Red Hat JBoss Enterprise Application Platform 8.0

Red Hat JBoss Enterprise Application Platform 入门

快速启动并运行 Red Hat JBoss Enterprise Application Platform。学习管理任务，如基本安装、管理和配置。使用 JBoss EAP 快速入门开始编写 Jakarta EE 应用程序

Last Updated: 2024-03-28

Red Hat JBoss Enterprise Application Platform 8.0 Red Hat JBoss Enterprise Application Platform 入门

快速启动并运行 Red Hat JBoss Enterprise Application Platform。学习管理任务，如基本安装、管理和配置。使用 JBoss EAP 快速入门开始编写 Jakarta EE 应用程序

法律通告

Copyright © 2024 Red Hat, Inc.

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

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

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

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

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

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

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

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

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

All other trademarks are the property of their respective owners.

摘要

快速启动并运行 Red Hat JBoss Enterprise Application Platform。学习管理任务，如基本安装、管理和配置。使用 JBoss EAP 快速入门开始编写 Jakarta EE 应用。

目录

提供有关 JBOSS EAP 文档的反馈

要报告错误或改进文档，请登录到 Red Hat JIRA 帐户并提交问题。如果您没有 Red Hat Jira 帐户，则会提示您创建一个帐户。

流程

1. 单击以下链接 以创建 **ticket**。
2. 在 **Summary** 中输入问题的简短描述。
3. 在 **Description** 中提供问题或功能增强的详细描述。包括一个指向文档中问题的 URL。
4. 点 **Submit** 创建问题，并将问题路由到适当的文档团队。

使开源包含更多

红帽致力于替换我们的代码、文档和 Web 属性中存在问题的语言。我们从这四个术语开始：master、slave、黑名单和白名单。由于这一努力的精力，这些更改将在即将发布的版本中逐渐实施。有关让我们的语言更加包含的更多详情，请参阅我们的CTO Chris Wright 信息。

第 1 章 管理 JBOSS EAP

1.1. 下载并安装 JBOSS EAP

.zip 文件选项是一个下载及安装 JBoss EAP 的快速的、独立于平台的方法。

1.1.1. 下载 JBoss EAP

在安装 JBoss EAP 前，您必须先下载 JBoss EAP **.zip** 文件。

先决条件

- 确认您的系统满足 JBoss EAP 支持的配置。
- 安装最新的更新和勘误表补丁。
- 为安装目录设置读写访问权限。
- 安装所需的 Java 开发套件(JDK)。
- 可选：设置 **JAVA_HOME** 和 **PATH** 环境变量。

流程

1. 登录红帽客户门户。
2. 点 **Downloads**。
3. **Product Downloads** 列表中，单击 **Red Hat JBoss Enterprise Application Platform**。
4. 在 **Version** 下拉菜单中，选择 **8.0**。
5. 在列表中找到 **Red Hat JBoss Enterprise Application Platform 8.0**，然后点 **Download** 链接。 '.zip'文件下载到您的系统上。

其他资源

- 红帽客户门户网站.

1.1.2. 安装 JBoss EAP

您可以通过将软件包内容解压缩到所需的文件位置来安装 JBoss EAP **.zip** 文件。

先决条件

- 下载 JBoss EAP。
- 确认您的系统满足 JBoss EAP 支持的配置。
- 安装最新的更新和勘误表补丁。
- 为安装目录设置读写访问权限。
- 安装所需的 Java 开发套件(JDK)。

- 可选：设置 **JAVA_HOME** 和 **PATH** 环境变量。

流程

1. 将 **.zip** 文件移动到要安装 JBoss EAP 的服务器和位置。
2. 解压缩 **.zip** 文件。
   a. 在 Linux 上，使用以下命令：

   ```
   $ unzip jboss-eap-8.0.0.zip
   ```

   b. 在 Windows 服务器上，右键单击 **.zip** 文件，并选择 **Extract All**。
   解压缩 **.zip** 文件所创建的目录是 JBoss EAP 安装的顶级目录。此目录称为 ***EAP_HOME***。

1.2. 启动和停止 JBOSS EAP

启动 JBoss EAP 的方法取决于您是将 JBoss EAP 作为独立服务器还是在受管域中运行。

停止 JBoss EAP 的方法取决于您在运行 JBoss EAP 的交互式实例还是后台实例。

1.2.1. 将 JBoss EAP 作为独立服务器启动

您可以将 JBoss EAP 作为独立服务器运行，来管理单个 JBoss EAP 实例。

服务器以暂停状态启动，且不会在所有需要的服务启动之前接受请求。在启动所需的服务后，服务器过渡到正常的运行状态，并可开始接受请求。

此启动脚本使用 ***EAP_HOME*/bin/standalone.conf** 文件，或使用用于 Windows 服务器的 **standalone.conf.bat** 来设置默认首选项，如 JVM 选项。您可以自定义此文件中的设置。

注意

要查看终端中的启动脚本参数列表，请使用 **--help** 参数。

JBoss EAP 默认使用 **standalone.xml** 配置文件，但您可以使用其他配置文件来启动 JBoss EAP。

先决条件

- 安装 JBoss EAP。

流程

1. 打开终端。
2. 使用以下脚本将 JBoss EAP 启动为独立服务器：

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

   a. 对于 Windows Server，请使用 ***EAP_HOME*\bin\standalone.bat** 脚本。

其他资源

- 单机服务器配置文件.
- 服务器运行时参数和交换机。

1.2.2. 为受管域中的服务器启动 JBoss EAP

您可以在受管域操作模式下运行 JBoss EAP，来使用单个域控制器管理多个 JBoss EAP 实例。

服务器以暂停状态启动，不接受请求，直到所有需要的服务都启动为止。在所需的服务都启动后，服务器将过渡到正常的运行状态，并开始接受请求。

您必须在域中的任何服务器组中的服务器启动之前启动域控制器。

先决条件

- 安装 JBoss EAP。

流程

1. 打开终端。
2. 首先启动域控制器，然后使用以下脚本启动每个关联的主机控制器：

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

   - 对于 Windows Server，请使用 ***EAP_HOME*\bin\domain.bat** 脚本。

此启动脚本使用 ***EAP_HOME*/bin/domain.conf** 文件，或使用用于 Windows 服务器的 **domain.conf.bat** 来设置默认首选项，如 JVM 选项。您可以自定义此文件中的设置。

JBoss EAP 默认使用 **host.xml** 主机配置文件，但您可以使用其他配置文件来启动 JBoss EAP。

设置受管域时，必须将额外的参数传给启动脚本。

注意

如需所有可用启动脚本参数及其目的的完整列表，请使用 **--help** 参数。

其他资源

- 受管域配置文件.
- 服务器运行时参数和交换机。

1.2.3. 停止 JBoss EAP 的交互式实例

您可以从启动它的终端停止独立服务器或域控制器的交互式实例。

先决条件

- 拥有 JBoss EAP 的运行实例。

流程

- 在您启动 JBoss EAP 的终端中按 **Ctrl+C**。

1.2.4. 停止 JBoss EAP 的后台实例

您可以连接到管理 CLI，来关闭受管域中正在运行的独立服务器的实例。

先决条件

- 有运行在终端中的 JBoss EAP 实例。

流程

1. 使用以下脚本启动管理 CLI：

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

2. 发出 **shutdown** 命令：

```
shutdown
```

当在受管域中的服务器上运行 JBoss EAP 实例时，您必须使用 **shutdown** 命令的 **--host** 参数来指定要关闭的主机名。

1.3. JBOSS EAP 管理

您可以使用命令行管理 CLI、基于 Web 的管理控制台、Java API 或 HTTP API 来配置 JBoss EAP。您使用这些管理接口所做的更改会自动永久保留，管理 API 会覆盖 XML 配置文件。管理 CLI 和管理控制台是首选的方法，不建议手动编辑 XML 配置文件。

JBoss EAP 使用简化的配置，并且每个独立服务器或受管域中的每台服务器都有一个配置文件。

- 独立服务器的默认配置存储在 ***EAP_HOME*/standalone/configuration/standalone.xml** 文件中。
- 受管域中服务器的默认配置存储在 ***EAP_HOME*/domain/configuration/domain.xml** 文件中。
- 主机控制器的默认配置存储在 ***EAP_HOME*/domain/configuration/host.xml** 文件中。

1.3.1. 管理用户

如果要远程访问管理 CLI 或者使用管理控制台，您必须添加管理用户，即使流量源自于本地主机，也会被视为远程访问。如果在添加管理用户之前尝试访问管理控制台，您会收到错误消息。

默认的 JBoss EAP 配置提供本地身份验证，使用户可以访问本地主机上的管理 CLI，而无需进行身份验证。

如果您使用图形安装程序安装 JBoss EAP，则图形安装程序在安装过程中创建一个管理用户。

1.3.2. 添加管理用户

您可以使用 **add-user** 脚本为 JBoss EAP 添加管理用户，该脚本是向属性文件添加新用户以进行立即身份验证的工具。

先决条件

- 您已安装了 JBoss EAP。

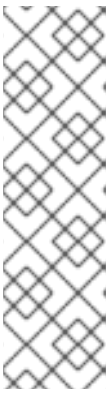
流程

1. 启动管理 CLI。
2. 运行 **add-user** 实用程序脚本并按照提示进行操作。

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

   - 对于 Windows Server，请使用 ***EAP_HOME*\bin\add-user.bat** 脚本。

3. 按 **ENTER** 选择默认选项 **a** 来添加管理用户。
   这会将用户添加到 *ManagementRealm*，并授权用户使用管理控制台或管理 CLI 来执行管理操作。另一个选择 **b** 将用户添加到 *ApplicationRealm* 中，该应用程序用于应用程序，不提供任何特定权限。
4. 输入用户名和密码。提示时必须确认密码。



注意

用户名只能以任何数字和顺序包含以下字符：

- 字母数字字符（a-z、A-Z、0-9）
- 短划线(-)、句点(.)、逗号(@)
- 反斜杠(\)
- 等号 (=)

   默认情况下，JBoss EAP 允许弱密码，但会有警告。

5. 输入以逗号分隔的用户所属组的列表。如果您不希望用户属于任何组，请按 **ENTER** 将其留空。
6. 检查信息并输入 **yes** 确认。
7. 确定此用户是否代表远程 JBoss EAP 服务器实例。对于基本管理用户，请输入 **no**。
   如果您要将用户添加到 *ManagementRealm*，其代表需要连接到域控制器的主机控制器，请对此提示回答 **yes**。您会获得一个编码的 secret 值，表示必须添加到主机控制器的 **host*.xml** 文件中的用户的密码。

您可以通过将参数传递给 **add-user** 脚本来以非交互方式创建用户。共享系统上不建议使用此方法，因为密码将在日志和历史记录文件中可见。

其他资源

- 以非交互方式运行 add-user 工具。

1.3.3. 以非交互方式运行 add-user 工具

您可以通过在命令行中传递参数，以非交互方式运行 **add-user** 脚本。必须至少提供用户名和密码。

警告

不建议在共享系统上使用这个方法，因为密码在日志和历史记录文件中可见。

创建属于多个组的用户
以下命令添加一个管理用户 **mgmtuser1**，属于 **guest** 和 **mgmtgroup** 组：

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

指定备选属性文件
默认情况下，使用 **add-user** 脚本创建的用户和组信息存储在服务器配置目录中的属性文件中。

用户信息存储在以下属性文件中：

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

组信息存储在以下属性文件中：

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

以下命令添加了一个新用户，为用户属性文件指定了不同的名称和位置：

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

新用户已添加到位于 ***/path/to /standaloneconfig/*newname.properties** 和 ***/path/to /domainconfig/*newname.properties** 的用户属性文件中。请注意，这些文件必须已经存在，否则您将看到错误。

注意

如需所有可用 **add-user** 参数及其目的的完整列表，请使用 **--help** 参数。

其他资源

- 添加用户参数.

