Red Hat JBoss Enterprise Application Platform 7.4

入门指南

下载、安装、启动、停止和维护 Red Hat JBoss Enterprise Application Platform。

Last Updated: 2024-06-14

Red Hat JBoss Enterprise Application Platform 7.4 入门指南

下载、安装、启动、停止和维护 Red Hat JBoss Enterprise Application Platform。

法律通告

摘要

本指南的目的是让您能够快速启动并运行 JBoss EAP。本指南涵盖了管理任务，如基本安装、管理和配置 JBoss EAP。本指南还 帮助开发人员 开始使用 JBoss EAP 快速入门编写 Jakarta EE 应用程序。如需更多信息，请参阅整个 JBoss EAP 文档套件。

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. 请包含 文档 **URL**、章节编号 并描述问题。
3. 在 **Summary** 中输入问题的简短描述。
4. 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。
5. 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

红帽承诺替换我们的代码、文档和网页属性中存在问题的语言。我们从这四个术语开始： master、slave、blacklist 和 whitelist。这些更改将在即将发行的几个发行本中逐渐实施。详情请查看 CTO Chris Wright 信息。

第 1 章 管理 JBOSS EAP

1.1. 下载并安装 JBOSS EAP

压缩文件选项是下载和安装 JBoss EAP 的快速、平台独立方式。

1.1.1. 下载 JBoss EAP

您必须先下载 JBoss EAP 压缩文件，然后才能安装 JBoss EAP。

先决条件

- 确认您的系统符合 JBoss EAP 支持的配置。
- 安装最新的更新和勘误补丁。
- 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： 对于 Windows Server，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. 在产品下载 列表中，点击 **Red Hat JBoss Enterprise Application Platform**。
4. 在 **Version** 下拉菜单中，选择 **7.4**。
5. 在列表中找到 **Red Hat JBoss Enterprise Application Platform 7.4.0**，点 **Download** 链接。压缩文件已下载到您的系统。

其他资源

- 要访问红帽产品下载，请访问红帽客户门户网站。

1.1.2. 安装 JBoss EAP

您可以通过将软件包内容提取到所需的文件位置来安装 JBoss EAP 压缩文件。

先决条件

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

- 对于 Windows Server，设置 **JAVA_HOME** 和 **PATH** 环境变量。

流程

1. 将压缩文件移到您要安装 JBoss EAP 的服务器和位置。
2. 展开压缩文件。
   a. 在 Linux 中，使用以下命令：

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

   b. 在 Windows Server 上，右键单击压缩文件并选择" 提取所有"。
   通过提取压缩文件创建的目录是 JBoss EAP 安装的顶级目录。此目录被称为 ***EAP_HOME***。

其他资源

- 有关使用图形安装程序或 RPM 软件包安装方法安装 JBoss EAP 的更多信息，请参阅 安装指南。

1.2. 启动和停止 JBOSS EAP

启动 JBoss EAP 的方法取决于您将 JBoss EAP 作为单机服务器运行，还是在受管域中的服务器上运行。

停止 JBoss EAP 的方法取决于您运行 JBoss EAP 交互式或后台实例。

1.2.1. 将 JBoss EAP 作为独立服务器启动

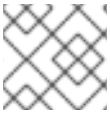
您可以将 JBoss EAP 作为单机服务器运行，以管理单个 JBoss EAP 实例。

JBoss EAP 在以下平台上被支持：

- Red Hat Enterprise Linux
- Windows Server
- Oracle Solaris

服务器以暂停状态启动，且不接受请求，直到所有需要的服务启动为止。在所需的服务启动后，服务器会进入正常的运行状态，并开始接受请求。

此启动脚本使用 ***EAP_HOME*/bin/standalone.conf** 文件或 **standalone.conf.bat** 用于 Windows Server，以设置默认的首选项，如 JVM 选项。您可以自定义此文件中的设置。



注意

要在终端中查看启动脚本参数列表，请使用 **--help** 参数。

JBoss EAP 默认使用 **standalone.xml** 配置文件，但您可以使用不同的配置文件来启动它。

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 作为单机服务器启动：

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

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

其他资源

- 有关可用独立配置文件和使用它们的更多信息，请参阅 _Standalone Server Configuration Files 部分。
- 有关所有可用启动脚本参数及其目的的完整列表，请参阅 Server Runtime Arguments 部分。

1.2.2. 在受管域中启动 JBoss EAP

您可以在受管域中运行 JBoss EAP，以使用单个域控制器管理多个 JBoss EAP 实例。

JBoss EAP 在以下平台上被支持：

- Red Hat Enterprise Linux
- Windows Server
- Oracle Solaris

服务器以暂停状态启动，且不接受请求，直到所有需要的服务启动为止。在所需的服务启动后，服务器会进入正常的运行状态，并开始接受请求。

您必须启动域控制器，然后域中的任何服务器组中的服务器。

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
```

   a. 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 **domain.conf.bat** 用于 Windows Server，以设置默认的首选项，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启动该文件。

在设置受管域时，必须将附加参数传递给启动脚本。

其他资源

- 有关受管域配置文件的更多信息，请参阅 托管域配置文件 部分。
- 有关所有可用启动脚本参数及其目的的完整列表，请使用 **--help** 参数或查看 Server Runtime Arguments 部分。

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单机服务器或域控制器的交互式实例。

先决条件

- 您启动了 JBoss EAP 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，以关闭受管域中单机服务器或服务器的运行实例。

先决条件

- 您有一个在终端中运行的 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 关闭命令 ：

```
shutdown
```

在受管域中运行 JBoss EAP 实例时，必须使用 **--host** 参数和 **shutdown** 命令指定要关闭的主机名。

1.3. JBOSS EAP 管理

JBoss EAP 使用简化的配置，每个单机服务器或受管域有一个配置文件。单机服务器的默认配置存储在 ***EAP_HOME*/standalone/configuration/standalone.xml** 文件中，受管域的默认配置则存储在 ***EAP_HOME*/domain/configuration/domain.xml** 文件中。此外，主机控制器的默认配置存储在 ***EAP_HOME*/domain/configuration/host.xml** 文件中。

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。使用这些管理接口所做的更改将自动保留，并且管理 API 将覆盖 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 支持使用 YAML 文件为独立服务器修改 XML 配置。如需更多信息，请参阅使用 YAML 文件更新独立服务器配置。

注意

受管域中的服务器 不支持 YAML 配置。

1.3.1. 管理用户

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

但是，如果您要远程访问管理 CLI，则必须添加管理用户，或使用管理控制台，即使流量来自本地主机上也被视为远程访问。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。

如果使用图形安装程序安装 JBoss EAP，则在安装过程中创建管理用户。

本指南介绍了使用 **add-user** 脚本对 JBoss EAP 进行简单的用户管理，此脚本可用于将新用户添加到用于开箱即用身份验证的属性文件中。

有关更高级的身份验证和授权选项，如 LDAP 或基于角色的访问控制(RBAC)，请参见 JBoss EAP *安全架构*的核心管理身份验证部分。

1.3.1.1. 添加管理用户

1. 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
```

注意

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

2. 按 **ENTER**，选择默认选项 **a** 以添加管理用户。
此用户将添加到 *ManagementRealm* 中，并将获得使用管理控制台或管理控制台执行管理操作的授权。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。

3. 输入所需的用户名和密码。系统将提示您确认密码。

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，但会发出警告。

如需了解有关更改此默认行为的详细信息，请参阅 JBoss EAP *配置指南*的设置附加用户实用程序密码限制一节。

4. 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** 将它留空。

5. 检查信息并输入 **yes** 进行确认。

6. 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**.
可能需要添加到 *ManagementRealm* 的一种用户是代表另一个 JBoss EAP 实例的用户，它必须能够进行身份验证以作为群集成员加入。如果出现这种情况，则在此提示中回答 **yes**，系统会为您提供一个表示用户密码的散列化机密值，该值需要添加到其他配置文件中。

