Introduction to CDI

- What is CDI?
- Naming and Injecting
- Conversations

Copyright © Descriptor Systems, 2001-2014. Course materials may not be reproduced in whole or in part without prior written consent of Joel Barnum

Copyright © Descriptor Systems, 2001-2014. Course materials may not be reproduced in whole or in part without prior written consent of Joel Barnum

What is CDI?

- The **Contexts and Dependency Injection** specification is a part of JEE designed to help you integrate tiers, e.g. Web and domain logic (EJB)
- CDI is a standard part of JEE 6 and later

CDI was originally known as JSR-299. It borrows concepts from the Spring and Seam frameworks.

Copyright © Descriptor Systems, 2001-2014. Course materials may not be reproduced in whole or in part without prior written consent of Joel Barnum

CDI Features

- Annotations to name beans
- Annotations to perform dependency injection
- Conversational scope

Copyright © Descriptor Systems, 2001-2014. Course materials may not be reproduced in whole or in part without prior written consent of Joel Barnum

CDI Annotations vs JSF Annotations

- JSF2 annotations predate CDI and are still supported for environments that do not completely implement JEE, e.g. Tomcat
- If you are deploying to a full JEE server, Oracle recommends that you use CDI for overlapping annotations

Copyright © Descriptor Systems, 2001-2014. Course materials may not be reproduced in whole or in part without prior written consent of Joel Barnum

Using CDI With Tomcat

- Even though Tomcat doesn't natively support CDI, you can install a CDI implementation into a WAR file:
    - Download an implementation, e.g. Weld and include JARs in WEB-INF/lib
    - Configure a context listener
    - Provide at least an empty beans.xml file in WEB-INF (required for **all** CDI applications, not just for Tomcat)

There are other CDI implementations besides JBoss Weld, notably Apache OpenWebBeans.

Copyright © Descriptor Systems, 2001-2014. Course materials may not be reproduced in whole or in part without prior written consent of Joel Barnum

Naming Managed Beans

- CDI provides the **@Named** annotation that lets you assign names to beans managed by CDI
- By default, the bean name is the class name with a lower-case first character

```
1    @Named
2    public class Bean1
3    {
4    . . .
5    }
```

```
<h:outputText
   value=
  "#{MyBean.someProperty}" />
```

```
1    @Named("MyBean")
2    public class Bean2
3    {
4    . . .
5    }
```

This annotation is in the javax.inject package.

The common usage of this annotation is to make the bean available to an expression language, e.g. the JSF EL.

This annotation is similar to the JSF2 @ManagedBean and Spring @Component annotations.

Copyright © Descriptor Systems, 2001-2014. Course materials may not be reproduced in whole or in part without prior written consent of Joel Barnum

Injecting a Bean Reference

- CDI provides the **@Inject** annotation to initialize a reference to a bean or other resource
- CDI uses the type of the field to find a bean implementation to inject

```
1    public class Test
2    {
3       @Inject
4       private Bean1 bean1;
5
6       @Inject
7       private Bean2 bean2;
8
9       . . .
10   }
```

This annotation is in the javax.inject package.

This annotation is similar to the JSF2 @ManagedProperty and JEE @Resource annotations.

It's common to use interfaces with injection and type the field to an interface - CDI finds a class that implements the interface an injects an object of that type. CDI lets you specify more precisely WHICH implementation to inject by using "qualifiers". A qualifier is a custom annotation that you can attach to an implementation class.

Copyright © Descriptor Systems, 2001-2014. Course materials may not be reproduced in whole or in part without prior written consent of Joel Barnum

Specifying a Bean Scope

- CDI provides annotations to let you specify the scope of beans

Annotation	Description
@RequestScoped	Bean survives a single HTTP request
@SessionScoped	Bean survives an HTTP session
@ApplicationScoped	Bean lifetime same as app's
@ConversationScoped	Bean survives a CDI conversation

```
1    @Named
2    @RequestScoped
3    public class Bean1
4    {
5    . . .
6    }
```

2 - 8

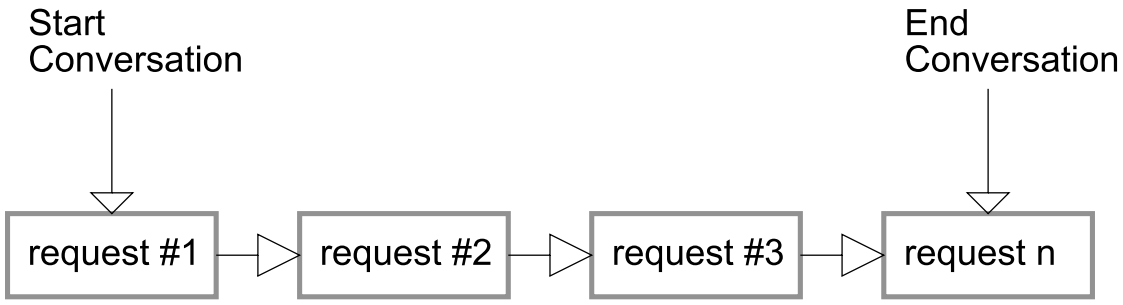
This annotation is in the javax.enterprise.context package.

