Red Hat Fuse 7.9

Apache Karaf へのインストール

Apache Karaf コンテナーへの Red Hat Fuse のインストール

Last Updated: 2021-08-16

Red Hat Fuse 7.9 Apache Karaf へのインストール

Apache Karaf コンテナーへの Red Hat Fuse のインストール

Enter your first name here. Enter your surname here.
Enter your organisation's name here. Enter your organisational division here.
Enter your email address here.

法律上の通知

概要

Red Hat Fuse をインストールして、特定の環境に合わせてインストールを調整するのは簡単です。

目次

前書き

Red Hat Fuse は、軽量で柔軟なインテグレーションプラットフォームであり、オンプレミスまたはクラウドで、組織全体における迅速なインテグレーションを可能にします。

Fuse は Apache Camel をベースとし、パターンベースのインテグレーション、豊富なコネクターカタログ、および広範なデータ変換機能を活用して、ユーザーによるあらゆるインテグレーションを可能にします。

多様性を受け入れるオープンソースの強化

Red Hat では、コード、ドキュメント、Web プロパティーにおける配慮に欠ける用語の置き換えに取り組んでいます。まずは、マスター (master)、スレーブ (slave)、ブラックリスト (blacklist)、ホワイトリスト (whitelist) の 4 つの用語の置き換えから始めます。この取り組みは膨大な作業を要するため、今後の複数のリリースで段階的に用語の置き換えを実施して参ります。詳細は、CTO である Chris Wright のメッセージ をご覧ください。

第1章 APACHE KARAF への開発用の FUSE のインストール

Karaf で実行される Fuse アプリケーションを開発するには、以下のトピックの説明にしたがって、Fuse をローカルにインストールします。

- 「Fuse on Apache Karaf をインストールする準備」
- 「Fuse on Apache Karaf のインストール」
- 「Fuse on Karaf のオフライン実行」
- 「スタンドアロンの Apache ディストリビューションを必要に応じて使用」

1.1. FUSE ON APACHE KARAF をインストールする準備

Fuse on Apache Karaf をインストールする準備を行うには、システムがハードウェア要件を満たし、サポートされているプラットフォームであり、サポートされている Java ランタイムを備えていることを確認します。また、Web サービス、API、およびトランスポートプロトコルにサポートされている標準ソフトウェアを使用する予定であることも確認します。

手順

1. Fuse をインストールするシステムに以下があることを確認してください。
   - 250 MB の空きディスク容量
   - 2 GB の RAM

   このハードウェア要件は、Fuse on Apache Karaf をフルインストールするためのものです。さらに、Fuse を実行しているシステムでは、キャッシュ、永続メッセージストア、およびその他の機能のために容量が必要です。実際の要件は、Fuse アプリケーションの動作によって異なります。
2. Fuse をインストールする予定のシステムがサポートされるプラットフォームを実行していることを確認します。Red Hat は、「Fuse でサポートされる構成」に記載されているプラットフォームで Fuse 製品をテストおよびサポートしています。
3. Fuse on Apache Karaf によってサポートされる Java ランタイムがシステムで稼働していることを確認します。「Red Hat Fuse でサポートされる構成 」の「サポート対象の Java バージョン」を参照してください。
4. Java ランタイムが、空白文字が含まれるディレクトリーパスにインストールされていないことを確認します。たとえば、**C:\Program Files\Java\jdk8** というパスは許可されません。パスに空白文字があると、起動時に Fuse on Apache Karaf で予期せぬエラーが発生します。
5. 「Red Hat Fuse でサポートされる標準 」のリストを確認し、サポートされる標準ソフトウェアを使用していることを確認します。

1.2. FUSE ON APACHE KARAF のインストール

Red Hat カスタマーポータルから Fuse 7.9 on Karaf の標準インストールパッケージをダウンロードできます。このパッケージは Apache Karaf コンテナーの標準アセンブリーをインストールし、完全な Fuse テクノロジースタックを提供します。

Fuse 機能およびバンドルのカスタムサブセットが含まれる Fuse 7.9 の独自のカスタムアセンブリーを

作成できます。**カスタム** クイックスタートは、Maven を使用して Red Hat Fuse のカスタムアセンブリーを作成する方法を表しています。Fuse Software Downloads ページにある利用可能なダウンロード可能なファイルからすべてのクイックスタートをインストールできます。

前提条件

Fuse をインストールする予定のシステムは、「Fuse on Apache Karaf をインストールする準備」 に記載されているハードウェアおよびソフトウェアの要件を満たしている必要があります。

手順

1. ブラウザーで Fuse Software Downloads ページに移動します。
Red Hat カスタマーポータルにまだログインしていない場合は、ログインを求めるプロンプトが表示され、ダウンロードページが表示されます (アカウントは Red Hat Fuse サブスクリプションに関連付けられている必要があります)。

2. Fuse **Software Downloads** ページの **Red Hat Fuse 7.9 on Karaf Installer**の右側にある **Download** をクリックし、ローカル zip ファイルを取得します。

3. zip ファイルの内容を、すべてのパーミッションのあるディレクトリーに展開します。
パス名に空白文字や **#**、**%**、**^**、**"** などの特殊文字が含まれるディレクトリーに、この Zip ファイルを**展開しない** でください。たとえば、**C:\Documents and Settings\Greco#Roman\Desktop\fuse** に展開しないでください。

