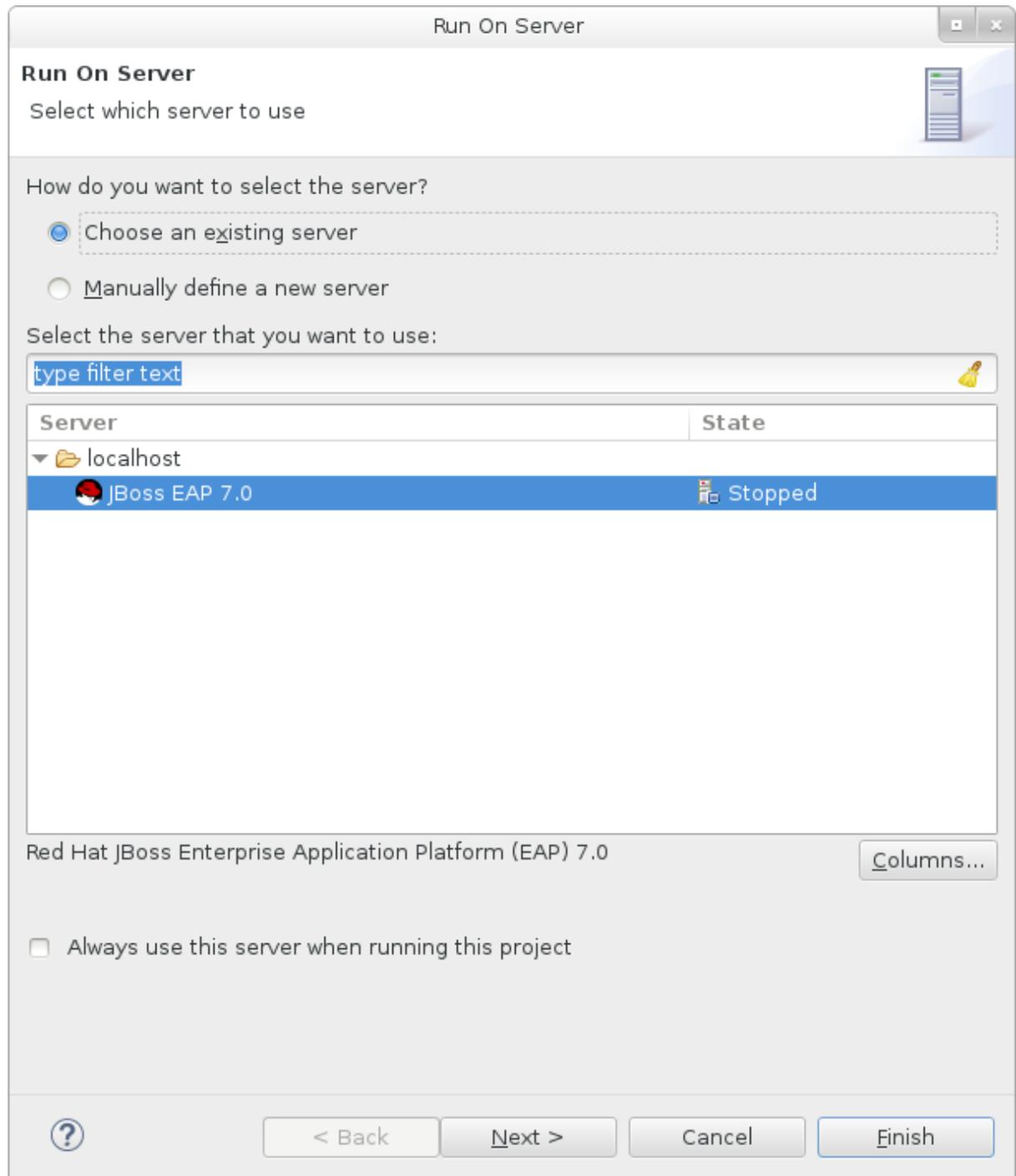
Running the **helloworld** quickstart is a simple way to verify that the JBoss EAP server is configured and running correctly.

1. If you have not yet defined a server, [add the JBoss EAP server to JBoss Developer Studio](#).
2. Right-click the **jboss-helloworld** project in the **Project Explorer** tab and select **Run As** → **Run on Server**.

