Sebastian Hempel

Sebastian Hempel

IT-Consulting Hempel

https://it-hempel.de

https://github.com/ithempel

@ithempel

Java-Entwickler seit 2003

> *Just enough application server.*

OpenSource Projekt von Red Hat

Version 1.0.0 am 27.06.2015

montly releases - 2016.8.1 vom 17.08.2016

http://wildfly-swarm.io

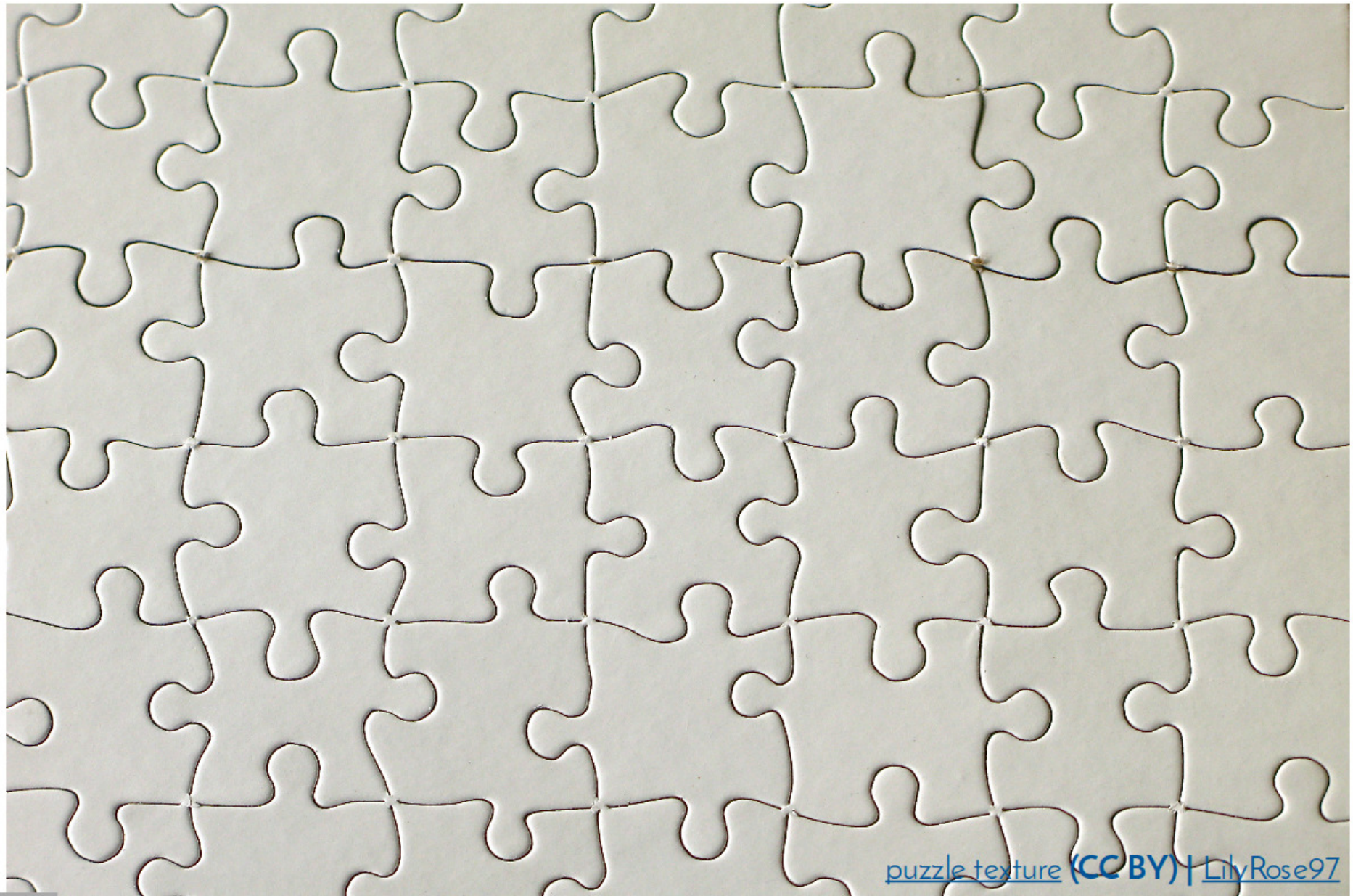
@wildflyswarm

Group: WildFly Swarm

Java EE Application

bekannte APIs
vorhandene Erfahrung
gewohnte Umgebung
MicroService APIs

Fractions



Maven Plugin

Uberjar

Umzugskarton (CC BY-SA) | HornM201

einfach Web-Applikation

WAR → MicroService

Maven

WildFly Swarm Plugin

```
<plugin>
  <groupId>org.wildfly.swarm</groupId>
  <artifactId>wildfly-swarm-plugin</artifactId>
  <version>${wildfly-swarm.version}</version>
  <executions>
    <execution>
      <goals>
        <goal>package</goal>
      </goals>
    </execution>
  </executions>
</plugin>
```

Demo

bestehende Anwendung

Anpassung `pom.xml`

Erstellung einer Uberjar

Maven Plugin Configuration - I

`fractionDetectMode` `fractions`

```
<configuration>
  <fractionDetectMode>force</fractionDetectMode>  (1)
  <fractions>
    org.wildfly.swarm:jaxrs  (2)
  </fractions>
</configuration>
```

(1) Wie sollen Fractions erkannt werden?

(2) `group:name:version` von Fractions

Maven Plugin Configuration - II

`mainClass`

```
<configuration>
  <mainClass>de.ithempel.swarm.Main</mainClass>  (1)
</configuration>
```

(1) Klasse mit user-defined main

Java EE MicroService

Entwicklung mit WildFly Swarm
Auswahl der Fractions
Konfiguration der Fractions

Maven Dependencies

Dependency → Fraction
bewusste Auswahl der APIs

BOM - bill of material

```