1.3.4. 管理 CLI

管理命令行界面(CLI)是 JBoss EAP 的命令行工具。

使用管理 CLI 启动和停止服务器、部署和删除应用、配置系统设置，以及执行其他管理任务。您可以在批处理模式下执行操作，允许多个任务作为一个组来运行。

有许多常见的终端命令可用，如 **ls** (list)、**cd** （更改目录）和 **pwd** （打印工作目录）。管理 CLI 也支持 tab 自动完成功能。

启动管理 CLI

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

注意

对于 Windows Server，请使用 ***EAP_HOME*\bin\jboss-cli.bat** 脚本。

连接到正在运行的服务器

```
connect
```

您可以使用 ***EAP_HOME*/bin/jboss-cli.sh --connect** 命令来一步启动管理 CLI 并连接。

显示帮助
使用以下命令提供常规帮助：

```
help
```

在命令中使用 **--help** 标志，以接收有关使用该特定命令的说明。例如，要接收有关使用 **deploy** 的信息，请使用以下命令：

```
deploy --help
```

退出管理 CLI
使用以下命令退出管理 CLI：

```
quit
```

查看系统设置
以下命令使用 **read-attribute** 操作来显示是否启用了示例数据源：

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

在受管域中运行服务器时，您必须在命令前使用**/profile=*PROFILE_NAME*** 指定要更新的配置集。

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

查看子系统配置
以下命令使用 **read-resource-description** 操作来显示给定子系统配置的描述，其中包括是否需要资源，是否有属性替换，等等：

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

更新系统设置
以下命令使用 **write-attribute** 操作来禁用示例数据源：

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

-

启动服务器
在受管域中运行时，使用以下命令启动和停止服务器：

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

1.3.5. 管理控制台

管理控制台是用于 JBoss EAP 的基于 Web 的管理工具。

使用管理控制台启动和停止服务器、部署和删除应用程序、调优系统设置，并对服务器配置进行持久性修改。当用户进行需要重启或重新加载服务器的任何更改时，管理控制台还可以执行管理任务，并发出实时通知。

在受管域中，同一域中的服务器实例和服务器组通过域控制器的管理控制台集中管理。

对于运行在本地主机上使用默认管理端口的 JBoss EAP 实例，您可以通过 Web 浏览器访问管理控制台 http://localhost:9990/console/index.html。您必须使用具有访问管理控制台权限的用户角色进行身份验证。

管理控制台提供下列选项卡，用于浏览和管理 JBoss EAP 单机服务器或受管域。

Home
: 了解如何完成几个常见配置和管理任务。参加导览，熟悉 JBoss EAP 管理控制台。

Deployments
: 添加、移除和启用部署。在受管域中，将部署分配到服务器组。

配置
: 配置可用的子系统，提供 Web 服务、消息传递或高可用性等功能。在受管域中，管理包含不同子系统配置的配置文件。

Runtime
: 查看运行时信息，如服务器状态、JVM 使用量和服务器日志。在受管域中，管理您的主机、服务器组和服务器。

Patching
: 将补丁应用到您的 JBoss EAP 实例。

Access control
: 在使用基于角色的访问控制时，将角色分配给用户和组。

1.3.5.1. 更新管理控制台中的资源属性

如果您有所需的权限，您可以在管理控制台中编辑资源属性。

先决条件

- JBoss EAP 正在运行。
- 您有修改所选资源的适当权限。
- 您已创建了用户。

流程

1. 登录到管理控制台。对于在默认端口中运行的本地服务器，您可以访问管理控制台 http://localhost:9990/console/index.html。
2. 进入您要修改的资源的管理控制台的适当部分。
3. 点 **Edit**。
4. 进行必要的更改。
必填字段标有星号(*)。您可以通过单击 **Help** 来查看属性描述。

注意

根据属性类型，输入字段可以是文本字段、ON/OFF 字段或下拉菜单。在某些文本字段中，当您输入时，配置中其他位置的值可能会显示为建议。

5. 点击 **Save**。
6. 如有必要，重新加载服务器以使更改生效。
当您进行更改需要重新加载才能生效时，会打开弹出窗口。要重新加载单机服务器，请单击弹出窗口中的 **Reload**。要在受管域中重新加载服务器，请单击 **Topology**，选择适当的服务器，然后从下拉列表中选择 **Reload**。

要查看您执行的最新配置操作的历史记录，请点通知图标。

1.3.5.2. 启用或禁用管理控制台

您可以通过设置 **/core-service=management/management-interface=http-interface** 资源的 **console-enabled** 布尔值属性来启用或禁用管理控制台。对于域模式的 master 主机，请使用 **/host=master/core-service=management/management-interface=http-interface**。

注意

启用或禁用管理控制台后，您必须重新启动或重新加载 JBoss EAP 实例。

启用管理控制台示例

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

禁用管理控制台示例

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

1.3.5.3. 更改管理控制台的语言

默认情况下，管理控制台的语言设置是英语。您可以选择使用以下语言之一：

- 德语(de)
- 简体中文(zh-Hans)
- 巴西葡萄牙语(pt-BR)

- 法语(fr)
- 西班牙语(es)
- 日语(ja)

前提条件

- JBoss EAP 正在运行。
- 您已创建了用户。

流程

1. 登录到管理控制台。对于在默认端口中运行的本地服务器，您可以访问管理控制台 http://localhost:9990/console/index.html。
2. 单击 **Settings**。
3. 从 **Locale** 列表中选择所需的语言。
4. 点击 **Save**。确认框会通知您需要重新载入应用程序。
5. 单击 **Yes** 。系统会自动刷新您的 Web 浏览器以使用所选区域设置。

1.3.5.4. 自定义管理控制台标题

您可以自定义管理控制台标题，以便快速轻松地识别每个 JBoss EAP 实例。

前提条件

- JBoss EAP 正在运行。
- 您已创建了用户。

流程

1. 登录到管理控制台。对于在默认端口中运行的本地服务器，您可以访问管理控制台 http://localhost:9990/console/index.html。
2. 单击 **Settings**，再修改 **Title** 字段中的标题。
3. 点击 **Save**。
   确认框会通知您您必须重新加载管理控制台。
4. 单击 **Yes** 。
   系统会自动刷新您的 Web 浏览器，新标题会显示在标签页标头中。

1.3.6. 独立服务器配置文件

独立配置文件位于 ***EAP_HOME*/standalone/configuration/** 目录中。对于五个预定义的配置集（*default*, *ha*, *full*, *full-ha*, *load-balancer*），每个都有单独的文件。以下是启动 JBoss EAP 时可以使用管理 CLI 进行修改的示例配置文件。

表 1.1. 独立配置文件

配置文件	用途
standalone.xml	此独立配置文件是 JBoss EAP 启动单机服务器时使用的默认配置。此配置对应于 Jakarta EE Web 和 Core Profiles，并包含有关服务器的所有信息，包括子系统、网络、部署、套接字绑定和其他可配置的详细信息。此配置不提供消息传递或高可用性所需的子系统。
standalone-ha.xml	此单机配置文件包含所有默认子系统，并添加 **modcluster** 和 **jgroups** 子系统，以实现高可用性。它不提供消息传递所需的子系统。
standalone-full.xml	此单机配置文件包含所有默认子系统，并添加 **messaging-activemq** 和 **iiop-openjdk** 子系统。它对应于 Jakarta EE 完整的配置文件，不提供高可用性所需的子系统。
standalone-full-ha.xml	此独立配置文件包括对每一种可能子系统的支持，包括消息传递和高可用性方面的支持。
standalone-load-balancer.xml	此单机配置文件包含使用内置 mod_cluster 前端负载平衡器对其他 JBoss EAP 实例进行负载平衡所需的最小子系统。

默认情况下，将 JBoss EAP 作为单机服务器启动使用 **standalone.xml** 文件。若要使用其他配置启动 JBoss EAP，可使用 **--server-config** 参数：例如，

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

1.3.7. 受管域配置文件

受管域配置文件位于 ***EAP_HOME*/domain/configuration/** 目录中。以下是启动 JBoss EAP 时可以使用管理 CLI 进行修改的示例配置文件。

表 1.2. 受管域配置文件

配置文件	用途
domain.xml	这是受管域的主配置文件。只有域 master 会读取此文件。此文件包含所有配置集（*default*, *ha*, *full*, *full-ha*, *load-balancer*）的配置。
host.xml	此文件包含特定于受管域中物理主机的配置详细信息，如网络接口、套接字绑定、主机名称和其他特定于主机的详细信息。**host.xml** 文件包含 **host-master.xml** 和 **host-slave.xml** 的所有功能，这些已在此表中进行了描述。
host-master.xml	此文件仅包含将服务器作为受管域控制器运行所需的配置详细信息。**host-master.xml** 文件将其自身定义为域控制器，不定义任何服务器实例。
host-slave.xml	此文件仅包含作为受管域主机控制器运行服务器所需的配置详细信息。它没有定义域控制器，您必须为 **host-slave.xml** 要连接的域控制器配置地址。此 xml 文件代表一个示例配置，其中 **host-slave.xml** 运行在一台机器上，并由远程域控制器管理。机器充当一个主机控制器，来定义和启动服务器实例。域控制器管理这些服务器实例。

默认情况下，在受管域中启动 JBoss EAP 将使用 **host.xml** 文件。若要使用其他配置启动 JBoss EAP，可使用 **--host-config** 参数：例如，

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

1.3.8. 备份配置数据

要恢复您的 JBoss EAP 服务器配置，您必须在以下位置备份数据：

- ***EAP_HOME*/standalone/configuration/**
  - 备份整个目录，以保存单机服务器的用户数据、服务器配置和日志记录设置。
- ***_EAP_HOME*/standalone/data**
  - 为在 data/content 目录中限制的受管部署备份数据。
- **EAP_HOME/standalone/deployments**
  - 备份独立服务器的部署。
- ***EAP_HOME*/domain/configuration/**
  - 备份整个目录，以保存用户和配置文件数据、域和主机配置，以及受管域的日志记录设置。
- **EAP_HOME/domain/data**
  - 备份 data/content 目录中限制的受管域和部署的数据。
- ***EAP_HOME*/modules/**
  - 备份任何自定义模块。
- ***EAP_HOME*/welcome-content/**
  - 备份任何自定义欢迎内容。
- ***EAP_HOME*/bin/**
  - 备份任何自定义脚本或启动配置文件。

1.3.9. 配置文件快照

为了协助服务器维护和管理，JBoss EAP 在启动时创建原始配置文件的时间戳版本。

管理操作的任何其他配置更改将导致自动备份原始文件，保留实例的工作副本供引用和回滚。此外，还可以生成配置快照，它们是当前服务器配置的即时副本。这些快照可由管理员保存和加载。

以下示例使用 **standalone.xml** 文件，但同一进程适用于 **domain.xml** 和 **host.xml** 文件。

创建快照
使用管理 CLI 为当前配置生成快照。

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

```
    "result" => "EAP_HOME/standalone/configuration/standalone_xml_history/snapshot/20151022-
133109702standalone.xml"
}
```

列出快照
使用管理 CLI 列出所有快照。

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

删除快照
使用管理 CLI 删除快照。

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

1.3.10. 使用快照启动服务器

您可以使用快照或自动保存的配置版本来启动服务器。

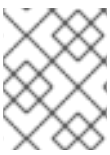
先决条件

- 您已安装了 JBoss EAP。
- 您已生成了一个配置文件的快照。

流程

1. 进入 ***EAP_HOME*/standalone/configuration/standalone_xml_history** 目录，找到要加载的快照或保存的配置文件。
2. 启动服务器并指向所选配置文件。传递与配置目录相关的文件路径 ***EAP_HOME*/standalone/configuration/**。

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



注意

在受管域中运行服务器时，请使用 **--host-config** 和 **--domain-config=<config>** 参数来指定配置文件。

1.3.11. 属性替换

您可以在 JBoss EAP 中使用表达式来定义可替换属性，以替换配置中的文字值。

在 **standalone*.xml** 或 **domain.xml** 配置文件中使用属性替换，将属性替换为系统属性中找到的值。系统属性定义在 EAP 配置文件 xml 文件中，也可以在命令行终端中输入 **-D** 命令来定义。

要确定给定子系统中是否允许属性替换，请使用以下命令显示子系统配置的描述：

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

如果 **expressions-allowed** 属性被设为 **true**，则允许属性替换。

表达式的格式为 **${*PARAMETER*:*DEFAULT_VALUE*}**。如果设置了指定参数，则将使用参数的值。否则，将使用提供的默认值。

解析表达式支持的源是系统属性和环境变量。使用环境变量解析表达式时，请使用 **${env.LANG}** 格式。

来自 **standalone.xml** 配置文件的以下示例，将 **public** 接口的 **inet-address** 设为 **127.0.0.1**，除非设置了 **jboss.bind.address** 参数。

