Red Hat Fuse 7.6

시작하기

Red Hat Fuse로 빠른 시작!

Last Updated: 2023-08-11

Red Hat Fuse 7.6 시작하기

Red Hat Fuse로 빠른 시작!

법적 공지

초록

Fuse on Spring Boot, Fuse on Apache Karaf, Fuse on JBoss Enterprise Application Platform을 사용해 보십시오.

차례

머리말 **3**

1장. FUSE ON SPRING BOOT 시작하기 **4**
1.1. FUSE ON SPRING BOOT 정보 4
1.2. 부스터 프로젝트 생성 4
1.3. 부스터 프로젝트 빌드 5

2장. FUSE ON KARAF 시작하기 **8**
2.1. FUSE ON KARAF 정보 8
2.2. KARAF에 FUSE 설치 9
2.3. KARAF에서 첫 번째 FUSE 애플리케이션 빌드 11

3장. JBOSS EAP에서 FUSE 시작하기 **15**
3.1. JBOSS EAP의 FUSE 정보 15
3.2. JBOSS EAP에 FUSE 설치 15
3.3. JBOSS EAP에서 첫 번째 FUSE 애플리케이션 빌드 18

4장. MAVEN을 로컬로 설정 **21**
4.1. MAVEN 설정 준비 21
4.2. MAVEN에 RED HAT 리포지토리 추가 22
4.3. 로컬 MAVEN 리포지토리 사용 23
4.4. 환경 변수 또는 시스템 속성을 사용하여 MAVEN 미러 설정 24
4.5. MAVEN 아티팩트 및 좌표 정보 26

머리말

Fuse를 시작하려면 Spring Boot, JBoss EAP 또는 Apache Karaf와 관계없이 원하는 컨테이너에 대한 파일을 다운로드하여 설치해야 합니다. 여기에서는 각 컨테이너를 위한 첫 번째 Fuse 애플리케이션을 설치, 개발 및 구축하는 방법을 안내합니다.

- 1장. *Fuse on Spring Boot 시작하기*
- 2장. *Fuse on Karaf 시작하기*
- 3장. *JBoss EAP에서 Fuse 시작하기*
- 4장. *Maven을 로컬로 설정*

1장. FUSE ON SPRING BOOT 시작하기

Spring Boot에서 Fuse 애플리케이션을 개발하려면 Spring Boot에서 실행되는 Fuse 샘플 부스터 프로젝트를 생성하고 빌드하여 시작하십시오. 다음 주제에서는 세부 정보를 제공합니다.

- 1.1절. “Fuse on Spring Boot 정보”
- 1.2절. “부스터 프로젝트 생성”
- 1.3절. “부스터 프로젝트 빌드”

1.1. FUSE ON SPRING BOOT 정보

Spring Boot 는 잘 알려진 Spring 컨테이너의 진화입니다. Spring Boot 컨테이너의 고유한 품질은 컨테이너 기능이 작은 청크로 분할되어 독립적으로 배포할 수 있다는 것입니다. 이를 통해 특정 종류의 서비스에 맞게 작은 설치 공간을 갖춘 컨테이너를 배포할 수 있으며 *마이크로 서비스 아키텍처* 의 패러다임에 맞게 정확히 필요한 컨테이너를 배포할 수 있습니다.

이 컨테이너 기술의 특징은 다음과 같습니다.

- 확장 가능한 클라우드 플랫폼(Kubernetes 및 OpenShift)에서 실행하는 데 특히 적합합니다.
- 작은 풋프린트(마이크로 서비스 아키텍처의 경우)입니다.
- *구성에 대한 규칙에* 최적화되어 있습니다.
- 애플리케이션 서버가 필요하지 않습니다. JVM에서 직접 Spring Boot 애플리케이션 Jar를 실행할 수 있습니다.

1.2. 부스터 프로젝트 생성

개발자가 독립 실행형 애플리케이션을 실행할 수 있도록 Fuse booster 프로젝트가 있습니다. 여기에 제공된 지침은 부스터 프로젝트 인 회로 차단기 부스터 중 하나를 생성하는 방법을 안내합니다. 이 실습에서는 Spring Boot에서 Fuse의 유용한 구성 요소를 보여줍니다.

넷플릭스/Hystrix 회로 차단기를 사용하면 분산 애플리케이션이 네트워크 연결 및 백엔드 서비스의 임시 사용할 수 없는 중단을 처리할 수 있습니다. 회로 차단기 패턴의 기본 개념은 종속 서비스의 손실이 자동으로 감지되고 백엔드 서비스를 일시적으로 사용할 수 없는 경우 대체 동작이 프로그램될 수 있다는 것입니다.

Fuse 회로 차단기 부스터는 두 가지 관련 서비스로 구성됩니다.

- **이름** 서비스인 백엔드 서비스는 이름을 Cryostat로 반환합니다.
- **인사말** 서비스인 frontend 서비스는 **name** 서비스를 호출하여 이름을 가져온 다음 문자열 **Hello, NAME** 을 반환합니다.