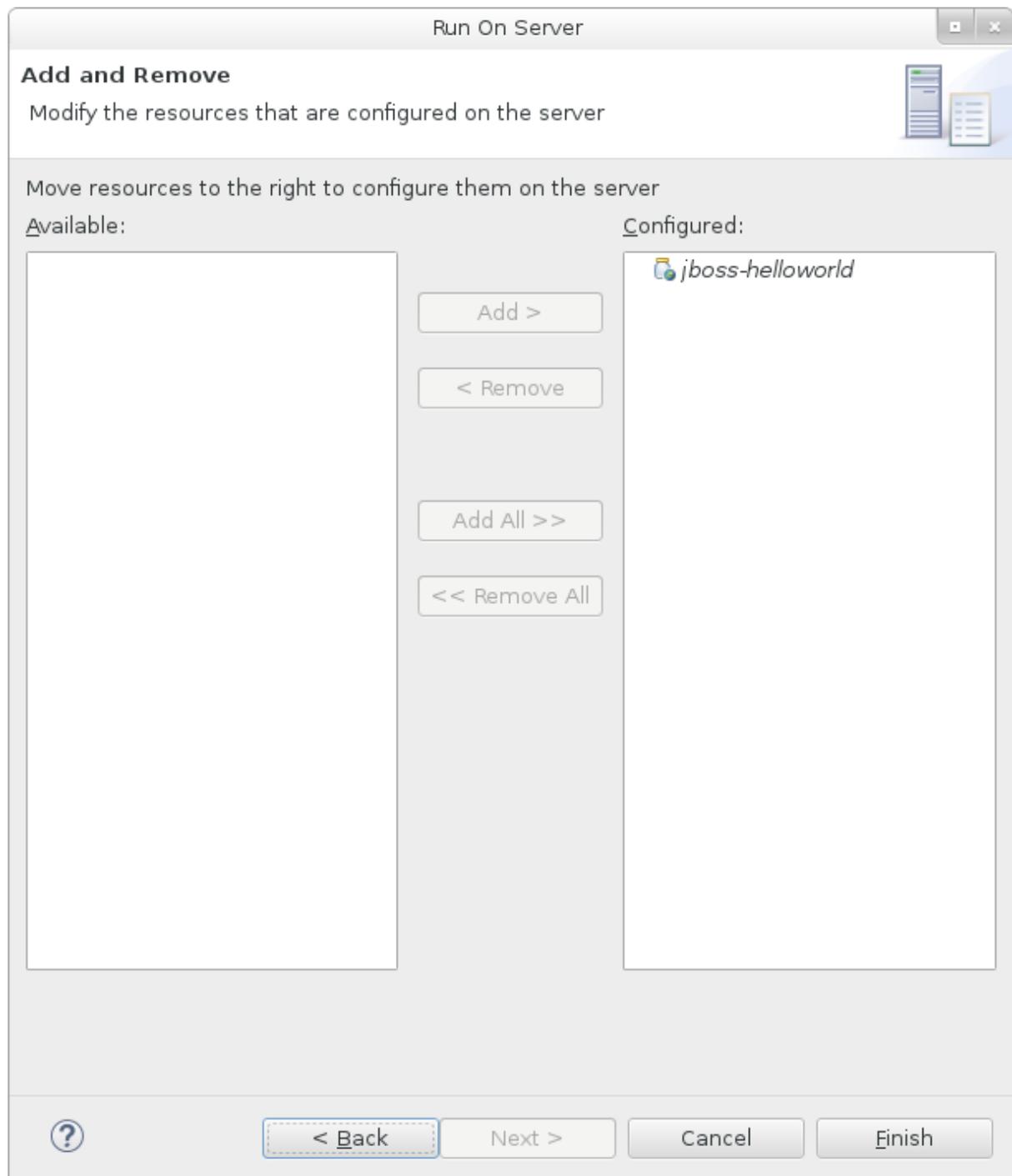
Figure 3.9. Run As - Run on Server

3. Select **JBoss EAP 7.0** from the server list and click **Next**.

Figure 3.10. Run on Server



4. The **jboss-helloworld** quickstart is already listed to be configured on the server. Click **Finish** to deploy the quickstart.

Figure 3.11. Modify Resources Configured on the Server

5. Verify the results.

- In the **Server** tab, the **JBoss EAP 7.0** server status changes to **Started**.
- The **Console** tab shows messages detailing the JBoss EAP server start and the **helloworld** quickstart deployment.

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

- The **helloworld** application is available at <http://localhost:8080/jboss-helloworld> and displays the text **Hello World!**.