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

在将 EAP 作为独立服务器启动时，您可以使用以下命令设置 **jboss.bind.address** 参数：

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

注意

对于部署，源可以是部署存档中的 **META-INF/jboss.properties** 文件中的属性。对于支持子部署的部署类型，如果属性文件位于外部部署中，则解析范围仅限于所有子部署，如 EAR。如果属性文件在子部署中，则解析的范围仅限于该子部署。

1.3.12. 嵌套表达式

您可以嵌套表达式，它允许更高级地使用表达式来代替固定值。

嵌套表达式的格式类似于普通表达式的格式，但一个表达式被嵌入在另一个表达式中，例如：

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

JBoss EAP 会递归评估嵌套表达式，因此首先评估 *inner* 表达式，然后评估 *outer* 表达式。表达式也可以是递归的，其中表达式被解析为另一个表达式，其然后再被解析。允许表达式的任何位置都允许嵌套表达式，但管理 CLI 命令除外。

例如，如果数据源定义密码被屏蔽，您可以使用嵌套的表达式。数据源的配置可能包含以下行：

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

系统属性(**datasource_name**)使用嵌套表达式替换了 **ds_ExampleDS** 的值。以下行是数据源的示例配置：

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

JBoss EAP 首先评估表达式 **${datasource_name}**，然后将其输入到更大的表达式并评估生成的表达式。此配置的优点在于数据源的名称是从固定配置中提取的。

1.3.13. 基于描述符的属性替换

部署基于描述符的属性替换根据描述符替换属性，以便您可以删除应用程序和构建链中有关环境的假设。

特定环境的配置可以在部署描述符中指定，而不是注释或构建系统脚本。您可以在文件中提供配置，或者作为参数在命令行中提供。

应用程序配置（如数据源连接参数）通常会因开发、测试和生产环境而异。构建系统脚本有时可以容纳这种差异，因为 Jakarta EE 规范不包含将这些配置外部化的方法。借助 JBoss EAP，您可以使用基于描述符的属性替换在外部管理配置。

spec-descriptor-property-replacement 标志控制 Jakarta EE 描述符替换，JBoss EAP 默认 *禁用* 它。启用后，您可以在以下部署描述符中替换属性：

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

您可以使用以下管理 CLI 命令，来在 Jakarta EE 描述符中启用或禁用属性替换：

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

jboss-descriptor-property-replacement 标志控制特定于 JBoss 的描述符替换，而 JBoss EAP 默认启用它。启用后，您可以在以下部署描述符中替换属性：

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

使用以下管理 CLI 命令，来在 JBoss EAP 特定描述符中启用或禁用属性替换：

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

注解**-property-replacement** 标志控制注释内的属性替换，它默认不启用。启用后，您可以替换应用程序类中注释属性中的属性。

使用以下管理 CLI 命令，来在注释中启用或禁用属性替换：

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

1.4. JBOSS EAP 网络和端口配置

您可以使用 JBoss EAP 配置各种服务的网络可访问性，并使用端口偏移来在使用相同的接口的同一计算机上轻松地运行多个 JBoss EAP 实例。网络配置是按照接口和套接字绑定进行组织的。

请使用以下有关这些网络和端口配置的详细信息，来成功运行 JBoss EAP。

1.4.1. 接口

JBoss EAP 在整个配置中引用了命名接口。您可以将 JBoss EAP 配置为引用具有逻辑名称的单个接口声明，而不用在每次使用时需要接口的全部详情。

您还可以在受管域中更轻松地配置，其中网络接口详情在多个机器之间可能会有所不同。每个服务器实例可以对应一个逻辑名称组。

standalone.xml、**domain.xml** 和 **host.xml** 文件都包含接口声明。根据使用的默认配置，有几个预配置的接口名称。**management** 接口可用于需要管理层的所有组件和服务，包括 HTTP 管理端点。 **public** 接口可用于所有应用相关的网络通信。**unsecure** 接口用于标准配置中的 IIOP 套接字。 **private** 接口用于标准配置中的 JGroups 套接字。

1.4.1.1. 默认接口配置

JBoss EAP 包括以下四个默认接口：

```
<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>
```

默认情况下，JBoss EAP 将这些接口绑定到 **127.0.0.1**，但可以通过设置适当的属性在运行时覆盖这些值。例如，通过以下命令将 JBoss EAP 启动为单机服务器时可以设置 **public** 接口的 **inet-address** ：

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

或者，您也可以在 server start 命令行上使用 **-b** 参数。

重要

如果您修改 JBoss EAP 使用的默认网络接口或端口，您也必须更改任何使用修改后的接口或端口的脚本。其中包括 JBoss EAP 服务脚本，以及在访问管理控制台或管理 CLI 时指定正确的接口和端口。

其他资源

- 服务器运行时参数和交换机。

1.4.1.2. 可选的接口配置

通过为物理接口指定逻辑名称和选择条件来声明网络接口。选择条件可以引用通配符地址，或者指定接口或地址必须是一个有效匹配的一个或多个特征集。

您可以使用管理控制台或管理 CLI 配置接口。本节后面的信息包括添加和更新接口的多个示例。首先显示管理 CLI 命令，后跟对应的配置 XML。

其他资源

- 默认接口配置.

1.4.1.2.1. 具有 NIC 值的接口

您可以使用以下示例添加一个 NIC 值为 **eth0** 的新接口。

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

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

1.4.1.2.2. 具有多个条件值的接口

如果其正在运行，您可以使用以下示例添加一个与正确子网上任何接口或地址匹配的新接口，支持多播，且不是点对点。

```
/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. 更新接口属性

在本例中，您可以更新 **public** 接口的默认 **inet-address** 值，保持 **jboss.bind.address** 属性，以便可以在运行时设置此值。

```
/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. 受管域中服务器的其他接口

您可以使用以下代码，向受管域中的服务器添加更多接口。

```
/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. 套接字绑定

使用套接字绑定和套接字绑定组来定义网络端口以及它们与 JBoss EAP 配置所需的网络接口的关系。套接字绑定是套接字的命名配置。套接字绑定组是套接字绑定声明的集合，这些声明按照逻辑名称分组。

这允许配置的其他部分根据其逻辑名称引用套接字绑定，而不必在每次使用套接字配置的完整详情。

您可以在 **standalone.xml** 和 **domain.xml** 配置文件中找到这些命名配置的声明。单机服务器仅包含一个套接字绑定组，而受管域则可包含多个组。您可以为受管域中的每个服务器组创建一个套接字绑定组，或者在多个服务器组之间共享套接字绑定组。

JBoss EAP 默认使用的端口取决于使用的套接字绑定组以及您各个部署的要求。

JBoss EAP 配置的套接字绑定组中可以定义三种类型的套接字绑定：

进站套接字绑定

socket-binding 元素用于为 JBoss EAP 服务器配置入站套接字绑定。默认 JBoss EAP 配置提供多个预配置的 **socket-binding** 元素，例如用于 HTTP 和 HTTPS 流量的元素。

远程出站套接字绑定

remote-destination-outbound-socket-binding 元素用于为远程到 JBoss EAP 服务器的目的地配置出站套接字绑定。默认 JBoss EAP 配置提供一个示例远程目标套接字绑定，可用于邮件服务器。

本地出站套接字绑定

local-destination-outbound-socket-binding 元素用于为属于 JBoss EAP 服务器本地的目的地配置出站套接字绑定。预计通常不会使用这种套接字绑定。

其他资源

- 套接字绑定属性.

1.4.2.1. 管理端口

默认情况下，JBoss EAP 8.0 将端口 **9990** 用于本地管理（由管理 CLI 使用）以及基于 Web 的管理控制台使用的 HTTP 管理。用作 JBoss EAP 6 中的原生管理端口的端口 **9999** 不再使用，但在需要时仍可启用。

如果为管理控制台启用了 HTTPS，则默认使用端口 **9993**。

1.4.2.2. 默认的套接字绑定

JBoss EAP 为预定义的五个配置集（*default*, *ha*, *full*, *full-ha*, *load-balancer*）的每一个都提供了一个套接字绑定组。

重要

如果您修改 JBoss EAP 使用的默认网络接口或端口，您也必须更改任何使用修改后的接口或端口的脚本。其中包括 JBoss EAP 服务脚本，以及在访问管理控制台或管理 CLI 时指定正确的接口和端口。

其他资源

- 默认套接字绑定参考。

1.4.2.2.1. 独立服务器的套接字绑定组

作为单机服务器运行时，每个配置文件仅定义一个套接字绑定组。每个独立配置文件（**standalone.xml**、**standalone-ha.xml**、**standalone-full.xml**、**standalone-full-ha.xml**、**standalone-load-balancer.xml**）定义其对应配置集所使用的技术的套接字绑定。

例如，默认的独立配置文件(**standalone.xml**)指定以下套接字绑定：

```
<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. 受管域中的套接字绑定组

在受管域中运行时，所有套接字绑定组都在 **domain.xml** 文件中定义。有五个预定义的套接字绑定组：

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

每个套接字绑定组都指定其对应配置集所使用的技术套接字绑定。例如，**full-ha-sockets** 套接字绑定组定义几个 **jgroups** 套接字绑定，供 *full-ha* 配置文件用于高可用性。

```
<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>
```

注意

管理接口的套接字配置在域控制器的 **host.xml** 文件中定义。

1.4.2.3. 配置套接字绑定

在定义套接字绑定时，您可以配置 **port** 和 **interface** 属性，以及多播设置，如 **multicast-address** 和 **multicast-port**。

流程

可以使用管理控制台或管理 CLI 配置套接字绑定。下列步骤介绍了添加套接字绑定组、添加套接字绑定和使用管理 CLI 配置套接字绑定设置。

1. 添加新套接字绑定组。

注意

在将 JBoss EAP 实例作为独立服务器运行时，您无法添加额外的套接字绑定。您可以删除、添加或修改现有的套接字绑定。

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

2. 添加套接字绑定。

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

3. 将套接字绑定更改为使用默认接口，由套接字绑定组设置。

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

以下示例显示了在上述步骤完成后 XML 配置可以如何进行。

```
<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>
```

其他资源

- 套接字绑定属性.

1.4.2.4. 端口偏移

端口偏移是一个数字偏移值，添加到该服务器的套接字绑定组中指定的所有端口值中。这允许服务器继承其套接字绑定组中定义的端口值，使用偏移来确保它与同一主机和接口上的任何其他服务器不会冲突。例如，如果套接字绑定组的 HTTP 端口为 **8080**，并且服务器使用端口偏移 **100**，则其 HTTP 端口为 **8180**。

本本节后面的信息是使用管理 CLI 对受管域中的服务器设置 **250** 端口偏移的示例。

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

端口偏移可用于受管域中的服务器和在同一主机上运行多个单机服务器。

使用 **jboss.socket.binding.port-offset** 属性启动单机服务器时，您可以传递端口偏移。

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

端口偏移使用 JBoss 配置文件中的系统属性名称定义。您可以更改系统属性名称，或将其删除并写死端口偏移设置。

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

1.4.3. IPv6 地址

默认情况下，JBoss EAP 配置为使用 IPv4 地址运行。以下流程描述了如何配置 JBoss EAP 以使用 IPv6 地址运行。

1.4.3.1. 为 IPv6 地址配置 JVM 堆栈

您可以将 JBoss EAP 配置为使用 IPv6 运行。

流程

要更新启动配置来在 IPv6 地址上运行，请完成以下步骤。

1. 打开启动配置文件。
   - 作为单机服务器运行时，编辑 ***EAP_HOME*/bin/standalone.conf** 文件（或对于 Windows Server，**standalone.conf.bat**）。
   - 在受管域中运行时，编辑 ***EAP_HOME*/bin/domain.conf** 文件（或对于 Windows Server，**domain.conf.bat**）。
2. 将 **java.net.preferIPv4Stack** 属性设置为 **false**。