也可以通过向 **add-user** 脚本传递参数，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。如需更多信息，请参阅使用非交换方式运行 Add-User 实用程序。

1.3.1.2. 使用交换方式运行 Add-User 实用程序

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

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。

创建一个属于多个组的用户
以下命令添加了一个管理用户, **mgmtuser1**，它带有 **guest** 和 **mgmtgroup** 组。

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

指定替代属性文件
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**

这些默认目录和属性文件名可以被覆盖。以下命令添加新用户，为用户属性文件指定不同的名称和位置。

```
$ 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。

有关所有可用 **add-user** 参数及其目的的完整列表，请使用 **--help** 参数或查看 Add-user 参数 部分。

1.3.2. 管理接口

1.3.2.1. 管理 CLI

管理命令行界面(CLI)是 JBoss EAP 的命令行管理工具。

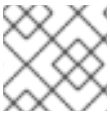
使用管理 CLI 启动和停止服务器、部署和取消部署应用、配置系统设置，以及执行其他管理任务。操作可以在批处理模式下执行，允许以组形式运行多个任务。

许多常见的终端命令可用，如 **ls**、**cd** 和 **pwd**。管理 CLI 也支持 tab 自动完成功能。

有关使用管理 CLI 的详细信息，包括命令和操作、语法以及批处理模式下运行，请参阅 JBoss EAP *管理 CLI 指南*。

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
```

或者，您可以使用 ***EAP_HOME*/bin/jboss-cli.sh --connect** 命令启动管理 CLI 并在一个步骤中进行连接。

显示帮助
使用以下命令获取一般帮助：

```
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，若要接收关于使用 **deploy** 的信息，请执行以下命令：

```
deploy --help
```

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
}
```

在受管域中运行时，您必须通过 **/profile=*PROFILE_NAME*** 命令指定要更新的配置集。

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

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
管理 CLI 也可用于在受管域中运行时启动和停止服务器。

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

1.3.2.2. 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、部署和取消部署应用、调优系统设置，以及对服务器配置进行持续修改。管理控制台还具备执行管理任务的功能，当当前用户执行的任何更改都要求重新启动或重新加载服务器实例时，实时通知功能。

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

对于使用默认管理端口在本地主机上运行的 JBoss EAP 实例，可通过位于 http://localhost:9990/console/index.html 的 Web 浏览器访问管理控制台。您将需要使用具有访问管理控制台权限的用户进行身份验证。

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。

Configuration
: 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。

1.3.3. 配置文件

1.3.3.1. 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。

表 1.1. 独立配置文件

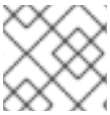
配置文件	用途
standalone.xml	此独立配置文件是启动单机服务器时使用的默认配置。它包含有关服务器的所有信息，包括子系统、网络、部署、套接字绑定和其他可配置的详细信息。它不提供消息传递或高可用性所需的子系统。
standalone-ha.xml	此单机配置文件包含所有默认子系统，并添加 **modcluster** 和 **jgroups** 子系统，以实现高可用性。它不提供消息传递所需的子系统。
standalone-full.xml	此单机配置文件包含所有默认子系统，并添加 **messaging-activemq** 和 **iiop-openjdk** 子系统。它不提供高可用性所需的子系统。
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
```

1.3.3.1.1. 使用 YAML 文件更新独立服务器配置

使用 YAML 文件配置单机服务器可外部化自定义流程，并提高服务器升级的速度。使用此功能时，服务器以只读模式启动。这意味着，在服务器重启后，对配置的更改不会保留。



注意

受管域中的服务器 不支持 YAML 配置。

用户可以修改 YAML 文件中的各种资源。YAML 文件支持以下资源：

- **core-service**
- **interface**
- **socket-binding-group**
- **subsystem**
- **system-property**

YAML 文件 不支持 以下资源：

- 扩展 ：向服务器添加扩展。不支持这个元素，因为它可能需要缺少的模块。
- 部署 ：将部署添加到服务器。这个元素不被支持，因为它除了配置外还需要更多大量更改。
- **deployment-overlay** ：将 deployment-overlays 添加到服务器。这个元素不被支持，因为它除了配置外还需要更多大量更改。

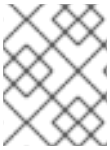
- 路径 ：在解析 YAML 文件时定义就绪。

YAML root 节点是 **wildfly-configuration**。您可以跟踪模型树来修改资源。如果已存在资源（由 XML 配置文件或以前的 YAML 文件创建），您可以使用模型树更新它。如果资源不存在，您可以使用模型树创建它。

定义新的 PostGresql 数据源的 YAML 配置文件示例

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

上面的示例定义了名为 **postgresql** 的 **jdbc-driver**，以及名为 **PostgreSQLDS** 的数据源。



注意

您不能使用 YAML 配置文件来管理模块。相反，您需要手动或通过管理 CLI 创建或调配 **org.postgresql.jdbc** 模块。

1.3.3.1.2. 使用标签进行 YAML 文件操作

您可以使用标签对 YAML 配置文件执行几个操作。

- **!undefine**: undefine a attribute

  undefine CONSOLE 日志记录器级别 YAML 配置文件示例

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

- **!remove** ：删除资源

  删除嵌入的 Artemis 代理并连接到远程代理 YAML 配置文件示例

```
wildfly-configuration:
 socket-binding-group:
  standard-sockets:
   remote-destination-outbound-socket-binding:
    remote-artemis:
     host: localhost
     port: 61616
 subsystem:
  messaging-activemq:
   server:
    default: !remove
   remote-connector:
    artemis:
     socket-binding: remote-artemis
   pooled-connection-factory:
    RemoteConnectionFactory:
     connectors:
      - artemis
     entries:
      - "java:jboss/RemoteConnectionFactory"
      - "java:jboss/exported/jms/RemoteConnectionFactory"
     enable-amq1-prefix: false
     user: admin
     password: admin
  ejb3:
   default-resource-adapter-name: RemoteConnectionFactory
  ee:
   service:
    default-bindings:
     jms-connection-factory: "java:jboss/RemoteConnectionFactory"
```

- **!list-add** : 向列表添加一个元素（使用可选索引）

  将 RemoteTransactionPermission 添加到权限列表 YAML 配置文件示例

```
wildfly-configuration:
  subsystem:
    elytron:
     permission-set:
      default-permissions:
       permissions: !list-add
        - class-name: org.wildfly.transaction.client.RemoteTransactionPermission
```

```
module: org.wildfly.transaction.client
target-name: "*"
index: 0
```

注意

如果未定义 **index** 属性，该条目将附加到列表的末尾。

1.3.3.1.3. 使用 YAML 文件启动单机服务器

您可以使用 YAML 配置文件启动单机服务器。

流程

1. 打开终端。
2. 使用以下命令启动带有 YAML 文件的单机服务器：

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

--yaml or **-y** 参数允许您传递 YAML 文件列表。您必须使用分号(;)对于 Windows Server 或基于 Unix 的操作系统的冒号(:)来分隔每个 YAML 文件路径。您可以使用绝对路径、相对于当前执行目录的路径，或者相对于独立配置目录的路径。

操作按照定义文件的顺序以及 XML 配置定义初始操作后的顺序应用。

1.3.3.2. 受管域配置文件

受管域配置文件位于 ***EAP_HOME*/domain/configuration/** 目录中。

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，如下所述。
host-master.xml	此文件仅包含将服务器作为主域控制器运行所需的配置详细信息。
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。

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
```

1.3.3.3. 备份配置数据

为了稍后恢复 JBoss EAP 服务器配置，应备份以下位置中的项目：

- ***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*/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*/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。

1.3.3.4. 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。

进行快照
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 列出已执行的所有快照。

```
: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)
```

使用快照启动服务器
可以使用快照或自动保存的配置启动服务器。

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

在受管域中运行时，请使用 **--host-config** 参数来指定配置文件。

1.3.3.5. 属性替换

JBoss EAP 允许您使用表达式来定义可在配置中替换字面值的可替换属性。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包括系统属性、环境变量和密码库。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。