These are similar to the equivalent JSF2 annotations, but are from a CDI package rather than a JSF package.

Copyright © Descriptor Systems, 2001-2014. Course materials may not be reproduced in whole or in part without prior written consent of Joel Barnum

Intro to Conversational Scope

- As with a session, CDI **conversations** can span multiple requests
- Unlike JEE sessions, conversations are tied to your business logic, rather than to a simple timeout

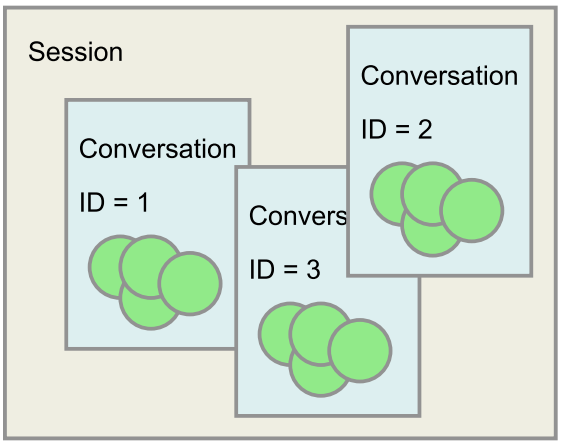


CDI provides facilities to start and end conversations base on your application's needs. CDI then manages the conversation for you -- no more issues with session memory leaks.

Copyright © Descriptor Systems, 2001-2014. Course materials may not be reproduced in whole or in part without prior written consent of Joel Barnum

How Do Conversations Work?

- CDI essentially partitions and manages the session so that conversations act as an application unit of work
- CDI partitions the session using a **conversation ID** -- each conversation has its own set of name/object pairs



2 - 10

Since CDI uses sessions behind the scenes to implement conversations, there are a couple of implications:

- Objects stored in a conversation must be serializable.

- The session timeout must exceed the maximum lifetime of any conversation.

Copyright © Descriptor Systems, 2001-2014. Course materials may not be reproduced in whole or in part without prior written consent of Joel Barnum

Types of Conversations

- CDI recognizes **transient** and **long-running** conversations
- Every request establishes a transient conversation, but you must manually establish a long-running conversation
- A transient conversation survives a redirect, but no longer
- Long-running conversations can span multiple requests

Copyright © Descriptor Systems, 2001-2014. Course materials may not be reproduced in whole or in part without prior written consent of Joel Barnum

The Conversation Object

- CDI provides the **Conversation** type which lets you control the boundaries of a conversation
- Applications typically inject a Converstation object reference

javax.enterprise.context.Conversation
begin():void begin(id:String):void end():void getId():String getTimeout():long isTransient():boolean setTimeout(timeout:long):void

Copyright © Descriptor Systems, 2001-2014. Course materials may not be reproduced in whole or in part without prior written consent of Joel Barnum

Starting a Conversation

```
1    @ConversationScoped
2    @Name("shoppingCart")
3    public class ShoppingCart
4    {
5       @Inject
6       private Conversation conv;
7
8       public void addItem()
9       {
10         conv.begin();
11         . . .
12      }
13   }
```


The @Begin annotation converts the current transient conversation into a long-running conversation.

Copyright © Descriptor Systems, 2001-2014. Course materials may not be reproduced in whole or in part without prior written consent of Joel Barnum

Ending a Conversation

```
1    @ConversationScoped
2    @Name("shoppingCart")
3    public class ShoppingCart
4    {
5       @Inject
6       private Conversation conv;
7    . . .
8       public void confirmOrder()
9       {
10         . . .
11         conv.end();
12      }
13   }
```


At the end of your unit-of-work, you can simply indicate that you want the conversation to terminate when the "action" method returns.

Copyright © Descriptor Systems, 2001-2014. Course materials may not be reproduced in whole or in part without prior written consent of Joel Barnum

Conversations and Multiple Tabs

- In a CDI application using conversations, each tab (window) has its own conversation, with its own conversation ID
- Thus users can work concurrently in multiple tabs on the same application without confusion

Copyright © Descriptor Systems, 2001-2014. Course materials may not be reproduced in whole or in part without prior written consent of Joel Barnum

Chapter Summary

In this chapter, you learned:

- The basics the CDI specification
- How to use CDI to name and inject beans
- How to use conversational scope

Copyright © Descriptor Systems, 2001-2014. Course materials may not be reproduced in whole or in part without prior written consent of Joel Barnum

Copyright © Descriptor Systems, 2001-2014. Course materials may not be reproduced in whole or in part without prior written consent of Joel Barnum