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

3. 附加 **java.net.preferIPv6Addresses** 属性，并将它设为 **true**。

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

下例演示了在进行上述更改后，启动配置文件中的 JVM 选项如何显示。

```
#
# 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. 更新了 IPv6 地址的默认接口值

配置中的默认接口值可以更改为 IPv6 地址。例如，以下管理 CLI 命令将 **management** 接口设为 IPv6 环回地址(**::1**)。

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

运行上述命令后，以下示例显示了 XML 配置可能的样子。

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

1.5. JBOSS EAP 服务器配置的优化

将最新的更新应用到 JBoss EAP，以保持对安全 CVE 和其他客户报告的 bug 修复。

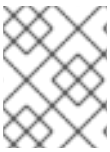
一旦安装了 JBoss EAP 服务器，并且创建了管理用户，请优化您的服务器配置。

常见优化包括：

- 设置 **ulimits**，以确保您的操作系统提供 web 连接所需的足够文件描述符
- 调整线程池大小

第 2 章 使用 JBOSS EAP 开发应用程序

您可以使用带有 Eclipse 集成开发环境(IDE)和 JBoss EAP 8.0 快速入门示例的 JBoss 工具开始开发应用程序。



注意

JBoss 工具在 JBoss EAP 8.0 中已弃用。不会对这个功能进行任何增强，它可能会在以后的版本中删除。

2.1. 概述

JBoss 工具是基于 Eclipse 插件的集合，增强了对 JBoss EAP 技术的支持。JBoss 工具可与 Eclipse 集成开发环境(IDE)一起使用。

注意

JBoss 工具在 JBoss EAP 8.0 中已弃用。不会对这个功能进行任何增强，它可能会在以后的版本中删除。

重要

JBoss 工具是一个社区项目，不受红帽支持。有关建立和运行 JBoss 工具实例的帮助，请参考 社区网站。要下载 JBoss 工具，请参阅 JBoss 工具下载。

JBoss EAP 8.0 提供了很多快速启动代码示例，可帮助用户使用不同的 Jakarta EE 技术开始编写应用程序。您可以使用 JBoss 工具运行快速入门示例。

其他资源

- 有关测试的 JBoss 工具版本的更多信息，请参阅 Red Hat JBoss Enterprise Application Platform (EAP)和 JBoss 工具。

2.2. 设置开发环境

您必须设置开发环境，以使用 Eclipse IDE。

注意

JBoss 工具在 JBoss EAP 8.0 中已弃用。不会对这个功能进行任何增强，它可能会在以后的版本中删除。

流程

1. 下载并安装 JBoss 工具。
   具体说明请查看 JBoss 工具安装指南中的 安装方法 。

2. 在 JBoss 工具中建立 JBoss EAP 服务器。
   具体说明请参阅 JBoss 工具指南中的 如何：配置 IDE 以与 JBoss EAP 和 JBoss Web 框架工具一起使用。

其他资源

- 有关测试的 JBoss 工具版本的更多信息，请参阅 Red Hat JBoss Enterprise Application Platform (EAP)和 JBoss 工具。

2.3. 使用快速入门示例

JBoss EAP 提供的快速入门示例是 Maven 项目。

2.3.1. 关于 Maven

Apache Maven 是 Java 应用程序开发中使用的分布式构建自动化工具，用于创建、管理和构建软件项目。Maven 使用名为 Project Object Model(POM)文件的标准配置文件来定义项目并管理构建流程。poms 描述模块和组件依赖项，使用 XML 文件描述生成的项目打包和输出的构建顺序和目标。这可确保以正确、一致的方式构建项目。

Maven 使用存储库可实现此目的。Maven 存储库存储 Java 库、插件和其他构建构件。默认公共存储库是 Maven 2 Central Repository，但存储库可以是私有和内部存储库，目标为在开发团队之间共享通用工件。也可从第三方获取存储库。如需更多信息，请参阅 Apache Maven 项目和存储库简介指南。Jakarta EE 开发人员通常用来在 JBoss EAP 上构建应用程序。

2.3.2. 快速入门使用 Maven

构建应用程序并部署到 JBoss EAP 8.0 所需的工件和依赖项托管在公共存储库中。从 JBoss EAP 7 快速入门开始，不再需要配置 Maven **settings.xml** 文件，以在构建快速入门时使用这些存储库。Maven 存储库现在在 Quickstart 项目 POM 文件中配置。我们提供了这种配置方法，以便更轻松地开始快速入门，但通常不建议用于生产项目，因为它可能会减慢您的构建速度。

JBoss 工具包括 Maven，因此无需单独下载和安装它。

如果您计划使用 Maven 命令行来构建和部署应用，您必须首先从 Apache Maven 项目下载 Maven 并使用 Maven 文档中的说明进行安装。

注意

JBoss 工具在 JBoss EAP 8.0 中已弃用。不会对这个功能进行任何增强，它可能会在以后的版本中删除。

2.4. 下载并运行快速启动

2.4.1. 下载快速启动

JBoss EAP 随附一整套快速入门代码示例，旨在帮助用户开始使用各种 Jakarta EE 技术编写应用程序。快速启动可以从红帽客户门户网站下载。

流程

1. 登录到红帽客户门户上的 JBoss EAP 下载页面。