以下 **standalone.xml** 配置文件的示例将 **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>
```

将 EAP 启动为单机服务器时，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
```

嵌套表达式
表达式可以嵌套，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}}
```

嵌套表达式是递归评估的，因此首先评估*内嵌*表达式，然后评估*外部*表达式。表达式也可能是递归的，其中一个表达式解析为另外一个表达式，然后解析。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，如果数据源定义中使用的密码被屏蔽，则可以使用嵌套表达式。数据源的配置可能包含以下行：

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

ds_ExampleDS 的值可以替换为使用嵌套表达式的系统属性(**datasource_name**)。数据源的配置可以改为有以下行：

```
<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。

基于描述符的特征替换
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。

基于描述符的属性替换基于描述符的属性，允许您从应用和构建链中删除对环境相关的假设。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。

ee 子系统中有几个标记控制是否应用属性替换。

JBoss 特定的描述符替换由 **jboss-descriptor-property-replacement** 标志控制，默认情况下被*启用*。启用后，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**

以下管理 CLI 命令可用于启用或禁用特定于 JBoss 的描述符中的属性替换：

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

Jakarta EE 描述符替换由 **spec-descriptor-property-replacement** 标志控制，默认为 *禁用*。启用后，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**

以下管理 CLI 命令可用于在 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)
```

1.4. 网络和端口配置 JBOSS EAP

JBoss EAP 附带接口、套接字绑定和 IPv6 地址，以帮助简化配置。使用以下关于这些网络和端口配置的详细信息，以成功运行 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 配置为使用逻辑名称来引用单独的接口声明，而无需在每个用途中接口的完整详情。

您还可以在受管域中更轻松地配置，网络接口详情可能因多个机器而异。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

如果您修改了 JBoss EAP 使用的默认网络接口或端口，您还必须记得更改使用修改后的接口或端口的任何脚本。其中包括 JBoss EAP 服务脚本，以及记得在访问管理控制台或管理控制台或 CLI 时指定正确的接口和端口。

其他资源

- 有关服务器启动选项的更多信息，请参阅 Server Runtime Arguments。

1.4.1.2. 可选接口配置

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具有的一个或多个特征集，才能成为有效的匹配项。

接口可以使用管理控制台或管理 CLI 进行配置。以下是添加和更新接口的几个示例。首先显示管理 CLI 命令，后跟对应的配置 XML。

其他资源

- 有关所有可用接口选择标准的列表，请参阅 Interface Attributes 部分。

1.4.1.2.1. 带有 NIC 值的接口

您可以使用以下示例添加带有 NIC 值 **eth0** 的新接口。

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

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

1.4.1.2.2. 带有多个条件值的接口

如果运行多播，您可以使用以下示例添加匹配任何接口或正确子网上的接口或地址的新接口。

```
/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. 对 interface 属性的更新

在本例中，您可以更新 公共接口 的默认 **inet-address** 值，保留 **jboss.bind.address** 属性，以便您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. 受管域中的服务器的其他接口

您可以使用以下代码为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

通过套接字绑定和套接字绑定组，您可以定义网络端口及其与 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。

这些指定配置的声明可以在 **standalone.xml** 和 **domain.xml** 配置文件中找到。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：

入站套接字绑定

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。另一个示例包括在为 JBoss EAP *配置消息传递*的广播组小节。

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。
此元素的属性可在 Local Outbound Socket Binding Attributes 表中找到。

其他资源

- 要查看入站套接字绑定的属性，请参阅 Inbound Socket Binding Attributes 表。
- 要查看远程出站套接字绑定的属性，请参阅 Remote Outbound Socket Binding Attributes 表。
- 有关远程出站套接字绑定的其他示例，请参阅 为 JBoss EAP *配置 Messaging* 的 Remote Connections 部分的 Integrated Artemis 资源适配器。
- 要查看本地出站套接字绑定的属性，请参阅 Local Outbound Socket Binding Attributes 表。

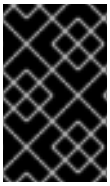
1.4.2.1. 管理端口

JBoss EAP 7 中整合了管理端口。默认情况下，JBoss EAP 7 将端口 **9990** 用于本地管理（由管理 CLI 使用）和 HTTP 管理（由基于 Web 的管理控制台使用）。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

如果您修改了 JBoss EAP 使用的默认网络接口或端口，您还必须记得更改使用修改后的接口或端口的任何脚本。其中包括 JBoss EAP 服务脚本，以及记得在访问管理控制台或管理控制台或 CLI 时指定正确的接口和端口。

其他资源

- 有关默认套接字绑定的详细信息，如默认端口和描述，请参阅 Default Socket Bindings 部分。

1.4.2.2.1. 独立服务器

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，默认的单机配置文件(**standalone.xml**)指定以下套接字绑定：

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

1.4.2.2.2. 受管域

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**。有关所有可用套接字绑定属性的详情，请查看 Socket Binding Attributes 部分。

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

当作为单机服务器运行时，无法执行此步骤。

```
/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>
```

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。这使得服务器能够继承其套接字绑定组中定义的端口值，并提供偏移以确保它不与同一主机上的任何其他服务器冲突。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**。

以下是使用管理 CLI 为受管域中的服务器设置端口偏移 **250** 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
```

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

要将启动配置更新为在 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$JAVA_OPTS" = "x" ]; then
   JAVA_OPTS="-Xms1303m -Xmx1303m -Djava.net.preferIPv4Stack=false"
   JAVA_OPTS="$JAVA_OPTS -
Djboss.modules.system.pkgs=$JBOSS_MODULES_SYSTEM_PKGS -Djava.awt.headless=true"
   JAVA_OPTS="$JAVA_OPTS -Djava.net.preferIPv6Addresses=true"
else
```

1.4.3.2. 默认接口值更新为 IPv6 地址

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 管理接口 设置为 IPv6 环回地址 (**::1**)。

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

运行此命令后，以下示例显示了 XML 配置的外观。

```
<interfaces>
```

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

1.5. 优化 JBOSS EAP 服务器配置

安装 JBoss EAP 服务器 后，并且创建了管理用户 后，红帽建议您优化服务器配置。

请查看*性能调优指南*中的信息，了解如何优化服务器配置，以避免在生产环境中部署应用程序时出现常见问题。常用的优化包括设置 ulimits, 启用垃圾收集, 创建 Java heap dumps, 以及调整线程池的大小。

最好为产品发布应用任何现有的补丁。EAP 的每个补丁都包含大量漏洞修复。如需更多信息，请参阅 JBoss EAP 的*补丁和升级指南*中的对 JBoss EAP 应用补丁。

第 2 章 使用 JBOSS EAP 开发应用程序

2.1. 概述

本指南提供有关使用红帽 CodeReady Studio 和 JBoss EAP 7 快速入门示例开始开发应用的信息。

红帽代码Ready Studio 是基于 Eclipse 的集成开发环境(IDE)，集成了 JBoss 应用程序开发插件。红帽 CodeReady Studio 可协助您的应用程序开发，帮助特定 JBoss 向导的可用性，以及将应用部署到 JBoss EAP 服务器的功能。JBoss EAP 7 提供了许多快速入门代码示例，可帮助用户使用不同的 Jakarta EE 技术开始编写应用。

2.2. 设置开发环境

1. 下载并安装 Red Hat CodeReady Studio。
   具体步骤，请参阅 Red Hat CodeReady Studio *安装指南*中的使用安装程序独立安装 CodeReady Studio。

2. 在红帽 CodeReady Studio 中设置 JBoss EAP 服务器。
   具体步骤，请参阅 *CodeReady Studio 工具入门指南* 中的通过 IDE 下载、安装和设置 JBoss EAP。

