Red Hat Fuse 7.13

Spring Boot 中使用 Fuse

在 Spring Boot 中使用 Red Hat Fuse

Last Updated: 2024-06-12

Red Hat Fuse 7.13 Spring Boot 中使用 Fuse

在 Spring Boot 中使用 Red Hat Fuse

法律通告

摘要

在 Spring Boot 中使用 Fuse。

目录

前言

要开始使用 Fuse，您需要下载并安装 Spring Boot 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。

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 SPRING BOOT 中使用 FUSE

要在 Spring Boot 上开发 Fuse 应用程序，请首先生成和构建在 Spring Boot 上运行的 Fuse 示例提升项目。以下主题提供详情：

- 第 1.1 节 “关于 Spring Boot 上的 Fuse”
- 第 1.2 节 “生成您的 booster 项目”
- 第 1.3 节 “构建您的 booster 项目”

1.1. 关于 SPRING BOOT 上的 FUSE

Spring Boot 是知名 Spring 容器的一个演进。Spring Boot 容器的一个独特的质量是容器功能被分成几个小块，可以独立部署。通过此质量，您可以部署资源占用少、对于特定类型的服务专用的容器，并且这种能力是需要满足 *微服务架构* 范式的内容。

此容器技术的主要特性是：

- 特别适合在可扩展的云平台(Kubernetes 和 OpenShift)上运行。
- 占用空间较小（适用于微服务架构）。
- 与 *配置相比，针对约定进行了优化*。
- 不需要应用服务器。您可以在 JVM 中直接运行 Spring Boot 应用程序 Jar。

1.2. 生成您的 BOOSTER 项目

Fuse booster 项目存在，可帮助开发人员开始使用运行独立应用程序。此处提供的说明指导您生成其中一个增强项目，即 Circuit Breaker booster。此练习演示了 Spring Boot 上 Fuse 的有用组件。

Netflix/Hystrix 断路器使分布式应用能够处理对后端服务的网络连接和临时不可用的中断。断路器模式的基本理念是，自动检测到依赖服务的丢失，如果后端服务暂时不可用，可以编程替代行为。

Fuse 断路器提升器由两个相关服务组成：

- 名称服务，将 **名称** 返回到 greet 的后端服务。
- 一个 **问候** 服务，即调用 **name** 服务的 frontend 服务以获取名称，然后返回字符串 **Hello, NAME**。

在此提升器演示中，Hystrix circuit breaker 在 **问候器** 服务和 **名称服务** 之间插入。如果 **后端名称** 服务不可用，则 **问候** 服务可能会回退到替代的行为并立即响应客户端，而不是在等待名称服务重启时被阻断。

先决条件

- **您必须有权访问 Red Hat Developer Platform。**
- **您必须有受支持的 Java Developer Kit (JDK)版本。详情请查看 支持的配置页面。**

- 您必须已安装并配置了 Apache Maven 3.3.x 或更高版本，如 在 本地设置 Maven所述。

流程

1. 进入 https://developers.redhat.com/launch。
2. 单击 START。

   启动程序向导会提示您登录到您的红帽帐户。
3. 单击 登录或注册 按钮，然后登录。
4. 在 Launcher 页面上，单击 Deploy an Example Application 按钮。
5. 在 Create Example Application 页面上，在 Create Example Application 字段中输入 name fuse-circuit-breaker。
6. 点 Select an Example。
7. 在 Example 对话框中，选择 Circuit Breaker 选项。此时会出现 选择运行时 下拉菜单。
   a. 从 Select a Runtime 下拉菜单中选择 Fuse。
   b. 从版本下拉菜单中，选择 7.13 （红帽 Fuse） （不要选择 2.21.2 (Community) 版本）。
   c. 点击 Save。
8. 在 Create Example Application 页面中，点 Download。
9. 当您看到您的应用程序 是 Ready 对话框时，点 Download.zip。您的浏览器下载生成的

booster 项目（打包为 ZIP 文件）。

10. 使用存档实用程序将生成的项目提取到本地文件系统中的方便位置。

1.3. 构建您的 BOOSTER 项目

这些说明指导您在 Spring Boot 上使用 Fuse 构建 Circuir Breaker booster。

先决条件

- 您必须已通过 Red Hat Developer Portal 生成并下载您的 booster 项目。
- 您必须有受支持的 Java Developer Kit (JDK)版本。详情请查看 支持的配置页面。
- 您必须已安装并配置了 Apache Maven 3.3.x 或更高版本，如 在 本地设置 Maven所述。

流程

1. 打开 shell 提示符并使用 Maven 从命令行构建项目：

```
cd fuse-circuit-breaker
```

```
mvn clean package
```

Maven 构建项目后，它会显示 Build Success 消息。

2. 打开一个新的 shell 提示符并启动名称服务，如下所示：

```
cd name-service
```

```
mvn spring-boot:run -DskipTests -Dspring-boot.run.arguments="--server.port=8081"
```

