Red Hat Fuse 7.7

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: 2023-09-20

Red Hat Fuse 7.7 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

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システムを調整するのは簡単です。

目次

はじめに

Red Hat Fuse は軽量で柔軟性に富んだインテグレーションプラットフォームで、オンプレミスとクラウドにおいて、企業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。

Apache Camel に基づく Fuse は、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、ユーザーがすべてを統合できるようにします。

第1章 APACHE KARAF への開発用 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、以下のトピックで説明するように Fuse をローカルでインストールします。

- 「Apache Karaf に Fuse をインストールする準備」
- 「Apache Karaf への Fuse のインストール」
- 「Fuse on Karaf のオフライン実行について」
- 「オプションでのスタンドアロン Apache ディストリビューションの使用」

1.1. APACHE KARAF に FUSE をインストールする準備

Apache Karaf に Fuse をインストールする準備をするには、システムをチェックして、ハードウェア要件を満たしていること、サポートされているプラットフォームであること、および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。Web サービス、API、およびトランスポートプロトコルに関して、サポート対象の標準ソフトウェアを使用する計画であることも確認してください。

手順

1. Fuse をインストールするシステムで、以下を確認します。

   - 250 MB の空きディスク領域がある
   - 2 GB の RAM がある

   このハードウェア要件は、Fuse on Apache Karaf の完全なインストール用です。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用に領域が必要です。実際の要件は、Fuse アプリケーションの用途によって異なります。

2. Fuse をインストールするシステムがサポート対象の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、Red Hat Fuse でサポートされる構成 に記載のプラットフォームで、Fuse 製品をテストおよびサポートします。

3. システムが Fuse on Apache Karaf でサポートされる Java ランタイムを実行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 で、サポート対象の Java バージョンを確認してください。

4. Java ランタイムが空白を含む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てください。たとえば、**C:\Program Files\Java\jdk8** は受け入れ可能なパスではありません。パスに空白があると、実行時に Fuse on Apache Karaf で予期しないエラーが発生します。

5. Red Hat Fuse でサポートされる構成 のリストを確認し、サポート対象の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. APACHE KARAF への FUSE のインストール

Red Hat カスタマーポータルから Fuse 7.7 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。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 7.12 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説明します。Fuse Software Downloads ページで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が、「Apache Karaf に Fuse 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要件を満たしている。

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ログインしていない場合は、ログインの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 (アカウントが 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 の **Software Downloads** ページの **Red Hat Fuse 7.12 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 ファイルのコンテンツを、すべてのパーミッションが設定されたディレクトリーにデプロイメントします。
   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** にデプロイメントしないでください。

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、コンテナーにはユーザーが定義されていません。この場合、フォアグラウンドでコンテナーを実行できますが、コンテナーにリモートでアクセスすることはできず、バックグラウンドで実行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. 以下の行を探します。

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

   c. 各行について、その行をコメント解除するために先頭の **#** 文字を削除します。

   d. 最初の行で、最初のインスタンスの **admin** を、**user1** などの希望するユーザー名に変更します。

e. 同じ行で、2 番目のインスタンスの **admin** を、そのユーザーのパスワードに変更します (例:**passw0rd**)。
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から閉じます。

6. Fuse を起動するには、Linux/Unix で **bin/fuse**、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 のオフライン実行について

Apache Karaf コンテナーを、インターネット接続なしでオフラインモードで実行できます。ただし、コンテナーにカスタムアプリケーションをデプロイする場合、これらのアプリケーションと共にコンテナーをオフラインモードで実行するためには、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しないといけない場合があります。

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、Apache Karaf コンテナーをデフォルト設定で実行するために必要な依存関係です。

- **ビルド時の依存関係** は、カスタムアプリケーションのビルドに必要な依存関係で、これにはサードパーティーのライブラリーが含まれます。

以下は、オフラインモードで実行できることと、(インターネット接続が利用できる) オンラインモードで何を実行する必要があるかの概要です。

- **Apache Karaf コンテナーをそのデフォルト設定で実行する** ことはオフラインモードでサポートされています。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任意の機能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)、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、一般的にオフラインモードでは **サポートされません**。最小限のビルド時依存関係のセットを持つアプリケーションをオフラインでデプロイできる場合があります。ただし、カスタムアプリケーションについては、通常、Apache Maven で JAR ファイルをダウンロードできるように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。

関連情報

「ローカル Maven リポジトリーの使用」

1.4. オプションでのスタンドアロン APACHE ディストリビューションの使用

Red Hat Fuse は、ダウンロードする追加パッケージを提供します。これには、Apache Camel および Apache CXF の標準ディストリビューションが含まれます。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 **7.12.0** を選択します。
4. Red Hat Fuse 7.12.0 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、ネストされた以下のアーカイブファイルが含まれます。
   - **apache-camel-2.23.2.fuse-7_11_1-00015-redhat-00002.zip**
   - **apache-cxf-3.3.6.fuse-7_11_1-00015-redhat-00002.zip**
5. これらのファイルを目的の場所にコピーし、プラットフォームに適したユーティリティーを使用してデプロイメントします。