2.3. 使用 QUICKSTART 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。

JBoss EAP 包括一个 Maven 存储库，其中包含 Jakarta EE 开发人员通常用于在 JBoss EAP 上构建应用程序的许多要求。

有关如何在 JBoss EAP 中使用 Maven 的更多信息，请参阅 JBoss EAP *开发指南*中的 Maven 与 JBoss EAP 搭配使用。

2.3.2. 通过 Quickstarts 使用 Maven

构建应用程序并部署到 JBoss EAP 7 所需的构件和依赖关系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。

红帽 CodeReady Studio 包含 Maven，因此无需单独下载和安装。

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。

2.3.3. 下载并运行快速入门

2.3.3.1. 下载 Quickstarts

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。快速入门可从红帽客户门户下载。

1. 在红帽客户门户上登录到 JBoss EAP 下载页面。
2. 在 **Version** 下拉菜单中选择 **7.4**。
3. 在列表中找到 **Red Hat JBoss Enterprise Application Platform 7.4.0 Quickstarts**条目，然后单击 **Download** 以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. 提取 ZIP 文件。

2.3.3.2. 在 Red Hat CodeReady Studio 中运行 Quickstarts

下载 Quickstarts 后，即可将它们导入到红帽 CodeReady Studio 中并部署到 JBoss EAP。

将 Quickstart 导入到 Red Hat CodeReady Studio

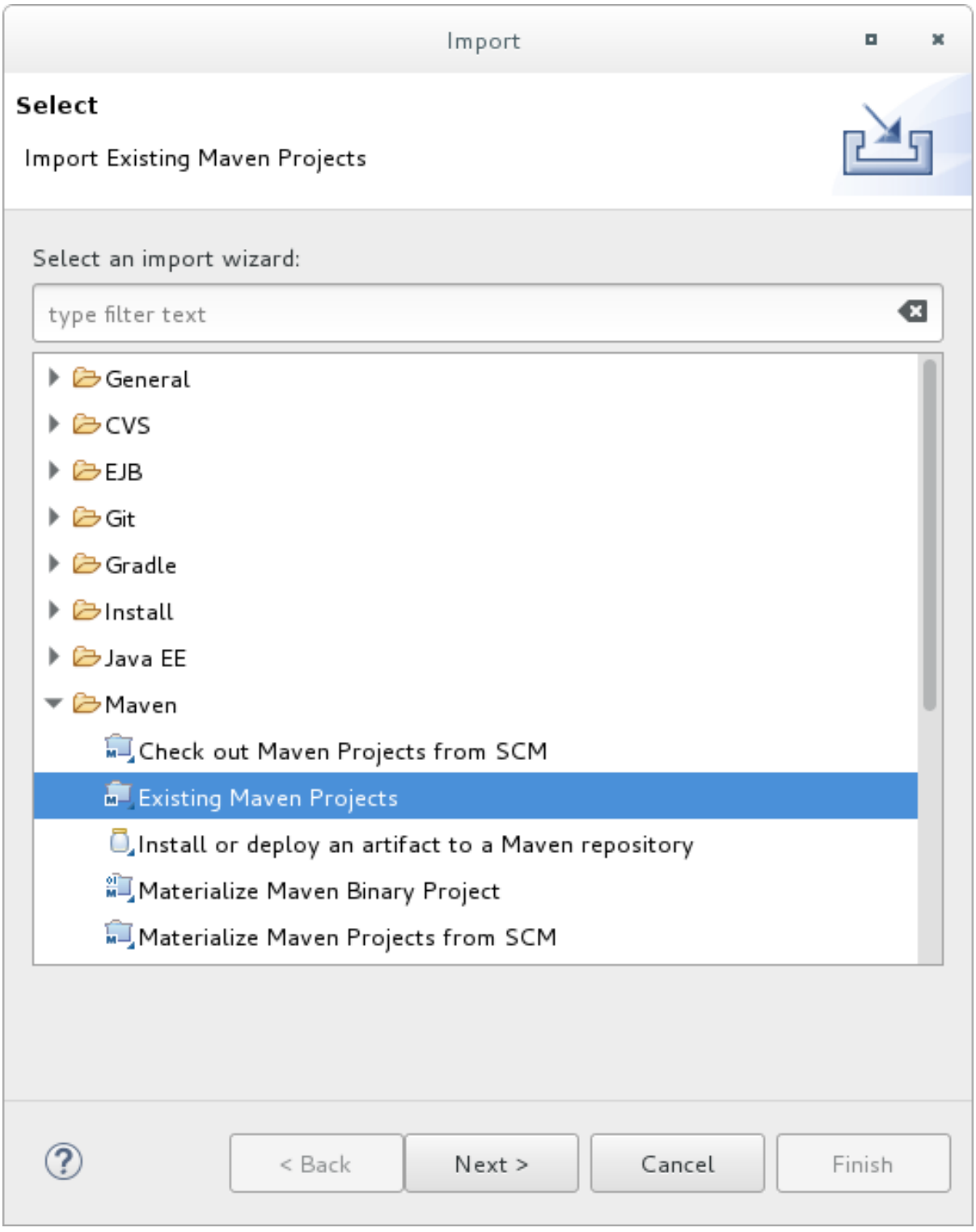
每个快速入门都附带了一个 POM 文件，其中包含其项目和配置信息。使用此 POM 文件，轻松将快速入门导入到红帽 CodeReady Studio。

重要

如果您将快速启动项目文件夹导入到 Red Hat CodeReady Studio 中，则 IDE 会生成无效的项目名称和 WAR 归档名称。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。

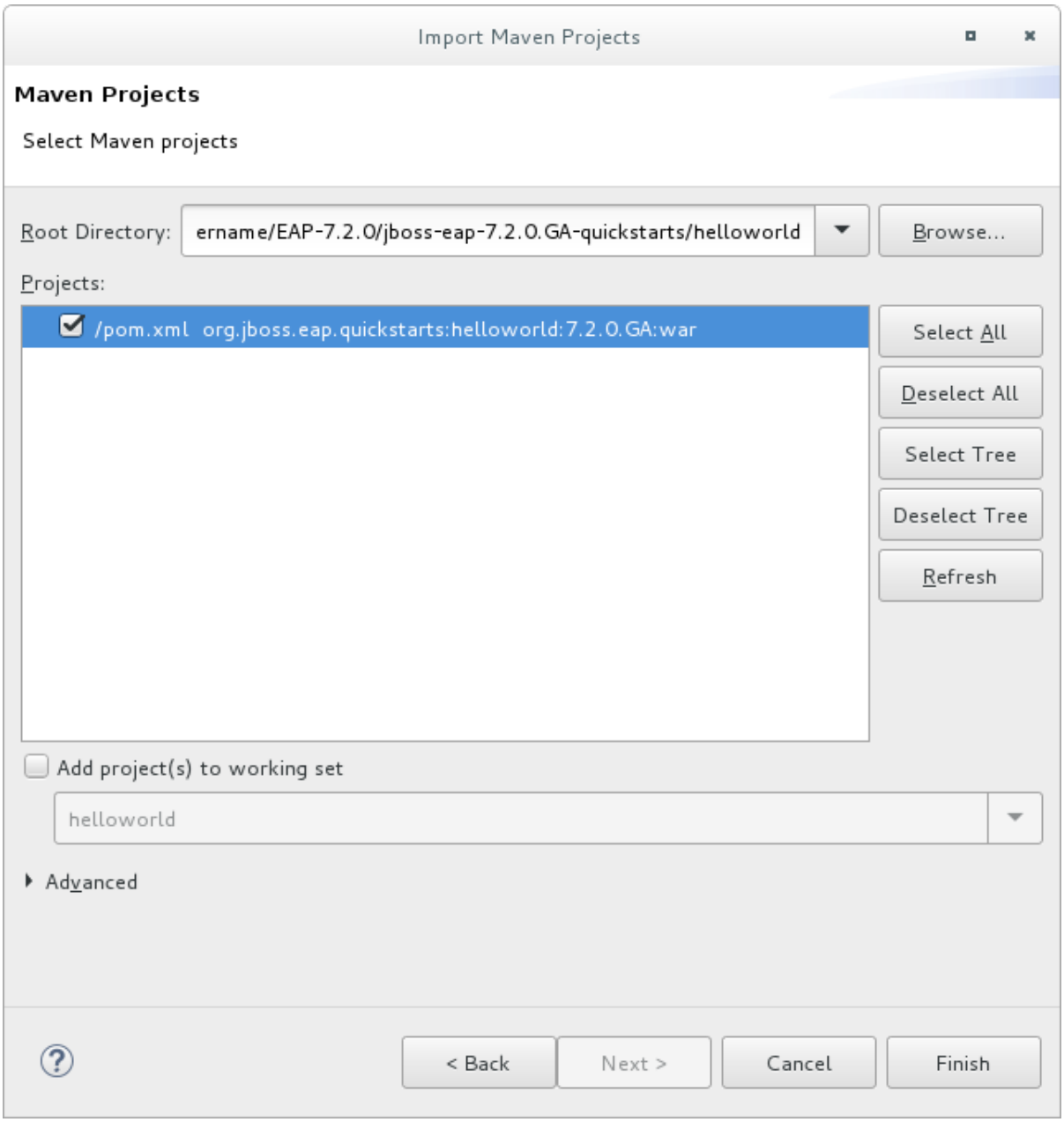
1. 启动 Red Hat CodeReady Studio。
2. 选择 **File → Import**。
3. 选择 **Maven → Existing Maven Projects**，然后点 **Next**。