이 부스터 데모에서 Hystrix 회로 차단기는 **인사말** 서비스와 **이름** 서비스 사이에 삽입됩니다. 백엔드 **이름** 서비스를 사용할 수 없게 되면 **인사말** 서비스가 대체 동작으로 대체되고, **이름** 서비스가 재시작될 때까지 기다리는 동안 차단되는 대신 즉시 클라이언트에 응답할 수 있습니다.

사전 요구 사항

- Red Hat Developer Platform 에 액세스할 수 있어야 합니다.

- 지원되는 JDK(Java Developer Kit) 버전이 있어야 합니다. 자세한 내용은 지원되는 구성 페이지를 참조하십시오.
- Apache Maven 3.3.x 이상이 있어야 합니다.

절차

1. https://developers.redhat.com/launch 로 이동합니다.
2. **START** 를 클릭합니다.
   시작 관리자 마법사에서 Red Hat 계정에 로그인하라는 메시지가 표시됩니다.
3. **로그인 또는 등록** 버튼을 클릭한 다음 로그인합니다.
4. **시작 관리자** 페이지에서 **예제 애플리케이션 배포** 버튼을 클릭합니다.
5. **Create Example Application** 페이지에서 Create **Example Application에 name, fuse-circuit-breaker 를 입력합니다**.
6. **Select an Example** 을 클릭합니다.
7. **예제** 대화 상자에서 **회로 차단기** 옵션을 선택합니다. **런타임 선택** 드롭다운 메뉴가 표시됩니다.
   a. **Select a Runtime** 드롭다운에서 **Fuse** 를 선택합니다.
   b. 버전 드롭다운 메뉴에서 **7.6(Red Hat Fuse)** 을 선택합니다(**2.21.2(Community)** 버전을 선택하지 마십시오.
   c. **저장**을 클릭합니다.
8. **예제 애플리케이션 생성** 페이지에서 **다운로드를** 클릭합니다.
9. **애플리케이션 준비** 상태 대화 상자가 표시되면 **Download.zip** 을 클릭합니다. 브라우저에서 생성된 booster 프로젝트를 다운로드합니다(ZIP 파일로 패키징됨).
10. 아카이브 유틸리티를 사용하여 생성된 프로젝트를 로컬 파일 시스템의 편리한 위치로 추출합니다.

1.3. 부스터 프로젝트 빌드

이 지침에서는 Spring Boot에서 Fuse on Spring Boot를 사용하여 회로 차단기를 빌드하는 방법을 안내합니다.

사전 요구 사항

- Red Hat 개발자 포털 을 통해 booster 프로젝트를 생성하고 다운로드해야 합니다.
- 지원되는 JDK(Java Developer Kit) 버전이 있어야 합니다. 자세한 내용은 지원되는 구성 페이지를 참조하십시오.
- Apache Maven 3.3.x 이상이 있어야 합니다.

절차

1. 쉘 프롬프트를 열고 Maven을 사용하여 명령줄에서 프로젝트를 빌드합니다.

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

```
mvn clean package
```

Maven이 프로젝트를 빌드하면 **Build Success** (빌드 성공) 메시지가 표시됩니다.

2. 다음과 같이 새 쉘 프롬프트를 열고 name 서비스를 시작합니다.

```
cd name-service
```

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

Spring Boot가 시작되면 다음과 유사한 출력이 표시됩니다.

```
...
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. 다음과 같이 새 쉘 프롬프트를 열고 인사말 서비스를 시작합니다.

```
cd greetings-service
```

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

Spring Boot가 시작되면 다음과 유사한 출력이 표시됩니다.

```
...
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)
```

인사말 서비스는 **http://localhost:8080/camel/greetings** URL에 REST 끝점을 노출합니다.

4. 웹 브라우저에서 URL을 열거나 다른 쉘 프롬프트를 열고 다음 **curl** 명령을 입력하여 REST 끝점을 호출합니다.

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

응답은 다음과 같습니다.

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

5. Camel Hystrix에서 제공하는 회로 차단기 기능을 설명하기 위해 이름 서비스가 실행 중인 쉘 프롬프트 창에서 **Ctrl-C** 를 입력하여 백엔드 이름 서비스를 종료합니다.
이제 이름 서비스를 사용할 수 없으므로 회로 차단기가 시작되어 인사말 서비스가 호출될 때 중단되지 않도록 합니다.

6. 웹 브라우저에서 **http://localhost:8080/camel/greetings** 를 열거나 다른 쉘 프롬프트 창에 다음 **curl** 명령을 입력하여 인사말 REST 끝점을 호출합니다.

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

응답은 다음과 같습니다.

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

인사말 서비스가 실행 중인 창에서 로그에 다음과 같은 메시지 시퀀스가 표시됩니다.