警告

アーカイブファイルを、パス名にスペースのあるディレクトリーにデプロイメントしないでください。たとえば、**C:\Documents and Settings\Greco Roman\Desktop\fuse** にデプロイメントしないでください。

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. 機能およびバンドルへのパッチ適用

パッチは、Fuse on Apache Karaf インストールにあるファイルの更新バージョンが含まれる ZIP アーカイブです。これには以下が含まれます。

- バンドル: 最も一般的で、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が機能オーバーライドファイルを変更する可能性があります。このファイルは **$FUSE_HOME/etc/org.apache.karaf.features.xml** です。これにより、特定の機能のバンドルをアップグレードしてホットフィックス方式で機能定義を変更したり、特定の機能が追加のバンドルを使用するようにしたりできます。

アップグレード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に対する修正が含まれます。これは、現在の Red Hat Fuse ディストリビューションに適用されることを想定しています。その主な目的は、既存のディストリビューションのバンドルとライブラリーの一部を更新することです。
- アップグレード: Fuse on Apache Karaf のアップグレードメカニズムを使用すると、更新されたバージョンの Fuse on Karaf を再インストールする必要なく、修正を Apache Karaf コンテナーに適用できます。アップグレードにより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、および静的ファイルを含む任意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。以下の手順を使用して、Red Hat Fuse on Apache Karaf をアップグレードすることもできます。アップグレードの詳細は、Apache Karaf での Fuse のアップグレード 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. 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ます。
2. アップグレードする前に、Fuse on Apache Karaf インストールの完全バックアップを作成します。

3. 端末を開き、Apache Karaf サーバーで Fuse を起動します。

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

4. **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。たとえば、patch-xxx.zip パッチファイルを追加するには、次のように入力します。

```
karaf@root()> patch:add 'file:///Downloads/patch-xxx.zip'
[name]              [installed] [rollup]   [description]
my-patch-x          false       false      my-patch-x
```

5. **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る変更のログが生成されますが、実際にはコンテナーに何の変更も加えません。シミュレーションログを確認し、これらの変更を確認します。

6. **patch:list** コマンドを入力し、追加されたパッチのリストを表示します。このリストで、[name] 見出しの下にあるエントリーはパッチ ID です。

```
patch:list
[name]                  [installed]   [description]
my-patch-x                false
```

7. **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。

```
patch:install my-patch-x
```

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.
```

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。

以下のトピックでは、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」
- 「Maven への Red Hat リポジトリーの追加」
- 「ローカル Maven リポジトリーの使用」
- 「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構築します。

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。
   デフォルトの動作では、プロジェクトのビルド中、Maven は外部リポジトリーを検索し、必要なアーティファクトをダウンロードします。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 リポジトリーの使用」 を参照してください。

3.2. MAVEN への RED HAT リポジトリーの追加

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、ユーザーのホームディレクトリーの **.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** にあるシステムレベルの **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把握している。

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 リポジトリーの使用

インターネットへ接続せずに Apache Karaf 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実行時に、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 は以下の場所で **settings.xml** を探します。

- 指定の URL を検索します。
- 見つからない場合は **${user.home}/.m2/settings.xml** を検索します。

- 見つからない場合は **${maven.home}/conf/settings.xml** を検索します。
- 見つからない場合は **${M2_HOME}/conf/settings.xml** を検索します。
- どの場所に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、ローカルリポジトリーで現在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、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は、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、ミラーが使用されません。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**
  たとえば、**-DmavenMirrorUrl=my-mirror::http://mirror.net/repository** となります。
- **-DmavenMirrorUrl=mirrorUrl**
  たとえば、**-DmavenMirrorUrl=http://mirror.net/repository** となります。この例では、<mirror> の <id> は mirror 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** が基本的なビルディングブロックです。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

Maven は、**packaging** の値、または **packaging** 値と **classifier** 値の両方を使用して基本的なコーディネートを拡張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。たとえば、**org.fusesource.example** です。

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 つの部分を使用でき、最後の部分には数字以外の文字を使用できます。たとえば **1.0-SNAPSHOT** の場合は、最後の部分が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。

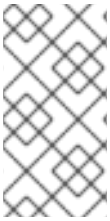
次に示すように、アーティファクトの POM ファイル内の要素で、アーティファクトのグループ ID、アーティファクト ID、パッケージング、およびバージョンを定義します。

```
<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

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。バンドルは特定タイプの JAR ファイルであり、**jar** はデフォルトの Maven パッケージタイプであるためです。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。