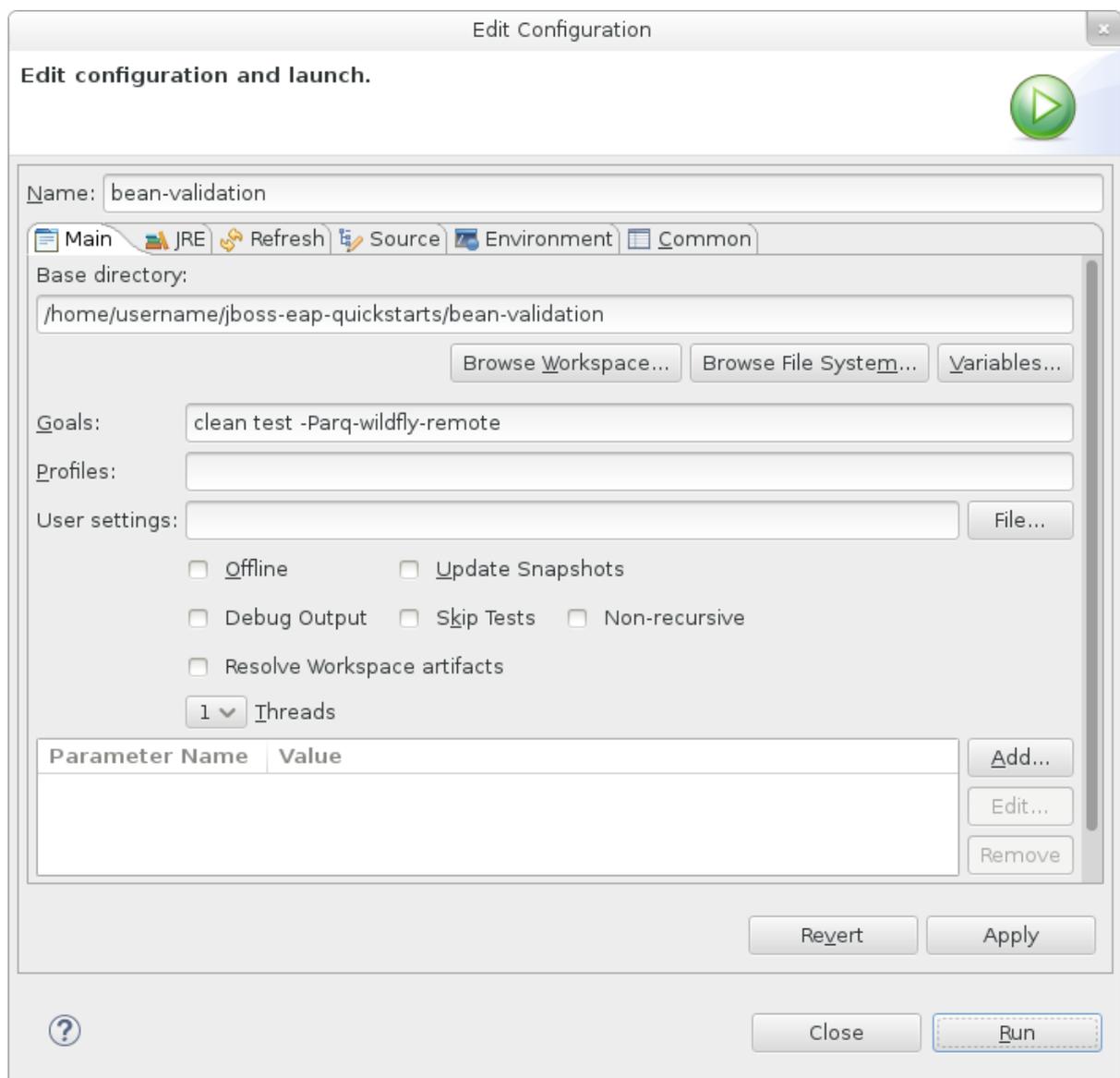
Run the *bean-validation* Quickstart

Some quickstarts, such as the **bean-validation** quickstart, do not provide a user interface layer and instead provide Arquillian tests to demonstrate functionality.