2. 在 **Version** 下拉菜单中选择 **8.0**。
3. 在列表中找到 **Red Hat JBoss Enterprise Application Platform 8.0.0 Quickstarts**条目，然后点 **Download** 下载包含快速入门的 **.zip** 文件。
4. 将 **.zip** 文件保存到所需的目录中。

5. 解压缩 **.zip** 文件。

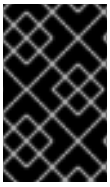
2.4.2. 将快速启动导入到 JBoss 工具

下载快速启动后，可以将它们导入到 JBoss 工具，并部署到 JBoss EAP。

注意

JBoss 工具在 JBoss EAP 8.0 中已弃用。不会对这个功能进行任何增强，它可能会在以后的版本中删除。

每个快速入门都附带了一个 POM 文件，其中包含其项目和配置信息。使用此 POM 文件轻松将快速启动导入到 JBoss 工具中。



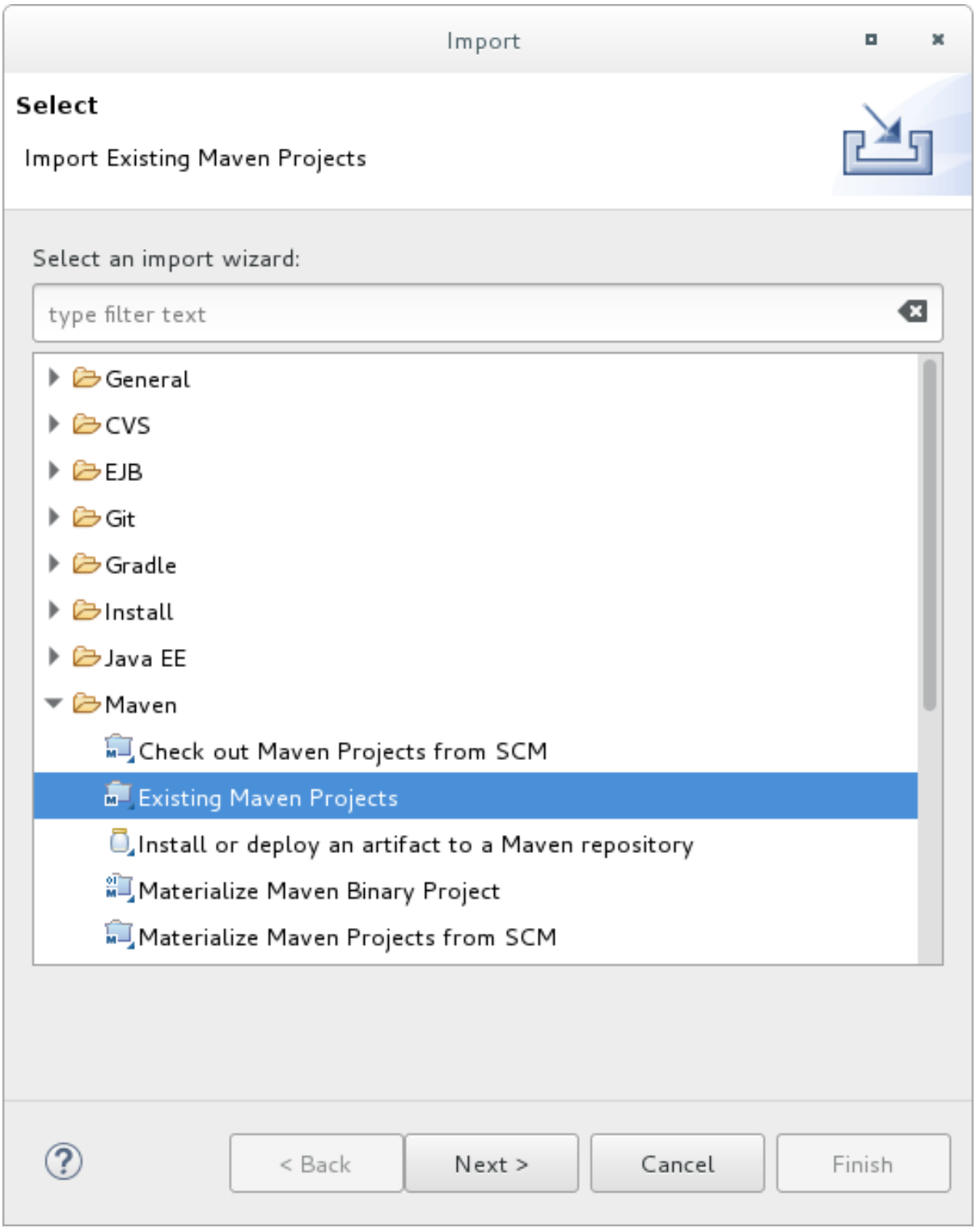
重要

如果在将其导入到 JBoss 工具中时，您的快速启动项目文件夹位于 IDE 工作区内，则 IDE 会生成一个无效的项目名称和 WAR 存档名称。在开始之前，请确保快速启动项目文件夹位于 IDE 工作区之外。

流程

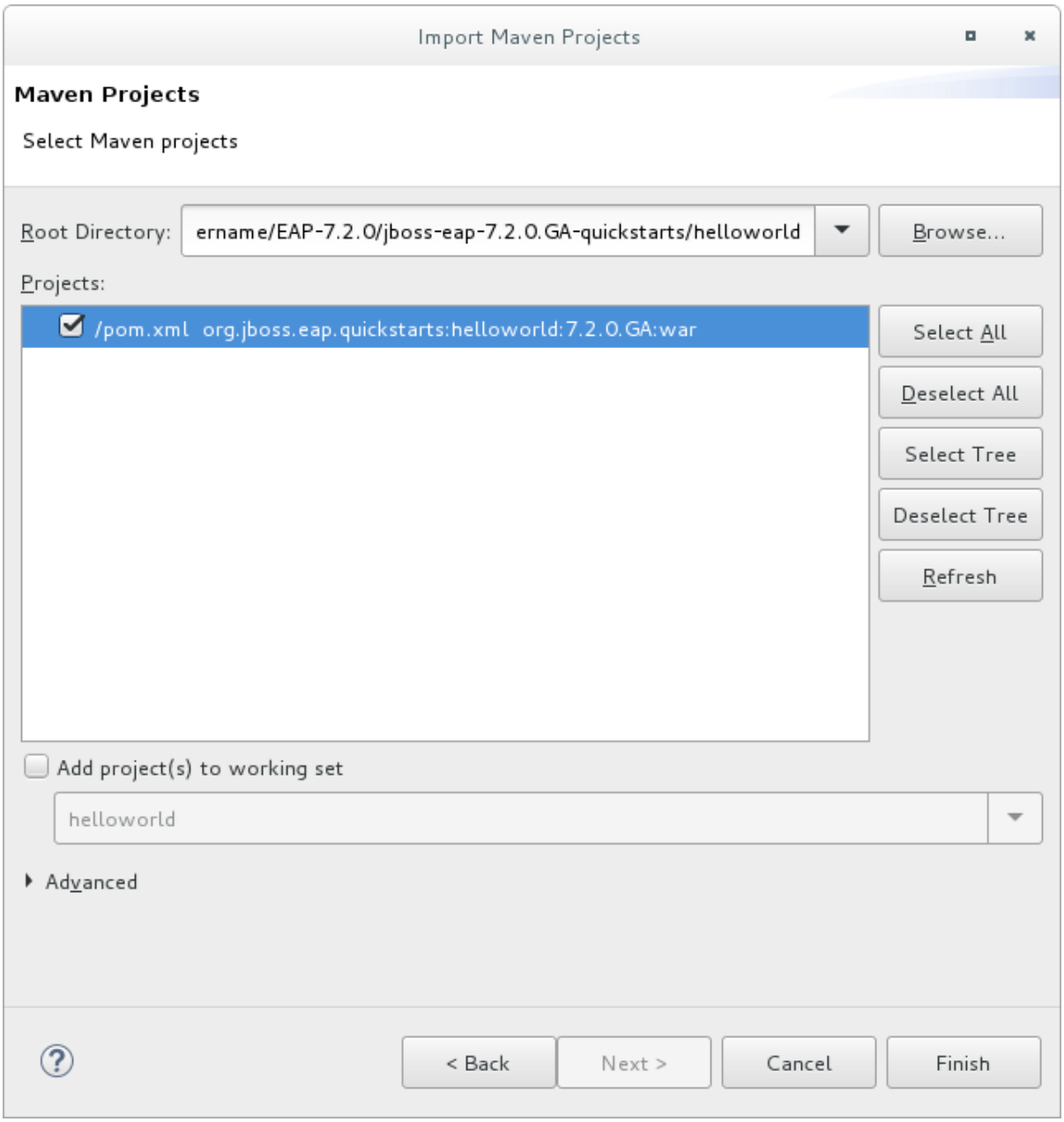
1. 启动 JBoss 工具。
2. 选择 **File → Import**。
3. 选择 **Maven → Existing Maven Projects**，然后点 **Next**。

图 2.1. 导入现有 Maven 项目



4. 浏览到所需的快速入门目录（如 **helloworld** 快速启动），然后单击 **OK**。**Projects** 列表框填充了所选快速启动项目的 **pom.xml** 文件。

图 2.2. 选择 **Maven Projects**



5. 点 **Finish**。

其他资源

- 有关测试的 JBoss 工具版本的更多信息，请参阅 Red Hat JBoss Enterprise Application Platform (EAP)和 JBoss 工具。

2.4.3. 运行 *helloworld* 快速启动

运行 **helloworld** 快速入门是一种简单的方式，可以验证 JBoss EAP 服务器是否已正确配置和运行。

注意

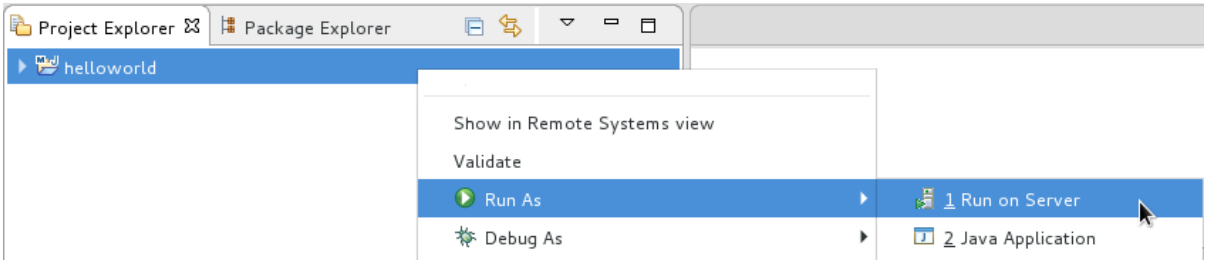
JBoss 工具在 JBoss EAP 8.0 中已弃用。不会对这个功能进行任何增强，它可能会在以后的版本中删除。

流程

1. 如果您尚未定义服务器，则将 JBoss EAP 服务器添加到 JBoss 工具中。请参阅 JBoss 工具指南中的 如何：配置 IDE 以与 JBoss EAP 和 JBoss Web 框架工具包一起工作 。

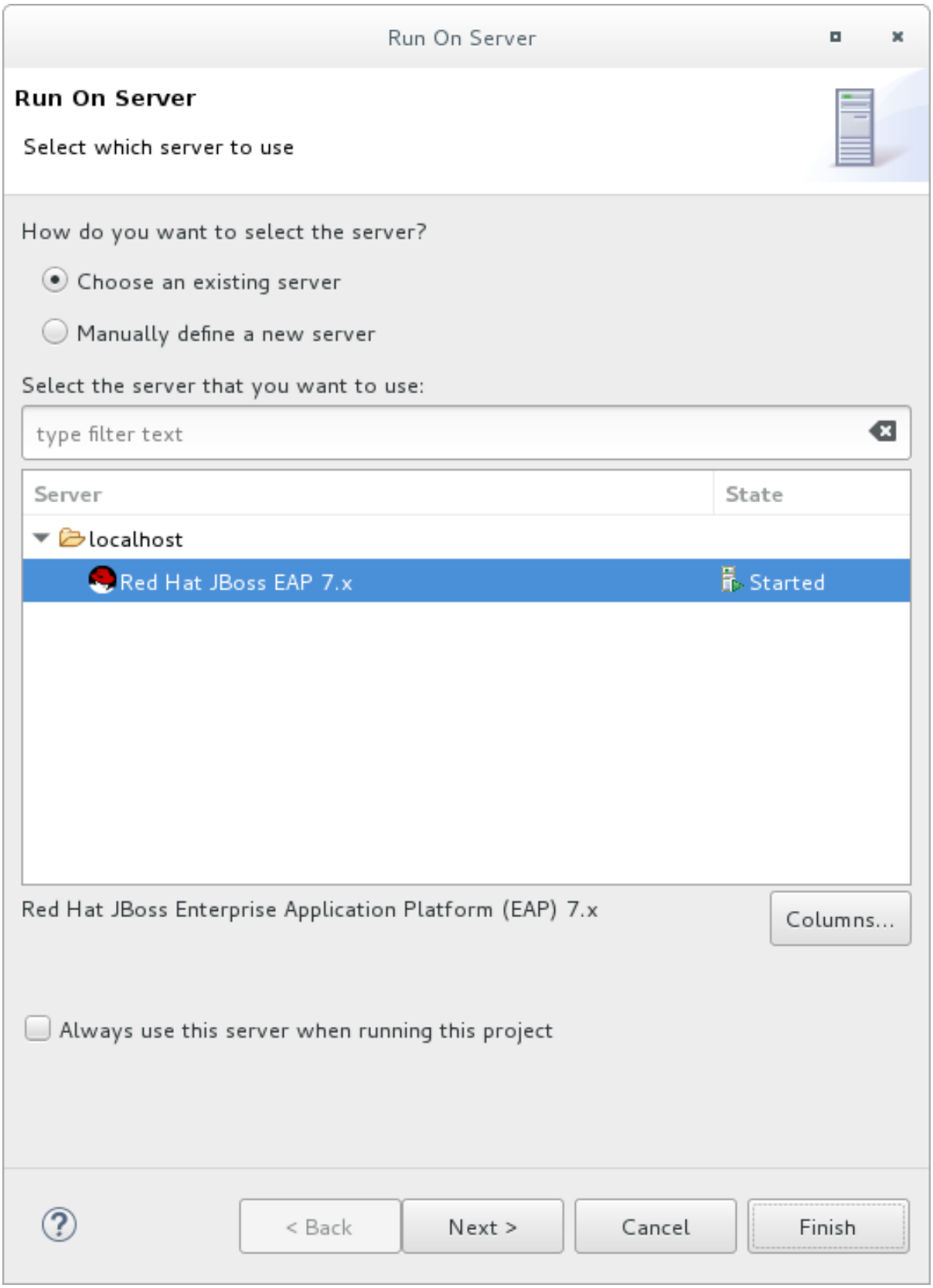
2. 右键点击 **Project Explorer** 选项卡中的 **helloworld** 项目，然后选择 **Run As → Run on Server**。

   图 **2.3. Run As - 在服务器中运行**



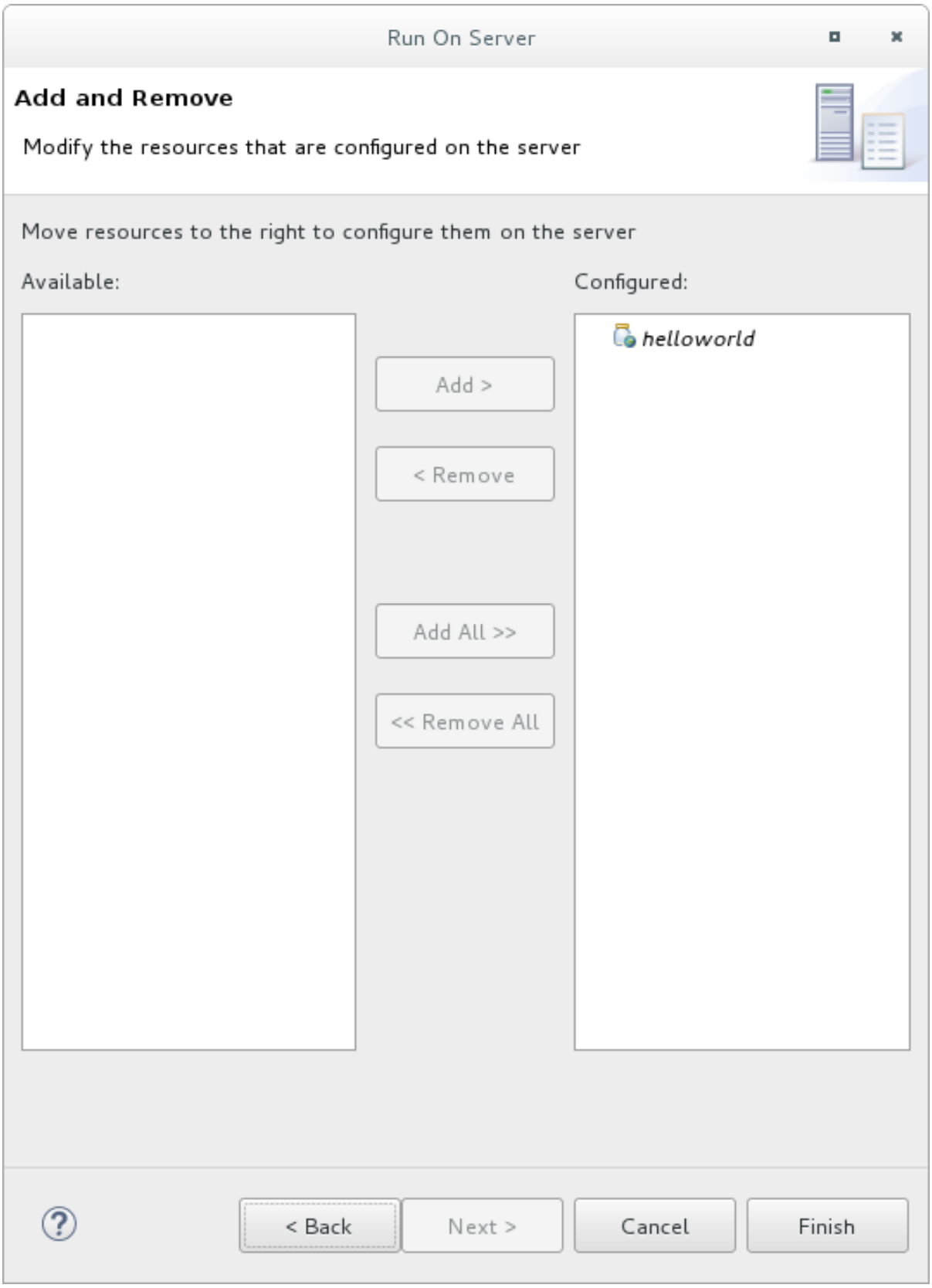
3. 从服务器列表中选择 JBoss EAP 8.0 服务器，然后单击 **Next**。

图 2.4. 在服务器上运行



4. **helloworld** 快速入门已经列出，需要配置在服务器上。点 **Finish** 以部署快速入门。

图 2.5. 修改服务器上配置的资源



5. 验证结果。

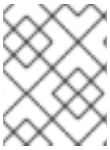
- 在 **Server** 选项卡中，JBoss EAP 8.0 服务器状态更改为 **Started**。
- **Console** 选项卡显示详细说明 JBoss EAP 服务器启动和 **helloworld** 快速启动部署的消息。

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

- **helloworld** 可以通过 http://localhost:8080/helloworld 访问，并显示文本 **Hello World!**。

2.4.4. 运行 *bean-validation* 快速启动

某些快速入门（如 **bean-validation** Quickstart）不提供用户界面层，而是提供 Arquillian 测试来演示功能。



注意

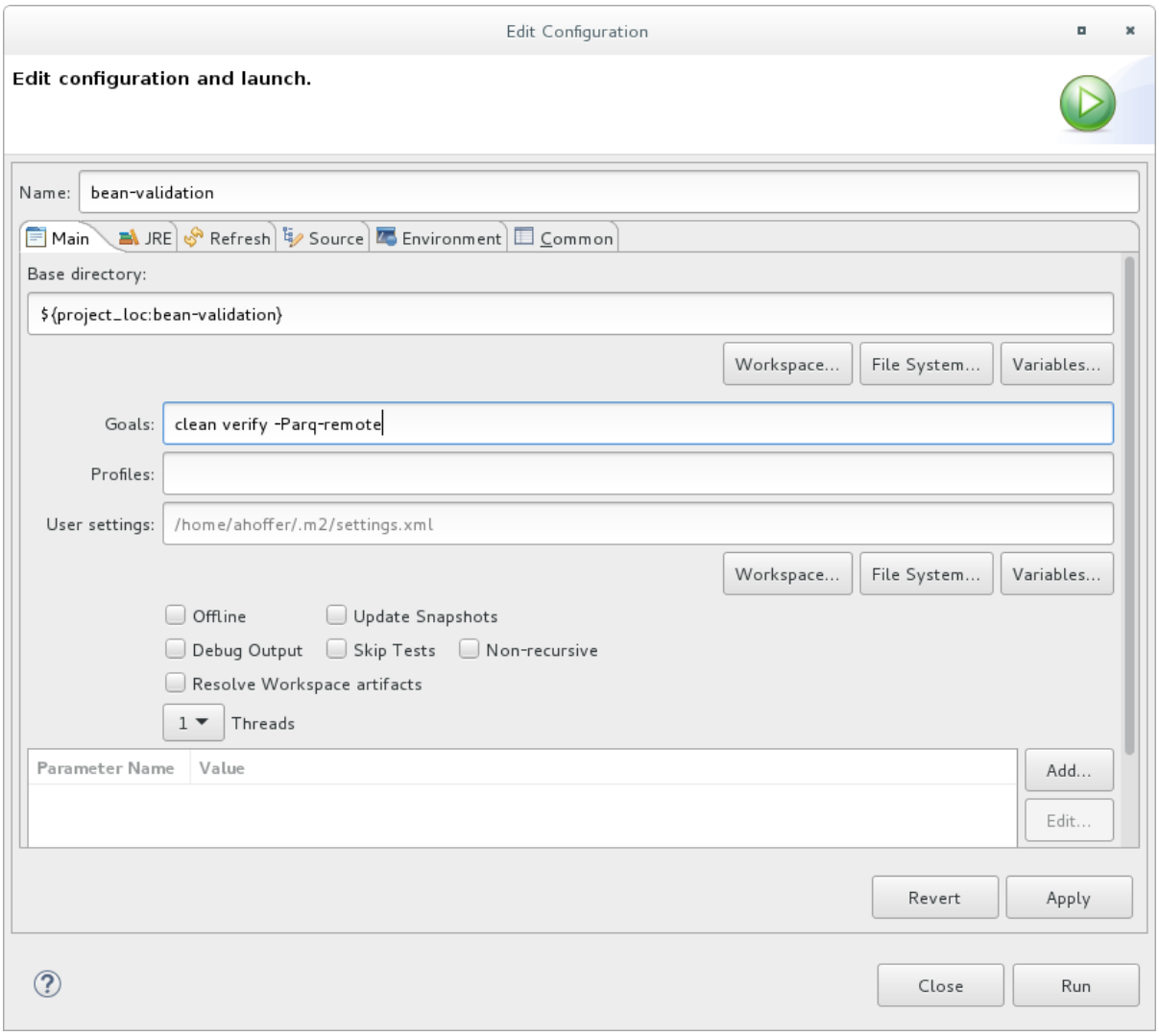
JBoss 工具在 JBoss EAP 8.0 中已弃用。不会对这个功能进行任何增强，它可能会在以后的版本中删除。

流程

1. 将 **bean-validation** 快速启动导入到 JBoss 工具中。
2. 在 **Servers** 选项卡中，右键单击服务器，然后选择 **Start** 以启动 JBoss EAP 服务器。如果您没有看到 **Servers** 选项卡或尚未定义服务器，请将 JBoss EAP 服务器添加到 JBoss 工具中。请参阅 JBoss 工具指南中的 如何：配置 IDE 以与 JBoss EAP 和 JBoss Web 框架工具包一起工作 。
3. 右键单击 **Project Explorer** 选项卡中的 **bean-validation** 项目，然后选择 **Run As → Maven Build**。
4. 在目标输入字段中输入以下内容，然后点 **Run**。

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

图 2.6. 编辑配置



5. 验证结果。
Console 标签页显示 **bean-validation** Arquillian 测试的结果：