4. IBM JDK を使用している場合は、以下の追加の手順を実行します。

   a. Fuse インストールディレクトリーの **/lib/endorsed** ディレクトリーで、**saaj-api .jar** ファイルを削除します。以下に例を示します。

   ```
   rm lib/endorsed/org.apache.servicemix.specs.saaj-api-1.3-2.9.0.jar
   ```

   b. **JAVA_OPTS** 環境変数を以下のように設定します。

   ```
   JAVA_OPTS=-Xshareclasses:none
   ```

   Karaf コンテナーを起動する前に、**JAVA_OPTS** 環境変数を設定する必要があります。

5. 管理者ユーザーを追加して、Fuse on Karaf コンテナーへのリモートアクセスを有効にし、Fuse Console にアクセスします。
デフォルトでは、コンテナーにユーザーは定義されていません。この場合、コンテナーをフォアグラウンドで実行できますが、コンテナーにリモートでアクセスしたり、バックグラウンドで実行したりすることはできません。以下の手順に従い、少なくとも **admin** ロールのあるユーザーを 1 人作成することが推奨されます。

   a. テキストエディターで、Fuse インストールディレクトリーにある **etc/users.properties** ファイルを開きます。

   b. 次の行を見つけます。

   ```
   #admin = admin,_g_:admingroup
   #_g_\:admingroup = group,admin,manager,viewer,systembundles,ssh
   ```

   c. 行ごとに、先頭の **#** 文字を削除して、行のコメントを解除します。

   d. 最初の行で、**admin** の最初のインスタンスを、選択したユーザー名 (**user1** など) に変更します。

e. 同じ行で、**admin** の 2 番目のインスタンスを、そのユーザーのパスワード (例: **passw0rd**) に変更します。
以下に例を示します。

```
user1 = passw0rd,_g_:admingroup
_g_\:admingroup = group,admin,manager,viewer,systembundles,ssh
```

f. ファイルを保存して閉じます。

6. Fuse を起動するには、Linux/Unix では **bin/fuse** を実行し、Windows では **bin\fuse.bat** を実行します。

7. 任意で、Fuse Console にアクセスするには、Web ブラウザーで提供された URL を開き、**etc/users.properties** ファイルで設定したユーザー名およびパスワードを使用してログインします。Fuse Console の使用に関する詳細は、『Managing Fuse』を参照してください。

1.3. FUSE ON KARAF のオフライン実行

Apache Karaf コンテナーは、インターネットに接続しないオフラインモードで実行できます。ただし、カスタムアプリケーションをコンテナーにデプロイする場合は、これらのアプリケーションでコンテナーをオフラインモードで実行する前に、ローカルの Maven リポジトリーに追加の依存関係をダウンロードする必要がある場合があります。

Apache Karaf コンテナーをオフラインモードで実行するには、以下の種類の依存関係を区別する必要があります。

- **ランタイム依存関係**は、デフォルト設定で Apache Karaf コンテナーを実行するために必要な依存関係です。
- **ビルドタイム依存関係**は、カスタムアプリケーションをビルドするために必要な依存関係であり、サードパーティのライブラリーが含まれる場合があります。

以下は、オフラインモードで実行できるものと、オンラインモードで実行する必要があるもの (インターネット接続を使用して) の概要になります。

- **デフォルト設定で Apache Karaf コンテナーを実行**することは、オフラインモードでサポートされます。Apache Karaf コンテナーのデフォルト設定は、**etc/org.apache.karaf.features.cfg** ファイルの **featuresBoot** プロパティーによって指定されます。必要な依存関係は、インストールの **system/** サブディレクトリーに提供されます。
- **追加機能のインストール** は、一般的にオフラインモードではサポートされて **いません**。原則では、**features:install** コマンドを使用して標準機能のリポジトリーから機能のいずれかをインストールできますが (**etc/org.apache.karaf.features.cfg** ファイルの **featuresRepositories** プロパティーによって指定された)、これらの機能の大部分はインターネットからダウンロードする必要があるため、オフラインモードではサポートされていません。
- **カスタムアプリケーションのデプロイ** は、通常オフラインモードではサポートされて **いません**。最小限のビルドタイム依存関係があるアプリケーションでは、場合によってはオフラインでデプロイできることがあります。しかし、カスタムアプリケーションには通常、JAR ファイルを Apache Maven でダウンロードできるようにするための、インターネット接続を必要とするサードパーティーの依存関係があります。

その他のリソース

「ローカル Maven リポジトリーの追加」

1.4. スタンドアロンの APACHE ディストリビューションを必要に応じて使用

Red Hat Fuse は、Apache Camel と Apache CXF の標準ディストリビューションが含まれる、ダウンロードする追加パッケージを提供します。ApacheCamel または ApacheCXF の標準のアップストリームディストリビューション (OSGiコンテナーなし) を使用する場合は、ダウンロードした **extras** パッケージのアーカイブバージョンを使用します。