```
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. 이 예제에 대한 자세한 내용은 http://localhost:8080/ 에서 **Circuit Breaker - Red Hat Fuse**페이지를 엽니다. **이 페이지에는 회로 차단기의 상태를 모니터링하는 Hystrix 대시보드에 대한 링크가 포함되어 있습니다.**

2장. FUSE ON KARAF 시작하기

Fuse on Karaf에 대한 정보와 함께 **Karaf** 컨테이너에 첫 번째 **Fuse** 애플리케이션을 설치, 개발 및 구축하기 위해 다음과 같은 정보와 지침을 제공합니다. 자세한 내용은 다음 항목을 참조하십시오.

- 2.1절. “Fuse on Karaf 정보”
- 2.2절. “Karaf에 Fuse 설치”
- 2.3절. “Karaf에서 첫 번째 Fuse 애플리케이션 빌드”

2.1. FUSE ON KARAF 정보

Apache Karaf는 **OSGi Alliance**의 **OSGi** 표준을 기반으로 합니다. **OSGi**는 통신 업계에서 시작되었으며 서버를 종료하지 않고도 즉시 업그레이드할 수 있는 게이트웨이 서버를 개발하는 데 사용되었습니다. *결과적으로 **OSGi** 컨테이너 기술은 다양한 다른 용도로 사용되고 있으며 모듈형 애플리케이션(예: **Eclipse IDE**)에 널리 사용됩니다.*

이 컨테이너 기술의 특징은 다음과 같습니다.

- *독립 실행형 모드에서 실행하는 데 특히 적합합니다.*
- *정교한 클래스 로드를 지원하는 **OSGi** 번들(**OSGi** 번들)에 대한 강력한 지원.*
- *여러 버전의 종속성을 컨테이너에 나란히 배포할 수 있습니다(실제로 어느 정도 주의가 필요합니다).*
- *핫 코드 스왑을 통해 컨테이너를 종료하지 않고 모듈을 업그레이드하거나 교체할 수 있습니다. 이는 고유한 기능이지만 제대로 작동하려면 상당한 노력이 필요합니다.*

*참고: **Spring Dynamic Modules (Spring-DM) (Apache Karaf**의 **OSGi** 서비스 계층과 **Spring XML** 통합)는 지원되지 않습니다. 대신 블루프린트 프레임워크를 사용해야 합니다. 블루프린트 **XML**을 사용하면*

Spring 프레임워크에서 Java 라이브러리를 사용할 수 없습니다. 최신 버전의 Spring은 블루프린트와 호환됩니다.

2.2. KARAF에 FUSE 설치

Karaf의 Fuse 7.6용 표준 설치 패키지는 Red Hat 고객 포털에서 다운로드할 수 있습니다. Karaf 컨테이너의 표준 어셈블리를 설치하고 전체 Fuse 기술 스택을 제공합니다.

사전 요구 사항

- Red Hat 고객 포털에 는 전체 서브스크립션 계정이 필요합니다.
- 고객 포털에 로그인해야 합니다.
- CodeReady Studio 설치 관리자를 다운로드해야 합니다.
- Fuse on Karaf 설치 프로그램을 다운로드해야 합니다.

절차

1. Apache Karaf에서 Fuse에 대해 다운로드한 .zip 아카이브 파일의 압축을 풀고 파일 시스템인 FUSE_INSTALL 의 편리한 위치에 압축을 풉니다.
2. Fuse 런타임에 관리자 사용자를 추가합니다.
   a. 텍스트 편집기에서 FUSE_INSTALL/etc/users.properties 파일을 엽니다.
   b. # admin = admin 으로 시작하는 행 시작 시 # 문자를 삭제합니다.
   c. # _g_\:admingroup 으로 시작하는 행 시작 시 # 문자를 삭제합니다.
   d. 다음과 같은 사용자 항목 및 관리자 그룹 항목이 있도록 사용자 항목의 사용자 이름, USERNAME, password, PASSWORD 를 사용자 지정합니다.

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

e. *etc/users.properties 파일을 저장합니다.*

3. *다음과 같이 CodeReady Studio 설치 프로그램을 실행합니다.*

```
java -jar DOWNLOAD_LOCATION/codereadystudio-12.14.0.GA-installer-standalone.jar
```

4. *설치하는 동안:*

a. *이용 약관에 동의합니다.*

b. *선호하는 설치 경로를 선택합니다.*

c. *Java 8 JVM을 선택합니다.*

d. *플랫폼 및 서버 선택 단계에서 추가 를 클릭하고 FUSE_INSTALL 디렉터리의 위치를 검색하여 Karaf 런타임에서 Fuse를 구성합니다.*

e. *Select additional features to Install 단계에서 Red Hat Fuse Tooling 을 선택합니다.*

5. *CodeReady Studio가 시작됩니다. 런타임 검색 대화 상자가 표시되면 OK 를 클릭하여 Karaf 런타임에서 Fuse를 만듭니다.*

6. *(선택 사항) 명령줄에서 Apache Maven을 사용하려면 Maven을 설치하고 구성해야 합니다.*

참고

CodeReady Studio만 사용하는 경우 CodeReady Studio에 Maven이 사전 설치되어 구성되어 있으므로 Maven을 설치할 필요가 없습니다. 그러나 명령줄에서 Maven을 호출하려면 이 단계를 수행해야 합니다.

