Context and Dependency Injection for Java EE (CDI)

Roberto Chinnici
Java EE 6 Specification Lead

What is CDI?

New API in Java EE 6

- Dependency injection
  - Builds on @Inject API
- Context/scope management
- Works with multiple bean types
- Includes ELResolver

Dependency Injection Basics (1)

Clients

- Injection point lists
  - > zero or more qualifiers
  - > a type

  @Inject @LoggedIn User user;
- The “what”: a User
- “Which one?”: the LoggedIn one
- Not string-based!

Dependency Injection Basics (2)

Other aspects

- Field-, method- or constructor-injection
- Separate from @Resource but the two can coexist
- @PostConstruct works as usual

Sample CDI Client Code (1)

Field and method injection

```
public class CheckoutHandler {

  @Inject @LoggedIn User user;

  @Inject PaymentProcessor processor;

  @Inject void setShoppingCart(@Default Cart cart) {
     …
  }

}
```

Sample CDI Client Code (2)

Constructor injection

```
public class CheckoutHandler {

  @Inject
  CheckoutHandler(@LoggedIn User user,
                  PaymentProcessor processor,
                  @Default Cart cart) {
    ...
  }

}
```

Sample CDI Client Code (3)

Multiple qualifiers and qualifiers with arguments

```
public class CheckoutHandler {

  @Inject
  CheckoutHandler(@LoggedIn User user,
                  @Reliable
                  @PayBy(CREDIT_CARD)
                  PaymentProcessor processor,
                  @Default Cart cart) {
    ...
  }

}
```

Declaring Qualifiers

With a meta-annotation

- Write your own annotation types and annotate them with @Qualifier
- E.g.

```
@Qualifier
@Retention(RUNTIME)
@Target({FIELD,TYPE})
public @interface Red {}
```

Managed Beans

Unified Component Model

- Anything injected is a “bean”
  - EJB session beans
  - Plain classes with @ManagedBean
  - Any class CDI can discover in a module

Sample CDI Bean (1)

A managed bean

```
@Reliable
@PayBy(CREDIT_CARD)
@ManagedBean
public class ReliableCreditCardPaymentProcessor
implements PaymentProcessor {

  void pay(Amount amount) throws PaymentException {
    ...
  }
}
```

Sample CDI Bean (2)

An EJB session bean

```
@Reliable
@PayBy(CREDIT_CARD)
@ManagedBean
@Stateless
public class ReliableCreditCardPaymentProcessor
implements PaymentProcessor {

  void pay(Amount amount) throws PaymentException {
    ...
  }
}
```

Configuration

There is none!

- CDI discovers bean in all modules in which CDI is enabled
- Beans are automatically selected for injection
- Possible to enable groups of bean selectively via a descriptor

Scopes

Automatic context management

- Beans can be declared in a scope
- The CDI runtime will make sure the right bean is created at the right time
- Clients do NOT have to be scope-aware

Sample CDI Bean (3)

A request-scoped managed bean

```
@Reliable
@PayBy(CREDIT_CARD)
@RequestScoped
@ManagedBean
public class ReliableCreditCardPaymentProcessor
implements PaymentProcessor {

  void pay(Amount amount) throws PaymentException {
    ...
  }
}
```

Built-in Scopes

General definitions specialized for different modules

- Everywhere:
  - > @ApplicationScoped
  - > @RequestScoped
- In a web app: @SessionScoped
- With JSF:
  - > @ConversationScoped
- Pseudo-scope: @Dependent

Decoupling

Clients completely decoupled from beans

- Clients only declare dependencies via injection points
- Bean selection is done by CDI
- No client knowledge of scope required
- CDI does proxying transparently when needed

Named Beans

Built-in support for the Unified EL

- Beans give themselves a name with @Named("cart")
- Then refer to it from a JSF or JSP page using the EL:

```
<h:commandButton
        value="Checkout"
        action="#{cart.checkout}"/>
```

Events

Even more decoupling!

- Annotation-based event model
- A bean @Observes an event

```
void onLogin(@Observes
        LoginEvent event) { … }
```

- Another bean fires an event using the Event.fire(T event) method

Built for the Future

Powerful SPI enables portable extensions

- Extensions can programmatically define new beans, injection points, qualifier types, events, event observers...
- Makes it easy to integrate any third-party framework
  - > Just model it as “beans”!

Much More...

CDI

- Producer methods and fields
- Bridging Java EE resources
- Alternatives
- Interceptors
- Decorators
- Stereotypes

Resources

Java EE 6 and GlassFish v3

- Java EE 6 Home
  java.sun.com/javaee
- Java EE 6 Downloads
  java.sun.com/javaee/downloads
- Upcoming Training
  java.sun.com/javaee/support/training

Java EE 6

- Sun GlassFish Enterprise Server v3 Home
  www.sun.com/glassfishv3
- Community Page
  glassfish.org
- The Aquarium Blog
  blogs.sun.com/theaquarium
- White Papers/Webinars
  http://www.sun.com/glassfish/resources

GlassFish

Thank You!

roberto.chinnici@sun.com