JUDCon
JBoss Users & Developers Conference
2012: India
Bridging The Gap
Integrating Spring and CDI

Ray Ploski
Director, Developer Programs & Strategy
@rayploski
# Protagonists

<table>
<thead>
<tr>
<th>Spring</th>
<th>Contexts &amp; Dependency Injection</th>
</tr>
</thead>
<tbody>
<tr>
<td>Since 2004</td>
<td>Since 2009</td>
</tr>
<tr>
<td>Popular Enterprise Java Framework</td>
<td>Standard Programming Model for Java EE 6</td>
</tr>
<tr>
<td>Conceived to provide an alternative to EJBs</td>
<td>Integrates with EJB</td>
</tr>
<tr>
<td>Dependency Injection</td>
<td>Dependency Injection</td>
</tr>
<tr>
<td>AOP</td>
<td>Cross Cutting Concerns</td>
</tr>
</tbody>
</table>
Commonalities

- Similar Concepts
  - **container** – managing object graphs according to a developer-provided blueprint
  - **bean** – a managed object
- Both Support
  - JSR–330 (@Inject)
  - JSR–250 (Java EE 5 injection, Lifecycle Annotations)
Differences

- Implicit (CDI) vs. explicit (Spring) blueprint
Seam–Spring Module

A utility library for integrating Spring and CDI programming models

Enables the use of Java business components across technology lines
Seam 3 Spring Module

**Seam 3**

- Seam3 – a library of CDI extensions
- Similar functionality to the Seam 2 Spring integration
  - With CDI specific undertones
- Will merge into Apache DeltaSpike
Seam 3 Spring Module

• Seam3 – a library of CDI extensions
• Similar functionality to the Seam 2 Spring integration
  • With CDI specific undertones
• Will merge into Apache DeltaSpike
Seam 3 Modules

• Solder
• Config
• Social
• JMS
• JCR
• Security
• I18N

• Mail
• Validation
• Wicket
• Catch
• Remoting
• REST
• Faces
• Spring

Drools
SwitchYard
jBPM
Errai
TBD
Ecetera
Seam 3 Modules

- Solder
- Config
- Social
- JMS
- JCR
- Security
- I18N

- Mail
- Validation
- Wicket
- Catch
- Remoting
- REST
- Faces
- Spring

DeltaSpike

Various JBoss Projects
- Drools
- SwitchYard
- jBPM
- Errai
- TBD
- Ecetera

TBD
Seam3 Spring Module
Design Goals

- Minimize changes to existing code
- Non-intrusive approach
- Highly customizable
- Mirror similar capabilities
Why?

• **Reuse** existing investments

• **Integrate groups** with various expertise

• **Simplify** integration of Spring and Java EE

• **Ease migration** of legacy Spring apps to Java EE 6
Seam 3 Spring Module Features

- Integrating Spring components into CDI
- Accessing Spring ApplicationContexts
  - created by the extension
  - access AC’s created by Spring components (e.g. web ApplicationContext)
- Accessing CDI beans
  - exposing Spring beans as CDI beans
Integrating Spring into CDI

1.) Get the ApplicationContext
Integrating Spring components into CDI

- Implemented as a Standard CDI Extension
- Spring contexts and Spring beans are installed as CDI beans
- Implemented using the resource producer pattern

- ApplicationContexts
- Spring Beans
  * Can be bootstrapped or external
ApplicationContext Management

Spring contexts are exposed as CDI beans

Created by Seam
Spring Module
- @Producer fields
- SPI Based

- or -

Created by Spring

@Produces @Web @SpringContext

ApplicationContext context;

- or -

e.g. a Web

ApplicationContext
@SpringContext Annotation

1.) Bootstrapped By Spring- CDI Extension

@Produces @SpringContext
@ApplicationScoped
@Configuration
(locations={"classpath:springCtx.xml"})
ApplicationContext springContext;
@SpringContext Annotation

2.) Used within a CDI Managed Bean

//Usage
@Inject @SpringContext
ApplicationContext context;
1.) Web Application Context (external)

// Spring Application Context loaded by Servlet container

@Produces @Web @SpringContext ApplicationContext context;
Integrating Spring into CDI

2.) Get access to the Spring Beans
2.) Use within a CDI Managed Bean

```java
//Usage
@Inject @SpringContext
ApplicationContext context;
```
Apply the *resource producer pattern* for creating Spring ApplicationContext CDI beans

```java
@Produces @SpringBean
UserService userService;
```

//Usage
@SessionScoped
public class AllocationManager {
	@Inject
	UserService userService;
}
Avoiding Bean Duplication

• Imported Spring beans may cause duplication issues
  • CDI will add any classes automatically as beans
  • Extension-added beans will be in conflict

• Options:
  • Solder @Veto annotation (recommended)
  • Using side-effects of @Typed and @Alternative
    (standard-based but we do not recommend)

• Proprietary extensions for bean scanning (e.g. Weld)
Bean Duplication and Vetoing

Importing beans may result in duplication ...

... but vetoing can solve this problem
In-Extension Context Bootstrapping

- Alternative to `@Produces` `@SpringContext` `@Configuration`

- Advantages relative to CDI lifecycle
  - Spring context is bootstrapped before bean discovery
  - Spring bean classes are automatically vetoed
Integrating CDI into Spring
CDI integration into Spring

- Uses standard Spring mechanisms
  - BeanFactories
  - BeanPostProcessors/BeanFactoryPostProcessors
- Most facilities are hidden behind the `<cdi:*/>` namespace
Accessing the Bean Manager

Usage:
@Autowired BeanManager beanManager;

<cdi:bean-manager/>

xmlns:cdi="http://www.jboss.org/schema/seam/spring"
Accessing CDI Beans

```xml
<cdi:bean-reference id="user" type="example.User">
    <cdi:qualifier type="example.BusinessUser"/>
</cdi:bean-reference>
```

```java
@Component
public class User {
    @Autowired User user;
}
```
Importing CDI Beans into Spring
Scoping

- Spring and CDI have different approaches to bean **scoping**
  - Spring – singletons, application-scoped
  - CDI – prototype, 'default-scoped'

- The module preserves the original scope
  - `@SpringContext` beans are `@ApplicationScoped`
  - `@SpringBean` are default-scoped and fall back to the native Spring scope (AOP scope proxy required)
  - CDI beans are prototype-scoped and fall back to the native CDI scope (through their scoped proxy)
Beyond CDI

• This module can serve as basis for integrating any Java EE component with Spring
  • EJBs
  • JAX-RS
  • any component that understands CDI
Roadmap

• Current version: 3.1.0.Final (included in Seam 3.1.0.Final)
• Upcoming versions:
  ○ 3.1.x - minor features, bug fixes
  ○ 3.2.x (March/April 2012) - additional features
    ▪ bean autoimport
    ▪ eventing bridge
    ▪ bridging Spring AOP/CDI interceptors, decorators
    ▪ Seam Persistence integration (transactions, Persistence Contexts)
• Post 3.2
  ○ integration into Apache DeltaSpike
ENTERPRISE APPLICATION PLATFORM
6.0

JBoss AS 7.1
SUBSCRIPTIONS = DEVELOPER PRODUCTIVITY

Frees you to focus on the new stuff

NOT JUST FOR ADMINS

“Q: I’m writing a WS-Policy and...”

KILL DRUDGERY

ACCELERATE YOUR PROJECT!

Fight your bugs, not others’

http://www.redhat.com/jboss/benefits/
Thank You!


Ray Ploski

Director, Developer Programs & Strategy

@rayploski