2.3. KARAF에서 첫 번째 FUSE 애플리케이션 빌드

이 지침에서는 Karaf에서 첫 번째 Fuse 애플리케이션을 빌드하는 데 도움이 됩니다.

사전 요구 사항

- Red Hat 고객 포털에 는 전체 서브스크립션 계정이 필요합니다.
- 고객 포털에 로그인해야 합니다.
- CodeReady Studio 설치 관리자를 다운로드해야 합니다.
- Karaf에 Fuse 를 다운로드하여 성공적으로 설치해야 합니다.

절차

1. CodeReady Studio에서 다음과 같이 새 프로젝트를 생성합니다.
   a. File(파일)NewNew Cryostat Integration Project 를 선택합니다.
   b. 프로젝트 이름 필드에 fuse-camel-cbr 을 입력합니다.
   c. 다음을 클릭합니다.
   d. 대상 환경 선택 창에서 다음 설정을 선택합니다.
      - 배포 플랫폼으로 Standalone 을 선택합니다.
      - 런타임 환경으로 Karaf/Fuse 를 선택하고 런타임 (선택 사항) 드롭다운 메뉴를 사용하여 fuse-karaf-7.6.0.fuse-760025-redhat-00001 런타임 서버를 대상 런타임으로 선택합니다.

e.

*대상 런타임을 선택하면 **Camel** 버전이 자동으로 선택되고 필드가 회색으로 표시됩니다.*

f.

다음을 클릭합니다.

g.

***Advanced Project Setup** 창에서 **Beginner CryostatContent Based Router - 블루프린트 DSL** 템플릿을 선택합니다.*

h.

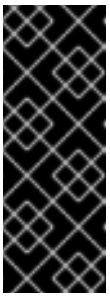
완료를 클릭합니다.

i.

*연결된 **Fuse** 통합 화면을 열라는 메시지가 표시되면 예를 클릭합니다.*

j.

***CodeReady Studio**가 필요한 아티팩트를 다운로드하고 백그라운드에서 프로젝트를 빌드하는 동안 기다립니다.*



중요

***CodeReady Studio**에서 **Fuse** 프로젝트를 처음 빌드하는 경우, 원격 **Maven** 리포지토리에서 종속성을 다운로드하므로 마법사가 프로젝트 생성을 완료하는 데 몇 분이 걸립니다. 프로젝트가 백그라운드에서 빌드하는 동안 마법사를 중단하거나 **CodeReady Studio**를 종료하지 마십시오.*

2.

다음과 같이 서버에 프로젝트를 배포합니다.

a.

*서버 보기(**Fuse Integration** 관점의 왼쪽 모서리)에서 서버가 아직 시작되지 않은 경우 **fuse-karaf-7.6.0.fuse-760025-redhat-00001 Runtime Server** 서버를 선택하고 녹색 화살표를 클릭하여 시작합니다.*

참고

*대화 상자가 표시되면 경고: **'localhost'** 호스트의 진위를 설정할 수 없습니다. 예 를 클릭하여 서버에 연결하고 **Karaf** 콘솔에 액세스합니다.*

b.

콘솔 보기에 다음과 같은 메시지가 표시될 때까지 기다립니다.

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

c.

서버가 시작되면 서버 보기로 다시 전환하고 서버 를 마우스 오른쪽 버튼으로 클릭하고 컨텍스트 메뉴에서 추가 및 제거를 선택합니다.

d.

*추가 및 제거 대화 상자에서 **fuse-camel-cbr** 프로젝트를 선택하고 **Add >** 버튼을 클릭합니다.*

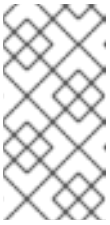
e.

완료를 클릭합니다.

f.

*터미널 보기로 이동하여 **bundle:list | tail** 을 입력하여 프로젝트의 **OSGi** 번들이 시작되었는지 확인할 수 있습니다. 다음과 같은 몇 가지 출력이 표시됩니다.*

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



참고

*Camel 경로가 시작되는 즉시 **Fuse** 설치에 디렉터리, **work/cbr/input** 가 생성됩니다(**fuse-camel-cbr** 프로젝트에없음).*

3.

*프로젝트의 **src/main/data** 디렉토리에서 찾은 파일을 **FUSE_INSTALL/work/cbr/input** 디렉터리에 복사합니다. 시스템 파일 브라우저(**Eclipse** 제외)에서 이 작업을 수행할 수 있습니다.*

4.

*잠시 기다린 다음 **FUSE_INSTALL/work/cbr/output** 디렉토리에서 국가별로 구성된 동일한 파일을 확인합니다.*

a.

work/cbr/output/others**의 **order1.xml

b.

work/cbr/output/uk**의 **order2.xml** 및 **order4.xml

c.

work/cbr/output/us의 order3.xml 및 order5.xml

5.