```
-------------------------------------------------------
 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. 从命令行运行快速启动

您可以使用 Maven 从命令行轻松构建和部署快速入门。如果您尚未安装 Maven，请参见 Apache Maven 项目以下载和安装它。

快速入门的根目录中提供了一个 **README.md** 文件，其中包含系统要求、配置 Maven、添加用户和运行快速入门的一般信息。

每个快速入门还包含自己的 **README.md** 文件，它提供特定的指令和 Maven 命令来运行该快速入门。

流程

1. 检查 *helloworld* 快速启动根目录中的 **README.md** 文件。
2. 启动 JBoss EAP 服务器。

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

3. 前往 *helloworld* quickstart 目录。
4. 使用快速启动的 **README.md** 文件中提供的 Maven 命令构建和部署快速入门。

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

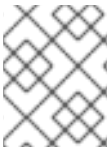
5. *helloworld* 应用可以通过 http://localhost:8080/helloworld 访问并显示文本 **Hello World!**。

2.5. 回顾快速启动示例

2.5.1. 浏览 helloworld 快速启动

helloworld 快速入门演示了如何将简单的 Servlet 部署到 JBoss EAP。业务逻辑封装在服务中，作为上下文和依赖注入(CDI) Bean 提供，并注入到 Servlet 中。此快速入门是一个起点，可确保您已正确配置和启动服务器。

使用命令行构建和部署此快速启动的详细说明，请参见 **helloworld** 快速启动目录根目录下的 **README.html** 文件。本主题展示了如何使用 JBoss 工具来运行快速启动，并假设您已安装了 JBoss 工具、配置了 Maven，导入并成功运行了 **helloworld** 快速启动。



注意

JBoss 工具在 JBoss EAP 8.0 中已弃用。不会对这个功能进行任何增强，它可能会在以后的版本中删除。

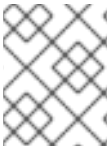
2.5.1.1. 检查目录结构

helloworld 快速启动的代码可以在 ***QUICKSTART_HOME*/helloworld/** 目录中找到。**helloworld** 快速启动由 Servlet 和一个 CDI bean 组成。它还包含应用的 **WEB-INF/** 目录中的 **beans.xml** 文件，其版本号为 1.1，并且 **bean-discovery-mode** 是 **all**。此标志文件将 WAR 识别为 bean 存档，并告知 JBoss EAP 在此应用程序中查找 bean，并激活 CDI。

src/main/webapp/ 目录包含快速启动的文件。本例的所有配置文件都位于 **src/main/webapp/** 中的 **WEB-INF/** 目录中，包括 **beans.xml** 文件。**src/main/webapp/** 目录还包括 **index.html** 文件，该文件使用简单的 meta refresh 将用户的浏览器重定向到 Servlet，它位于 http://localhost:8080/helloworld/HelloWorld。quickstart 不需要 **web.xml** 文件。

2.5.1.2. 回顾 HelloWorldServlet.java 代码

软件包声明和导入已从这些列表中排除。Quickstart 源代码中提供了完整的列表。



注意

JBoss 工具在 JBoss EAP 8.0 中已弃用。不会对这个功能进行任何增强，它可能会在以后的版本中删除。

先决条件

- 安装 JBoss 工具。具体说明请查看 JBoss 工具安装指南中的 安装方法 。
- 运行 **helloworld** 快速入门。
- 打开 Web 浏览器并在 http://localhost:8080/helloworld 访问应用，以验证 **helloworld** quickstart 已成功部署到 JBoss EAP。

流程

1. 查看 **HelloWorldServlet** 代码。
HelloWorldServlet.java 文件位于 **src/main/java/org/jboss/as/quickstarts/helloworld/** 目录中。此 servlet 将信息发送到浏览器。

示例：HelloWorldServlet 类代码

```
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 详情

此 servlet 会向您的浏览器发送信息。

表 2.1. HelloWorldServlet 详情

行	备注
43	您只需要添加 **@WebServlet** 注释，并提供用于访问 servlet 的 URL 映射。
46-48	每个网页都需要正确构成 HTML。这个快速入门使用静态字符串来编写最小标头和页脚输出。
50-51	这些行注入生成实际消息的 HelloService CDI bean。只要我们不更改 HelloService 的 API，这种方法允许我们以后在不更改视图层的情况下更改 HelloService 的实施。
58	此行调用 服务以生成消息"Hello World"，并将它写入到 HTTP 请求。

1. 检查 **HelloService** 代码。
   HelloService.java 文件位于 **src/main/java/org/jboss/as/quickstarts/helloworld/** 目录中。此服务只需返回一条消息。不需要 XML 或注解注册。

   示例：HelloService 类代码

```
public class HelloService {

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

其他资源

- 有关测试的 JBoss 工具版本的更多信息，请参阅 Red Hat JBoss Enterprise Application Platform (EAP)和 JBoss 工具。

2.5.2. 探索 numberguess 快速启动

numberguess 快速启动演示了如何将简单的非持久性应用创建和部署至 JBoss EAP。使用 JSF 视图显示信息，业务逻辑被封装到两个 CDI bean 中。在 **numberguess** 快速启动中，您有十次尝试猜测 1 到 100 之间的数字。在每次尝试后，您都会被告知您的猜测过高还是过低。

numberguess Quickstart 的代码可以在 ***QUICKSTART_HOME*/numberguess/** 目录中找到，其中 ***QUICKSTART_HOME*** 是您下载并解压缩 JBoss EAP 快速入门的目录。 **numberguess** 快速启动由多个 bean、配置文件和 Facelets (JSF)视图组成，并被打包为 WAR 模块。

使用命令行构建和部署此快速启动的详细说明，请参阅 **numberguess** quickstart 目录的 **README.html** 文件。以下示例使用 JBoss 工具来运行快速启动。

注意

JBoss 工具在 JBoss EAP 8.0 中已弃用。不会对这个功能进行任何增强，它可能会在以后的版本中删除。

2.5.2.1. 检查 numberguess 配置文件

本例的所有配置文件都位于 ***QUICKSTART_HOME*/numberguess/src/main/webapp/WEB-INF/** 目录中。

先决条件

- 安装 JBoss 工具。具体说明请查看 JBoss 工具安装指南中的 安装方法 。
- 运行 **numberguess** Quickstart。
- 打开 Web 浏览器并访问 URL http://localhost:8080/numberguess 来访问这个应用，以验证 **numbergues** quickstart 已成功部署到 JBoss EAP。

注意

JBoss 工具在 JBoss EAP 8.0 中已弃用。不会对这个功能进行任何增强，它可能会在以后的版本中删除。

流程

1. 检查 **face-config.xml** 文件。
   该快速启动使用 **faces-config.xml** 文件名的 JSF 2.2 版本。Facelets 的标准化版本是 JSF 2.2 中的默认视图处理程序，因此不需要配置。此文件仅包含 root 元素，只是指示应用中应启用 JSF 的标志文件。

```
<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. 检查 **beans.xml** 文件。
   beans.xml 文件包含版本号 1.1，并且 **bean-discovery-mode** 包含 **all**。此文件是一种标志文件，将 WAR 识别为 bean 存档，并告知 JBoss EAP 在此应用程序中查找 Bean ，并激活 CDI。

```
<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>
```

注意

此快速入门不需要 **web.xml** 文件。

其他资源

- 有关测试的 JBoss 工具版本的更多信息，请参阅 Red Hat JBoss Enterprise Application Platform (EAP)和 JBoss 工具。

2.5.2.2. 检查 JSF 代码

JSF 将 **.xhtml** 文件扩展用于源文件，但使用 **.jsf** 扩展提供呈现的视图。**home.xhtml** 文件位于 **src/main/webapp/** 目录中。

示例：JSF 源代码

