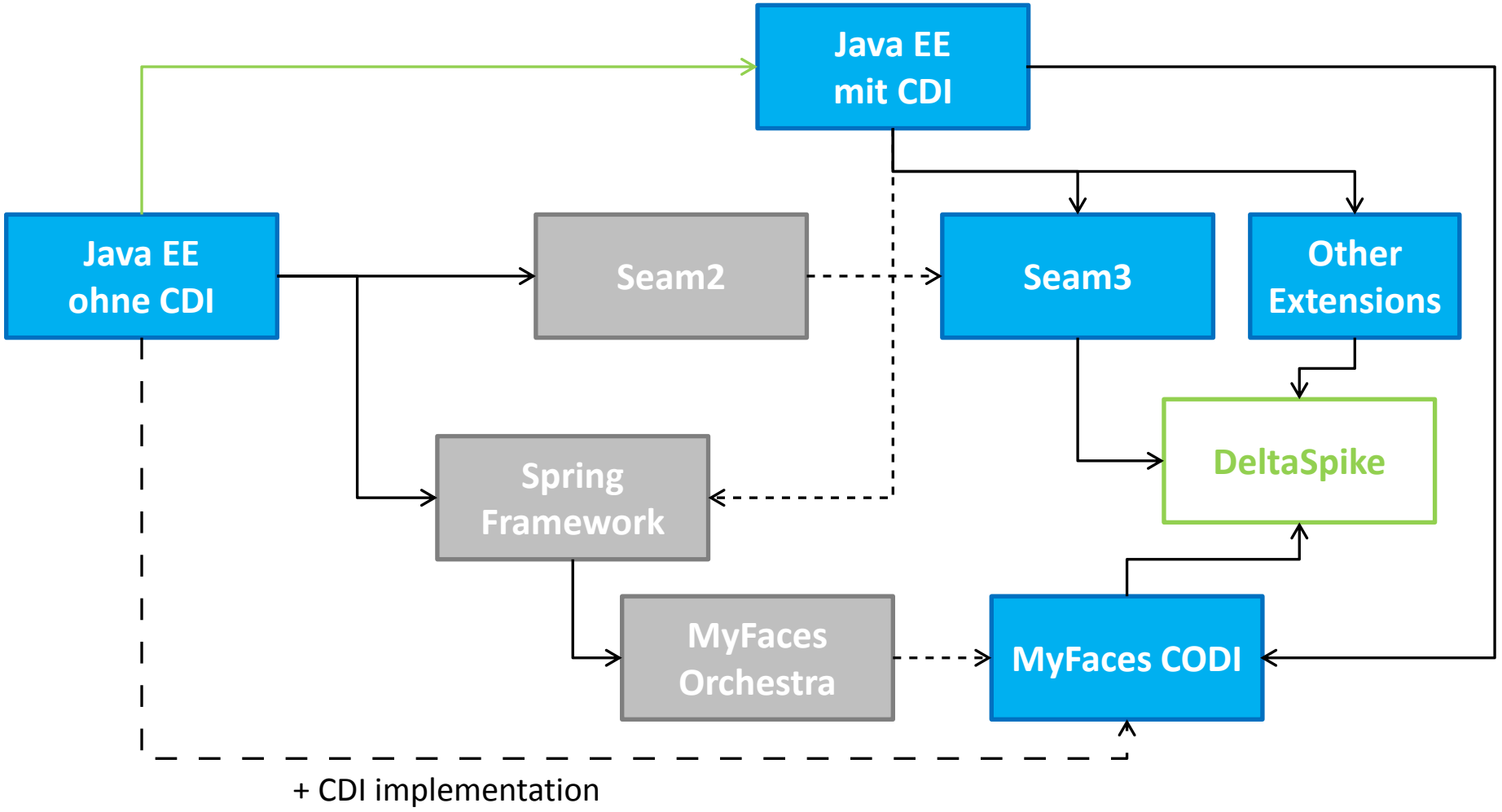
Flexibilität mit CDI & Apache DeltaSpike

Agenda

- Historie
- DeltaSpike ist…
- Portabilität
- Bestandteile von DeltaSpike
- DeltaSpike hilft…
- CDI in EE-Servern

Historie


Java EE
mit CDI
Java EE
ohne CDI
Seam2
Seam3
Other
Extensions
DeltaSpike
Spring
Framework
MyFaces
Orchestra
MyFaces CODI
+ CDI implementation

DeltaSpike ist...

