



Red Hat Fuse 7.13

在 JBoss EAP 中使用 Fuse

在 EAP 上开始使用红帽 Fuse

Red Hat Fuse 7.13 在 JBoss EAP 中使用 Fuse

在 EAP 上开始使用红帽 Fuse

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

开始在 JBoss 企业应用平台上使用 Fuse 构建应用程序。

目录

前言	3
使开源包含更多	4
第 1 章 在 JBOSS EAP 上使用 FUSE	5
1.1. 关于 JBOSS EAP 上的 FUSE	5
1.2. 在 JBOSS EAP 上安装 FUSE	5
1.3. 在 JBOSS EAP 上构建第一个 FUSE 应用程序	6
第 2 章 在本地设置 MAVEN	8
2.1. 准备设置 MAVEN	8
2.2. 将红帽软件仓库添加到 MAVEN	8
2.3. 使用本地 MAVEN 存储库	10
2.4. 使用环境变量或系统属性设置 MAVEN 镜像	10
2.5. 关于 MAVEN 工件和协调	11

前言

要开始使用 Fuse，您需要下载并安装 JBoss EAP 容器的文件。此处的信息和说明指导您安装、开发和构建第一个 Fuse 应用程序。

- [第 1 章 在 JBoss EAP 上使用 Fuse](#)
- [第 2 章 在本地设置 Maven](#)

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。这些更改将在即将发行的几个发行本中逐渐实施。详情请查看我们的 [CTO Chris Wright 信息](#)。

第1章 在 JBOSS EAP 上使用 FUSE

本章介绍了 JBoss EAP 上的 Fuse，并解释了如何在 JBoss EAP 容器上安装、开发和构建第一个 Fuse 应用程序。

详情请查看以下主题：

- [第 1.1 节 “关于 JBoss EAP 上的 Fuse”](#)
- [第 1.2 节 “在 JBoss EAP 上安装 Fuse”](#)
- [第 1.3 节 “在 JBoss EAP 上构建第一个 Fuse 应用程序”](#)

1.1. 关于 JBOSS EAP 上的 FUSE

JBoss Enterprise Application Platform (EAP) 基于 [Eclipse Foundation](#) 中的 [Jakarta EE](#) 技术（之前是 Java EE），最初创建为开发企业应用程序的用例而创建。JBoss EAP 的特征是实施服务和标准化的 Java API（例如，持久性、消息传递、安全性等）的定义模式。近年来，此技术已演变为更加轻量级，为企业 Java Bean 的依赖性注入和简化的注解推出 CDI。

此容器技术的主要特性是：

- 特别适用于在独立模式运行。
- 许多标准服务（如持久性、消息传递、安全性等）预先配置并提供开箱即用。
- 应用程序 WAR 通常比较小且轻量级（因为容器中预装了许多依赖项）。
- 标准化、向后兼容的 Java API。

1.2. 在 JBOSS EAP 上安装 FUSE

JBoss EAP 上 Fuse 7.13 的标准安装软件包可从红帽客户门户下载。它安装 JBoss EAP 容器的标准装配，并提供完整的 Fuse 技术堆栈。

先决条件

- 您必须 [在红帽客户门户网站中](#) 有一个完整的订阅帐户。
- 您必须登录到客户门户网站。
- 您必须已下载 [JBoss EAP](#)。
- 您必须在 [JBoss EAP](#) 上下载 Fuse。
- 您必须在 [JBoss EAP Update 16](#) 上下载 Fuse。

流程

1. 从 shell 提示符运行 JBoss EAP 安装程序，如下所示：

```
java -jar DOWNLOAD_LOCATION/jboss-eap-7.4.16-installer.jar
```

2. 在安装过程中：

- a. 接受条款和条件。
 - b. 为 JBoss EAP 运行时选择您首选的安装路径 **EAP_INSTALL**。
 - c. 创建管理用户，并小心记录这些管理用户凭证。
 - d. 您可以在剩余的屏幕上接受默认设置。
3. 打开 shell 提示符，并将目录更改为 **EAP_INSTALL**。
 4. 在 **EAP_INSTALL** 目录中，在 EAP 安装程序上运行 Fuse，如下所示：

```
java -jar DOWNLOAD_LOCATION/fuse-eap-installer-7.13.0.jar
```

