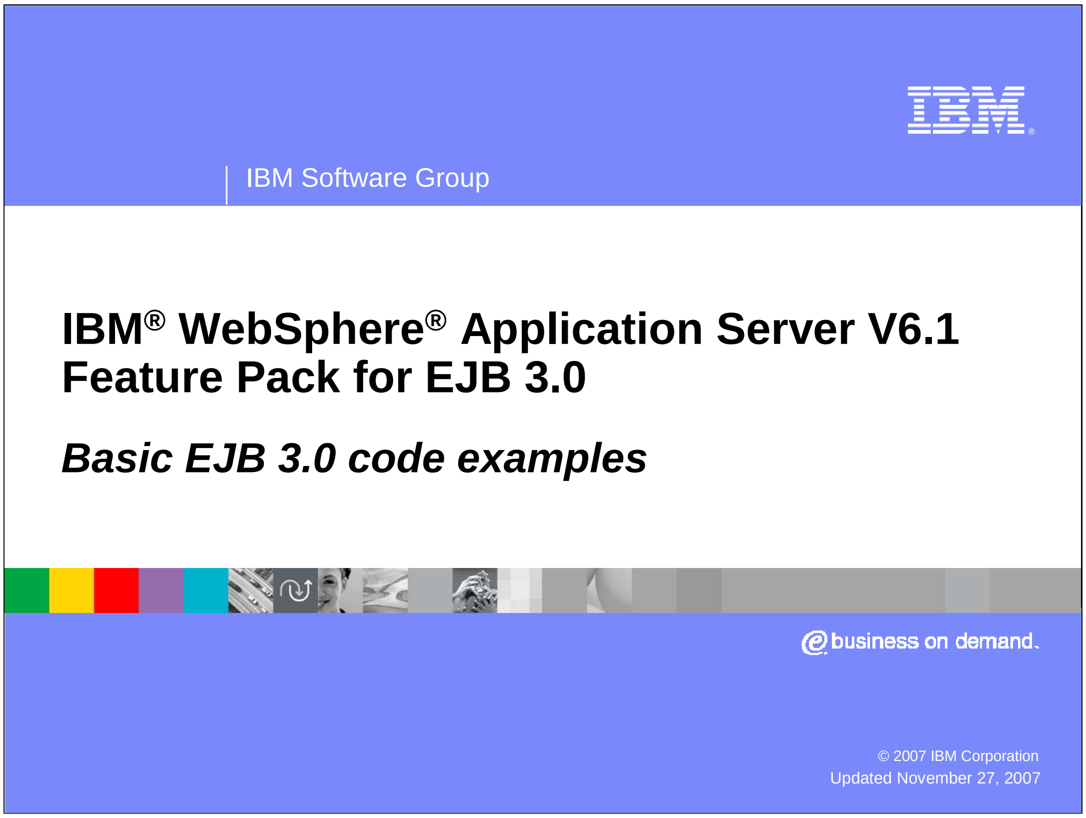


IBM Software Group

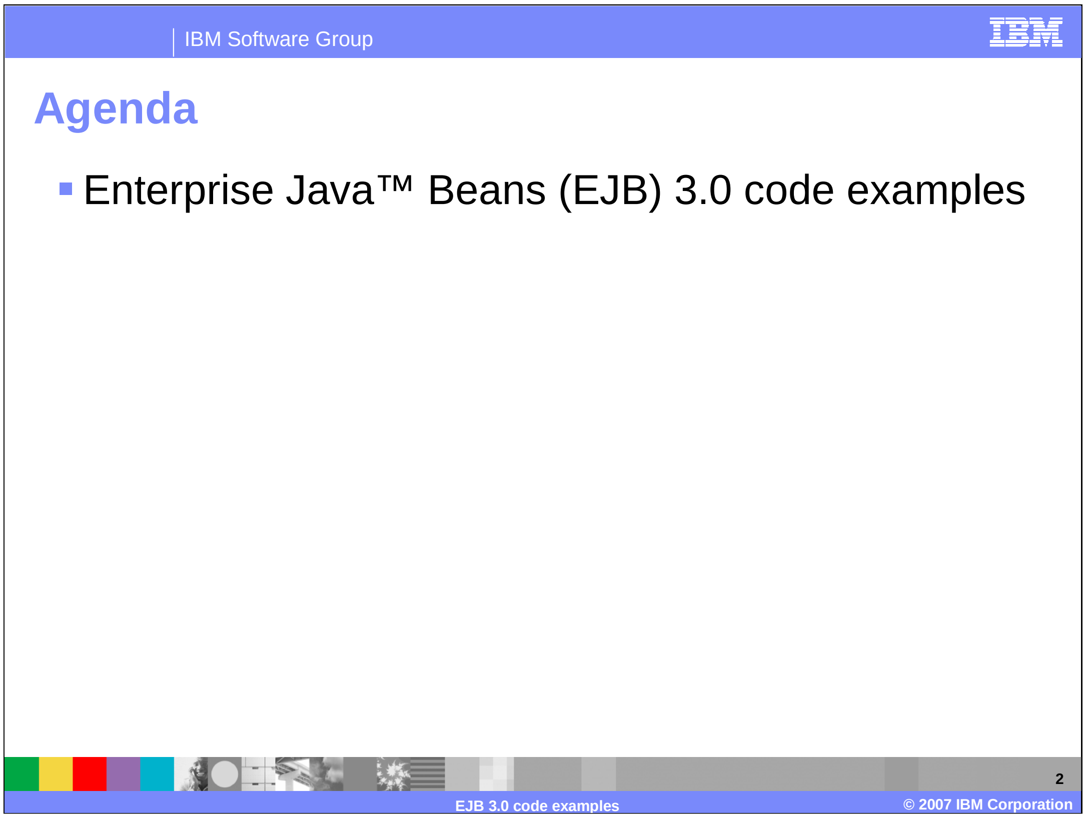
IBM® WebSphere® Application Server V6.1 Feature Pack for EJB 3.0

