



Red Hat Fuse 7.4

スタートガイド

Red Hat Fuse を今すぐ使用する

Red Hat Fuse 7.4 スタートガイド

Red Hat Fuse を今すぐ使用する

法律上の通知

Copyright © 2021 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.

概要

Fuse on Spring Boot、Fuse on Apache Karaf、および Fuse on JBoss Enterprise Application Platform を今すぐ使用します。

目次

前書き	3
第1章 FUSE ON SPRING BOOT の使用	4
1.1. FUSE ON SPRING BOOT	4
1.2. ブースタープロジェクトの生成	4
1.3. ブースタープロジェクトのビルド	5
第2章 FUSE ON KARAF の使用	8
2.1. FUSE ON KARAF	8
2.2. FUSE ON KARAF のインストール	8
2.3. KARAF で初めて FUSE アプリケーションをビルド	9
第3章 FUSE ON JBOSS EAP の使用	12
3.1. FUSE ON JBOSS EAP	12
3.2. FUSE ON JBOSS EAP のインストール	12
3.3. JBOSS EAP で最初の FUSE アプリケーションをビルド	14
第4章 MAVEN のローカルでの設定	16
4.1. MAVEN 設定の準備	16
4.2. RED HAT リポジトリを MAVEN へ追加	16
4.3. ローカル MAVEN リポジトリの追加	18
4.4. MAVEN アーティファクトおよびコーディネート	18

前書き

Fuse を使用するには、Spring Boot、JBoss EAP、または Apache Karaf に関わらず、該当のコンテナ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、これらの各コンテナに初めて 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 Spring Boot の使用](#)
- [2章 Fuse on Karaf の使用](#)
- [3章 Fuse on JBoss EAP の使用](#)
- [4章 Maven のローカルでの設定](#)

第1章 FUSE ON SPRING BOOT の使用

Spring Boot で Fuse アプリケーションを開発するには、最初に Spring Boot で実行する Fuse のサンプルブースタープロジェクトを生成およびビルドします。詳細は以下を参照してください。

- [「Fuse on Spring Boot」](#)
- [「ブースタープロジェクトの生成」](#)
- [「ブースタープロジェクトのビルド」](#)

1.1. FUSE ON SPRING BOOT

[Spring Boot](#) は、よく知られている Spring コンテナがさらに進化したものです。Spring Boot コンテナの特徴は、コンテナ機能が独立してデプロイできる小さなチャンクに分割されていることです。これにより、特定のサービスに特化し、小さなフットプリントでコンテナをデプロイできます。これは、マイクロサービスアーキテクチャーのパラダイムに対応するために必要になります。

このコンテナ技術の特徴は次のとおりです。

- 特に、スケーラブルなクラウドプラットフォーム (Kubernetes および OpenShift) での実行に適しています。
- マイクロサービスアーキテクチャーに適した小さなフットプリント。
- 「**設定より規約**」 (Convention over Configuration) 向けに最適化されています。
- アプリケーションサーバーは必要ありません。Spring Boot アプリケーション Jar を直接 JVM で実行できます。

1.2. ブースタープロジェクトの生成

Fuse ブースタープロジェクトは、スタンドアロンアプリケーションの実行を手助けする開発者向けのプロジェクトです。ここでは、ブースタープロジェクトの1つである Circuit Breaker ブースターの生成手順を説明します。この演習では、Fuse on Spring Boot の便利なコンポーネントを使用します。

[Netflix/Hystrix](#) サーキットブレーカーは、ネットワーク接続の中断や、バックエンドサービスの一時的な利用停止に分散アプリケーションが対応できるようにします。サーキットブレーカーパターンの基本概念は、バックエンドサービスが一時的に利用できなくなった場合に、依存するサービスの損失が自動的に検出され、代替動作をプログラムで作成できることです。