5. (可选) 为了从命令行使用 Apache Maven，您需要安装并配置 Maven，如在 [本地设置 Maven](#) 所述。
6. 在 JBoss EAP Update 16 补丁上应用 Fuse。
具体步骤请查看 [Red Hat JBoss EAP 补丁和升级指南](#)。

1.3. 在 JBOSS EAP 上构建第一个 FUSE 应用程序

这组说明可帮助您在 JBoss EAP 上构建第一个 Fuse 应用程序。

先决条件

- 在 [红帽客户门户网站](#) 中 您需要一个全订阅帐户。
- 您必须登录到客户门户网站。
- 您必须在 [JBoss EAP](#) 上下载并成功安装了 Fuse。
- 您必须已下载并成功安装了 [Joss Tools 安装程序](#)。

流程

1. 在 IDE 环境中，创建一个新项目，如下所示：
 - a. 选择 **File→New→Fuse Integration Project**。
 - b. 在 **Project Name** 字段中，输入 **eap-camel**。
 - c. 点击 **Next**。
 - d. 在 **Select a Target Environment** 窗格中选择以下设置：
 - 选择 **Standalone** 作为部署平台。
 - 选择 **EAP 上的 Wildfly/Fuse** 作为运行时环境，并使用 **Runtime (可选)** 下拉菜单选择 **JBoss EAP 7.x Runtime** 服务器作为目标运行时。
 - e. 选择目标运行时后，会自动为您选择 **Camel Version**，字段将灰掉。
 - f. 点击 **Next**。
 - g. 在 **Advanced Project Setup** 窗格中，选择 **Spring Bean - Spring DSL** 模板。

h. 点 **Finish**。



重要

如果您是第一次构建 Fuse 项目，向导将 *需要几分钟时间* 才能完成生成项目。这是因为它从远程 Maven 存储库下载依赖项。不要在项目在后台构建时中断向导或关闭窗口。

i. 如果系统提示您打开关联的 Fuse 集成透视图，请单击 **Yes**。

j. 等待 JBoss 工具下载所需的工件并在后台构建项目。

2. 将项目部署到服务器，如下所示：

a. 在 **Servers** 视图(Fuse Integration 视角的左下角)中，如果服务器尚未启动，请选择 **Red Hat JBoss EAP 7.4 Runtime** 服务器，然后单击绿色箭头以启动它。

b. 等待 **Console** 视图中看到类似如下的消息：

```
14:47:07,283 INFO [org.jboss.as] (Controller Boot Thread) WFLYSRV0025: JBoss EAP 7.4.0.GA (WildFly Core 10.1.11.Final-redhat-00001) started in 3301ms - Started 314 of 576 services (369 services are lazy, passive or on-demand)
```

c. 服务器启动后，切回到 **Servers** 视图，右键单击服务器，然后从上下文菜单中选择 **Add and Remove**。

d. 在 **Add and Remove** 对话框中，选择 **eap-camel** 项目并点 **Add >**。

e. 点 **Finish**。

3. 验证项目是否正常工作，如下所示：

a. 浏览到以下 URL 以访问 **eap-camel** 项目中运行的服务：<http://localhost:8080/camel-test-spring?name=Kermit>

b. 浏览器窗口应当显示响应 **Hello Kermit**。

4. 取消部署项目，如下所示：

a. 在 **Servers** 视图中，选择 **Red Hat JBoss EAP 7.4 Runtime** 服务器。

b. 右键单击服务器，然后从上下文菜单中选择 **Add and Remove**。

c. 在 **Add and Remove** 对话框中，选择 **eap-camel** 项目，再点 **< Remove**。

d. 点 **Finish**。

第 2 章 在本地设置 MAVEN

典型的 Fuse 应用程序开发使用 Maven 来构建和管理项目。

以下主题描述了如何在本地设置 Maven：

- [第 2.1 节 “准备设置 Maven”](#)
- [第 2.2 节 “将红帽软件仓库添加到 Maven”](#)
- [第 2.3 节 “使用本地 Maven 存储库”](#)
- [第 2.4 节 “使用环境变量或系统属性设置 Maven 镜像”](#)
- [第 2.5 节 “关于 Maven 工件和协调”](#)

2.1. 准备设置 MAVEN

Maven 是一个来自 Apache 的免费开源构建工具。通常，您可以使用 Maven 构建 Fuse 应用程序。

流程

1. 从 [Maven 下载页面](#) 下载最新版本的 Maven。
2. 确定您的系统已连接到互联网。
在构建项目时，默认行为是 Maven 搜索外部存储库并下载所需的工件。Maven 查找可通过互联网访问的存储库。