Basic EJB 3.0 code examples

business on demand.

Updated November 27, 2007

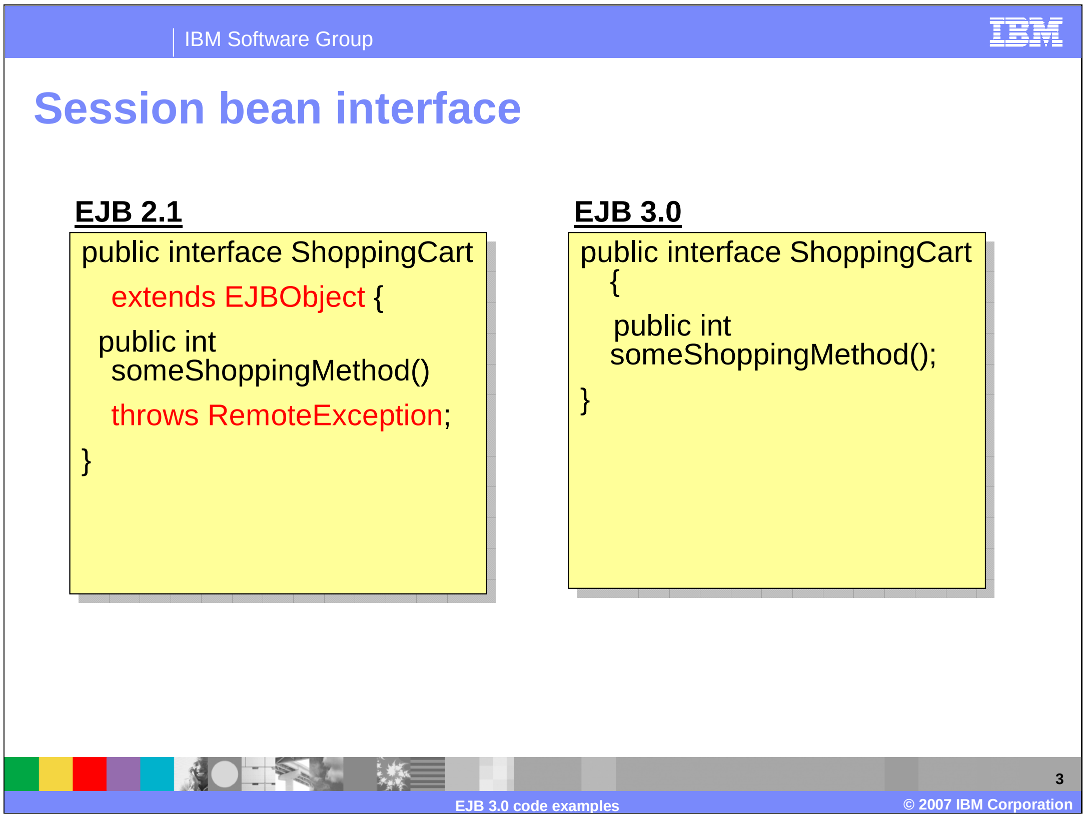
This presentation will cover basic EJB 3.0 code examples.