<dependencyManagement>
  <dependencies>
    <dependency>
      <groupId>org.wildfly.swarm</groupId>
      <artifactId>bom</artifactId>
      <version>${wildfly-swarm.version}</version>
      <type>pom</type>
      <scope>import</scope>
    </dependency>
  </dependencies>
</dependencyManagement>
```

Auswahl der Fractions

```
<dependency>
  <groupId>org.wildfly.swarm</groupId>
  <artifactId>jaxrs</artifactId>
</dependency>
<dependency>
  <groupId>org.wildfly.swarm</groupId>
  <artifactId>jaxrs-cdi</artifactId>
</dependency>
<dependency>
  <groupId>org.wildfly.swarm</groupId>
  <artifactId>jpa</artifactId>
</dependency>
```

Demo

eigenständige Anwendung

spezielle `pom.xml`

Paketierung über `war`

Customizing Fractions

user-provided main
Konfiguration des Containers

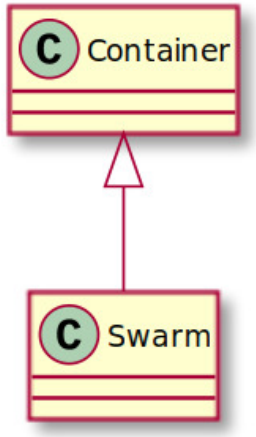
user-provided main

```
static void main(String…args)
```

Konfiguration in Maven Plugin

Container

Repräsentiert WildFly
Start / Stop
Deploy



DataSource Fraction

```
DatasourcesFraction dataSource = new DatasourcesFraction();  (1)
dataSource.jdbcDriver("h2", d -> {  (2)
    d.driverClassName("org.h2.Driver");
    d.xaDatasourceClass("org.h2.jdbcx.JdbcDataSource");
    d.driverModuleName("com.h2database.h2");
});
dataSource.dataSource("StudentDS", ds -> {  (3)
    ds.driverName("h2");
    ds.connectionUrl("jdbc:h2:mem:test;DB_CLOSE_DELAY=-1;DB_CLOSE_ON_EXIT=FALSE");
        ds.userName("sa");
        ds.password("sa");
});
```

1. Erstellung der DataSource Fraction
2. Definition des H2 JDCB-Treibers, der als JBoss Modul vorliegt
3. Hinzufügen einer DataSource `StudentDS`

WildFly Modul

`module.xml` in `src/main/resources`

Dependency für JAR(s)