Fuse サーキットブローカーブースターは2つの関連サービスで構成されます。

- 呼び名を返すバックエンドサービスである **name** サービス。
- 名前を取得するよう **name** サービスを呼び出し、文字列 **Hello, NAME** を返すフロントエンドサービスである **greetings** サービス。

このブースターデモンストレーションでは、Hystrix サーキットブレーカーは **greetings** サービスと **name** サービスとの間に挿入されます。バックエンドの **name** サービスが利用できなくなると、**name** サービスは再起動するまでの間にブロックされるのではなく、**greetings** サービスが代替動作にフォールバックして即座にクライアントに応答することができます。

前提条件

- [Red Hat Developer Platform](#) にアクセスできる必要があります。
- サポートされるバージョンの Java Developer Kit (JDK) が必要です。詳細は「[Red Hat Fuse でサポートされる構成](#)」を参照してください。
- [Apache Maven 3.3.x](#) 以上が必要です。

手順

1. <https://developers.redhat.com/launch> に移動します。
2. **START** をクリックします。
ランチャーウィザードによって、Red Hat アカウントにログインするよう要求されます。
3. **Log in or register** ボタンをクリックし、ログインします。
4. **Launcher** ページで **Deploy an Example Application** ボタンをクリックします。
5. **Create Example Application** ページで **Create Example Application as** フィールドに名前 **fuse-circuit-breaker** を入力します。
6. **Select an Example** をクリックします。
7. **Example** ダイアログで、**Circuit Breaker** オプションを選択します。**Select a Runtime** ドロップダウンメニューが表示されます。
 - a. **Select a Runtime** ドロップダウンメニューで **Fuse** を選択します。
 - b. バージョンのドロップダウンメニューで **7.3.0 (Red Hat Fuse)** を選択します。**2.21.2 (Community)** バージョンは選択しないでください。
 - c. **Save** をクリックします。
8. **Create Example Application** ページで **Download** をクリックします。
9. **Your Application is Ready** ダイアログが表示されたら、**Download.zip** をクリックします。ブラウザが生成されたブースタープロジェクト (ZIP ファイルとしてパッケージ) をダウンロードします。
10. アーカイブユーティリティーを使用して、生成されたプロジェクトをローカルファイルシステムの任意の場所に展開します。

1.3. ブースタープロジェクトのビルド

これらの手順は、Fuse on Spring Boot で Circuit Breaker ブースターをビルドする方法を説明します。

前提条件

- [Red Hat Developer Portal](#) からブースタープロジェクトの生成およびダウンロードが完了している必要があります。
- サポートされるバージョンの Java Developer Kit (JDK) が必要です。詳細は「[Red Hat Fuse でサポートされる構成](#)」を参照してください。
- [Apache Maven 3.3.x](#) 以上が必要です。

手順

1. シェルプロンプトを開き、Maven を使用してコマンドラインからプロジェクトをビルドします。

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

```
mvn clean package
```

Maven によってプロジェクトがビルドされた後、**Build Success** メッセージが表示されます。

2. 新しいシェルプロンプトを開き、以下のように name サービスを起動します。

```
cd name-service
```

```
mvn spring-boot:run -DskipTests -Dserver.port=8081
```

Spring Boot が起動すると、以下のような出力が表示されます。

```
...
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. 新しいシェルプロンプトを開き、以下のように greetings サービスを起動します。

```
cd greetings-service
```

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

Spring Boot が起動すると、以下のような出力が表示されます。

```
...
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 サービスは <http://localhost:8080/camel/greetings> URL で REST エンドポイントを公開します。

- web ブラウザーで URL を開くか、別のシェルプロンプトで以下の **curl** コマンドを入力して、REST エンドポイントを呼び出します。

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

応答は次のとおりです。

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

- Camel Hystrix によって提供されるサーキットブレーカー機能を実証するには、name サービスが実行されているシェルプロンプトウィンドウで **Ctrl-C** を入力し、バックエンド name サービスを中止します。