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**。

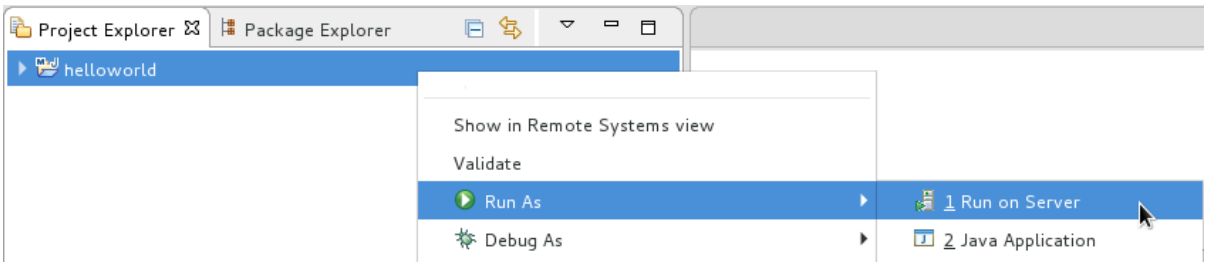
运行 *helloworld* Quickstart

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。

1. 如果您尚未定义服务器，请将 JBoss EAP 服务器添加到红帽 CodeReady Studio。具体步骤，请参阅 *CodeReady Studio 工具入门指南* 中的通过 IDE 下载、安装和设置 JBoss EAP 。

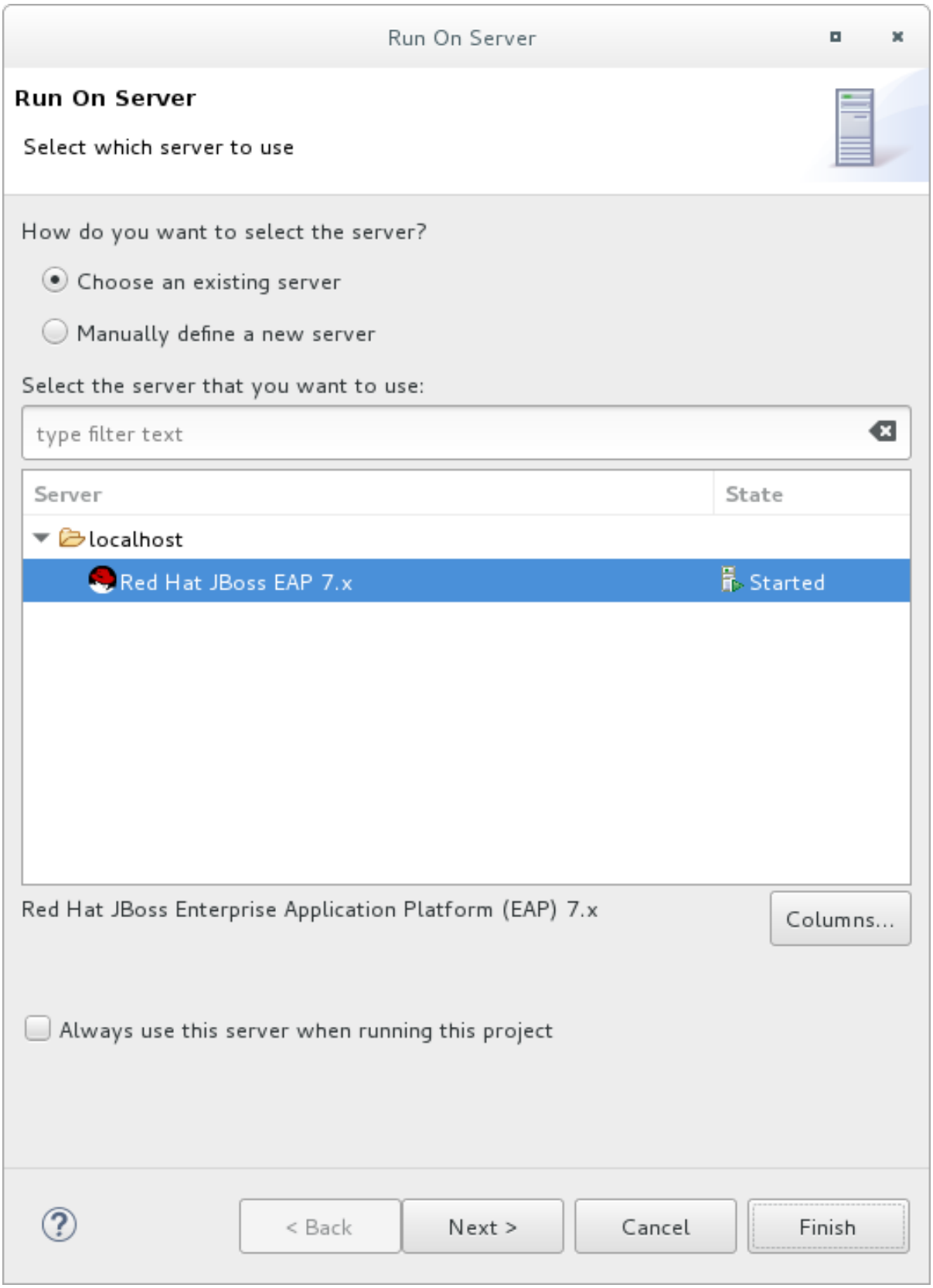
2. 右键点击 **Project Explorer** 选项卡中的 **helloworld** 项目，然后选择 **Run As → Run on Server**。