다음과 같이 프로젝트 배포를 취소합니다.

a.

*서버 보기에서 **Red Hat Fuse 7+ Runtime Server** 서버를 선택합니다.*

b.

서버를 마우스 오른쪽 버튼으로 클릭하고 컨텍스트 메뉴에서 추가 및 제거를 선택합니다.

c.

*추가 및 제거 대화 상자에서 **fuse-camel-cbr** 프로젝트를 선택하고 **<** 제거 버튼을 클릭합니다.*

d.

완료를 클릭합니다.

3장. JBOSS EAP에서 FUSE 시작하기

이 장에서는 Fuse on JBoss EAP를 소개하고 JBoss EAP 컨테이너에서 첫 번째 Fuse 애플리케이션을 설치, 개발 및 빌드하는 방법을 설명합니다.

자세한 내용은 다음 항목을 참조하십시오.

- 3.1절. “JBoss EAP의 Fuse 정보”
- 3.2절. “JBoss EAP에 Fuse 설치”
- 3.3절. “JBoss EAP에서 첫 번째 Fuse 애플리케이션 빌드”

3.1. JBOSS EAP의 FUSE 정보

Eclipse Foundation 의 Jakarta EE 기술(이전에는 Java EE)을 기반으로 하는 JBoss EAP(Enterprise Application Platform)는 엔터프라이즈 애플리케이션 개발을 위한 사용 사례를 해결하기 위해 원래 생성되었습니다. JBoss EAP는 서비스 및 표준화된 Java API(예: 지속성, 메시징, 보안 등)를 구현하기 위한 잘 정의된 패턴이 특징입니다. 최근 몇 년 동안 이 기술은 보다 경량화되었으며 엔터프라이즈 Java 빈을 위한 종속성 주입 및 단순화된 주석을 위한 CDI가 도입되었습니다.

이 컨테이너 기술의 특징은 다음과 같습니다.

- 독립 실행형 모드에서 실행하는 데 특히 적합합니다.
- 많은 표준 서비스(예: 지속성, 메시징, 보안 등) 사전 구성되어 즉시 제공됩니다.
- 애플리케이션 WAR는 일반적으로 작고 경량(많은 종속성이 컨테이너에 사전 설치됨)입니다.
- 표준화된 이전 호환 Java API.

3.2. JBOSS EAP에 FUSE 설치

JBoss EAP의 Fuse 7.6용 표준 설치 패키지는 Red Hat 고객 포털에서 다운로드할 수 있습니다. JBoss EAP 컨테이너의 표준 어셈블리를 설치하고 전체 Fuse 기술 스택을 제공합니다.

사전 요구 사항

- *Red Hat 고객 포털에 는 전체 서브스크립션 계정이 있어야 합니다.*
- *고객 포털에 로그인해야 합니다.*
- *JBoss EAP 및 JBoss EAP 7.2 업데이트 05 이 있어야 합니다.*
- *Fuse on JBoss EAP 를 다운로드해야 합니다.*
- *CodeReady Studio 설치 관리자를 다운로드해야 합니다.*

절차

1. *다음과 같이 셸 프롬프트에서 JBoss EAP 설치 프로그램을 실행합니다.*

```
java -jar DOWNLOAD_LOCATION/jboss-eap-7.2.0-installer.jar
```

2. *설치하는 동안:*
   a. *이용 약관에 동의합니다.*
   b. *JBoss EAP 런타임으로 선호하는 설치 경로 EAP_INSTALL 을 선택합니다.*
   c. *관리 사용자를 만들고 나중에 이러한 관리 사용자 자격 증명을 주의 깊게 기록해 두십시오.*
   d. *나머지 화면에서 기본 설정을 허용할 수 있습니다.*

3. *쉘 프롬프트를 열고 디렉터리를 **EAP_INSTALL** 으로 변경합니다.*

4. ***EAP_INSTALL** 디렉토리에서 **JBoss EAP 7.2 Update 05**를 적용합니다. 예를 들면 다음과 같습니다.*

```
bin/jboss-cli.sh "patch apply jboss-eap-7.2.x-patch.zip"
```

5. ***EAP_INSTALL** 디렉터리에서 다음과 같이 **EAP** 설치 프로그램에서 **Fuse**를 실행합니다.*

```
java -jar DOWNLOAD_LOCATION/fuse-eap-installer-7.6.0.jar
```

6. *다음과 같이 **CodeReady Studio** 설치 프로그램을 실행합니다.*

```
java -jar DOWNLOAD_LOCATION/codereadystudio-12.14.0.GA-installer-standalone.jar
```