您可以更改此行为，以便 Maven 仅搜索位于本地网络上的存储库。也就是说，Maven 可以在离线模式下运行。在离线模式中，Maven 会在其本地存储库中查找工件。请参阅 [第 2.3 节 “使用本地 Maven 存储库”](#)。

2.2. 将红帽软件仓库添加到 MAVEN

要访问位于 Red Hat Maven 存储库中的工件，您需要将这些存储库添加到 Maven 的 **settings.xml** 文件中。Maven 在用户主目录的 **.m2** 目录中查找 **settings.xml** 文件。如果没有用户指定的 **settings.xml** 文件，Maven 将使用 **M2_HOME/conf/settings.xml** 中的系统级 settings.xml 文件。

前提条件

您知道要在其中添加红帽软件仓库的 **settings.xml** 文件的位置。

流程

在 **settings.xml** 文件中，为红帽 软件仓库 添加软件仓库元素，如下例所示：

```
<?xml version="1.0"?>
<settings>

<profiles>
  <profile>
    <id>extra-repos</id>
    <activation>
      <activeByDefault>true</activeByDefault>
    </activation>
```

```
<repositories>
<repository>
  <id>redhat-ga-repository</id>
  <url>https://maven.repository.redhat.com/ga</url>
  <releases>
    <enabled>true</enabled>
  </releases>
  <snapshots>
    <enabled>>false</enabled>
  </snapshots>
</repository>
<repository>
  <id>redhat-ea-repository</id>
  <url>https://maven.repository.redhat.com/earlyaccess/all</url>
  <releases>
    <enabled>true</enabled>
  </releases>
  <snapshots>
    <enabled>>false</enabled>
  </snapshots>
</repository>
<repository>
  <id>jboss-public</id>
  <name>JBoss Public Repository Group</name>
  <url>https://repository.jboss.org/nexus/content/groups/public/</url>
</repository>
</repositories>
<pluginRepositories>
<pluginRepository>
  <id>redhat-ga-repository</id>
  <url>https://maven.repository.redhat.com/ga</url>
  <releases>
    <enabled>true</enabled>
  </releases>
  <snapshots>
    <enabled>>false</enabled>
  </snapshots>
</pluginRepository>
<pluginRepository>
  <id>redhat-ea-repository</id>
  <url>https://maven.repository.redhat.com/earlyaccess/all</url>
  <releases>
    <enabled>true</enabled>
  </releases>
  <snapshots>
    <enabled>>false</enabled>
  </snapshots>
</pluginRepository>
<pluginRepository>
  <id>jboss-public</id>
  <name>JBoss Public Repository Group</name>
  <url>https://repository.jboss.org/nexus/content/groups/public</url>
</pluginRepository>
</pluginRepositories>
</profile>
</profiles>
```

```

<activeProfiles>
  <activeProfile>extra-repos</activeProfile>
</activeProfiles>

</settings>

```

2.3. 使用本地 MAVEN 存储库

如果您在没有互联网连接的情况下运行容器，并且需要部署一个具有离线依赖项的应用程序，您可以使用 Maven 依赖项插件将应用的依赖项下载到 Maven 离线存储库中。然后，您可以将此自定义 Maven 离线存储库分发到没有互联网连接的机器。

流程

1. 在包含 **pom.xml** 文件的项目目录中，运行以下命令来下载 Maven 项目的存储库，如下所示：

```

mvn org.apache.maven.plugins:maven-dependency-plugin:3.1.0:go-offline -
Dmaven.repo.local=/tmp/my-project

```

在本例中，构建项目所需的 Maven 依赖项和插件将下载到 **/tmp/my-project** 目录中。

2. 将此自定义 Maven 离线存储库在内部分发到没有互联网连接的任何机器。

2.4. 使用环境变量或系统属性设置 MAVEN 镜像

在运行应用程序时，您需要访问 Red Hat Maven 存储库中的工件。这些存储库添加到 Maven 的 **settings.xml** 文件中。Maven 检查 **settings.xml** 文件的以下位置：

- 查找指定的 url
- if not found looks for **\${user.home}/.m2/settings.xml**
- 如果没有找到 **\${maven.home}/conf/settings.xml** 的查找
- 如果未找到，则查找 **\${M2_HOME}/conf/settings.xml**
- 如果没有找到位置，则会创建空的 **org.apache.maven.settings.Settings** 实例。