これで name サービスが利用できなくなるため、呼び出されたときに greetings サービスがハングしないよう、サーキットブレーカーが作動します。

- web ブラウザーで <http://localhost:8080/camel/greetings> を開くか、別のシェルプロンプトウィンドウに以下の **curl** コマンドを入力して、greetings REST エンドポイントを呼び出します。

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

応答は次のとおりです。

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

greetings サービスが実行されているウィンドウで、ログに以下のメッセージシーケンスが表示されます。

```
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!!!!
```

- この例に関する詳細は、**greetings-service** の実行中に <http://localhost:8080/> で **Circuit Breaker - Red Hat Fuse** ページを開いてください。このページには、サーキットブレーカーの状態を監視する Hystrix ダッシュボードへのリンクが含まれます。

第2章 FUSE ON KARAF の使用

ここで説明する情報や手順は、Fuse On Karaf について学び、Karaf コンテナで初めて Fuse アプリケーションをインストール、開発、およびビルドするのに役立ちます。以下を参照してください。

- [「Fuse on Karaf」](#)
- [「Fuse on Karaf のインストール」](#)
- [「Karaf で初めて Fuse アプリケーションをビルド」](#)

2.1. FUSE ON KARAF

Apache Karaf は、OSGi Alliance の [OSGi 標準](#) をベースとしています。OSGi は電気通信業界で開発され、[ホットコードスワッピング](#) と呼ばれる機能により、サーバーをシャットダウンする必要がなくすぐにアップグレードできる、ゲートウェイサーバーの開発に使用されました。その後、OSGi コンテナ技術はその他多くの用途に使用され、一般的に [Eclipse IDE](#) などのモジュール化されたアプリケーションで使用されます。

このコンテナ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。
- モジュール化 (OSGi バンドル) の強力なサポート、および高度なクラスローディングサポート。
- 1つのコンテナに複数のバージョンの依存関係を一緒にデプロイできます (ただし、注意して行う必要があります)。
- ホットコードスワッピングにより、コンテナをシャットダウンせずにモジュールをアップグレードまたは置き換えられます。これは独自の機能ですが、適切に行うには多大な労力が必要になります。

2.2. FUSE ON KARAF のインストール

Red Hat カスタマーポータルから Fuse 7.4 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 Karaf コンテナ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。
- [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。
- [Fuse on Karaf 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. ダウンロードした Fuse on Apache Karaf の **.zip** アrchive ファイルを、ファイルシステム **FUSE_INSTALL** の任意の場所に展開します。
2. Fuse ランタイムに管理ユーザーを追加します。

- a. テキストエディターで **FUSE_INSTALL/etc/users.properties** ファイルを開きます。
 - b. **#admin = admin** で始まる行の最初の **#** 文字を削除します。
 - c. **#g\:admingroup** で始まる行の最初の **#** 文字を削除します。
 - d. ユーザーエントリーのユーザー名 **USERNAME** とパスワード **PASSWORD** をカスタマイズし、以下のようなユーザーエントリーおよび管理グループエントリーにします (連続した行)。


```
USERNAME = PASSWORD,_g\:admingroup
_g\:admingroup = group,admin,manager,viewer,systembundles,ssh
```
 - e. **etc/users.properties** ファイルを保存します。
3. 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-12.12.0.GA-installer-standalone.jar
```

4. 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** で、**Add** をクリックして **FUSE_INSTALL** ディレクトリー の場所を確認し、Fuse on Karaf 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。
5. 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 Fuse on Karaf ランタイムを作成します。
6. (任意手順): コマンドラインから Apache Maven を使用するには、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、Maven をインストールする必要は厳密的にはありません。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。

2.3. KARAF で初めて FUSE アプリケーションをビルド

次の手順は、Karaf 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。

- [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。
- ダウンロードした [Fuse on Karaf インストーラー](#) が正常に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. **fuse-camel-cbr** 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。
 - **Karaf/Fuse on Karaf** 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 **Red Hat JBoss Middleware > Red Hat Fuse 7+ 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** ペインで **Beginner**→**Content Based Router - Blueprint DSL** テンプレートを選択します。
 - h. **Finish** をクリックします。
 - 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。



重要

CodeReady Studio で初めて Fuse プロジェクトをビルドする場合、リモート Maven リポジトリから依存関係をダウンロードするため、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。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。

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左下隅) で **fuse-karaf-7.4.0.fuse-740028-redhat-00001 Runtime Server** 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。



注記

Warning: The authenticity of host 'localhost' can't be established.というダイアログが表示されたら、**Yes** をクリックしてサーバーに接続し、Karaf コンソールにアクセスします。

- 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。