7. *설치하는 동안:*

   a. *이용 약관에 동의합니다.*

   b. *선호하는 설치 경로를 선택합니다.*

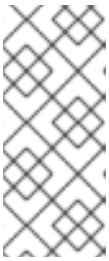
   c. ***Java 8 JVM**을 선택합니다.*

   d. ***Select Platforms and Servers** 단계에서 **Add** 를 클릭하고 **EAP_INSTALL** 디렉터리의 위치를 검색하여 **JBoss EAP** 런타임을 구성합니다.*

   e. ***Select additional features to Install** 단계에서 **Red Hat Fuse Tooling** 을 선택합니다.*

8. ***CodeReady Studio**가 시작됩니다. 런타임 검색 대화 상자가 표시되면 **OK** 를 클릭하여 **JBoss EAP** 런타임을 생성합니다.*

9. *(선택 사항) 명령줄에서 **Apache Maven**을 사용하려면 **Maven**을 설치하고 구성해야 합니다.*



참고

CodeReady Studio만 사용하는 경우 CodeReady Studio에 Maven이 사전 설치되어 구성되어 있으므로 Maven을 설치할 필요가 없습니다. 그러나 명령줄에서 Maven을 호출하려면 이 단계를 수행해야 합니다.

3.3. JBOSS EAP에서 첫 번째 FUSE 애플리케이션 빌드

이 지침에서는 JBoss EAP에서 첫 번째 Fuse 애플리케이션을 빌드하는 데 도움이 됩니다.

사전 요구 사항

- Red Hat 고객 포털에 는 전체 서브스크립션 계정이 필요합니다.
- 고객 포털에 로그인해야 합니다.
- JBoss EAP에 Fuse 를 다운로드하여 성공적으로 설치해야 합니다.
- CodeReady Studio 설치 프로그램을 다운로드하여 성공적으로 설치해야합니다.

절차

1. CodeReady Studio에서 다음과 같이 새 프로젝트를 생성합니다.
   a. File(파일)NewNew Cryostat Integration Project 를 선택합니다.
   b. 프로젝트 이름 필드에 192.0.2. -camel 을 입력합니다.
   c. 다음을 클릭합니다.
   d. 대상 환경 선택 창에서 다음 설정을 선택합니다.

- *배포 플랫폼으로 **Standalone** 을 선택합니다.*

- ***EAP**에서 런타임 환경으로 **Wildfly/Fuse** 를 선택하고 런타임 (선택 사항) 드롭다운 메뉴를 사용하여 **JBoss EAP 7.x** 런타임 서버를 대상 런타임으로 선택합니다.*

e. *대상 런타임을 선택하면 **Camel** 버전이 자동으로 선택되고 필드가 회색으로 표시됩니다.*

f. *다음을 클릭합니다.*

g. ***Advanced Project Setup** 창에서 **Spring Cryostat - Spring DSL** 템플릿을 선택합니다.*

h. *완료를 클릭합니다.*

중요

***CodeReady Studio**에서 **Fuse** 프로젝트를 처음 빌드하는 경우 마법사가 프로젝트 생성을 완료하는 데 몇 분이 걸립니다. 이는 원격 **Maven** 리포지토리에서 종속성을 다운로드하기 때문입니다. 프로젝트가 백그라운드에서 빌드하는 동안 마법사를 중단하거나 **CodeReady Studio**를 종료하지 마십시오.*

i. *연결된 **Fuse** 통합 화면을 열라는 메시지가 표시되면 예를 클릭합니다.*

j. ***CodeReady Studio**가 필요한 아티팩트를 다운로드하고 백그라운드에서 프로젝트를 빌드하는 동안 기다립니다.*

2. *다음과 같이 서버에 프로젝트를 배포합니다.*

a. *서버 보기(**Fuse Integration** 관점의 오른쪽 하단)에서 서버가 아직 시작되지 않은 경우 **Red Hat JBoss EAP 7.2** 런타임 서버를 선택하고 녹색 화살표를 클릭하여 시작합니다.*

b. *콘솔 보기에 다음과 같은 메시지가 표시될 때까지 기다립니다.*

> *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. *서버가 시작된 후 서버 보기로 다시 전환한 후 서버 를 마우스 오른쪽 버튼으로 클릭하고 컨텍스트 메뉴에서 추가 및 제거를 선택합니다.*

d. *추가 및 제거 대화 상자에서 **Cryostat -camel** 프로젝트를 선택하고 **Add >** 를 클릭합니다.*

e. *완료를 클릭합니다.*

3. *다음과 같이 프로젝트가 작동하는지 확인합니다.*

a. *다음 **URL**로 이동하여 **ExternalIP -camel** 프로젝트에서 실행되는 서비스에 액세스합니다. **http://localhost:8080/camel-test-spring?name=Kermit***

b. *브라우저 창에 **Hello Kermit** 이 표시되어야 합니다.*

4. *다음과 같이 프로젝트 배포를 취소합니다.*

a. *서버 보기에서 **Red Hat JBoss EAP 7.2** 런타임 서버를 선택합니다.*

b. *서버를 마우스 오른쪽 버튼으로 클릭하고 컨텍스트 메뉴에서 추가 및 제거를 선택합니다.*

c. *추가 및 제거 대화 상자에서 **Cryostat -camel** 프로젝트를 선택하고 **< 제거**를 클릭합니다.*

d. *완료를 클릭합니다.*

4장. MAVEN을 로컬로 설정

일반적인 Fuse 애플리케이션 개발에서는 Maven을 사용하여 프로젝트를 빌드하고 관리합니다.