手順

1. Red Hat カスタマーポータル にログインします。
2. Red Hat カスタマーポータル→Downloads→Red Hat Fuse→Downloads ページに移動します。
3. **Software Downloads** ページの **Version** ドロップダウンリストから **7.9.0** を選択します。
4. Red Hat Fuse 7.9.0 Extras アーカイブをダウンロードします。
   extras アーカイブファイルには、その中にネストされた以下のアーカイブファイルが含まれています。
   - **apache-camel-2.23.2.fuse-790054-redhat-00001.zip**
   - **apache-cxf-3.3.6.fuse-790049-redhat-00001.zip**
5. これらのファイルを任意の場所にコピーし、プラットフォームの適切なユーティリティーを使用して展開します。

警告

パス名に空白文字が含まれているフォルダーにアーカイブファイルを展開しないでください。たとえば、**C:\Documents and Settings\Greco Roman\Desktop\fuse** に展開しないでください。

第2章 FUSE ON APACHE KARAF へのホットフィックスパッチの適用

2.1. 機能とバンドルへのパッチ適用

パッチは、Fuse on Apache Karaf インストールに存在する更新されたバージョンのファイルが含まれる ZIP アーカイブです。これらには以下が含まれます。

- バンドル: 最も一般的であり、最も単純なケースでは、ホットフィックスパッチに単一のバンドルが含まれる場合があります。
- **$FUSE_HOME/etc** および **$FUSE_HOME/bin** ディレクトリーにそれぞれ存在する設定ファイルおよびスクリプト。
- 通常のバンドルではなく、**$FUSE_HOME/lib** ディレクトリーに存在するライブラリー。
- 機能定義の変更: 通常、Karaf 機能は **$FUSE_HOME/system** ディレクトリーで利用できる記述子に含まれますが、ホットフィックスパッチはこれらのファイルを変更しません。代わりに、ホットフィックスパッチは、**$FUSE_HOME/etc/org.apache.karaf.features.xml** である機能オーバーライドファイルを変更することがあります。これにより、指定の機能バンドルをアップグレードするか、指定の機能バンドルが追加バンドルを使用するようにすることで、機能の定義をホットフィックスで変更できるようになります。

アップグレードパッチとホットフィックスパッチの違い

- ホットフィックスパッチ: ホットフィックスパッチには、1 つまたは複数の重大なバグのみに対する修正が含まれています。これらは、現在の Red Hat Fuse ディストリビューション上に適用することを目的としています。その主な目的は、既存ディストリビューションのバンドルとライブラリの一部を更新することです。
- アップグレード: Fuse on Apache Karaf アップグレードメカニズムにより、更新されたバージョンの Fuse on Karaf を再インストールしなくても、修正を Apache Karaf コンテナーに適用できます。また、アップグレードが原因でデプロイされたアプリケーションで問題が発生した場合に、アップグレードをロールバックすることもできます。Fuse on Apache Karaf のアップグレードプロセスでは、バンドル JAR、設定ファイル、静的ファイルなどのファイルが更新されます。

Fuse on Apache Karaf スタンドアロンでは、Karaf コンソールのパッチシェルからコマンドを使用してパッチを適用できます。このアプローチは破壊的ではなく、元に戻すことができます。以下の手順は、Red Hat Fuse on Apache Karaf のアップグレードにも使用できます。アップグレードの詳細は、『Upgrading Fuse on Apache Karaf』を参照してください。

2.2. RED HAT FUSE ON APACHE KARAF へのホットフィックスパッチの適用

ホットフィックスメカニズムを使用して、利用可能な機能定義とバンドルを同時に更新できます。Fuse on Apache Karaf インストールにホットフィックスパッチを適用する手順は次のとおりです。

手順

1. アップグレードする前に、Fuse on Apache Karaf インストールの完全なバックアップを作成します。
2. ターミナルを開き、Apache karaf サーバーで Fuse を起動します。

```
[user@FUSE_HOME/bin ~] $ ./fuse
```

3. 任意手順: カスタマーポータルから必要なパッチをダウンロードし、ステップ 5 に進みます。

4. **patch:find** コマンドを入力し、Maven リポジトリーで利用可能なパッチを検索します。以下に例を示します。

```
karaf@root()> patch:find
Found new remote patch at mvn:org.jboss.redhat-fuse/fuse-karaf-patch-repository/7.8.0.fuse-sb2-780040/zip
You can add the patch using "patch:add mvn:org.jboss.redhat-fuse/fuse-karaf-patch-repository/7.8.0.fuse-sb2-780040/zip" command, or simply use "patch:find --add" option.
```

注記

--add オプションを指定して **patch:find** コマンドを使用し、最新のパッチを見つけ、コンテナーの環境に追加できます。

5. **patch:add** コマンドを入力して、コンテナーの環境にパッチを追加します。以下に例を示します。

```
karaf@root()> patch:add mvn:org.jboss.redhat-fuse/fuse-karaf-patch-repository/7.8.0.fuse-sb2-780040/zip
[name]                                          [installed] [rollup] [description]
[CVEs]
fuse-karaf-maintenance-patch-7.8.0.fuse-sb2-780040 false       false    fuse-karaf-maintenance-patch-7.8.0.fuse-sb2-780040 CVE-2020-28052

Current patch mechanism version: 7.8.0.fuse-780038
New patch mechanism version detected: 7.8.0.fuse-780040
Please run "patch:update" command to upgrade patching mechanism to version 7.8.0.fuse780040
```