- Portable CDI-Erweiterung
  (für Apache OpenWebBeans und JBoss Weld)
- Mögliche Basis für eigene CDI-Erweiterungen
- Beispiel-Quelle für die Verwendung von CDI
  (API und SPI)
- Sammlung der besten Konzepte aus
  - Apache MyFaces CODI
  - JBoss Seam3
  - u.v.m.

Getestete Portabilität

- Implementierungen
  - Apache OpenWebBeans (>= 1.1.4)
  - JBoss Weld (>= 1.1.10)
- EE-Server
  - Apache TomEE 1.6.0+
  - JBoss AS7 und WildFly8
  - Oracle Weblogic 12.1.3 (derzeit indirekt)
  - IBM WebSphere (derzeit indirekt)

- DeltaSpike Weld 1.1.16
- DeltaSpike Deploy
- DeltaSpike OWB (nightly)
- DeltaSpike OWB 1.1.1
- DeltaSpike OWB 1.1.2
- DeltaSpike OWB 1.1.3
- DeltaSpike OWB 1.1.4
- DeltaSpike OWB 1.1.4 (JDK 1.7)
- DeltaSpike OWB 1.1.5
- DeltaSpike OWB 1.1.6
- DeltaSpike OWB 1.1.7
- DeltaSpike OWB 1.1.8
- Deltaspike OWB 1.1.9-SNAPSHOT
- DeltaSpike OWB 1.2.0
- DeltaSpike OWB 1.2.2-SNAPSHOT
- DeltaSpike RAT-Check
- DeltaSpike Weld (nightly)
- DeltaSpike Weld 1.1.10

- DeltaSpike Weld 1.1.11
- DeltaSpike Weld 1.1.12
- DeltaSpike Weld 1.1.13
- DeltaSpike Weld 1.1.14
- DeltaSpike Weld 1.1.15
- DeltaSpike Examples
- DeltaSpike Weld 1.1.3
- DeltaSpike Weld 1.1.3.SP1
- DeltaSpike Weld 1.1.4
- DeltaSpike Weld 1.1.5
- DeltaSpike Weld 1.1.5.AS71
- DeltaSpike Weld 1.1.6
- DeltaSpike Weld 1.1.7
- DeltaSpike Weld 1.1.8
- DeltaSpike Weld 1.1.9
- DeltaSpike Weld 1.1.9 (JDK 1.7)
- DeltaSpike Weld 2.0.0
- DeltaSpike Weld 2.0.0.SP1
- DeltaSpike Weld 2.0.1

- DeltaSpike_AS7
- DeltaSpike_Control_Build
- DeltaSpike_TomEE

DS Bestandteile im Überblick

- Core (v0.1+)
- CDI-Control (v0.2+)
- Module

DS Module - 1

- Security (v0.2+)
- JPA (Transaction) (v0.3+)
- JSF (v0.4+)
- Partial-Bean (v0.4+)
- Bean-Validation (v0.5+)
- Data (Query) (v0.5+)
- Servlet (v0.5+)

DS Module - 2

- Scheduler (v0.6+)
- Test-Control (v0.6+)

CDI und DS - Let's rock!

CDI SPI

- Sehr umfangreich und flexibel
- Zusammenhänge und Regeln in manchen Fällen wichtig (und oftmals vernachlässigt)

Fast richtig

- Beispiele
  - Falscher Bean-Lookup
  - Falscher Umgang mit Dependent-Beans
  - u.v.m.
- Folgen
  - Einfache Use-Cases funktionieren (trotz Fehler)
  - Unerwartete Probleme können später auftreten

Manueller Bean-Lookup

- Aufgabenstellung:
  Manueller Bean-Lookup statt Injection
- 1. Schritt eindeutig:

```
Set<Bean<?>> beans =
  beanManager.getBeans(type, qualifiers);
```

Manueller Bean-Lookup – Schritt 2

- Falsche Annahme:
  "Eindeutige Type/Qualifier Kombinationen führen immer nur zu einem Bean"
  ➔ Falsche Folgerung – es genügt:

```
Bean<?> bean = beans.iterator().next();
```