H2 JDBC-Treiber

modules/com/h2database/h2/main/module.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<module xmlns="urn:jboss:module:1.3" name="com.h2database.h2">
  <resources>
    <artifact name="com.h2database:h2:${h2.version}" />
  </resources>
  <dependencies>
    <module name="javax.api" />
    <module name="javax.transaction.api" />
    <module name="javax.servlet.api" optional="true" />
  </dependencies>
</module>
```

Start und Deploy

Container erstellen
Fractions hinzufügen
Starten
Anwendung deployen

Demo

"Microservice"

`main` Methode

Fractions hinzufügen

Start und Deploy

ShrinkWrap der Applikation

JAR Archive

Packen der Anwendung

WAR vs. JAR Project

WAR: Swarm deployt

JAR: manuelles Deployment

Shrink Wrap

> *The ShrinkWrap project provides a simple API to programmatically assemble archives in code.*

Teil des Arquillian Projekts

Shrink Wrap

JAX-RS Archiv mit WEB-INF Resource

```
JAXRSArchive deployment = ShrinkWrap.create(JAXRSArchive.class, "student.war");  (1)

deployment.addPackages(true, "de.ithempel.swarm.business.student");  (2)

deployment.addClass(StudentApplication.class);  (3)

deployment.addAsWebInfResource(new ClassLoaderAsset("META-INF/persistence.xml",
    Main.class.getClassLoader()), "classes/META-INF/persistence.xml");  (4)
```

1. Erstellen eines Archivs
2. Hinzufügen eines Packages (rekursiv)
3. Hinzufügen einer einzelnen Klasse
4. Hinzufügen einer Resource (`persistence.xml`)

Deploy

Shrink Warp Archiv
Deployment in Container

Demo

Erstellung eines Archivs
Deployment des Archivs

Non Java EE APIs

"Microservice APIs" Monitoring

Netflix OSS

Ribbon
Hystrix
RxJava

Topology

Abstraktion für

Service discovery
Service registration

Implementierungen

JGroups
Consul

Logstash

Sendet Logmeldungen an Logstash

Konfiguration: Server und Port

Monitor

Informationen über den "Node"

Node
Heap
Threads

Erweiterbar um eigene Checks

Jolokia

JMX / MBeans über REST

Konfiguration des Paths
Zugriff auf alle MBeans

Command Line

Swarm Container

Parameter der Kommandozeile

Listener

```
-b 127.0.0.18
```

Configuration File

```
-c configuration/standalone.xml
```

Properties

```
-Dconfig.filepath=/configuration
```

```
-Psystem.properties
```

Swarm Properties

`swarm.bind.address`

`swarm.port.offset`

`swarm.http.port`

`swarm.context.path` (default: /)

Project Stages

Konfiguration pro Stage
Parameter for CDI Inject

Stages File

project-stages.yml

```yaml
logger:
  level: DEBUG
swarm:
  port:
    offset: 10
---
project:
  stage: development
logger:
  level: TRACE
swarm:
  port:
    offset: 50
---
project:
  stage: production
logger:
  level: INFO
swarm:
  port:
    offset: 100
```

Nutzung in Konfiguration

standalone.xml

```
<subsystem xmlns="urn:jboss:domain:logging:3.0">
  <console-handler name="CONSOLE">
      <level name="${logger.level:INFO}"/>
      <formatter>
          <named-formatter name="COLOR-PATTERN"/>
      </formatter>
  </console-handler>
  <root-logger>
      <level name="${logger.level:INFO}"/>
      <handlers>
          <handler name="CONSOLE"/>
      </handlers>
  </root-logger>
  [...]
</subsystem>
```

Injection von Parametern

bis 1.0.0.Final

```
@Inject
@ConfigValue("logger.level")
String loggingLevel;
```

ab 2016.8.1

```
@Inject
@ConfigurationValue("logger.level")
String loggingLevel;
```

WildFly Swam 2016.8.1

Microprofile
JDBC Driver Fractions

`@ConfigurationValue`

MicroProfile Initiative

> *An open forum to optimize Enterprise Java for a microservices architecture by innovating across multiple implementations and collaborating on common areas of interest with a goal of standardization.*

> *The first release of the MicroProfile is expected to be available in September, with Red Hat's implementation to be based on WildFly Swarm.*

Informationen

https://microprofile.io

Group: Microprofile

Any Questions?