注記

patch:add コマンドを使用する代わりに、.zip パッチファイルを **FUSE_HOME/patches** ディレクトリーにコピーしてパッチファイルを自動的に追加することもできます。

6. 任意手順: **patch:update** コマンドを入力し、パッチメカニズム自体を更新します。

```
karaf@root()> patch:update
Current patch mechanism version: 7.8.0.fuse-780038
New patch mechanism version detected: 7.8.0.fuse-780040
Uninstalling patch features in version 7.8.0.fuse-780038
Installing patch features in version 7.8.0.fuse-780040
```

7. **patch:simulate** コマンドを入力して、パッチのインストールをシミュレートします。これにより、パッチのインストール時にコンテナーに加えられた変更のログが生成されますが、コンテナーに実際の変更は加えられません。シミュレーションログを確認して、これらの変更を理解してください。

8. **patch:list** コマンドを入力し、追加されたパッチの一覧を表示します。このリストで、[name] 見出しのエントリーはパッチ ID になります。

```
karaf@root()> patch:list
[name]                                          [installed] [rollup] [description]
[CVEs]
fuse-karaf-maintenance-patch-7.8.0.fuse-sb2-780040 false       false    fuse-karaf-
maintenance-patch-7.8.0.fuse-sb2-780040 CVE-2020-28052
```

9. パッチに明示的な CVE メタデータが含まれる場合は、**patch:show** コマンドを入力して詳細を表示できます。

```
karaf@root()> patch:show fuse-karaf-maintenance-patch-7.8.0.fuse-sb2-780040
Patch ID: fuse-karaf-maintenance-patch-7.8.0.fuse-sb2-780040
Patch Commit ID: a2d7cf58e21116cde66c97232aea4be1ec304400
#### 1 CVE fix:
 - CVE-2020-28052: bouncycastle: password bypass in OpenBSDBCrypt.checkPassword
utility possible
   Bugzilla link: https://bugzilla.redhat.com/show_bug.cgi?id=1912881
   CVE link: https://cve.mitre.org/cgi-bin/cvename.cgi?name=2020-28052
```

10. **patch:install** コマンドを入力し、適用するパッチのパッチ ID を指定して、コンテナーにパッチを適用します。以下に例を示します。

```
patch:install fuse-karaf-maintenance-patch-7.8.0.fuse-sb2-780040
```

2.3. パッチのロールバック

以下のように、**patch:rollback** コマンドを使用して、インストールされたホットフィックスパッチをロールバックし、パッチ適用前の動作に戻すことができます。

手順

1. **patch:list** コマンドを入力し、直近にインストールされたパッチのパッチ ID を取得します。
2. 更新されたバンドルをロールバックするには、以下のコマンドを入力します。

```
karaf@root()> patch:rollback my-patch-x
INFO : org.jboss.fuse.modules.patch.patch-management (2): Rolling back non-rollup patch
"my-patch-x"
removing overriden feature: hawtio-rbac/2.0.0.fuse-000117
refreshing features
Enter feature:info command to view the information about the feature.
```

```
karaf@root()> feature:info hawtio-rbac
Feature hawtio-rbac 2.0.0.fuse-000117
Details:
    Installs the hawtio RBAC enabler bundle(s)
Feature has no configuration
Feature has no configuration files
Feature has no dependencies.
Feature contains followed bundles:
    mvn:io.hawt/hawtio-osgi-jmx/2.0.0.fuse-000117
Feature has no conditionals.
```

2.4. RED HAT FUSE アプリケーションへのパッチ適用

新しい **patch-maven-plugin** メカニズムを使用すると、パッチを Red Hat Fuse アプリケーションに適用できます。このメカニズムにより、異なる Red Hat Fuse の BOM によって提供される個々のバージョンを変更できます (たとえば、**fuse-springboot-bom** と **fuse-karaf-bom** など)。

2.4.1. patch-maven-plugin

patch-maven-plugin は以下の操作を実行します。

- 現在の Red Hat Fuse BOM に関連するパッチメタデータを取得します。
- BOM からインポートされた **<dependencyManagement>** に、バージョンの変更を適用します。

patch-maven-plugin がメタデータを取得したら、プラグインが宣言されたプロジェクトの管理された依存関係および直接の依存関係すべてに対して繰り返し処理を行い、CVE/patch メタデータを使用して、一致する依存関係バージョンを置き換えます。バージョンが置き換えられたら、Maven ビルドが続行され、標準の Maven プロジェクトのステージに進みます。

2.4.2. Red Hat Fuse アプリケーションへのパッチ適用

patch-maven-plugin の目的は、Red Hat Fuse BOM にある依存関係のバージョンを、アプリケーションに適用するパッチのパッチメタデータに指定されたバージョンに更新することです。

手順

以下の手順では、アプリケーションにパッチを適用する方法を説明します。

1. **patch-maven-plugin** をプロジェクトの **pom.xml** ファイルに追加します。**patch-maven-plugin** のバージョンは、Fuse BOM のバージョンと同じである必要があります。

```
<build>
  <plugins>
    <plugin>
      <groupId>org.jboss.redhat-fuse</groupId>
      <artifactId>patch-maven-plugin</artifactId>
      <version>${version.org.jboss-redhat-fuse}</version>
      <extensions>true</extensions>
    </plugin>
  </plugins>
</build>
```

