2.– 5. September 2013
in Nürnberg

Herbstcampus

Wissenstransfer
par excellence

Schmiedekunst

Forge Beanstest Plugin für DI in Junit-Tests

Christian Brandenstein und Sandro Sonntag

adorsys GmbH & Co. KG

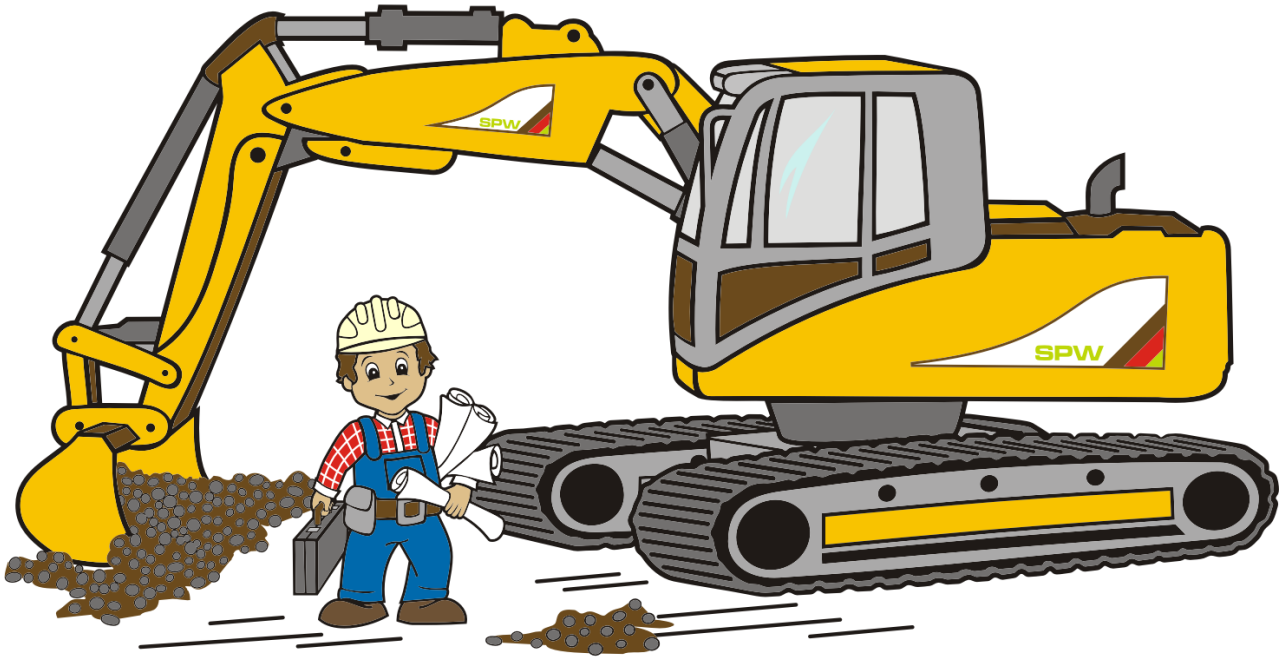
adorsys

Produktivität ist King – viele Verbesserungen der Java Plattform

Aber, unser Werkzeug zum Boostrapping in JEE Projekten ist immer noch die Schaufel...

auch Copy & Paste genannt ☺

Und das Werkzeug der „Nachbarn“...

