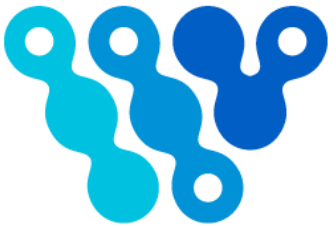
CDI @ Apache

OpenWebBeans and DeltaSpike Deep Dive

Mark Struberg
Gerhard Petracek



OpenWebBeans

Agenda

- CDI and its terms
- Why OpenWebBeans?
- Portable CDI Extensions
- CDI by example with DeltaSpike

CDI is a ...

- JCP specification started in ~2007 Contexts and Dependency Injection for the Java EE platform (CDI) as JSR-299
- component model designed for Java EE (can be used with Java SE)

CDI Features

- Type-safe Dependency Injection
- Interceptors
- Decorators
- Events
- SPI for implementing "Portable Extensions"
- Unified EL integration

What is Dependency Injection?

- "Inversion Of Control" object creation
- No more hardcoded dependencies when working with Interfaces

  ```
  MailService ms = new VerySpecialMailService();
  ```

- Basically the old Factory Pattern
- Hollywood Principle:
  "Don't call us, we call you!"
- Macho Principle
  "Dude, gimme that bloody stuff!"

Singletons and Contexts

- What is a 'Singleton'
- "exactly one single instance **in a well specified context**"

Built-in CDI Scopes

- NormalScoped with well defined lifecycle:
  - @ApplicationScoped
  - @SessionScoped
  - @RequestScoped
  - @ConversationScoped
- 'Pseudo Scope':
  - @Dependent

Terms - Managed Bean

- ... a Java Class and all it's rules to create (contextual) instances of that bean.
- 'Managed Beans' in JSR-299 and JSR-346 doesn't mean JavaBeans!
- Interface

Bean<T> extends Contextual<T>

Terms - Contextual Instance

- ... a Java instance created with the rules of the Managed Bean Bean<T>
- Contextual Instances usually don't get injected directly!

Terms - Contextual Reference

- ... a proxy for a Contextual Instance.
- Proxies will automatically be created for injecting @NormalScope beans and allow decoupled scope handling

Bootstrapping & Runtime

- Creating the meta information at startup
  - Bean meta-data can be changed
  - Fail fast (e.g. AmbiguousResolutionException)
- Contextual Instance creation at runtime
  - based on the Managed Beans
  - the Context will maintain the instances
- Well defined contextual instance termination

Why Apache OpenWebBeans?

- Fast
- Stable
- Modular Plugin Architecture
- Usable
  (e.g. alternative approach for BDAs)