다음 주제는 Maven을 로컬로 설정하는 방법을 설명합니다.

- 4.1절. “Maven 설정 준비”
- 4.2절. “Maven에 Red Hat 리포지토리 추가”
- 4.3절. “로컬 Maven 리포지토리 사용”
- 4.4절. “환경 변수 또는 시스템 속성을 사용하여 Maven 미러 설정”
- 4.5절. “Maven 아티팩트 및 좌표 정보”

4.1. MAVEN 설정 준비

Maven은 Apache의 무료 오픈 소스 빌드 툴입니다. 일반적으로 Maven을 사용하여 Fuse 애플리케이션을 빌드합니다.

절차

1. Maven 다운로드 페이지에서 Maven의 최신 버전을 다운로드합니다.
2. 시스템이 인터넷에 연결되어 있는지 확인합니다.

프로젝트를 빌드하는 동안 기본 동작은 Maven이 외부 리포지토리를 검색하고 필요한 아티팩트를 다운로드하는 것입니다. Maven은 인터넷을 통해 액세스할 수 있는 리포지토리를 찾습니다.

Maven이 로컬 네트워크에 있는 리포지토리만 검색하도록 이 동작을 변경할 수 있습니다. 즉 Maven은 오프라인 모드에서 실행할 수 있습니다. 오프라인 모드에서 Maven은 로컬 리포지토리

에서 아티팩트를 찾습니다. 4.3절. “로컬 Maven 리포지토리 사용”을 참조하십시오.

4.2. MAVEN에 RED HAT 리포지토리 추가

Red Hat Maven 리포지토리에 있는 아티팩트에 액세스하려면 해당 리포지토리를 Maven의 settings.xml 파일에 추가해야 합니다. Maven은 사용자의 홈 디렉터리의 .m2 디렉터리에서 settings.xml 파일을 찾습니다. 사용자가 settings.xml 파일이 지정되지 않은 경우 Maven은 M2_HOME/conf/ settings.xml 에서 시스템 수준 settings.xml 파일을 사용합니다.

사전 요구 사항

Red Hat 리포지토리를 추가할 settings.xml 파일의 위치를 알고 있습니다.

절차

settings.xml 파일에서 다음 예와 같이 Red Hat 리포지토리의 리포지토리 요소를 추가합니다.

```
<?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>
```

4.3. 로컬 MAVEN 리포지토리 사용

인터넷 연결 없이 **Apache Karaf** 컨테이너를 실행 중이고 오프라인에서 사용할 수 없는 종속성이 있는 애플리케이션을 배포해야 하는 경우 **Maven** 종속성 플러그인을 사용하여 애플리케이션의 종속 항목을 **Maven** 오프라인 리포지토리로 다운로드할 수 있습니다. 그런 다음 이 사용자 지정 **Maven** 오프라인 리포지토리를 인터넷 연결이 없는 시스템에 배포할 수 있습니다.

절차

1. **pom.xml** 파일이 포함된 프로젝트 디렉터리에서 다음과 같은 명령을 실행하여 **Maven** 프로젝트의 리포지토리를 다운로드합니다.

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

이 예에서는 프로젝트를 빌드하는 데 필요한 **Maven** 종속 항목 및 플러그인이 **/tmp/my-project** 디렉터리로 다운로드됩니다.

2. **etc/org.ops4j.pax.url.mvn.cfg** 파일을 편집하여 **org.ops4j.pax.url.mvn.offline** 을 **true**로 설정합니다. 이렇게 하면 오프라인 모드가 활성화됩니다.

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

3. 이 사용자 지정 **Maven** 오프라인 리포지토리를 인터넷 연결이 없는 모든 시스템에 내부적으로 배포합니다.

4.4. 환경 변수 또는 시스템 속성을 사용하여 MAVEN 미러 설정

애플리케이션을 실행하는 경우 **Red Hat Maven** 리포지토리에 있는 아티팩트에 액세스해야 합니다. 이러한 리포지토리는 **Maven**의 **settings.xml** 파일에 추가됩니다. **Maven**은 다음 위치에서 **settings.xml** 파일을 확인합니다.

- 지정된 **URL**을 찾습니다.
- **if not found looks for ${user.home}/.m2/settings.xml**
- **if not found looks for ${maven.home}/conf/settings.xml**
- 찾을 수 없는 경우 **${M2_HOME}/conf/settings.xml**
- 위치를 찾을 수 없는 경우 비어 있는 **org.apache.maven.settings.Settings** 인스턴스가 생성됩니다.