2.4.1. 关于 Maven 镜像

Maven 使用一组远程存储库来访问工件，这些工件目前在本地存储库中不可用。存储库列表几乎始终包含 Maven Central 存储库，但对于 Red Hat Fuse，它还包含 Maven 红帽存储库。在某些情况下，无法访问不同的远程存储库，您可以使用 Maven 镜像的机制。镜像将特定的存储库 URL 替换为不同的存储库 URL，因此搜索远程工件时的所有 HTTP 流量都可以定向到单个 URL。

2.4.2. 在 settings.xml 中添加 Maven 镜像

要设置 Maven 镜像，请在 Maven 的 **settings.xml** 中添加以下部分：

```

<mirror>
  <id>all</id>
  <mirrorOf>*</mirrorOf>

```

```
<url>http://host:port/path</url>
</mirror>
```

如果在 **settings.xml** 文件中找不到以上部分，则不会使用 mirror。要在不提供 XML 配置的情况下指定全局镜像，您可以使用系统属性或环境变量。

2.4.3. 使用环境变量或系统属性设置 Maven 镜像

要使用环境变量或系统属性设置 Maven 镜像，您可以添加：

- 名为 **MAVEN_MIRROR_URL** 的环境变量到 **bin/setenv** 文件
- 名为 **mavenMirrorUrl** 到 **etc/system.properties** 文件的系统属性

2.4.4. 使用 Maven 选项指定 Maven 镜像 url

要使用替代的 Maven 镜像 url，在环境变量或系统属性之外，在运行应用程序时使用以下 maven 选项：

- **-DmavenMirrorUrl=mirrorId::mirrorUrl**
for example, **-DmavenMirrorUrl=my-mirror::http://mirror.net/repository**
- **-DmavenMirrorUrl=mirrorUrl**
例如, **-DmavenMirrorUrl=http://mirror.net/repository**。在本例中，<mirror> 的 <id> 只是一个镜像(mirror)。

2.5. 关于 MAVEN 工件和协调

在 Maven 构建系统中，基本构建块是一个 *工件*。构建后，工件的输出通常是一个存档，如 JAR 或 WAR 文件。

Maven 的一个关键方面是能够找到工件和管理它们之间的依赖关系。*Maven 协调*是一组用于标识特定工件位置的值。基本协调有三个值，格式为：

groupId:artifactId:version

有时，Maven 通过 *打包值*或使用*打包*值和 *分类器*值增加一个基本的协调。*Maven 协调*可以具有以下形式之一：

```
groupId:artifactId:version
groupId:artifactId:packaging:version
groupId:artifactId:packaging:classifier:version
```

以下是值的描述：

groupId

定义工件名称的范围。您通常使用所有或部分软件包名称作为组 ID。例如，**org.fusesource.example**。

artifactId

定义相对于组 ID 的工件名称。

version

指定工件的版本。版本号最多可有四个部分：**n.n.n.n**，其中版本号的最后一部分可以包含非数字字符。例如，**1.0-SNAPSHOT** 的最后一部分是字母数字字符串 **0-SNAPSHOT**。

打包

定义构建项目时生成的打包实体。对于 OSGi 项目，打包是 **捆绑包**。默认值为 **jar**。

分类器

可让您区分从同一 POM 构建但具有不同内容的工件。

工件的 POM 文件中的元素定义工件的组 ID、工件 ID、打包和版本，如下所示：

```
<project ... >
...
<groupId>org.fusesource.example</groupId>
<artifactId>bundle-demo</artifactId>
<packaging>bundle</packaging>
<version>1.0-SNAPSHOT</version>
...
</project>
```

要定义对上述工件的依赖项，您可以在 POM 文件中添加以下 `dependencies` 元素：

```
<project ... >
...
<dependencies>
<dependency>
  <groupId>org.fusesource.example</groupId>
  <artifactId>bundle-demo</artifactId>
  <version>1.0-SNAPSHOT</version>
</dependency>
</dependencies>
...
</project>
```



注意

不需要在前面的依赖项中指定 **bundle** 软件包类型，因为捆绑包只是特定类型的 JAR 文件，**jar** 是默认的 Maven 软件包类型。但是，如果您需要在依赖项中明确指定打包类型，您可以使用 `type` 元素。