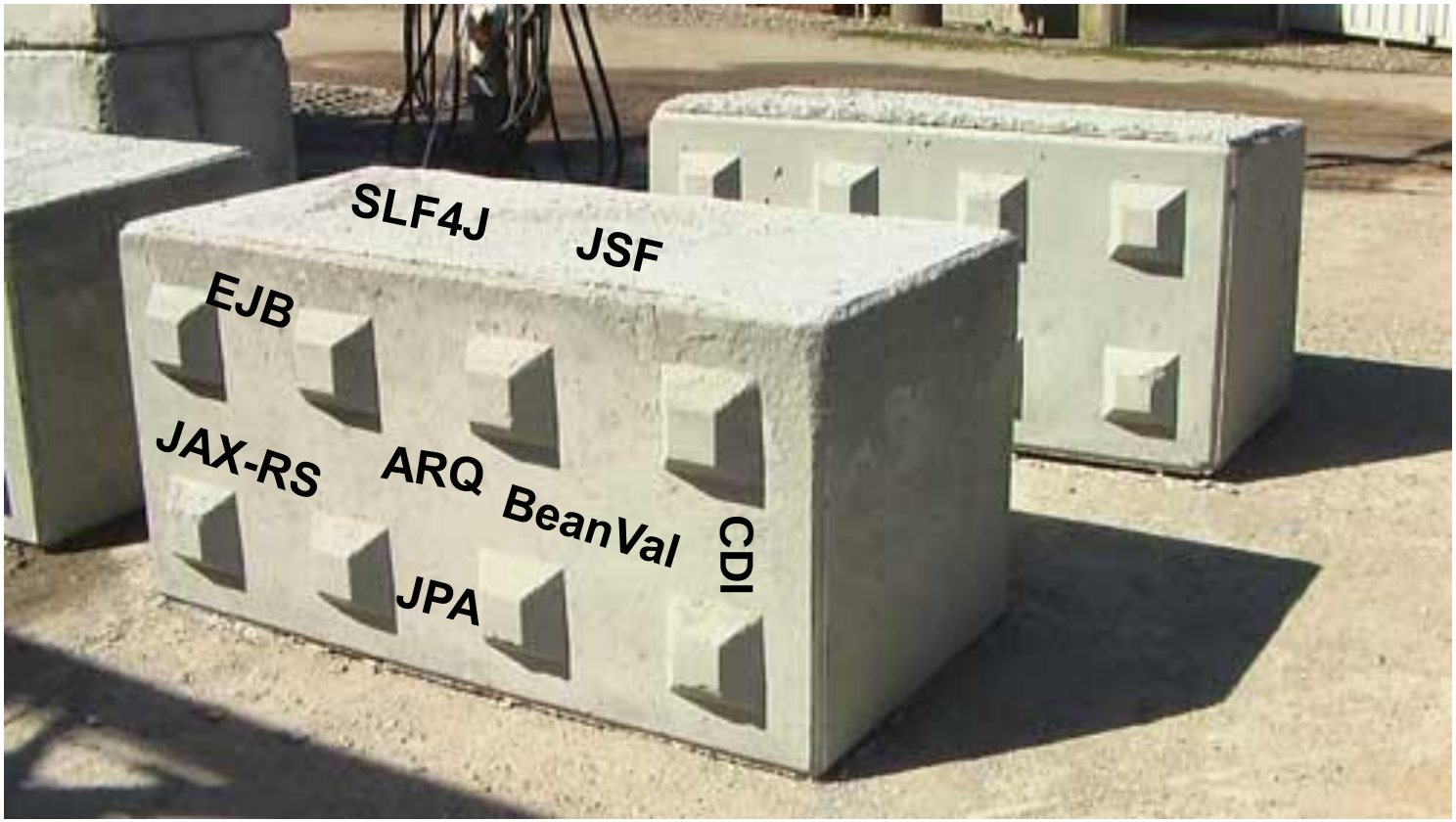


springroo

Play!

RAILS

Aber wir haben doch Maven-Archetypes..



Was wir wirklich brauchen ist ein RAD Baukasten...

Zeit für ein neues Tool ☺

Was steckt drin ?

- Scaffolding / Generierungstool
  - Generierung von Javacode und Konfigurationen
- Unterstützt die gängigen JEE APIs
- Maven Unterstützung
- Hervorragender GIT Support
- Unterstützt die gängigen OSS Applicationserver

- Pluginsystem (CDI, baut auf JBoss Modules)
- Ökosystem - zentrales Pluginrepository

Forge ist DRY für typische Entwicklungsaufgaben

Heute

Mit Forge

+ Forge Scripting

JBoss Arquillian

- Write Real (JEE) Tests ...