图 2.3. Run As - 在服务器中运行



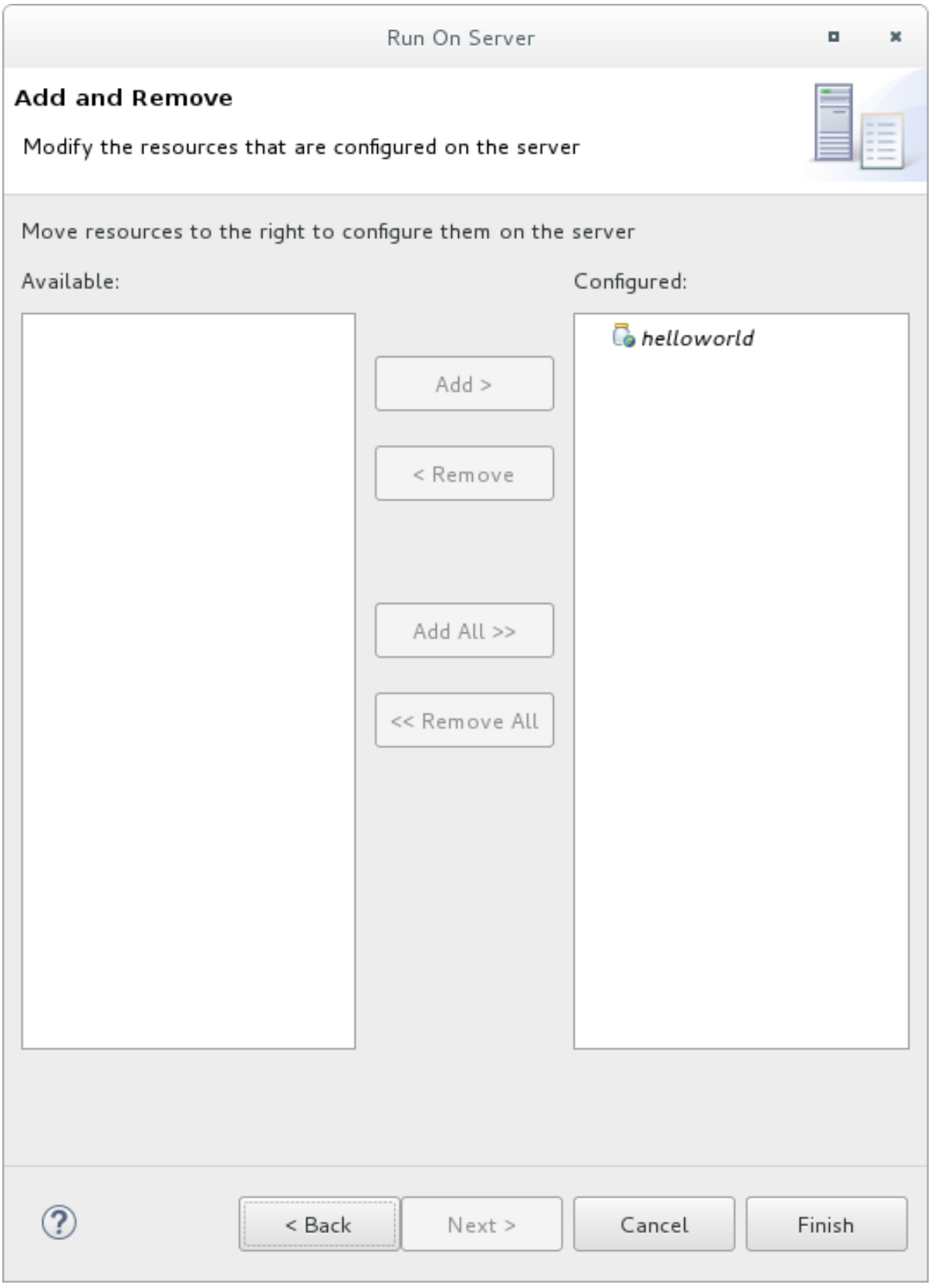
3. 从服务器列表中选择 JBoss EAP 7.4 服务器，然后点 **Next**。

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 7.4 服务器状态将更改为 **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!**。

有关 **helloworld** 快速入门的更多详细信息，请参阅 helloworld Quickstart。

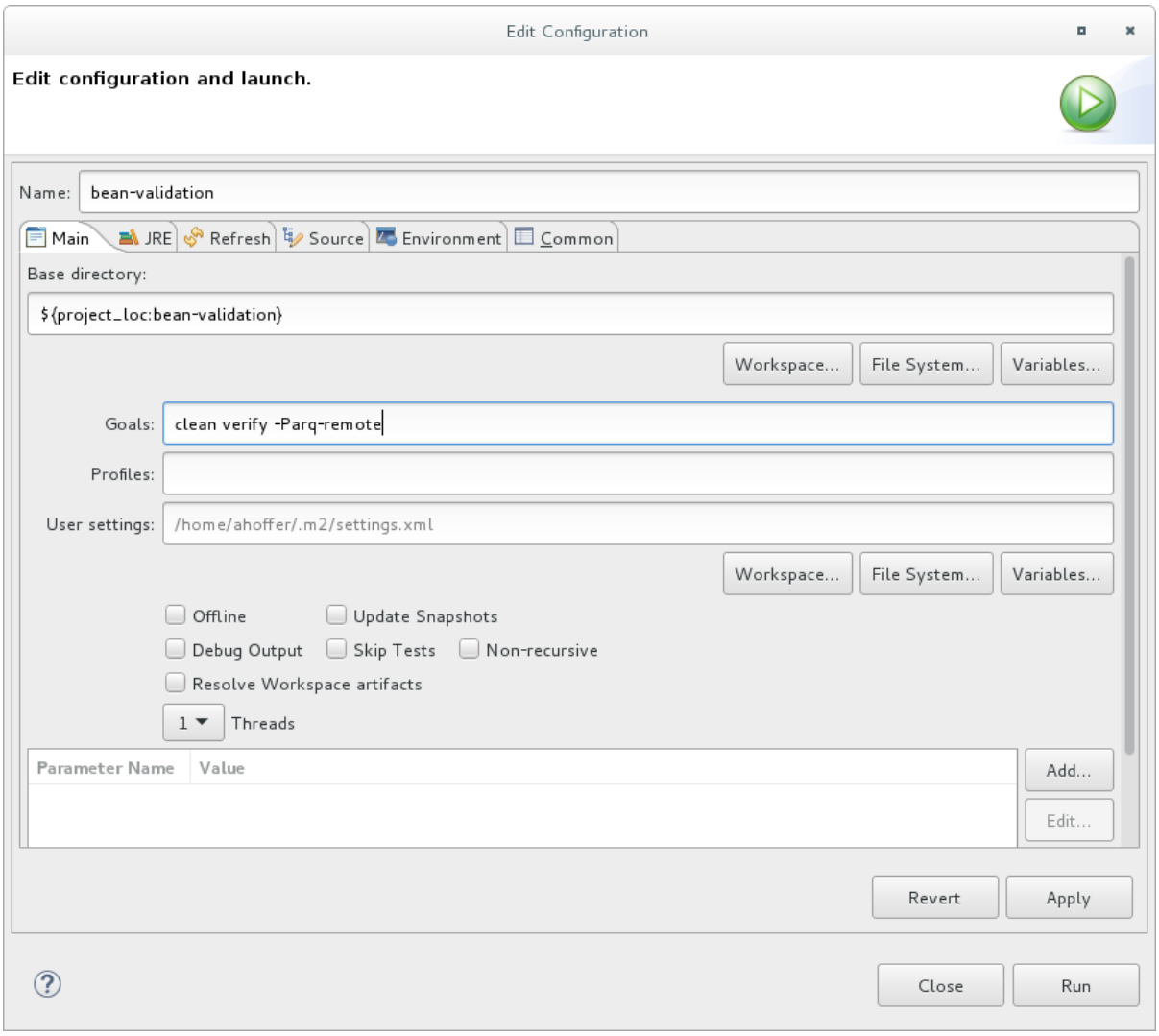
运行 *bean-validation* Quickstart

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。

1. 将 **bean-validation** 快速入门导入到红帽 CodeReady Studio。
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。如果您未看到 **Servers** 选项卡或尚未定义服务器，请将 JBoss EAP 服务器添加到红帽 CodeReady Studio。具体步骤，请参阅 *CodeReady Studio 工具入门指南* 中的通过 IDE 下载、安装和设置 JBoss EAP。
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] ------------------------------------------------------------------------
```

2.3.3.3. 从命令行运行 Quickstarts

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。

从命令行运行 ***helloworld*** **Quickstart**

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!**。

2.4. 查看 QUICKSTART 示例

2.4.1. 探索 helloworld Quickstart

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，该服务作为 Jakarta 上下文和依赖注入 Bjection Bjection 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。本节介绍如何使用 Red Hat CodeReady Studio 运行 Quickstart，并假设您已安装红帽 CodeReady Studio、配置 Maven，并导入并成功运行 **helloworld** quickstart。

先决条件

- 安装 Red Hat CodeReady Studio。具体步骤，请参阅 Red Hat CodeReady Studio *安装指南*中的使用安装程序独立安装 CodeReady Studio。
- 运行 **helloworld** 快速入门。具体步骤，请参阅在 Red Hat CodeReady Studio 中运行 Quickstarts。
- 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。

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** 快速入门由一个 Servlet 以及 Jakarta 上下文和依赖注入 Bjection Ban 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 在此应用中查找 bean，并激活 Jakarta 上下文和依赖注入。

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。

检查代码
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。

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 }
```