Agenda

- Enterprise Java™ Beans (EJB) 3.0 code examples

In this presentation, you will see several basic code examples. Many of them will show EJB 2.1 code alongside equivalent EJB 3.0 code, to help you understand the differences between the two programming models.



Session bean interface

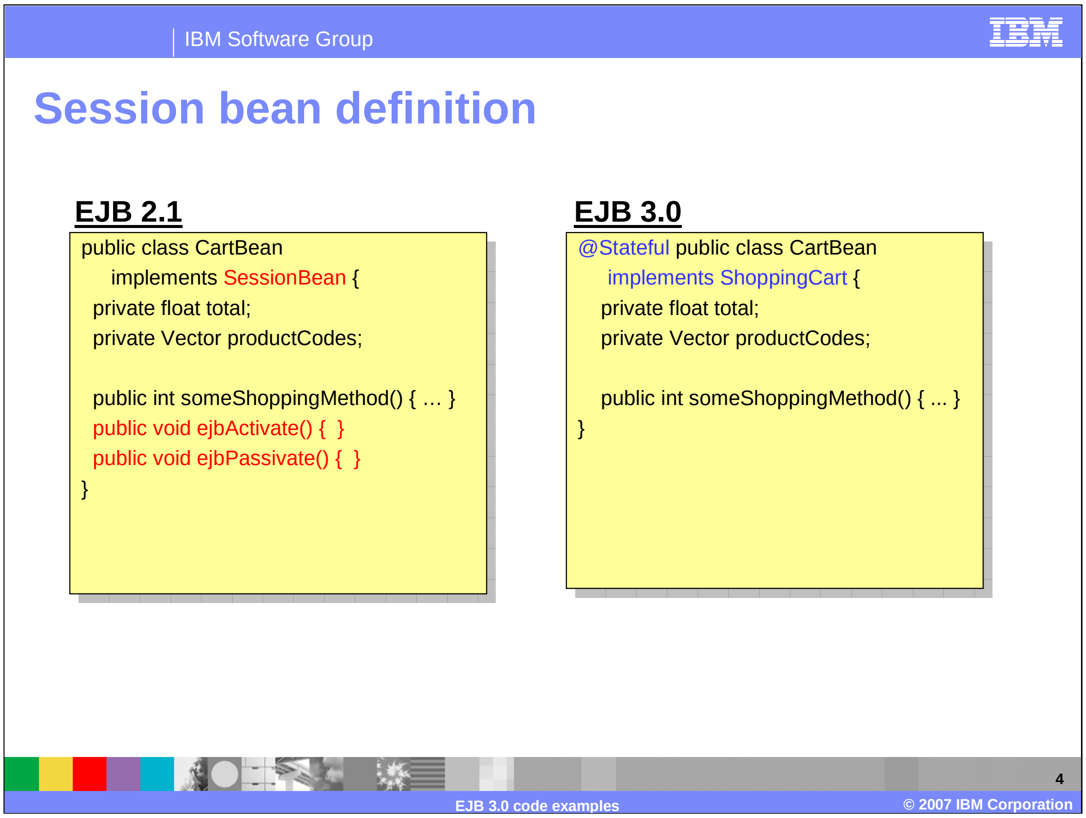
EJB 2.1

```
public interface ShoppingCart
   extends EJBObject {
  public int
   someShoppingMethod()
   throws RemoteException;
}
```