2. **mvn clean deploy**、**mvn validate**、または **mvn dependency:tree** コマンドの 1 つを実行すると、プラグインはプロジェクトモジュールを検索して、Red Hat Fuse BOM のいずれかが使用されているかどうかを確認します。以下の 2 つのみがサポートされる BOM とみなされます。
   - **org.jboss.redhat-fuse:fuse-karaf-bom**: Fuse Karaf BOM の場合
   - **org.jboss.redhat-fuse:fuse-springboot-bom**: Fuse Spring Boot BOM の場合
3. 上記の BOM がいずれも見つからない場合は、プラグインによって以下のメッセージが表示されます。

```
$ mvn clean install
[INFO] Scanning for projects...
[INFO]

========== Red Hat Fuse Maven patching ==========

[INFO] [PATCH] No project in the reactor uses Fuse Karaf or Fuse Spring Boot BOM.
Skipping patch processing.
[INFO] [PATCH] Done in 3ms
```

4. Fuse BOM が両方が見つかった場合は、**patch-maven-plugin** は以下の警告により停止します。

```
$ mvn clean install
[INFO] Scanning for projects...
[INFO]

========== Red Hat Fuse Maven patching ==========

[WARNING] [PATCH] Reactor uses both Fuse Karaf and Fuse Spring Boot BOMs. Please
use only one. Skipping patch processing.
[INFO] [PATCH] Done in 3ms
```

5. **patch-maven-plugin** は以下の Maven メタデータ値のいずれかを取得しようと試みます。

   - Fuse Karaf BOM を使用するプロジェクトでは、**org.jboss.redhat-fuse/fuse-karaf-patch-metadata/maven-metadata.xml** は解決されています。これは、**org.jboss.redhat-fuse:fuse-karaf-patch-metadata:RELEASE**コーディネートのアーティファクトのメタデータです。

   - Fuse Spring Boot BOM プロジェクトを使用するプロジェクトでは、**org.jboss.redhat-fuse/fuse-springboot-patch-metadata/maven-metadata.xml** は解決されています。これは、**org.jboss.redhat-fuse:fuse-springboot-patch-metadata:RELEASE** コーディネートのアーティファクトのメタデータです。

     Maven によって生成されたメタデータの例

```
<?xml version="1.0" encoding="UTF-8"?>
<metadata>
  <groupId>org.jboss.redhat-fuse</groupId>
  <artifactId>fuse-springboot-patch-metadata</artifactId>
  <versioning>
    <release>7.8.1.fuse-sb2-781025</release>
    <versions>
      <version>7.8.0.fuse-sb2-780025</version>
      <version>7.7.0.fuse-sb2-770010</version>
      <version>7.7.0.fuse-770010</version>
      <version>7.8.1.fuse-sb2-781025</version>
    </versions>
    <lastUpdated>20201023131724</lastUpdated>
  </versioning>
</metadata>
```

6. **patch-maven-plugin** はメタデータを解析し、現在のプロジェクトに適用可能なバージョンを選択します。これは、バージョン **7.8.xxx** の Fuse BOM を使用する Maven プロジェクトでのみ

可能です。バージョン範囲 7.8 および 7.9 以降と一致するメタデータのみが適用可能で、メタデータの最新バージョンのみが取得されます。

7. **patch-maven-plugin** は、前の手順で見つかった **groupId**、**artifactId**、および **version** によって特定されたパッチメタデータをダウンロードする際に使用されるリモート Maven リポジトリーのリストを収集します。これらの Maven リポジトリーは、アクティブなプロファイルのプロジェクトの **<repositories>** 要素にリストされているもので、**settings.xml** ファイルからのリポジトリーもリストされています。

```
$ mvn clean install
[INFO] Scanning for projects...
[INFO]

========== Red Hat Fuse Maven patching ==========

[INFO] [PATCH] Reading patch metadata and artifacts from 2 project repositories
[INFO] [PATCH]  - local-nexus: http://everfree.forest:8081/repository/maven-releases/
[INFO] [PATCH]  - central: https://repo.maven.apache.org/maven2
Downloading from local-nexus: http://everfree.forest:8081/repository/maven-releases/org/jboss/redhat-fuse/fuse-springboot-patch-metadata/maven-metadata.xml
...
```