- Korrektur:
  `@Alternative` führt zu mehreren `Bean<T>`
  ➔ Richtige Verwendung:

```
Bean<?> bean = beanManager.resolve(beans);
```

DS-Core hilft mit BeanProvider - 1

- Manueller Bean-Lookup vereinfacht
  - `getContextualReference`
  - `getContextualReferences`
  - `getDependent`
- Lookup per
  - Typ
  - Name
- Optionale Ergebnisse möglich

DS-Core hilft mit BeanProvider - 2

- Lookup mit Default-Qualifier
```
BeanProvider.getContextualReference(type)
```
- Lookup mit 1-n Qualifier/n
```
BeanProvider.getContextualReference(
   type, qualifiers)
```
- Lookup optionales Bean mit Default-Qualifier
```
BeanProvider.getContextualReference(
   type, true)
```

DS hat geholfen
Fall gelöst!

CDI-Beans anpassen - 1

- CDI-Bootstrapping-Prozess stellt viele hilfreiche Events zur Verfügung
- Die Erzeugung von eigenen Metadaten zur Veränderung von Beans kann komplex werden

CDI-Beans anpassen - 2

- Beispiele
  - **`BeforeBeanDiscovery`**
    ➔ externe Klassen können hinzugefügt werden (**`addAnnotatedType`**)
  - **`ProcessAnnotatedType`**
    ➔ Beans können verändert (**`setAnnotatedType`**) oder exkludiert (**`veto`**) werden
  - **`AfterBeanDiscovery`**
    ➔ Beans "nachträglich" hinzufügen (**`addBean`**)

DS-Core hilft mit AnnotatedTypeBuilder

- Aufgabenstellung:
  Interceptor autom. hinzufügen
- Dynamische Veränderung von Bean-Metadaten
- Beispiel

```
AnnotatedType<T> annotatedType =
processAnnotatedType.getAnnotatedType();

processAnnotatedType.setAnnotatedType(
  new AnnotatedTypeBuilder<T>()
    .readFromType(annotatedType)
    .addToClass(
      new TransactionalLiteral())
    .create());
```

DS hat geholfen
Fall gelöst!

Beans de-/aktivieren

- Aufgabenstellung:
  Beans (bedingt) entfernen
- Kontextabhängige Deaktivierung
  - `ifProjectStage`
  - `exceptIfProjectStage`
  - `onExpression`

DS hilft mit @Exclude

- Deaktivierung ohne Bedingung

```
@Exclude
public class MyManualTestService
  extends MyManualService { /*...*/ }
```

- Aktivierung für Unit-Tests

```
@Exclude(exceptIfProjectStage =
  ProjectStage.UnitTest.class)
public class MyManualTestService
  extends MyManualService { /*...*/ }
```

DS hat geholfen
Fall gelöst!

CDI in EE Servern

Java EE5

- CDI kann manuell konfiguriert werden
  (siehe Konfiguration für Servlet-Container)
- CDI-Containerstart erfolgt später
  (als in EE6+ Servern)
- Apache DeltaSpike für EE6+
  (Apache MyFaces CODI für EE5+)
- Apache OpenWebBeans durch Plug-ins sehr flexibel
  (bzw. EJB-Injection in CDI-Beans)

Java EE6

- CDI 1.0
- Upgrade abhängig vom Server
- Die ersten Server-Versionen hatten oft Probleme
- Viele Einschränkungen durch BDA-Regeln
- In Weld-basierten Servern kann OWB helfen (http://os890.blogspot.com/2013/04/one-container-everywhere.html)
- Gute Kompatibilität mit Apache DeltaSpike

Java EE7

- CDI 1.1
- Aktuelle:
  - Oracle Glassfish 4
  - JBoss WildFly8
- Einschränkungen durch BDA-Regeln in vielen Fällen entschärft
- Sehr früh mit Apache DeltaSpike getestet (Apache MyFaces CODI ab 1.0.6+)

Q&A!?!

Links zu weiteren Details

- Apache DeltaSpike
  http://deltaspike.apache.org
- Erweiterungen
  https://github.com/os890
- Professioneller Support
  http://www.irian.at