Red Hat Fuse 7.9

Fuse on JBoss EAP のスタートガイド

Red Hat Fuse on EAP を今すぐ使用する

Last Updated: 2021-08-16

Red Hat Fuse 7.9 Fuse on JBoss EAP のスタートガイド

Red Hat Fuse on EAP を今すぐ使用する

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

Fuse on JBoss Enterprise Application Platform でアプリケーションの構築を開始します。

目次

前書き

Fuse を使用するには、JBoss EAP コンテナーのファイルをダウンロードおよびインストールする必要があります。本書では、初めて Fuse アプリケーションをインストール、開発、および構築するための情報および手順を提供します。

- 1章*Fuse on JBoss EAP の使用*
- 2章*Maven のローカルでの設定*

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 FUSE ON JBOSS EAP の使用

本章では、Fuse on JBoss EAP を紹介し、JBoss EAP コンテナーで初めて Fuse アプリケーションをインストール、開発、およびビルドする方法を説明します。

詳細は以下のトピックを参照してください。

- 「Fuse on JBoss EAP」
- 「Fuse on JBoss EAP のインストール」
- 「JBoss EAP で最初の Fuse アプリケーションをビルド」

1.1. FUSE ON JBOSS EAP

Eclipse Foundation の Jakarta EE の技術 (旧名称 Java EE) をベースとした JBoss Enterprise Application Platform (EAP) は、当初はエンタープライズアプリケーション開発のユースケースに対応するために作成されました。JBoss EAP は、サービスおよび標準化された Java API (永続性、メッセージング、セキュリティーなど) を実装する明確に定義されたパターンを特徴としています。近年ではディペンデンシーインジェクション (依存性の注入) の CDI の導入や、エンタープライズ Java Bean の簡素化されたアノテーションの導入により、この技術はより軽量化されました。

このコンテナー技術の特徴は次のとおりです。

- 特にスタンドアロンモードでの実行に適しています。
- 事前設定された多くの標準サービス (永続性、メッセージング、セキュリティーなど) をそのまま使用できます。
- 通常、アプリケーション WAR は小型および軽量です (多くの依存関係はコンテナーに事前インストールされているため)。
- 標準化された後方互換性のある Java API。

1.2. FUSE ON JBOSS EAP のインストール

Red Hat カスタマーポータルから Fuse 7.9 on JBoss EAP 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 JBoss EAP 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。

前提条件

- Red Hat カスタマーポータル のフルサブスクリプションアカウントが必要になります。
- カスタマーポータルにログインする必要があります。
- JBoss EAP がダウンロードされている必要があります。
- Fuse on JBoss EAP がダウンロードされている必要があります。
- CodeReady Studio インストーラー がダウンロードされている必要があります。

手順

1. 次のように、シェルプロンプトから JBoss EAP インストーラーを実行します。

```
java -jar DOWNLOAD_LOCATION/jboss-eap-7.4.0-installer.jar
```

2. インストール中、以下を行います。
   a. 契約条件に同意します。
   b. JBoss EAP ランタイムのインストールパス **EAP_INSTALL** を選択します。
   c. 管理ユーザーを作成し、後で必要になる管理ユーザーのクレデンシャル情報を注意して書き留めておきます。
   d. 残りの画面では、デフォルト設定を使用できます。
3. シェルプロンプトを開き、**EAP_INSTALL** ディレクトリーに移動します。
4. **EAP_INSTALL** ディレクトリーから、以下のように Fuse on EAP のインストーラーを実行します。

```
java -jar DOWNLOAD_LOCATION/fuse-eap-installer-7.9.0.jar
```

5. 以下のように、CodeReady Studio インストーラーを実行します。

```
java -jar DOWNLOAD_LOCATION/codereadystudio-12.19.1.GA-installer-standalone.jar
```

6. インストール中、以下を行います。
   a. 契約条件に同意します。
   b. インストールパスを選択します。
   c. Java 8 JVM を選択します。
   d. **Select Platforms and Servers** ステップで、**Add** をクリックして **EAP_INSTALL** ディレクトリーの場所を確認し、JBoss EAP ランタイムを設定します。
   e. **Select Additional Features to Install** で **Red Hat Fuse Tooling** を選択します。
7. CodeReady Studio が起動します。**Searching for runtimes** ダイアログが表示されたら **OK** をクリックして JBoss EAP ランタイムを作成します。
8. **(任意手順)** コマンドラインから Apache Maven を使用するには、「Setting up Maven locally」の説明どおりに Maven をインストールおよび設定する必要があります。