Hands on

- Entwicklung einer Wahl JEE App mit CDI, JPA Arquillian und Forge

Forge Beanstest Plugin

- Generiert einen einfachen JUnit-Runner („SimpleRunner.java“)
- Startet den Weld-SE-Container
- Registriert Testklasse als CDI-Bean
- Sämtliche Features von CDI SE im JUnit-Test

Installation

- Forge installieren:
  1. Forge downloaden und unzippen (forge.jboss.org)
  2. '$FORGE_HOME/bin' in den Pfad
- Forge starten (`forge` in Kommandozeile)
- In der Forge-Shell:

```
forge install-plugin beanstest
```

- Output (hoffentlich):

```
***SUCCESS*** Installed from
[https://github.com/ersatzcapo/forge-beanstest.git]
successfully.
```

Neues Projekt anlegen

- Neues Projekt in Forge anlegen:

```
new-project –named <name> --topLevelPackage <pkg name>
```

- Beantest initialisieren:

```
beanstest setup
```

- „CDIFacet“ wird installiert:
  1. beans.xml ins src/main/resources/META-INF
  2. cdi-api und jboss annotations dependency
- „CDITestFacet“ wird installiert:
  1. beans.xml ins src/test/resources/META-INF
  2. Weld SE und Junit dependency
  3. SimpleRunner.java
- Neuen Test anlegen:

```
beanstest new-test --type <test class>
```

Persistence Konfiguration

- Persistence Test-Setup kann angelegt werden:

```
beanstest test-persistence
```

- Hibernate und jpa-Dependencies werden addiert
- Hsqlb-Dependenies werden addiert
- Eine `persistence.xml` wird ins `META-INF` der Test-Resourcen hinzugefügt
- Folgende Dateien entstehen:
  1. PersistenceExtension
  2. MockJpaInjectionServices
  3. META-INF/services/javax.enterprise.inject.spi.Extension

Ablauf nach Start des CDI-Containers

- Liest Service-Extension-Datei ein und registriert damit die PersistenceExtension
- PersistenceExtension hat den Callback `beforeBeanDiscovery`, der vom Container gerufen wird
- Registriert im BeanManager einen `JpaInjectionService`, der vom Container gerufen wird, wenn er einen `PersistenceContext`-InjectionPoint befüllen will
- Der `MockJpaInjectionService` erzeugt bei Bedarf einen `EntityManager` und gibt diesen an den Container
- In der Test `persistence.xml` ist eine hsqld konfiguriert
- Können beliebige DBs konfiguriert werden

Mock Support mit mockito

- Anlegen eines Mockito-Mock mit

```
beanstest new-mockito --type <Zu mockende Klasse>
```

- Mockito-Dependency wird addiert
- Klasse `AlternativesProducer` wird erzeugt
- Ein Alternative-CDI-Stereotype wird erzeugt, Default ist `BeanstestAlternative`
- Der Alternative-Stereotype wird in der Test-`beans.xml` registriet
- In der Klasse `AlternativesProducer` werden die Mockito-Mocks erzeugt
- Weitere Mocks können mit dem selben Stereotype hinzugefügt werden

CDI-Scopes mocken

- RequestScope und SessionScope werden in Weld SE nicht unterstützt
- In älteren Versionen von Weld SE führte ein unbekannter Scope zu

```
ContextNotActiveException: WELD-001303 No active contexts
for scope type ...
```

- Abhilfe mit

```
beanstest mock-scopes
```

- Die `MockMissingScopesExtension` wird registriert
- Verhindert nur Exception, keine Simulation der Scopes

Unterschiede

- Beanstest-Klassen werden als Java-Code generiert
- Es wird kein Shrinkwrap-Archiv benötigt
- Muss somit auch nicht erweitert werden bei Änderungen
- Es können keine verschiedenen Container getestet werden
- Mock-Verwendung wird explizit unterstützt
- Funktioniert nicht mit war-Artifakten
- Ist mehr Entwicklungstool als Testtool
- Beanstest-Tests laufen schneller

Links

- Jboss Forge: http://forge.jboss.org/
- Arquillian Tutorial: http://arquillian.org/guides/get_started_faster_with_forge/
- Beantest Forge Plugin: https://github.com/ersatzcapo/forge-beanstest
- SLF4J Plugin: https://github.com/xandrox/forge-slf4jplugin
- Forge GWT Plugin: http://forge-gwtplugin.github.io/
- Forge Artikel: http://jaxenter.de/artikel/MaytheForgebewithyou

Vielen Dank!

Christian Brandenstein und Sandro Sonntag

adorsys GmbH & Co. KG