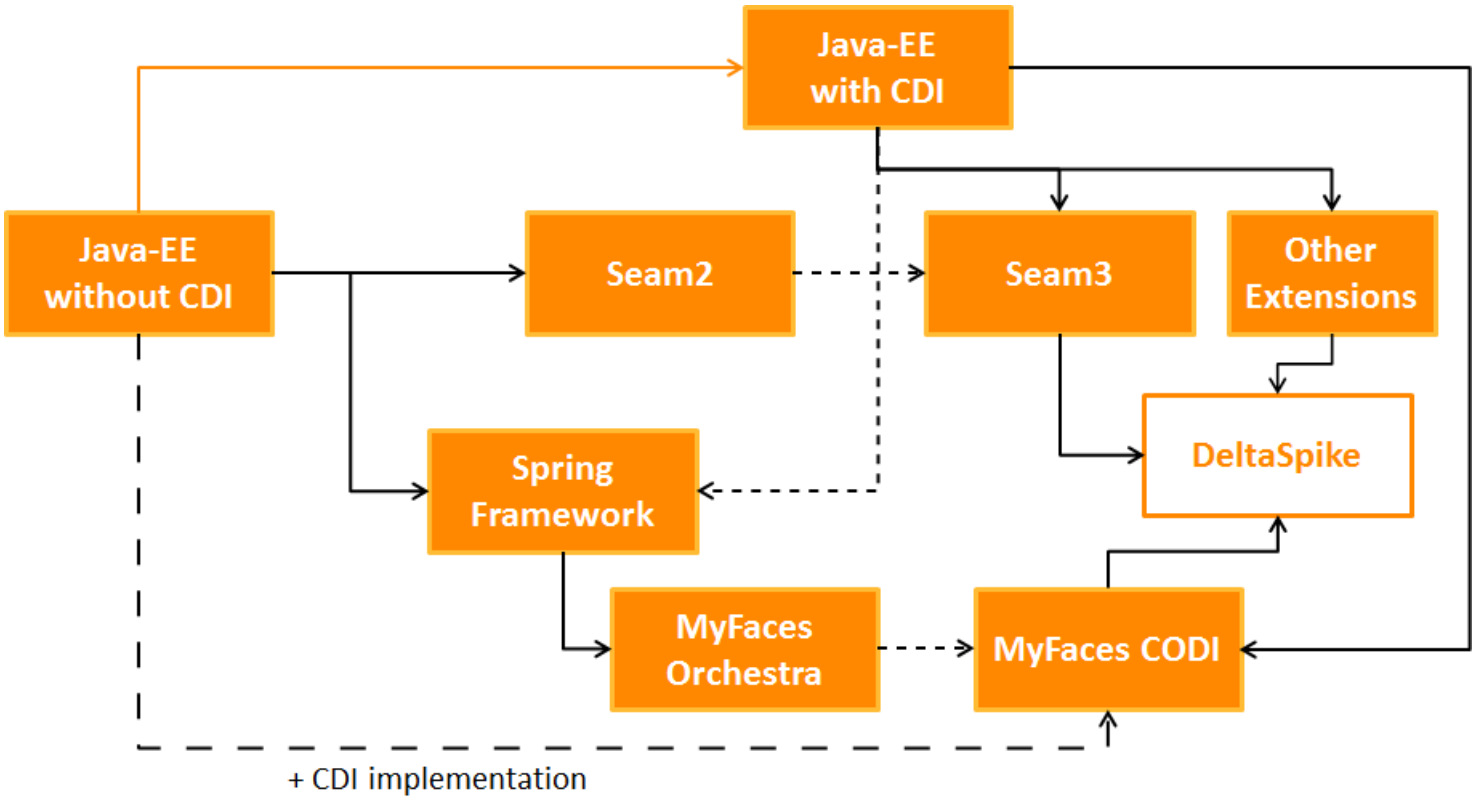
Portable CDI Extensions

- Apache MyFaces CODI
  http://myfaces.apache.org/extensions/cdi
- JBoss Seam3
  http://seamframework.org/Seam3
- Apache DeltaSpike
  http://incubator.apache.org/deltaspike

DeltaSpike closes the gaps between ...

- ... Java-EE and the needs of real-world applications
- ... different CDI communities

History of Apache DeltaSpike



CDI in Action with Apache DeltaSpike

DeltaSpike 0.3 - Overview

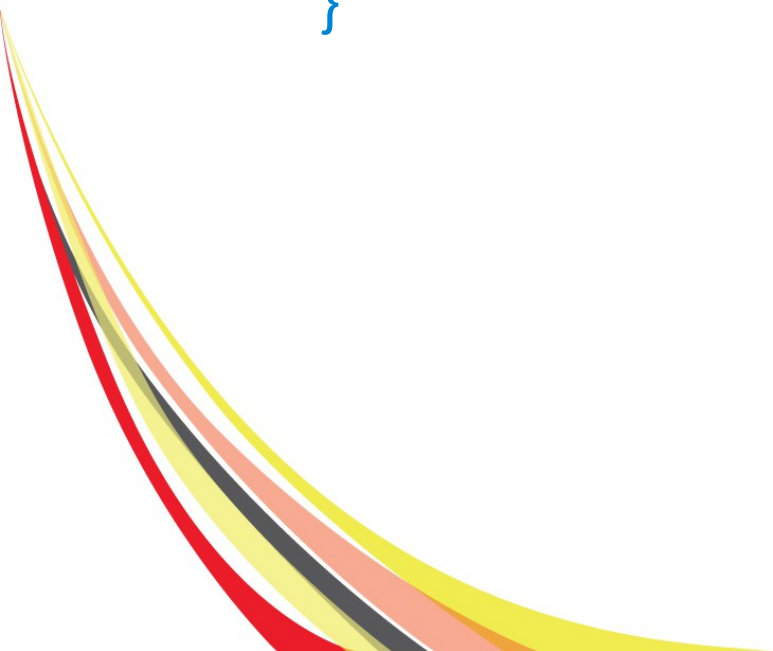
- Core
- JPA
- Security
- Container-Control