当 Spring Boot 启动时，您应该看到类似如下的输出：

```
...
2019-05-06 20:19:59.401  INFO 9553 --- [           main] o.a.camel.spring.SpringCamelContext
: Route: route1 started and consuming from: servlet:/name?httpMethodRestrict=GET
2019-05-06 20:19:59.402  INFO 9553 --- [           main] o.a.camel.spring.SpringCamelContext
: Total 1 routes, of which 1 are started
2019-05-06 20:19:59.403  INFO 9553 --- [           main] o.a.camel.spring.SpringCamelContext
: Apache Camel 2.21.0.fuse-730078-redhat-00001 (CamelContext: camel-1) started in 0.287
seconds
2019-05-06 20:19:59.406  INFO 9553 --- [           main] o.a.c.c.s.CamelHttpTransportServlet
: Initialized CamelHttpTransportServlet[name=CamelServlet, contextPath=]
2019-05-06 20:19:59.473  INFO 9553 --- [           main]
b.c.e.u.UndertowEmbeddedServletContainer : Undertow started on port(s) 8081 (http)
2019-05-06 20:19:59.479  INFO 9553 --- [           main]
com.redhat.fuse.boosters.cb.Application  : Started Application in 5.485 seconds (JVM
running for 9.841)
```

3. **打开一个新的 shell 提示符并启动问候服务，如下所示：**

```
cd greetings-service
```

```
mvn spring-boot:run -DskipTests
```

当 Spring Boot 启动时，您应该看到类似如下的输出：

```
...
2019-05-06 20:22:19.051  INFO 9729 --- [           main] o.a.c.c.s.CamelHttpTransportServlet
: Initialized CamelHttpTransportServlet[name=CamelServlet, contextPath=]
2019-05-06 20:22:19.115  INFO 9729 --- [           main]
b.c.e.u.UndertowEmbeddedServletContainer : Undertow started on port(s) 8080 (http)
2019-05-06 20:22:19.123  INFO 9729 --- [           main]
com.redhat.fuse.boosters.cb.Application  : Started Application in 7.68 seconds (JVM running
for 12.66)
```

greetings 服务通过 http://localhost:8080/camel/greetings URL 公开 REST 端点。

4. **通过在 web 浏览器中打开 URL 或打开另一个 shell 提示符并输入以下 curl 命令来调用 REST 端点：**

```
curl http://localhost:8080/camel/greetings
```

以下是响应：

```
{"greetings":"Hello, Jacopo"}
```

5. 要演示 Camel Hystrix 提供的断路器功能，请在名称服务的 shell 提示符窗口中键入 Ctrl-C 来终止后端名称服务。

现在，名称服务不可用，断路器在 中启动，以防止在调用时问候服务挂起。

6. 在网页浏览器中打开 http://localhost:8080/camel/greetings，或者在另一个 shell 提示符窗口中输入以下 curl 命令来调用问候 REST 端点：

```
curl http://localhost:8080/camel/greetings
```

以下是响应：

```
{"greetings":"Hello, default fallback"}
```

在运行 greetings 服务的窗口中，日志会显示以下信息序列：

```
2019-05-06 20:24:16.952  INFO 9729 --- [-CamelHystrix-2] route2                   : Try
to call name Service
2019-05-06 20:24:16.956  INFO 9729 --- [-CamelHystrix-2]
o.a.c.httpclient.HttpMethodDirector      : I/O exception (java.net.ConnectException) caught
when processing request: Connection refused (Connection refused)
2019-05-06 20:24:16.956  INFO 9729 --- [-CamelHystrix-2]
o.a.c.httpclient.HttpMethodDirector      : Retrying request
2019-05-06 20:24:16.957  INFO 9729 --- [-CamelHystrix-2]
o.a.c.httpclient.HttpMethodDirector      : I/O exception (java.net.ConnectException) caught
when processing request: Connection refused (Connection refused)
2019-05-06 20:24:16.957  INFO 9729 --- [-CamelHystrix-2]
o.a.c.httpclient.HttpMethodDirector      : Retrying request
2019-05-06 20:24:16.957  INFO 9729 --- [-CamelHystrix-2]
o.a.c.httpclient.HttpMethodDirector      : I/O exception (java.net.ConnectException) caught
when processing request: Connection refused (Connection refused)
2019-05-06 20:24:16.957  INFO 9729 --- [-CamelHystrix-2]
o.a.c.httpclient.HttpMethodDirector      : Retrying request
2019-05-06 20:24:16.964  INFO 9729 --- [-CamelHystrix-2] route2                   : We
are falling back!!!!
```

7. 有关此示例的更多信息，请打开 Circuit Breaker - Red Hat Fuse 页面，地址为 http://localhost:8080/ （同时 问候服务正在运行 ）。此页面包含一个到监控断路器状态的 Hystrix 仪表板的链接。

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 to 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：

groupdId

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子字符串 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

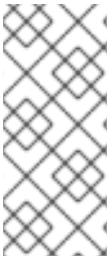
可让您区分从同一 POM 构建但具有不同内容的工件。

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。