表 2.1. HelloWorldServlet Details

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 HelloService Jakarta Contexts 和 Dependency Injection bean，用于生成实际消息。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。

2. 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 + "!";
    }
}
```

2.4.2. 探索 numberguess Quickstart

numberguess quickstart 演示了如何将简单的非持久性应用创建和部署至 JBoss EAP。信息通过 Jakarta Server Faces 视图显示，业务逻辑则封装在两个 Jakarta Contexts 和 Dependency Injection 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** quickstart 由多个 Bean、配置文件和 Facelets Jakarta Server Faces 视图组成，并打包为一个 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 Red Hat CodeReady Studio 运行 Quickstart。

先决条件

- 安装 Red Hat CodeReady Studio。具体步骤，请参阅 Red Hat CodeReady Studio *安装指南*中的使用安装程序独立安装 CodeReady Studio。
- 运行 **numberguess** Quickstart。具体步骤请参阅 在 Red Hat CodeReady Studio 中运行 Quickstarts，并将 **helloworld** 替换为说明中的 **numbergues**。
- 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。

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

1. 检查 **face-config.xml** 文件。
   此快速入门使用 Jakarta Server Faces 2.2 版本的 **face-config.xml** 文件名。Facelet 的标准化版本是 Jakarta Server Faces 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**。此文件是一个标志文件，将 WAR 识别为 Bean 存档，并告知 JBoss EAP 在此应用中查找 bean，并激活 Jakarta 上下文和依赖注入。

```
<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。

2.4.2.1. 检查 Jakarta 服务器 Faces 代码

Jakarta Server Faces 将 **.xhtml** 文件扩展用于源文件，但通过 **.jsf** 扩展提供渲染的视图。**home.xhtml** 文件位于 **src/main/webapp/** 目录中。

示例：Jakarta Server Faces 源代码

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

```
55 <h:inputText id="inputGuess" value="#{game.guess}"
56  required="true" size="3"
57  disabled="#{game.number eq game.guess}"
58  validator="#{game.validateNumberRange}" />
59  <h:commandButton id="guessButton" value="Guess"
60   action="#{game.check}"
61   disabled="#{game.number eq game.guess}" />
62 </div>
63 <div>
64 <h:commandButton id="restartButton" value="Reset"
65 action="#{game.reset}" immediate="true" />
66 </div>
67 </h:form>
68
69 </div>
70
71 <br style="clear: both" />
72
73 </body>
74</html>
```

以下行号与在 Red Hat CodeReady Studio 中查看文件时看到的行号对应。

表 2.2. Jakarta Server Faces 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。