1. Import the **bean-validation** quickstart into JBoss Developer Studio.
2. In the **Servers** tab, right-click on the server and choose **Start** to start the JBoss EAP server. If you do not see a **Servers** tab or have not yet defined a server, [add the JBoss EAP server to Red Hat JBoss Developer Studio](#).
3. Right-click on the **jboss-bean-validation** project in the **Project Explorer** tab and select **Run As → Maven Build**.
4. Enter the following in the **Goals** input field and then click **Run**.

```
clean test -Parq-wildfly-remote
```

Figure 3.12. Edit Configuration



5. Verify the results.
The **Console** tab shows the results of the **bean-validation** Arquillian tests:

```

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

Results :

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

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

```

3.3.3.3. Run the Quickstarts from the Command Line

You can easily build and deploy the quickstarts from the command line using Maven. If you do not yet have Maven installed, see the [Apache Maven](#) project to download and install it.

A **README.md** file is provided at the root directory of the quickstarts that contains general information about system requirements, configuring Maven, adding users, and running the quickstarts.

Each quickstart also contains its own **README.md** file that provides the specific instructions and Maven commands to run that quickstart.

Run the *helloworld* Quickstart from the Command Line

1. Review the **README.md** file in the root directory of the *helloworld* quickstart.
2. Start the JBoss EAP server.

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

3. Navigate to the *helloworld* quickstart directory.
4. Build and deploy the quickstart using the Maven command provided in the quickstart's **README.md** file.

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

5. The *helloworld* application is now available at <http://localhost:8080/jboss-helloworld> and displays the text **Hello World!**.

APPENDIX A. REFERENCE MATERIAL

A.1. SERVER RUNTIME ARGUMENTS

The application server startup script accepts arguments and switches at runtime. This allows the server to start under alternative configurations to those defined in the `standalone.xml`, `domain.xml`, and `host.xml` configuration files.

Alternative configurations might include starting the server with an alternative socket bindings set or a secondary configuration.

The available parameters list can be accessed by passing the help switch `-h` or `--help` at startup.

Table A.1. Runtime Switches and Arguments

Argument or Switch	Operating Mode	Description
<code>--admin-only</code>	Standalone	Set the server's running type to ADMIN_ONLY . This will cause it to open administrative interfaces and accept management requests, but not start other runtime services or accept end user requests.
<code>--admin-only</code>	Domain	Set the host controller's running type to ADMIN_ONLY causing it to open administrative interfaces and accept management requests but not start servers or, if this host controller is the master for the domain, accept incoming connections from slave host controllers.
<code>-b=<value></code> , <code>-b <value></code>	Standalone, Domain	Set system property jboss.bind.address , which is used in configuring the bind address for the public interface. This defaults to 127.0.0.1 if no value is specified. See the -b<interface>=<value> entry for setting the bind address for other interfaces.
<code>-b<interface>=<value></code>	Standalone, Domain	Set system property jboss.bind.address.<interface> to the given value. For example, -bmanagement=IP_ADDRESS
<code>--backup</code>	Domain	Keep a copy of the persistent domain configuration even if this host is not the domain controller.
<code>-c=<config></code> , <code>-c <config></code>	Standalone	Name of the server configuration file to use. The default is standalone.xml .
<code>-c=<config></code> , <code>-c <config></code>	Domain	Name of the server configuration file to use. The default is domain.xml .

Argument or Switch	Operating Mode	Description
--cached-dc	Domain	If the host is not the domain controller and cannot contact the domain controller at boot, boot using a locally cached copy of the domain configuration.
--debug [<port>]	Standalone	Activate debug mode with an optional argument to specify the port. Only works if the launch script supports it.
-D<name>[=<value>]	Standalone, Domain	Set a system property.
--domain-config=<config>	Domain	Name of the server configuration file to use. The default is domain.xml .
-h, --help	Standalone, Domain	Display the help message and exit.
--host-config=<config>	Domain	Name of the host configuration file to use. The default is host.xml .
--interprocess-hc-address=<address>	Domain	Address on which the host controller should listen for communication from the process controller.
--interprocess-hc-port=<port>	Domain	Port on which the host controller should listen for communication from the process controller.
--master-address=<address>	Domain	Set system property jboss.domain.master.address to the given value. In a default slave host controller config, this is used to configure the address of the master host controller.
--master-port=<port>	Domain	Set system property jboss.domain.master.port to the given value. In a default slave host controller config, this is used to configure the port used for native management communication by the master host controller.
--read-only-server-config=<config>	Standalone	Name of the server configuration file to use. This differs from --server-config and -c in that the original file is never overwritten.