```
Karaf started in 1s. Bundle stats: 12 active, 12 total
```

- 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 **fuse-camel-cbr** プロジェクトを選択し、**Add** > ボタンをクリックします。
- e. **Finish** をクリックします。
- f. **Terminal** ビューに移動し、**bundle:list | tail** を入力して、プロジェクトの OSGi バンドルが起動したかどうかをチェックします。以下のような出力が表示されるはずです。

```
...
228 | Active | 80 | 1.0.0.201505202023 | org.osgi:org.osgi.service.j
232 | Active | 80 | 1.0.0.SNAPSHOT | Fuse CBR Quickstart
```



注記

Camel ルートが起動すると、すぐに **fuse-camel-cbr** プロジェクトにディレクトリー **work/cbr/input** が作成されます。

3. **Project Explorer** ビューで **Refresh** をクリックし、新たに作成された **work/cbr/input** ディレクトリーを表示します。
4. プロジェクトの **src/main/data** ディレクトリーにあるファイルを **work/cbr/input** ディレクトリーにコピーします。
5. しばらく待ってから再度 **Project Explorer** ビューを更新し、国別に分類された同じファイルが **work/cbr/output** ディレクトリーにあることを確認します。
- a. **work/cbr/output/others** の **order1.xml**
 - b. **work/cbr/output/uk** の **order2.xml** および **order4.xml**
 - c. **work/cbr/output/us** の **order3.xml** および **order5.xml**
6. 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 Fuse 7+ Runtime Server** 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 **fuse-camel-cbr** プロジェクトを選択し、< **Remove** ボタンをクリックします。
 - d. **Finish** をクリックします。

第3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。

以下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」](#)

3.1. 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。

3.2. 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 7.4 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](#) および [JBoss EAP 7.2 Update 01](#) 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-7.2.0-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 JBoss EAP 7.2 Update 01 を適用します。以下に例を示します。

```
bin/jboss-cli.sh "patch apply jboss-eap-7.2.1-patch.zip"
```

5. **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-7.4.0.jar
```

6. 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-12.12.0.GA-installer-standalone.jar
```

7. 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。
8. 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。
9. (任意手順): コマンドラインから Apache Maven を使用するには、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。

3.3. 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。
- [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。
- 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。

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左下隅) で **Red Hat JBoss EAP 7.2 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 7.2.0.GA (WildFly Core 6.0.11.Final-redhat-00001) started in 13948ms - Started 495 of 680 services (326 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. ブラウザーウィンドウには、**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 7.2 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4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 リポジトリを Maven へ追加」](#)
- [「ローカル Maven リポジトリの追加」](#)
- [「Maven アーティファクトおよびコーディネート」](#)

4.1. 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 リポジトリの追加」](#) を参照してください。

4.2. RED HAT リポジトリを MAVEN へ追加

Red Hat Maven リポジトリ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>
```

```

    </pluginRepositories>
  </profile>
</profiles>

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

</settings>

```

4.3. ローカル MAVEN リポジトリの追加

インターネットへ接続せずに Apache Karaf コンテナを実行し、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。

4.4. 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できます。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。

groupId

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。