2.4.2.2. 检查类**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。

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。如需有关限定符的更多信息，请参阅 JBoss EAP *开发指南*中的使用限定符解决 Ambiguous Injection。**@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。只有在您想要使用 Jakarta Expression Language（本例中为 **#{game}**）使 bean 可访问 Jakarta Server Faces 视图时，才需要此注解。

```
@SuppressWarnings("serial")
@Named
@SessionScoped
public class Game implements Serializable {

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

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

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

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

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

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

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

   public Game() {
   }

   public int getNumber() {
      return number;
   }

   public int getGuess() {
      return guess;
   }

   public void setGuess(int guess) {
```

```
        this.guess = guess;
    }

    public int getSmallest() {
        return smallest;
    }

    public int getBiggest() {
        return biggest;
    }

    public int getRemainingGuesses() {
        return remainingGuesses;
    }

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

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

    /**
     * A Jakarta Server Faces validation method which checks whether the guess is valid. It
might not be valid because there are no guesses left,
     * or because the guess is not in range.
     *
     */
    public void validateNumberRange(FacesContext context, UIComponent toValidate, Object
value) {
```

```
        if (remainingGuesses <= 0) {
            FacesMessage message = new FacesMessage("No guesses left!");
            context.addMessage(toValidate.getClientId(context), message);
            ((UIInput) toValidate).setValid(false);
            return;
        }
        int input = (Integer) value;

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

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

附录 A. 开始使用 JBOSS EAP 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 单机服务器。这有助于配置服务器来满足您的需要。

A.1. 服务器运行时参数和交换机

在受管域中的单机服务器和服务器上，您可以将特定的服务器运行时参数与应用的启动脚本配合使用。脚本可以启动将替代配置用于 **standalone.xml**、**domain.xml** 和 **host.xml** 配置文件中定义的服务器。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 help switch、**-h** 或 **--help** 来访问可用的参数列表。

表 A.1. 运行时参数和交换机的描述：

参数或交换机	服务器类型	描述
--admin-only	独立模式	将服务器的运行类型设置为 **ADMIN_ONLY**。参数打开管理界面并接受管理请求，但 参数不会启动其他运行时服务或接受用户请求。要获得最佳性能，请使用 **--start-mode=admin-only** 参数。
--admin-only	域	将主机控制器的运行类型设置为 **ADMIN_ONLY** 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，它将接受来自从属主机控制器的传入连接。
-b=<value>, -b <value>	独立，域	设置系统属性 **jboss.bind.address**，您可以使用它来为公共接口配置绑定地址。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>	独立，域	将系统属性 **jboss.bind.address.<interface>** 设置为给定值。例如：**-bmanagement=*IP_ADDRESS***。
--backup	域	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>	独立模式	要使用的服务器配置文件的名称。默认值为 **standalone.xml**。
-c=<config>, -c <config>	域	要使用的服务器配置文件的名称。默认值为 **domain.xml**。
--cached-dc	域	如果主机不是域控制器，并且无法在引导时联系域控制器，则必须使用域配置的本地缓存副本引导。
--debug [<port>]	独立模式	使用可选参数激活调试模式以指定端口。只有启动脚本支持 Argument 时才有效参数。

参数或交换机	服务器类型	描述
-D<name>[=<value>]	独立，域	设置系统属性。
--domain-config=<config>	域	要使用的服务器配置文件的名称。默认为 **domain.xml**。
--git-repo	独立模式	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	独立模式	Git 存储库中要使用的分支或标签名称。此参数应命名现有的分支或标签名称，因为如果不存在，则*不会*创建该分支或标签名称。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	独立模式	到 Elytron 配置文件的 URL，该文件包含服务器在连接到远程 Git 存储库时使用的凭据。当远程 Git 存储库需要身份验证时，您可以使用 参数。Elytron 不支持 SSH。Elytron 只支持使用不使用密码的私钥的默认 SSH 身份验证。您不能将 参数与 本地存储库搭配使用。
-h, --help	独立，域	显示帮助信息并退出 help 索引。
--host-config=<config>	域	要使用的主机配置文件的名称。默认为 **host.xml**。
--interprocess-hc-address=<address>	域	主机控制器可以监听进程控制器的通信的地址。
--interprocess-hc-port=<port>	域	主机控制器可以监听进程控制器的通信的端口。
--master-address=<address>	域	将系统属性 **jboss.domain.master.address** 设置为给定的值。在默认的从属主机控制器配置中，您可以使用 参数配置 master 主机控制器的地址。
--master-port=<port>	域	将系统属性 **jboss.domain.master.port** 设置为给定的值。在默认的从属主机控制器配置中，您可以使用 参数配置供 master 主机控制器进行原生管理通信的端口。
--read-only-server-config=<config>	独立模式	要使用的服务器配置文件的名称。参数与 **--server-config** 和 **-c** 不同，这些参数不会覆盖原始文件。
--read-only-domain-config=<config>	域	要使用的域配置文件的名称。参数与 **--domain-config** 和 **-c** 不同，该参数不会覆盖初始文件。

参数或交换机	服务器类型	描述
--read-only-host-config=<config>	域	要使用的主机配置文件的名称。参数与 **--host-config** 不同，该参数不会覆盖初始文件。
-P=<url>, -P <url>, --properties=<url>	独立，域	从指定的 URL 加载系统属性。
--pc-address=<address>	域	进程控制器侦听来自其控制的进程的通信的地址。
--pc-port=<port>	域	进程控制器在其上侦听其控制进程的通信的端口。
-S<name>[=<value>]	独立模式	设置安全属性。
-secmgr	独立，域	运行安装有安全管理器的服务器。
--server-config=<config>	独立模式	要使用的服务器配置文件的名称。默认为 **standalone.xml**。
--start-mode=<mode>	独立模式	设置服务器的启动模式。您不能将参数用于 **--admin-only** 参数。您可以将以下条目与参数一起使用： • **Normal** ：服务器正常启动。 • 仅 **admin-only** ：服务器只能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。 • 暂停 ：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>	独立，域	设置系统属性 **jboss.default.multicast.address**，服务器在配置文件的 socket-binding 元素中配置多播地址。默认为 **230.0.0.4**。
-v, -V, --version	独立，域	显示应用服务器版本并退出。

警告

JBoss EAP 设置其所含配置文件，以处理交换机的行为。例如： **-b** 和 **-u**。如果您将配置文件更改为不再使用由交换机控制的系统属性，那么在 start 命令中添加系统属性将无法正常工作。

A.2. ADD-USER 参数

您可以将参数与 **add-user.sh** 脚本或 **add-user.bat** 脚本搭配使用，以配置这些节新用户如何将新用户添加到属性文件中以进行身份验证。

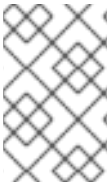
表 A.2. add-user 参数的描述

命令行参数	描述
-a	在应用域中创建用户。如果您没有在应用域中创建用户，则该脚本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/** 设置为默认 目录。
-sc <value>	另一种包含属性文件的单机服务器配置目录。如果省略该参数，则脚本会将 ***EAP_HOME*/standalone/configuration/** 设置为默认 目录。
-up, --user-properties <value>	备用用户属性文件的名称。您可以使用带有 **-sc** 或 **-dc** 参数来设置其他配置目录的参数，为文件设置绝对路径，或者指定文件名。
-g, --group <value>	要分配给用户的组的逗号分隔列表。
-gp, --group-properties <value>	备用组属性文件的名称。您可以使用带有 **-sc** 或 **-dc** 参数来设置其他配置目录的参数，为文件设置绝对路径，或者指定文件名。
-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**。
-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。

命令行参数	描述
-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 界面。



注意

表中的属性名称以 JBoss EAP 的顺序将它们列在其管理模式中。请参阅位于 ***EAP_HOME*/docs/schema/wildfly-config_5_0.xsd** 中的架构定义文件，以查看它们出现在 XML 中的元素。XML 元素列表必须与管理模型中显示的不同。

表 A.3. 接口属性描述：

Interface 属性	描述
any	指定接口必须至少满足一个，但不一定都是所选嵌套标准集合。
any-address	将通配符地址绑定到使用接口的套接字的空属性。该属性具有以下配置选项： ● 属性使用 IPv6 通配符地址(**::**)作为默认值。如果将 **java.net.preferIPv4Stack** 系统属性设置为 **true**，则套接字将使用 IPv4 通配符地址(**0.0.0.0**)。 ● 如果套接字绑定到双栈机器上的任何本地地址，则套接字接受 IPv6 和 IPv4 流量。 ● 如果套接字绑定到 IPv4(IPv4-mapped)任何本地地址，则套接字只接受 IPv4 流量。
inet-address	指定 IPv6 或 IPv4 十进制表示法中的 IP 地址，或者指定解析为 IP 地址的主机名。
link-local-address	空属性指定接口是否包括本地链路地址的条件。
loopback	空属性指定接口是否识别为回环接口的条件。
loopback-address	在计算机的回环接口上可能尚未配置回环地址。属性与 **inet-address** 类型不同，因为接口使用属性值，即使值包含没有 IP 地址的 NIC。
multicast	空属性指定接口是否支持多播条件。
name	接口的名称。
nic	网络接口的名称，如 **eth**0、**eth1** 或 **lo**。

Interface 属性	描述
nic-match	将计算机上可用的网络接口名称与可接受的接口匹配的正则表达式。
not	指示接口不能满足的选择条件的属性。
point-to-point	空属性指定接口是否被识别为点对点接口的条件。
public-address	空属性指定接口是否包含可公开路由的地址的条件。
site-local-address	空属性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，*以斜杠表示法* 写入，如 **192.168.0.0/16**。
up	空属性指定接口是否定位为 up 的条件。
virtual	空属性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。

存在以下类型的套接字绑定的具体属性：

- 入站套接字绑定
- 远程出站套接字绑定
- 本地出站套接字绑定

注意

表中的属性名称以 JBoss EAP 的顺序将它们列在其管理模式中。请参阅位于 ***EAP_HOME*/docs/schema/wildfly-config_5_0.xsd** 中的架构定义文件，以查看它们出现在 XML 中的元素。XML 元素列表必须与管理模型中显示的不同。

表 A.4. 描述入站套接字绑定、套接字绑定、 属性：

属性	描述
client-mappings	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的地地址。这个地址与出站接口匹配。通过将 client-mapping 属性与入站套接字绑定搭配使用，您可以应用使用网络地址转换或在多个网络接口上绑定的高级网络拓扑。您必须以声明的顺序评估每个映射，即第一个成功匹配来确定映射的目标。
fixed-port	使用 属性来确定端口值是否必须保持固定。即使将数字偏移应用到套接字组中的其他套接字，您可以使用 属性。

属性	描述
interface	用于设置套接字绑定到的接口的名称。您还可以使用 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，则该属性将使用所属套接字绑定组中的 **default-interface** 值。
multicast-address	套接字接收多播流量的多播地址。如果您没有为属性指定值，则不会配置套接字来接收多播功能。
multicast-port	套接字接收多播流量的端口。如果您配置了 **multicast-address** 属性，您必须配置 attrbiute。
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	套接字绑定的端口数量。如果您配置了服务器，以将 port-offset 应用到递增或减少所有端口值，您必须覆盖属性值。

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

属性	描述
fixed-source-port	确定端口值是否必须保留固定，即使您已将数字偏移应用到套接字组中的其他出站套接字。
host	出站套接字连接到的远程目的地的主机名或 IP 地址。
port	出站套接字连接到的远程目的地的端口号。
source-interface	JBoss EAP 用于出站套接字源地址的接口的名称。
source-port	JBoss EAP 的端口号将用作出站套接字的源端口。

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

属性	描述
fixed-source-port	确定端口值是否必须保留固定，即使您已将数字偏移应用到套接字组中的其他出站套接字。
socket-binding-ref	JBoss EAP 用来确定连接出站套接字的端口的本地套接字绑定名称。
source-interface	JBoss EAP 用于出站套接字源地址的接口的名称。
source-port	JBoss EAP 的端口号将用作出站套接字的源端口。

A.5. 默认套接字绑定

您可以为每个套接字绑定组设置默认套接字绑定。

JBoss EAP 有以下五种默认套接字绑定：

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

表 A.7. 默认 标准套接字套接字绑定 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。
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。

套接字绑定	端口	多播端口	描述
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 全套接字套接字绑定 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。
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. 默认负载均衡器 **-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。
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。

Name	端口	多播端口	描述
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。

更新于 2024-06-13