Argument or Switch	Operating Mode	Description
<code>--read-only-domain-config= <config></code>	Domain	Name of the domain configuration file to use. This differs from <code>--domain-config</code> and <code>-c</code> in that the initial file is never overwritten.
<code>--read-only-host-config= <config></code>	Domain	Name of the host configuration file to use. This differs from <code>--host-config</code> in that the initial file is never overwritten.
<code>-P=<url>, -P <url>, -- properties=<url></code>	Standalone, Domain	Load system properties from the given URL.
<code>--pc-address=<address></code>	Domain	Address on which the process controller listens for communication from processes it controls.
<code>--pc-port=<port></code>	Domain	Port on which the process controller listens for communication from processes it controls.
<code>-S<name>[=<value>]</code>	Standalone	Set a security property.
<code>-secmgr</code>	Standalone, Domain	Runs the server with a security manager installed.
<code>--server-config=<config></code>	Standalone	Name of the server configuration file to use. The default is <code>standalone.xml</code> .
<code>-u=<value>, -u <value></code>	Standalone, Domain	Set system property <code>jboss.default.multicast.address</code> , which is used in configuring the multicast address in the socket-binding elements in the configuration files. This defaults to <code>230.0.0.4</code> if no value is specified.
<code>-v, -V, --version</code>	Standalone, Domain	Display the application server version and exit.



WARNING

The configuration files that ship with JBoss EAP are set up to handle the behavior of the switches, for example, `-b` and `-u`. If you change your configuration files to no longer use the system property controlled by the switch, then adding it to the launch command will have no effect.

A.2. ADD-USER UTILITY ARGUMENTS

The following table describes the arguments available for the **add-user.sh** or **add-user.bat** script, which is a utility for adding new users to the properties file for out-of-the-box authentication.

Table A.2. Add-User Command Arguments

Command Line Argument	Description
-a	Create a user in the application realm. If omitted, the default is to create a user in the management realm.
-dc <value>	The domain configuration directory that will contain the properties files. If it is omitted, the default directory is EAP_HOME/domain/configuration/ .
-sc <value>	An alternative standalone server configuration directory that will contain the properties files. If omitted, the default directory is EAP_HOME/standalone/configuration/ .
-up, --user-properties <value>	The name of the alternative user properties file. It can be an absolute path or it can be a file name used in conjunction with the -sc or -dc argument that specifies the alternative configuration directory.
-g, --group <value>	A comma-separated list of groups to assign to this user.
-gp, --group-properties <value>	The name of the alternative group properties file. It can be an absolute path or it can be a file name used in conjunction with the -sc or -dc argument that specifies the alternative configuration directory.
-p, --password <value>	The password of the user.
-u, --user <value>	The name of the user. Only alphanumeric characters and the following symbols are valid: ,./=@\ .
-r, --realm <value>	The name of the realm used to secure the management interfaces. If omitted, the default is ManagementRealm .
-s, --silent	Run the add-user script with no output to the console.
-e, --enable	Enable the user.
-d, --disable	Disable the user.
-cw, --confirm-warning	Automatically confirm warning in interactive mode.
-h, --help	Display usage information for the add-user script.

A.3. INTERFACE ATTRIBUTES

Table A.3. Interface Attributes and Values

Interface Element	Description
any	Element indicating that part of the selection criteria for an interface should be that it meets at least one, but not necessarily all, of the nested set of criteria.
any-address	Empty element indicating that sockets using this interface should be bound to a wildcard address. The IPv6 wildcard address (::) will be used unless the java.net.preferIPv4Stack system property is set to true, in which case the IPv4 wildcard address (0.0.0.0) will be used. If a socket is bound to an IPv6 anylocal address on a dual-stack machine, it can accept both IPv6 and IPv4 traffic; if it is bound to an IPv4 (IPv4-mapped) anylocal address, it can only accept IPv4 traffic.
inet-address	Either an IP address in IPv6 or IPv4 dotted decimal notation, or a host name that can be resolved to an IP address.
link-local-address	Empty element indicating that part of the selection criteria for an interface should be whether or not an address associated with it is link-local.
loopback	Empty element indicating that part of the selection criteria for an interface should be whether or not it is a loopback interface.
loopback-address	A loopback address that may not actually be configured on the machine's loopback interface. Differs from inet-address type in that the given value will be used even if no NIC can be found that has the IP address associated with it.
multicast	Empty element indicating that part of the selection criteria for an interface should be whether or not it supports multicast.
nic	The name of a network interface (e.g. eth0, eth1, lo).
nic-match	A regular expression against which the names of the network interfaces available on the machine can be matched to find an acceptable interface.
not	Element indicating that part of the selection criteria for an interface should be that it does not meet any of the nested set of criteria.
point-to-point	Empty element indicating that part of the selection criteria for an interface should be whether or not it is a point-to-point interface.