注記

CodeReady Studio のみを使用する場合、CodeReady Studio にはMaven が事前インストールおよび設定されているため、厳密的には Maven をインストールする必要はありません。しかし、コマンドラインから Maven を呼び出す場合は、インストールを行う必要があります。

1.3. JBOSS EAP で最初の FUSE アプリケーションをビルド

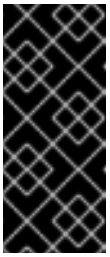
次の手順は、JBoss EAP で初めて Fuse アプリケーションをビルドする場合に便利です。

前提条件

- Red Hat カスタマーポータル のフルサブスクリプションアカウントが必要になります。
- カスタマーポータルにログインする必要があります。
- ダウンロードした Fuse on JBoss EAP が正常にインストールされている必要があります。
- CodeReady Studio インストーラー がダウンロードされ、正しくインストールされている必要があります。

手順

1. CodeReady Studio で以下のように新しいプロジェクトを作成します。
   a. **File→New→Fuse Integration Project** と選択します。
   b. **Project Name** フィールドに **eap-camel** を入力します。
   c. **Next** をクリックします。
   d. **Select a Target Environment** ペインで以下の設定を選択します。
      - **Standalone** をデプロイメントプラットフォームとして選択します。
      - **Wildfly/Fuse on EAP** をランタイム環境として選択し、**Runtime (optional)** ドロップダウンメニューを使用して**JBoss EAP 7.x Runtime** サーバーをターゲットランタイムとして選択します。
   e. ターゲットランタイムの選択後、**Camel Version** が自動的に選択され、フィールドがグレーアウトされます。
   f. **Next** をクリックします。
   g. **Advanced Project Setup** ペーンで **Spring Bean - Spring DSL**テンプレートを選択します。
   h. **Finish** をクリックします。



重要

CodeReady Studio で初めて Fuse プロジェクトをビルドする場合、ウィザードがプロジェクトの生成を完了するまで **数分** かかることがあります。これは、リモート Maven リポジトリーから依存関係をダウンロードするためです。プロジェクトがバックグラウンドでビルドされている間は、ウィザードを中断したり、CodeReady Studio を閉じたりしないでください。

   i. 関連する Fuse Integration パースペクティブを開くように要求された場合は、**Yes** をクリックします。
   j. CodeReady Studio が必要なアーティファクトをダウンロードし、バックグラウンドでプロジェクトをビルドする間待機します。
2. 以下のように、プロジェクトをサーバーにデプロイします。
   a. サーバーが起動していない場合は、**Servers** ビュー (Fuse Integration パースペクティブの右下隅) で **Red Hat JBoss EAP 7.3 Runtime** サーバーを選択し、緑色の矢印をクリックして起動します。

b. **Console** ビューに以下のようなメッセージが表示されるまで待機します。

> 14:47:07,283 INFO [org.jboss.as] (Controller Boot Thread) WFLYSRV0025: JBoss EAP 7.3.2.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. サーバーが起動した後、**Servers** ビューに切り替え、サーバーを右クリックしてコンテキストメニューで **Add and Remove** を選択します。

d. **Add and Remove** ダイアログで **eap-camel** プロジェクトを選択し、**Add >** をクリックします。

e. **Finish** をクリックします。

3. 以下のように、プロジェクトが動作していることを確認します。

a. URL http://localhost:8080/camel-test-spring?name=Kermit に移動し、**eap-camel** プロジェクトで実行されているサービスにアクセスします。

b. ブラウザーウインドウには、**Hello Kermit** が応答として表示されるはずです。

4. 以下のようにプロジェクトをアンデプロイします。

a. **Servers** ビューで **Red Hat JBoss EAP 7.3 Runtime** サーバーを選択します。

b. サーバーを右クリックし、コンテキストメニューで **Add and Remove** を選択します。

c. **Add and Remove** ダイアログで **eap-camel** プロジェクトを選択し、**< Remove** をクリックします。

d. **Finish** をクリックします。

第2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」

2.1. 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。

2.2. 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>
```

2.3. 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。

2.4. 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。

2.4.1. 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。

2.4.2. 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。

2.4.3. 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。

2.4.4. 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。

2.5. 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。

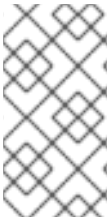
アーティファクトの POM ファイルの要素は、以下のようにアーティファクトのグループ ID、アーティファクト ID、パッケージング、およびバージョンを定義します。

```
<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。