4.4.1. Maven 미러 정보

Maven은 원격 리포지토리 세트를 사용하여 현재 로컬 리포지토리에서 사용할 수 없는 아티팩트에 액세스합니다. 리포지토리 목록에는 거의 항상 Maven Central 리포지토리가 포함되어 있지만 Red Hat Fuse의 경우 Maven Red Hat 리포지토리도 포함되어 있습니다. 불가능하거나 다른 원격 리포지토리에 액세스할 수 없는 경우 Maven 미러 메커니즘을 사용할 수 있습니다. 미러는 특정 저장소 URL을 다른 저장소 URL로 대체하므로 원격 아티팩트를 검색할 때 모든 HTTP 트래픽을 단일 URL로 전달할 수 있습니다.

4.4.2. settings.xml에 Maven 미러 추가

Maven 미러를 설정하려면 Maven의 settings.xml:에 다음 섹션을 추가합니다.

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

위 섹션이 settings.xml 파일에 없는 경우 미러가 사용되지 않습니다. XML 구성을 제공하지 않고 글로벌 미러를 지정하려면 시스템 속성 또는 환경 변수를 사용할 수 있습니다.

4.4.3. 환경 변수 또는 시스템 속성을 사용하여 Maven 미러 설정

환경 변수 또는 시스템 속성을 사용하여 Maven 미러를 설정하려면 다음을 추가할 수 있습니다.

- MAVEN_MIRROR_URL 이라는 환경 변수 bin/setenv 파일
- mavenMirrorUrl 이라는 시스템 속성 etc/system.properties 파일

4.4.4. Maven 옵션을 사용하여 Maven 미러 URL 지정

환경 변수 또는 시스템 속성에 지정된 대체 Maven 미러 URL을 사용하려면 애플리케이션을 실행할 때 다음 maven 옵션을 사용합니다.

- -DmavenMirrorUrl=mirrorId::mirrorUrl

  for example, -DmavenMirrorUrl=my-mirror::http://mirror.net/repository

- *-DmavenMirrorUrl=mirrorUrl*

  for example, -DmavenMirrorUrl=http://mirror.net/repository. 이 예에서 <mirror>의 <id>는 단지 미러일 뿐입니다.

4.5. MAVEN 아티팩트 및 좌표 정보

Maven 빌드 시스템에서 기본 빌딩 블록은 아티팩트 입니다. 빌드 후 아티팩트의 출력은 일반적으로 JAR 또는 WAR 파일과 같은 아카이브입니다.

Maven의 핵심 측면은 아티팩트를 찾고 해당 아티팩트 간의 종속성을 관리하는 기능입니다. Maven 좌표는 특정 아티팩트의 위치를 식별하는 값 집합입니다. 기본 조정에는 다음과 같은 세 가지 값이 있습니다.

groupId:artifactId:version

Maven은 패키지 값으로 기본 조정을 늘리거나 패키징 값과 분류자 값을 모두 사용하는 경우가 있습니다. Maven 조정에는 다음 양식 중 하나가 있을 수 있습니다.

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

값에 대한 설명은 다음과 같습니다.

groupdId

아티팩트 이름의 범위를 정의합니다. 일반적으로 패키지 이름의 전체 또는 일부를 그룹 ID로 사용합니다. 예: org.fusesource.example.

artifactId

그룹 ID를 기준으로 아티팩트 이름을 정의합니다.

버전

아티팩트의 버전을 지정합니다. 버전 번호는 최대 4개의 부분(n.n.n.n.n)을 가질 수 있습니다. 여기서 버전 번호의 마지막 부분은 숫자가 아닌 문자를 포함할 수 있습니다. 예를 들어 1.0-SNAPSHOT의 마지막 부분은 영숫자 하위 문자열인 0-SNAPSHOT 입니다.

패키지

*프로젝트를 빌드할 때 생성되는 패키지 엔터티를 정의합니다. **OSGi** 프로젝트의 경우 패키지는 번들 입니다. 기본값은 **Cryostat** 입니다.*

분류자

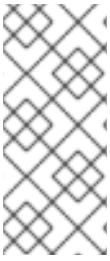
*동일한 **POM**에서 빌드되었지만 콘텐츠가 다른 아티팩트를 구분할 수 있습니다.*

*아티팩트의 **POM** 파일의 요소는 다음과 같이 아티팩트의 그룹 **ID**, 아티팩트 **ID**, 패키징 및 버전을 정의합니다.*

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

*이전 아티팩트에 대한 종속성을 정의하려면 **POM** 파일에 다음 종속성 요소를 추가합니다.*

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



참고

*번 들은 특정 종류의 **JAR** 파일일 뿐이고, **Cryostat**는 기본 **Maven** 패키지 유형이므로 이전 종속성에서 번들 패키지 유형을 지정할 필요가 없습니다. 종속성에서 명시적으로 패키징 유형을 지정해야 하는 경우 **type** 요소를 사용할 수 있습니다.*