Interface Element	Description
public-address	Empty element indicating that part of the selection criteria for an interface should be whether or not it has a publicly routable address.
site-local-address	Empty element indicating that part of the selection criteria for an interface should be whether or not an address associated with it is site-local.
subnet-match	A network IP address and the number of bits in the address' network prefix, written in <i>slash notation</i> (e.g. 192.168.0.0/16).
up	Empty element indicating that part of the selection criteria for an interface should be whether or not it is currently up.
virtual	Empty element indicating that part of the selection criteria for an interface should be whether or not it is a virtual interface.

A.4. SOCKET BINDING ATTRIBUTES

Table A.4. Socket Binding Attributes

Attribute	Description
client-mappings	Specifies the client mappings for this socket binding. A client connecting to this socket should use the destination address specified in the mapping that matches its desired outbound interface. This allows for advanced network topologies that use either network address translation, or have bindings on multiple network interfaces to function. Each mapping should be evaluated in declared order, with the first successful match used to determine the destination.
fixed-port	Whether the port value should remain fixed even if numeric offsets are applied to the other sockets in the socket group.
interface	Name of the interface to which the socket should be bound, or, for multicast sockets, the interface on which it should listen. This should be one of the declared interfaces. If not defined, the value of the default-interface attribute from the enclosing socket binding group will be used.
multicast-address	Multicast address on which the socket should receive multicast traffic. If unspecified, the socket will not be configured to receive multicast.
multicast-port	Port on which the socket should receive multicast traffic. Must be configured if 'multicast-address' is configured.

Attribute	Description
name	The name of the socket. Services needing to access the socket configuration information will find it using this name. This attribute is required.
port	Number of the port to which the socket should be bound. Note that this value can be overridden if servers apply a port-offset to increment or decrement all port values.

A.5. DEFAULT SOCKET BINDINGS

Table A.5. Default Socket Bindings

Name	Port	Multicast Port	Description	Socket Binding Groups
ajp	8009		Apache JServ Protocol. Used for HTTP clustering and load balancing.	standard-sockets, ha-sockets, full-sockets, full-ha-sockets
http	8080		The default port for deployed web applications.	standard-sockets, ha-sockets, full-sockets, full-ha-sockets
https	8443		SSL-encrypted connection between deployed web applications and clients.	standard-sockets, ha-sockets, full-sockets, full-ha-sockets
iiop	3528		CORBA services for JTS transactions and other ORB-dependent services.	full-sockets, full-ha-sockets
iiop-ssl	3529		SSL-encrypted CORBA services.	full-sockets, full-ha-sockets
jgroups-mping		45700	Multicast. Used to discover initial membership in a HA cluster.	ha-sockets, full-ha-sockets
jgroups-tcp	7600		Unicast peer discovery in HA clusters using TCP.	ha-sockets, full-ha-sockets
jgroups-tcp-fd	57600		Used for HA failure detection over TCP.	ha-sockets, full-ha-sockets

Name	Port	Multicast Port	Description	Socket Binding Groups
jgroups-udp	55200	45688	Multicast peer discovery in HA clusters using UDP.	ha-sockets, full-ha-sockets
jgroups-udp-fd	54200		Used for HA failure detection over UDP.	ha-sockets, full-ha-sockets
management-http	9990		Used for HTTP communication with the management layer.	standard-sockets, ha-sockets, full-sockets, full-ha-sockets
management-https	9993		Used for HTTPS communication with the management layer.	standard-sockets, ha-sockets, full-sockets, full-ha-sockets
modcluster		23364	Multicast port for communication between JBoss EAP and the HTTP load balancer.	ha-sockets, full-ha-sockets
txn-recovery-environment	4712		The JTA transaction recovery manager.	standard-sockets, ha-sockets, full-sockets, full-ha-sockets
txn-status-manager	4713		The JTA / JTS transaction manager.	standard-sockets, ha-sockets, full-sockets, full-ha-sockets

Revised on 2018-02-08 10:16:39 EST