8. 任意で、オフラインのリポジトリーを使用する場合は、**-Dpatch** オプションを使用して、**jboss-fuse/redhat-fuse** プロジェクトの **fuse-karaf/fuse-karaf-patch-repository** モジュールまたは **fuse-springboot/fuse-springboot-patch-repository** モジュールによって生成される ZIP ファイルを指定できます。これらの ZIP ファイルの内部構造は、Maven リポジトリーの構造と同じです。例を以下に示します。

```
$ mvn clean install -Dpatch=../../../test/resources/patch-3.zip
[INFO] Scanning for projects...
[INFO]

========== Red Hat Fuse Maven patching ==========

[INFO] [PATCH] Reading metadata and artifacts from /data/sources/github.com/jboss-fuse/redhat-fuse/fuse-tools/patch-maven-plugin/src/test/resources/patch-3.zip
Downloading from fuse-patch: zip:file:/tmp/patch-3.zip-1742974214598205745/org/jboss/redhat-fuse/fuse-springboot-patch-metadata/maven-metadata.xml
Downloaded from fuse-patch: zip:file:/tmp/patch-3.zip-1742974214598205745/org/jboss/redhat-fuse/fuse-springboot-patch-metadata/maven-metadata.xml (406 B at 16 kB/s)
Downloading from fuse-patch: zip:file:/tmp/patch-3.zip-1742974214598205745/org/jboss/redhat-fuse/fuse-springboot-patch-metadata/7.8.0.fuse-sb2-781023/fuse-springboot-patch-metadata-7.8.0.fuse-sb2-781023.xml
Downloaded from fuse-patch: zip:file:/tmp/patch-3.zip-1742974214598205745/org/jboss/redhat-fuse/fuse-springboot-patch-metadata/7.8.0.fuse-sb2-781023/fuse-springboot-patch-metadata-7.8.0.fuse-sb2-781023.xml (926 B at 309 kB/s)
[INFO] [PATCH] Resolved patch descriptor: /home/user/.m2/repository/org/jboss/redhat-fuse/fuse-springboot-patch-metadata/7.8.0.fuse-sb2-781023/fuse-springboot-patch-metadata-7.8.0.fuse-sb2-781023.xml
...
```

9. メタデータがリモートリポジトリー、ローカルリポジトリー、または ZIP ファイルからであるかどうかに関わらず、**patch-maven-plugin** によって分析されます。フェッチされたメタデー

タには CVE の一覧が含まれ、各 CVE には、影響を受ける Maven アーティファクトのリスト (glob パターンおよびバージョン範囲で指定) と、指定の CVE の修正が含まれるバージョンがあります。例を以下に示します。

```
<?xml version="1.0" encoding="UTF-8" ?>

<metadata xmlns="urn:redhat:fuse:patch-metadata:1">
   <product-bom groupId="org.jboss.redhat-fuse" artifactId="fuse-springboot-bom" versions="
[7.8,7.9)" />
   <cves>
      <cve id="CVE-2020-xyz" description="Jetty can be configured to listen on port 8080"
           cve-link="https://nvd.nist.gov/vuln/detail/CVE-2020-xyz"
           bz-link="https://bugzilla.redhat.com/show_bug.cgi?id=42">
         <affects groupId="org.eclipse.jetty" artifactId="jetty-*" versions="[9.4,9.4.32)"
fix="9.4.32.v20200930" />
         <affects groupId="org.eclipse.jetty.http2" artifactId="http2-*" versions="[9.4,9.4.32)"
fix="9.4.32.v20200930" />
      </cve>
   </cves>
   <fixes />
</metadata>
```

10. 最後に、現在のプロジェクトの管理された依存関係に繰り返し処理が行われるときに、パッチメタデータに指定された修正リストが参照されます。一致するこれらの依存関係 (および管理された依存関係) は、固定バージョンに変更されます。以下に例を示します。

```
$ mvn clean install -U
[INFO] Scanning for projects...
[INFO

========== Red Hat Fuse Maven patching ==========

[INFO] [PATCH] Reading patch metadata and artifacts from 2 project repositories
[INFO] [PATCH]  - local-nexus: http://everfree.forest:8081/repository/maven-releases/
[INFO] [PATCH]  - central: https://repo.maven.apache.org/maven2
Downloading from local-nexus: http://everfree.forest:8081/repository/maven-
releases/org/jboss/redhat-fuse/fuse-springboot-patch-metadata/maven-metadata.xml
Downloading from central: https://repo.maven.apache.org/maven2/org/jboss/redhat-
fuse/fuse-springboot-patch-metadata/maven-metadata.xml
Downloaded from local-nexus: http://everfree.forest:8081/repository/maven-
releases/org/jboss/redhat-fuse/fuse-springboot-patch-metadata/maven-metadata.xml (363 B
at 4.3 kB/s)
[INFO] [PATCH] Resolved patch descriptor: /home/user/.m2/repository/org/jboss/redhat-
fuse/fuse-springboot-patch-metadata/7.8.0.fuse-sb2-780032/fuse-springboot-patch-
metadata-7.8.0.fuse-sb2-780032.xml
[INFO] [PATCH] Patch metadata found for org.jboss.redhat-fuse/fuse-springboot-
bom/[7.8,7.9)
[INFO] [PATCH]  - patch contains 1 CVE fix
[INFO] [PATCH] Processing managed dependencies to apply CVE fixes...
(https://nvd.nist.gov/vuln/detail/CVE-2020-xyz, https://bugzilla.redhat.com/show_bug.cgi?
id=42_
[INFO] [PATCH] - CVE-2020-xyz: Jetty can be configured to expose itself on port 8080
[INFO] [PATCH]   Applying change org.eclipse.jetty/jetty-*/[9.4,9.4.32) -> 9.4.32.v20200930
[INFO] [PATCH]    - managed dependency: org.eclipse.jetty/jetty-alpn-
client/9.4.30.v20200611 -> 9.4.32.v20200930
...
```

```
[INFO] [PATCH]    - managed dependency: org.eclipse.jetty/jetty-openid/9.4.30.v20200611 ->
9.4.32.v20200930
[INFO] [PATCH]   Applying change org.eclipse.jetty.http2/http2-*/[9.4,9.4.32) ->
9.4.32.v20200930
[INFO] [PATCH]    - managed dependency: org.eclipse.jetty.http2/http2-
client/9.4.30.v20200611 -> 9.4.32.v20200930
...
[INFO] [PATCH] Done in 635ms

================================================
```