```
<html xmlns="http://www.w3.org/1999/xhtml"
 xmlns:ui="http://java.sun.com/jsf/facelets"
 xmlns:h="http://java.sun.com/jsf/html"
 xmlns:f="http://java.sun.com/jsf/core">

 <head>
 <meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1" />
 <title>Numberguess</title>
 </head>

 <body>
 <div id="content">
  <h1>Guess a number...</h1>
  <h:form id="numberGuess">

  <!-- Feedback for the user on their guess -->
 <div style="color: red">
  <h:messages id="messages" globalOnly="false" />
  <h:outputText id="Higher" value="Higher!"
     rendered="#{game.number gt game.guess and game.guess ne 0}" />
  <h:outputText id="Lower" value="Lower!"
     rendered="#{game.number lt game.guess and game.guess ne 0}" />
 </div>

 <!-- Instructions for the user -->
 <div>
 I'm thinking of a number between <span
 id="numberGuess:smallest">#{game.smallest}</span> and <span
 id="numberGuess:biggest">#{game.biggest}</span>. You have
 #{game.remainingGuesses} guesses remaining.
 </div>

 <!-- Input box for the users guess, plus a button to submit, and reset -->
 <!-- These are bound using EL to our CDI beans -->
 <div>
 Your guess:
 <h:inputText id="inputGuess" value="#{game.guess}"
  required="true" size="3"
  disabled="#{game.number eq game.guess}"
  validator="#{game.validateNumberRange}" />
  <h:commandButton id="guessButton" value="Guess"
   action="#{game.check}"
   disabled="#{game.number eq game.guess}" />
 </div>
 <div>
 <h:commandButton id="restartButton" value="Reset"
 action="#{game.reset}" immediate="true" />
 </div>
 </h:form>
```

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

下面的行号与在 JBoss 工具中查看文件时看到的行号相对应。

注意

JBoss 工具在 JBoss EAP 8.0 中已弃用。不会对这个功能进行任何增强，它可能会在以后的版本中删除。

表 2.2. JSF 详情

行	备注
36-40	这些消息可以发送给用户："Higher!"和"Lower!"。
45-48	用户猜测，可以猜到的数字范围会较小。这一句子会改变，确保他们知道有效猜测的范围。
55-58	此输入字段绑定至使用值表达式的 bean 属性。
58	验证器绑定用于确保用户不会意外输入他们可能猜到的范围之外的数字。如果验证器不在此处，用户可能会对不限号使用一个猜测。
59-61	必须有办法让用户将其猜测发送到服务器。在这里，我们绑定了 Bean 的操作方法。

2.5.2.3. 检查 numberguess 类文件

所有 **numberguess** 快速启动源文件都可在 ***QUICKSTART_HOME*/numberguess/src/main/java/org/jboss/as/quickstarts/numberguess/** 目录中找到。软件包声明和导入已从这些列表中排除。Quickstart 源代码中提供了完整的列表。

流程

1. 查看 **Random.java** Qualifier Code
   限定符用于消除两个 Bean 之间的不确定性，两者都有资格根据其类型注入。**@Random** 限定符用于注入随机数字。

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

}
```

2. 查看 **MaxNumber.java** Qualifier Code
@MaxNumber qualifier 用于注入允许的最大数量。

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

3. 查看 **Generator.java** Code
Generator 类通过制作者方法创建随机数，并通过相同方式公开可能的最大数量。此类为应用范围，因此每次都不会出现不同的随机值。

```
@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. 查看 **Game.java** 代码
会话范围的 **Game** 类是应用的主要入口点。它负责设置或重置游戏，捕获和验证用户的猜测，并通过 **FacesMessage** 向用户提供反馈。它使用构建后生命周期方法从 **@Random Instance<Integer>** bean 检索随机数来初始化游戏。

注意类中的 **@Named** 注释。只有在您希望使用 Jakarta Expression Language （在本例中为 **#{game}**）使 bean 可访问 JSF 时，才需要此注释。