Interceptors and Producers in action

@Transactional - 1

- Transactional bean in the application

```
@Transactional
public class MyBean {
    @Inject
    private EntityManager em;
}
```



@Transactional - 2

- Producer and disposer in the application

```
@Produces
@TransactionScoped
protected EntityManager defaultEntityManager() {
   return …;
}

protected void dispose(@Disposes EntityManager em) {
   if (em.isOpen()) {
      em.close();
   }
}
```

@Transactional - 3

- Interceptor annotation in DeltaSpike

```
@InterceptorBinding
public @interface Transactional {
   //...
}
```

@Transactional - 4

- Interceptor implementation in DeltaSpike

```
@Interceptor @Transactional
public class TransactionalInterceptor implements Serializable {

    @Inject private TransactionStrategy ts;

    @AroundInvoke
    public Object executeInTransaction(
        InvocationContext invocationContext)
        throws Exception {
            return ts.execute(invocationContext);
    }
}
```

+ config in the beans.xml

Qualifiers in action

@Transactional - 5

- Transactional bean in the application

```
@Transactional
public class MyBean {
    @Inject
    private @First EntityManager em1;

    @Inject
    private @Second EntityManager em2;
}
```

@Transactional - 6

- Producer implementations in the application

```
@Produces @First
protected EntityManager firstEntityManager() {
   //...
}

@Produces @Second
protected EntityManager secondEntityManager() {
   //...
}
```

@Transactional - 7

- Producers and disposers in the application

```
protected void disposeFirst(@Disposes @First
    EntityManager em) {
        if (em.isOpen()) {
            em.close();
        }
}
protected void disposeSecond(@Disposes @Second
    EntityManager em) {
        if (em.isOpen()) {
            em.close();
    }
}
```

@Transactional - 8

- Qualifier implementations in the application

```
@Qualifier
public @interface First {
}

@Qualifier
public @interface Second {
}
```

Events in action

@BeforeJsfRequest - 1

- Observer in the application

```
public void onBeforeJsfRequest(
  @Observes @BeforeJsfRequest
  FacesContext facesContext) {
    //…
}
```

@BeforeJsfRequest - 2

- Fired event in DeltaSpike

```
@Inject
@BeforeJsfRequest
private Event<FacesContext> beforeJsfRequestEvent;

this.beforeJsfRequestEvent.fire(facesContext);
```

@Specializes and @Alternative

@Specializes configs - 1

- Specialized type-safe config in the application

```
@Specializes
public class CustomWindowContextConfig
    extends WindowContextConfig {
        public int getWindowContextTimeoutInMinutes() {
            return 240;
        }
}
```

@Specializes configs - 2

- Config implementation in CODI (/DS)

```
@ApplicationScoped
public class WindowContextConfig {
    public int getWindowContextTimeoutInMinutes() {
        return 60;
    }
    public int getMaxWindowContextCount() {
        return 64;
    }
    //...
}
```

@Alternative - 1

- Alternative implementation in the application

```
@Alternative
@Exclude(
   exceptIfProjectStage = ProjectStage.Development.class)
public class MockedMailService
   implements MailService {
}
```

+ config in beans.xml

@Alternative - 2

- Primary implementation in the application

```
public interface MailService {
}

@ApplicationScoped
public class DefaultMailService
    implements MailService {
}
```

Apache DeltaSpike.Next

- Simple answer:
  There is no fixed master plan!
  The future depends on the community
  -> **get involved**!