パッチのスキップ

特定のパッチをプロジェクトに適用したくない場合、**patch-maven-plugin** は **skip** オプションを提供します。すでに **patch-maven-plugin** をプロジェクトの **pom.xml** ファイルに追加済みで、バージョンを変更したくない場合は、以下のいずれかの方法を使用してパッチをスキップできます。

- 以下のように、プロジェクトの **pom.xml** ファイルに skip オプションを追加します。

```
<build>
   <plugins>
      <plugin>
         <groupId>org.jboss.redhat-fuse</groupId>
         <artifactId>patch-maven-plugin</artifactId>
         <version>${version.org.jboss-redhat-fuse}</version>
         <extensions>true</extensions>
         <configuration>
            <skip>true</skip>
         </configuration>
      </plugin>
   </plugins>
</build>
```

- または、以下のように **mvn** コマンドの実行時に **-DskipPatch** オプションを使用します。

```
$ mvn dependency:tree -DskipPatch
[INFO] Scanning for projects...
[INFO]
[INFO] ------< org.jboss.redhat-fuse:cve-dependency-management-module1 >-------
[INFO] Building cve-dependency-management-module1 7.8.0.fuse-sb2-780033
[INFO] --------------------------------[ jar ]---------------------------------
...
```

上記の出力にあるように、**patch-maven-plugin** は呼び出されず、パッチはアプリケーションに適用されません。

第3章 MAVEN のローカルでの設定

一般的な Fuse アプリケーションの開発では、Maven を使用してプロジェクトをビルドおよび管理します。

以下のトピックで Maven をローカルで設定する方法を説明します。

- 「Maven 設定の準備」
- 「Red Hat リポジトリーを Maven へ追加」
- 「ローカル Maven リポジトリーの追加」
- 「環境変数またはシステムプロパティーを使用した Maven ミラーの設定」
- 「Maven アーティファクトおよびコーディネート」

3.1. MAVEN 設定の準備

Maven は、Apache の無料のオープンソースビルドツールです。通常は、Maven を使用して Fuse アプリケーションをビルドします。

手順

1. Maven ダウンロードページ から最新バージョンの Maven をダウンロードします。
2. システムがインターネットに接続していることを確認します。
   プロジェクトのビルド中、Maven が外部リポジトリーを探し、必要なアーティファクトをダウンロードするのがデフォルトの動作になります。Maven はインターネット上でアクセス可能なリポジトリーを探します。

   このデフォルト動作を変更し、Maven によってローカルネットワーク上のリポジトリーのみが検索されるようにすることができます。これは Maven をオフラインモードで実行できることを意味します。オフラインモードでは、Maven によってローカルリポジトリーのアーティファクトが検索されます。「ローカル Maven リポジトリーの追加」 を参照してください。

3.2. RED HAT リポジトリーを MAVEN へ追加

Red Hat Maven リポジトリーあるアーティファクトにアクセスするには、Red Hat Maven リポジトリーを Maven の **settings.xml** ファイルに追加する必要があります。Maven は、**.m2** ディレクトリーで **settings.xml** ファイルを探します。ユーザー指定の **settings.xml** ファイルがない場合、Maven は **M2_HOME/conf/settings.xml** のシステムレベルの **settings.xml** ファイルを使用します。

前提条件

Red Hat リポジトリーを追加する **settings.xml** ファイルがある場所を知っている必要があります。

手順

以下の例のように、**settings.xml** ファイルに Red Hat リポジトリーの **repository** 要素を追加します。

```
<?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>
```

3.3. ローカル MAVEN リポジトリーの追加

インターネットへ接続せずにコンテナーを実行し、オフライン状態では使用できない依存関係を持つアプリケーションをデプロイする場合、Maven 依存関係プラグインを使用してアプリケーションの依存関係を Maven オフラインリポジトリーにダウンロードすることができます。ダウンロード後、このカスタマイズされた Maven オフラインリポジトリーをインターネットに接続していないマシンに提供することができます。

手順

1. **pom.xml** ファイルが含まれるプロジェクトディレクトリーで、以下のようなコマンドを実行し、Maven プロジェクトのリポジトリーをダウンロードします。

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

   この例では、プロジェクトのビルドに必要な Maven 依存関係とプラグインは **/tmp/my-project** ディレクトリーにダウンロードされます。

2. **etc/org.ops4j.pax.url.mvn.cfg** ファイルを編集し、**org.ops4j.pax.url.mvn.offline** を true に設定します。これによりオフラインモードが有効になります。

   ```
   ##
   # If set to true, no remote repository will be accessed when resolving artifacts
   #
   org.ops4j.pax.url.mvn.offline = true
   ```