```
@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);
        }
    }
}
```

附录 A. JBOSS EAP 入门的参考信息

您可以使用参数、属性和默认套接字绑定来帮助您开始使用 JBoss EAP。例如，您可以使用参数将替代配置设置为默认的 JBoss EAP 独立服务器。这有助于配置服务器以满足您的需要。

A.1. 服务器运行时参数和开关

在受管域中的独立服务器和服务器上，您可以使用特定服务器运行时参数以及应用的启动脚本。脚本可以使用与 **standalone.xml**、**domain.xml** 和 **host.xml** 配置文件中定义的配置不同的配置来启动服务器。其他配置可能包括启动服务器，并设置替代套接字绑定或辅助配置。

在启动服务器之前，您可以通过在终端中发出帮助开关，**-h** 或 **--help** 来访问可用的参数列表。

表 A.1. 运行时参数和开关的描述：

参数或开关	服务器类型	描述
--admin-only	Standalone	将服务器的运行类型设为 **ADMIN_ONLY**。参数打开管理接口并接受管理请求，但该参数不会启动其他运行时服务或接受用户请求。为获得最佳性能，请使用 **--start-mode=admin-only** 参数。
--admin-only	Domain	将主机控制器的运行类型设为 **ADMIN_ONLY** 会导致主机控制器打开管理界面并接受管理请求，但主机控制器不会启动服务器。对于域的 master 主机控制器，它接受来自 slave 主机控制器的传入连接。
-b=<value> , **-b <value>**	Standalone, Domain	设置系统属性 **jboss.bind.address**，您可以使用它来配置公共接口的绑定地址。绑定地址默认为 **127.0.0.1**。有关为其他接口设置绑定地址，请参阅 **-b<interface>=<value>** 条目。
-b<interface>=<value>	Standalone, Domain	将系统属性 **jboss.bind.address.<interface>** 设为给定值。例如：**-bmanagement=*IP_ADDRESS***。
--backup	Domain	即使此主机不是域控制器，也保留永久域配置的副本。
-c=<config> , **-c <config>**	Standalone	要使用的服务器配置文件的名称。默认值为 **standalone.xml**。
-c=<config> , **-c <config>**	Domain	要使用的服务器配置文件的名称。默认值为 **domain.xml**。
--cached-dc	Domain	如果主机不是域控制器，并且无法在启动时联系域控制器，那么您必须使用域配置的本地缓存副本来启动。

参数或开关	服务器类型	描述
--debug [<port>]	Standalone	使用可选参数激活调试模式以指定端口。参数仅在启动脚本支持参数时才能工作。
-D<name>[=<value>]	Standalone, Domain	设置系统属性。
--domain-config=<config>	Domain	要使用的服务器配置文件的名称。默认为 **domain.xml**。
--git-repo	Standalone	用于管理和持久服务器配置数据的 Git 存储库的位置。如果为本地存储则可以是 **local**；或是到远程存储库的 URL。
--git-branch	Standalone	Git 存储库中要使用的分支或标签名称。您必须命名一个现有的分支或标签名称，如果不存在，则 *不会* 创建它。如果使用标签名称，请将存储库置于分离的 HEAD 状态，这意味着以后的提交不会附加到任何分支。标签名称为只读，通常在多个节点之间复制配置时使用。
--git-auth	Standalone	Elytron 配置文件的 URL 包含服务器连接到远程 Git 存储库时使用的凭证。当远程 Git 存储库需要身份验证时，您可以使用此参数。Elytron 不支持 SSH。Elytron 只支持使用私钥的默认 SSH 身份验证，而无需密码。您不能使用带有 **loca** 存储库的参数。
-h，--help	Standalone, Domain	显示帮助信息并退出帮助索引。
--host-config=<config>	Domain	要使用的主机配置文件的名称。默认值为 **host.xml**。
--interprocess-hc-address=<address>	Domain	主机控制器可以侦听来自进程控制器通信的地址。
--interprocess-hc-port=<port>	Domain	主机控制器可以侦听来自进程控制器通信的端口。
--master-address=<address>	Domain	将系统属性 **jboss.domain.master.address** 设为给定值。在默认的 slave 主机控制器配置中，您可以使用参数来配置 master 主机控制器的地址。
--master-port=<port>	Domain	将系统属性 **jboss.domain.master.port** 设为给定值。在默认的 slave 主机控制器配置中，您可以使用参数来配置 master 主机控制器用于本地管理通信的端口。

参数或开关	服务器类型	描述
--read-only-server-config=<config>	Standalone	要使用的服务器配置文件的名称。参数与 **--server-config** 和 **-c** 的不同之处在于，该参数不会覆盖原始文件。
--read-only-domain-config=<config>	Domain	要使用的域配置文件的名称。参数与 **--domain-config** 和 **-c** 的不同之处在于，参数不会覆盖初始文件。
--read-only-host-config=<config>	Domain	要使用的主机配置文件的名称。参数与 **--host-config** 的不同之处在于，参数不会覆盖初始文件。
-P=<url>, -P <url>, --properties=<url>	Standalone, Domain	从指定的 URL 加载系统属性。
--pc-address=<address>	Domain	进程控制器侦听来自其控制的进程的通信的地址。
--pc-port=<port>	Domain	进程控制器在其上侦听其控制进程的通信的端口。
-S<name>[=<value>]	Standalone	设置安全属性。
-secmgr	Standalone, Domain	运行安装有安全管理器的服务器。
--server-config=<config>	Standalone	要使用的服务器配置文件的名称。默认为 **standalone.xml**。
--start-mode=<mode>	Standalone	设置服务器的启动模式。您不能将参数与 **--admin-only** 参数一起使用。您可以将以下条目与参数一起使用： • **normal** ：服务器正常启动。 • **admin-only** ：服务器只在管理界面中打开并接受管理请求，但服务器不会启动其他运行时服务或接受最终用户请求。 • **suspend** ：服务器以暂停模式启动，但服务器不会接收服务请求，直到服务器恢复为止。
-u=<value> ， **-u <value>**	Standalone, Domain	设置系统属性 **jboss.default.multicast.address**，服务器用来在配置文件中的套接字绑定元素中配置多播地址。默认值为 **230.0.0.4**。
-v,-V,--version	Standalone, Domain	显示应用服务器版本并退出。

警告

JBoss EAP 设置其包含的配置文件，以处理开关的行为。例如：**-b** 和 **-u**。如果您将配置文件改为不再使用由开关控制的系统属性，那么将系统属性添加到 start 命令不能正常工作。

A.2. ADD-USER 参数

您可以将参数与 **add-user.sh** 脚本或 **add-user.bat** 脚本一起使用，以配置这些脚本如何将新用户添加到属性文件中以进行身份验证。

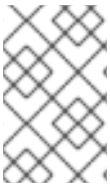
表 A.2. add-user 参数的描述

命令行参数	描述
-a	在应用程序域中创建用户。如果您没有在应用程序域中创建用户，则脚本会默认在管理域中创建用户。
-dc <value>	包含属性文件的域配置目录。如果省略该参数，则脚本会将 ***EAP_HOME*/domain/configuration/** 设为默认目录。
-sc <value>	另一种包含属性文件的独立服务器配置目录。如果省略该参数，则脚本会将 ***EAP_HOME*/standalone/configuration/** 设为默认目录。
-up,--user-properties <value>	备用用户属性文件的名称。您可以设置文件的绝对路径，或通过将参数与 **-sc** 或 **-dc** 参数一起使用来指定一个文件名称来设置一个替代配置目录。
-g,--group <value>	分配给用户的以逗号分隔的组的列表。
-gp, --group-properties <value>	备用组属性文件的名称。您可以设置文件的绝对路径，或通过将参数与 **-sc** 或 **-dc** 参数一起使用来指定一个文件名称来设置一个替代配置目录。
-p,--password <value>	用户的密码。
-u,--user <value>	用户名称。用户名只能以任何数字和顺序包含以下字符： • 字母数字字符（a-z、A-Z、0-9） • 短划线(-)、句点(.)、逗号(@) • 反斜杠(\) • 等号 (=)
-r、--realm <value>	用于保护管理接口的域名称。如果省略，默认值为 **ManagementRealm**。

命令行参数	描述
-s,**--silent**	运行对控制台不带输出的 **add-user** 脚本。
-e,**--enable**	启用用户。
-d,**--disable**	禁用用户。
-cw, **--confirm-warning**	以交互模式自动确认警告。
-h，**--help**	显示 **add-user** 脚本的使用信息。
-ds,**--display-secret**	以非交互模式打印机密值。

A.3. 接口属性

您可以使用接口属性来配置 JBoss EAP 接口。



注意

表中的属性名称按 JBoss EAP 在其管理模型中列出的顺序显示。请参阅 ***EAP_HOME*/docs/schema/wildfly-config_5_0.xsd** 中的模式定义文件，来查看 XML 中出现的元素。XML 元素列表必须与管理模型中显示的不同。

表 A.3. 接口属性的描述：

接口属性	描述
any	指定接口必须至少满足所选嵌套条件集的一个，但不一定是全部。
any-address	将通配符地址绑定到使用接口的套接字的空属性。属性有以下配置选项： ● 属性使用 IPv6 通配符地址(**::**)作为默认值。如果将 **java.net.preferIPv4Stack** 系统属性设为 **true**，则套接字使用 IPv4 通配符地址(**0.0.0.0**)。 ● 如果套接字绑定到双栈机器上的 IPv6 anylocal 地址，则该套接字接受 IPv6 和 IPv4 流量。 ● 如果套接字绑定到 IPv4 (IPv4-mapped) anylocal 地址，则套接字仅接受 IPv4 流量。
inet-address	以 IPv6 形式或 IPv4 点十进制表示法指定 IP 地址或指定解析到 IP 地址的主机名。
link-local-address	空属性，用来指定接口是否包括链接本地地址的条件。
loopback	空属性，用来指定接口是否被识别为回环接口的条件。

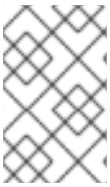
接口属性	描述
loopback-address	回环地址可能尚未在机器的回环接口上配置。属性与 **inet-address** 类型不同，因为接口使用了属性值，即使值包含没有 IP 地址的 NIC。
multicast	空属性，用于指定接口是否支持多播的条件。
name	接口的名称。
nic	网络接口的名称，如 **eth0** 、**eth1** 或 **lo**。
nic-match	正则表达式，用于将机器上可用网络接口的名称与可接受的接口匹配。
not	指示接口不能满足的选择条件的属性。
point-to-point	空属性，用来指定接口是否被识别为点对点接口的条件。
public-address	空属性，用来指定接口是否包含公开的可路由地址的条件。
site-local-address	空属性，用来指定接口是否包含站点本地地址的条件。
subnet-match	指定网络 IP 地址和地址网络前缀中的位数，用*斜杠表示法*，如 **192.168.0.0/16** 。
up	空属性，用于指定接口是否处于启用状态的条件。
virtual	空属性，用来指定接口是否包含标识为虚拟接口的条件。

A.4. 套接字绑定属性

您可以使用套接字绑定属性来配置 JBoss EAP 服务器的套接字绑定。

以下类型的套接字绑定存在特定的属性：

- 进站套接字绑定
- 远程出站套接字绑定
- 本地出站套接字绑定



注意

表中的属性名称按 JBoss EAP 在其管理模型中列出的顺序显示。请参阅 ***EAP_HOME*/docs/schema/wildfly-config_5_0.xsd** 中的模式定义文件，来查看 XML 中出现的元素。XML 元素列表必须与管理模型中显示的不同。

表 A.4. 入站套接字绑定的描述, **socket-binding**, 属性：

属性	描述
client-mapping	指定入站套接字绑定的客户端映射。连接到入站套接字的客户端必须使用映射中指定的目标地址。此地址与出站接口匹配。通过将 **client-mapping** 属性与入站套接字绑定一起使用，您可以应用高级网络拓扑，该拓扑使用网络地址转换或包含多个网络接口上的绑定。您必须按声明的顺序评估每个映射，即，第一个成功的匹配决定了映射的目标。
fixed-port	使用该属性来确定端口值是否必须保持固定。即使对套接字组中的其他套接字应用了数字偏移，您也可以使用该属性。
interface	设置套接字绑定的接口的名称的属性。您也可以使用该属性来设置多播套接字必须侦听的接口。如果您没有定义声明的接口，则该属性使用来自封闭套接字绑定组中的 **default-interface** 值。
multicast-address	套接字接收多播流量的多播地址。如果没有为该属性指定值，则不要将套接字配置为接收多播功能。
multicast-port	套接字在其上接收多播流量的端口。如果配置了 **multicast-address** 属性，则您必须配置该属性。
name	您必须设置套接字的名称。需要访问套接字配置信息的服务无法使用名称来查找套接字。
port	套接字绑定的端口号。如果您将服务器配置为应用 **port-offset** 来递增或递减所有端口值，则必须覆盖此属性值。

表 A.5. 远程出站套接字绑定的描述, **remote-destination-outbound-socket-binding**, 属性：

属性	描述
fixed-source-port	确定端口值是否必须保持固定，即使您将数字偏移应用到了套接字组中的其他出站套接字。
host	出站套接字连接的远程目标的主机名或 IP 地址。
port	出站套接字连接的远程目标的端口号。
source-interface	JBoss EAP 用于出站套接字的源地址的接口名称。
source-port	JBoss EAP 用作出站套接字的源端口的端口号。

表 A.6. 本地出站套接字绑定的描述, **local-destination-outbound-socket-binding**, 属性：

属性	描述
fixed-source-port	确定端口值是否必须保持固定，即使您将数字偏移应用到了套接字组中的其他出站套接字。

属性	描述
socket-binding-ref	JBoss EAP 用来确定连接出站套接字端口的本地套接字绑定的名称。
source-interface	JBoss EAP 用于出站套接字的源地址的接口名称。
source-port	JBoss EAP 用作出站套接字的源端口的端口号。

A.5. 默认的套接字绑定

您可以为每个套接字绑定组设置默认套接字绑定。

JBoss EAP 有以下五种类型默认套接字绑定：

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

表 A.7. 默认 **standard-sockets** 套接字绑定的描述：

套接字绑定	端口	描述
ajp	**8009**	Apache JServ 协议。用于 HTTP 集群和负载平衡。
http	**8080**	部署 Web 应用的默认端口。
https	**8443**	部署的 Web 应用程序和客户端之间的 SSL 加密连接。
management-http	**9990**	用于与管理层的 HTTP 通信。
management-https	**9993**	用于与管理层通信的 HTTPS。
txn-recovery-environment	**4712**	JTA 事务恢复管理器。
txn-status-manager	**4713**	JTA / JTS 事务管理器。

表 A.8. 默认 **ha-sockets** 套接字绑定的描述：

套接字绑定	端口	多播端口	描述
ajp	**8009**		Apache JServ 协议。用于 HTTP 集群和负载平衡。

套接字绑定	端口	多播端口	描述
http	**8080**		部署 Web 应用的默认端口。
https	**8443**		部署的 Web 应用程序和客户端之间的 SSL 加密连接。
jgroups-mping		**45700**	多播。用于发现 HA 集群中的初始成员资格。
jgroups-tcp	**7600**		使用 TCP 在 HA 集群中单播对等发现。
jgroups-udp	**55200**	**45688**	使用 UDP 在 HA 集群中进行多播对等发现。
management-http	**9990**		用于与管理层的 HTTP 通信。
management-https	**9993**		用于与管理层通信的 HTTPS。
modcluster		**23364**	用于 JBoss EAP 和 HTTP 负载平衡器之间通信的多播端口。
txn-recovery-environment	**4712**		JTA 事务恢复管理器。
txn-status-manager	**4713**		JTA / JTS 事务管理器。

表 A.9. 默认 **full-sockets** 套接字绑定的描述：

套接字绑定	端口	描述
ajp	**8009**	Apache JServ 协议。用于 HTTP 集群和负载平衡。
http	**8080**	部署 Web 应用的默认端口。
https	**8443**	部署的 Web 应用程序和客户端之间的 SSL 加密连接。
iiop	**3528**	用于 JTS 事务和其他 ORB 依赖服务的 CORBA 服务。
iiop-ssl	**3529**	SSL 加密的 CORBA 服务。
management-http	**9990**	用于与管理层的 HTTP 通信。
management-https	**9993**	用于与管理层通信的 HTTPS。
txn-recovery-environment	**4712**	JTA 事务恢复管理器。

套接字绑定	端口	描述
txn-status-manager	**4713**	JTA / JTS 事务管理器。

表 A.10. 默认 **full-ha-sockets** 套接字绑定的描述：

Name	端口	多播端口	描述
ajp	**8009**		Apache JServ 协议。用于 HTTP 集群和负载平衡。
http	**8080**		部署 Web 应用的默认端口。
https	**8443**		部署的 Web 应用程序和客户端之间的 SSL 加密连接。
iiop	**3528**		用于 JTS 事务和其他 ORB 依赖服务的 CORBA 服务。
iiop-ssl	**3529**		SSL 加密的 CORBA 服务。
jgroups-mping		**45700**	多播。用于发现 HA 集群中的初始成员资格。
jgroups-tcp	**7600**		使用 TCP 在 HA 集群中单播对等发现。
jgroups-udp	**55200**	**45688**	使用 UDP 在 HA 集群中进行多播对等发现。
management-http	**9990**		用于与管理层的 HTTP 通信。
management-https	**9993**		用于与管理层通信的 HTTPS。
modcluster		**23364**	用于 JBoss EAP 和 HTTP 负载平衡器之间通信的多播端口。
txn-recovery-environment	**4712**		JTA 事务恢复管理器。
txn-status-manager	**4713**		JTA / JTS 事务管理器。

表 A.11. 默认 **load-balancer-sockets** 套接字绑定的描述：

Name	端口	多播端口	描述
http	**8080**		部署 Web 应用的默认端口。

Name	端口	多播端口	描述
https	**8443**		部署的 Web 应用程序和客户端之间的 SSL 加密连接。
management-http	**9990**		用于与管理层的 HTTP 通信。
management-https	**9993**		用于与管理层通信的 HTTPS。
mcmp-management	**8090**		用于传输生命周期事件的 Mod-Cluster Management 协议(MCMP)连接的端口。
modcluster		**23364**	用于 JBoss EAP 和 HTTP 负载平衡器之间通信的多播端口。

第 3 章 JBOSS EAP 8.0 的 PACKAGE NAMESPACE CHANGE

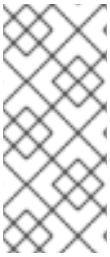
本节提供了 JBoss EAP 8.0 中用于 package 命名空间更改的额外信息。JBoss EAP 8.0 对 Jakarta EE 10 以及 Jakarta EE 10 API 的许多其他实现提供全面支持。用于 JBoss EAP 8.0 的 Jakarta EE 10 支持的重要更改是软件包命名空间更改。

3.1. JAVAX 到 JAKARTA 命名空间更改

Jakarta EE 8 和 EE 10 之间的关键区别在于，将 EE API Java 软件包从 **javax** 重命名为 **jakarta**
prerequisites。这遵循 Java EE 迁移到 Eclipse Foundation 并建立 Jakarta EE。

适应此命名空间更改是将应用程序从 JBoss EAP 7 迁移到 JBoss EAP 8 的最大任务。要将应用程序迁移到 Jakarta EE 10，您必须完成以下步骤：

- 将 **javax** 软件包中的 EE API 类的任何导入语句或其他源代码使用到 **jakarta** 软件包。
- 更新任何 EE 指定的系统属性或其他以 **javax** 开头的配置属性的名称，以从 **jakarta** 开始。
- 对于任何使用 **java.util.ServiceLoader** 机制启动的 EE 接口或抽象类，请将识别 META-INF/services/javax 的实施类从 **META-INF/services/javax.[rest_of_name]** 改为 **META-INF/services/jakarta.[rest_of_name]**。



注意

Red Hat Migration Toolkit 可帮助更新应用程序源代码中的命名空间。如需更多信息，请参阅 如何使用 Red Hat Migration Toolkit for Auto-Migration of a Application to the Jakarta EE 10 Namespace。如果源代码迁移不是一个选项，则 Open Source Eclipse Transformer 项目会提供字节代码转换工具，来将现有 Java 存档从 **javax** 命名空间转换为 **jakarta** 命名空间。

注意

这个更改不会影响作为 Java SE 一部分的 **javax** 软件包。

其他资源

- 如需更多信息，请参阅 javax 到 jakarta Package Namespace Change。

更新于 2024-02-08