EJB 3.0

```
public interface ShoppingCart
   {
    public int
   someShoppingMethod();
}
```

This first example shows a simple session bean interface. In EJB 2.1, your interface must extend EJBObject, and you are required to throw a RemoteException. The EJB 3.0 interface is just a plain Java interface, with no special requirements. In EJB 3.0, bean interfaces are optional. If you do not create one, it will be generated automatically at runtime. One reason that you might want to create your own interface is if you want to selectively expose the public methods on your bean. When an interface is automatically generated, all public methods are exposed.



Session bean definition

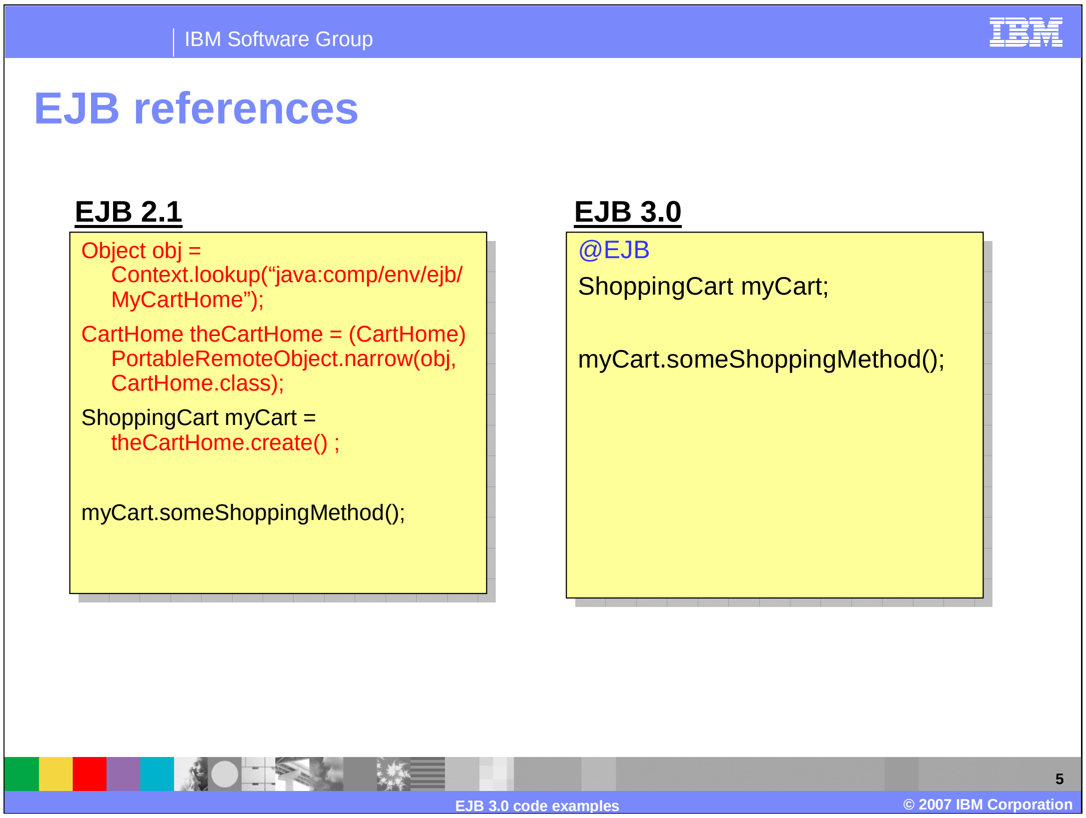
EJB 2.1

```
public class CartBean
    implements SessionBean {
  private float total;
  private Vector productCodes;

  public int someShoppingMethod() { … }
  public void ejbActivate() {  }
  public void ejbPassivate() {  }
}
```

EJB 3.0

```
@Stateful public class CartBean
    implements ShoppingCart {
    private float total;
    private Vector productCodes;

    public int someShoppingMethod() { ... }
}
```