3. このカスタマイズされた Maven オフラインリポジトリーを、インターネットに接続していない内部のマシンに提供します。

3.4. 環境変数またはシステムプロパティーを使用した MAVEN ミラーの設定

アプリケーションの稼働中に、Red Hat Maven リポジトリーにあるアーティファクトにアクセスする必要があります。これらのリポジトリーは、Maven の **settings.xml** ファイルに追加されます。Maven は **settings.xml** で以下の場所をチェックします。

- 指定の URL を検索します。
- 見つからなかった場合は **${user.home}/.m2/settings.xml** を検索します。
- 見つからなかった場合は **${maven.home}/conf/settings.xml** を検索します。

- 見つからなかった場合は **${M2_HOME}/conf/settings.xml** を検索します。
- どの場所も見つからない場合は、空の **org.apache.maven.settings.Settings** インスタンスが作成されます。

3.4.1. Maven ミラー

Maven では、一連のリモートリポジトリーを使用して、現在ローカルリポジトリーで利用できないアーティファクトにアクセスします。ほとんどの場合で、リポジトリーのリストには Maven Central リポジトリーが含まれますが、Red Hat Fuse では Maven Red Hat リポジトリーも含まれます。リモートリポジトリーへのアクセスが不可能な場合や許可されない場合に、Maven ミラーのメカニズムを使用できます。ミラーは、特定のリポジトリー URL を異なるリポジトリー URL に置き換えるため、リモートアーティファクトの検索時にすべての HTTP トラフィックを単一の URL に転送することができます。

3.4.2. Maven ミラーの settings.xml への追加

Maven ミラーを設定するには、以下のセクションを Maven の **settings.xml** に追加します。

```
<mirror>
    <id>all</id>
    <mirrorOf>*</mirrorOf>
    <url>http://host:port/path</url>
</mirror>
```

settings.xml ファイルに上記のセクションがない場合は、ミラーは使用されません。XML 設定を提供せずにグローバルミラーを指定するには、システムプロパティーまたは環境変数を使用します。

3.4.3. 環境変数またはシステムプロパティーを使用した Maven ミラーの設定

環境変数またはシステムプロパティーのいずれかを使用して Maven ミラーを設定するには、以下を追加します。

- 環境変数 **MAVEN_MIRROR_URL** を **bin/setenv** ファイルに追加します。
- システムプロパティー **mavenMirrorUrl** を **etc/system.properties** ファイルに追加します。

3.4.4. Maven オプションを使用した Maven ミラー URL の指定

環境変数またはシステムプロパティーによって指定された Maven ミラー URL ではなく、別の Maven ミラー URL を使用するには、アプリケーションの実行時に以下の Maven オプションを使用します。

- **-DmavenMirrorUrl=mirrorId::mirrorUrl**
  例: **-DmavenMirrorUrl=my-mirror::http://mirror.net/repository**
- **-DmavenMirrorUrl=mirrorUrl**
  例: **-DmavenMirrorUrl=http://mirror.net/repository**この例では、<mirror> の <id> はミラーになります。

3.5. MAVEN アーティファクトおよびコーディネート

Maven ビルドシステムでは、**アーティファクト** が基本のビルドブロックです。ビルド後のアーティファクトの出力は、通常 JAR や WAR ファイルなどのアーカイブになります。

Maven の主な特徴として、アーティファクトを検索し、検索したアーティファクト間で依存関係を管理できる機能が挙げられます。**Maven コーディネート** は、特定のアーティファクトの場所を特定する値のセットです。基本的なコーディネートには、以下の形式の 3 つの値があります。

groupId:artifactId:version

基本的なコーディネートに **packaging** の値、または **packaging** と **classifier** の値の両方を追加することがあります。Maven コーディネートには以下の形式のいずれかを使用できます。

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

値の説明は次のとおりです。

groupdId

アーティファクトの名前の範囲を定義します。通常、パッケージ名のすべてまたは一部をグループ ID として使用します。例: **org.fusesource.example**

artifactId

グループ名に関連するアーティファクト名を定義します。

version

アーティファクトのバージョンを指定します。バージョン番号には **n.n.n.n** のように最大 4 つの部分を含めることができ、最後の部分には数字以外の文字を含めることができます。たとえば **1.0-SNAPSHOT** の場合、最後の部分は英数字のサブ文字列である **0-SNAPSHOT** になります。

packaging

プロジェクトのビルド時に生成されるパッケージ化されたエンティティーを定義します。OSGi プロジェクトでは、パッケージングは **bundle** になります。デフォルト値は **jar** です。

classifier

同じ POM からビルドされた内容が異なるアーティファクトを区別できるようにします。

アーティファクトの POM ファイルの要素は、以下のようにアーティファクトのグループ ID、アーティファクト ID、パッケージング、およびバージョンを定義します。

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

前述のアーティファクトの依存関係を定義するには、以下の **dependency** 要素を POM ファイルに追加します。

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

```
    </dependency>
  </dependencies>
  ...
</project>
```

注記

バンドルは特定タイプの JAR ファイルで、**jar** はデフォルトの Maven パッケージタイプであるため、前述の依存関係に **bundle** パッケージを指定する必要はありません。ただし、依存関係でパッケージタイプを明示的に指定する必要がある場合は、**type** 要素を使用できます。