In this example, note that the EJB 3.0 session bean is a plain-old Java object (POJO) that implements its own interface, not “SessionBean”. It is recognized as a stateful session bean because of the “@Session” annotation that precedes the class definition. EJB 3.0 also does not require implementation of the callback methods you seen in the EJB 2.1 beans, ejbActivate() and ejbPassivate(). You need only implement them if you intend to use them. If you did not create a business interface, all of your public methods will be exposed by the automatically-generated interface. Alternately, you can use the “@BusinessMethod” annotation to selectively expose only certain methods. Note that in this basic example, the EJB 2.1 code would also require an XML deployment descriptor, while the EJB 3.0 code can stand on its own and run without any additional metadata.



EJB references

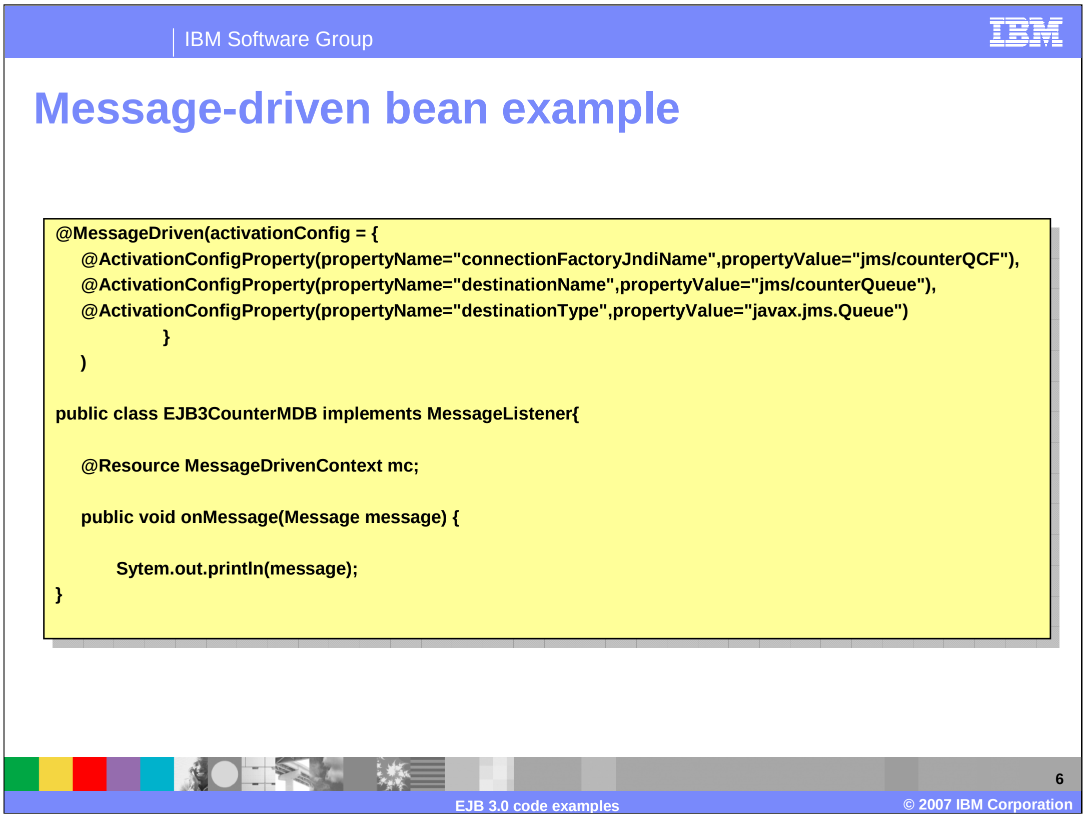
EJB 2.1

```
Object obj =
    Context.lookup(“java:comp/env/ejb/
    MyCartHome”);
CartHome theCartHome = (CartHome)
    PortableRemoteObject.narrow(obj,
    CartHome.class);
ShoppingCart myCart =
    theCartHome.create() ;


myCart.someShoppingMethod();
```

EJB 3.0

```
@EJB
ShoppingCart myCart;


myCart.someShoppingMethod();
```

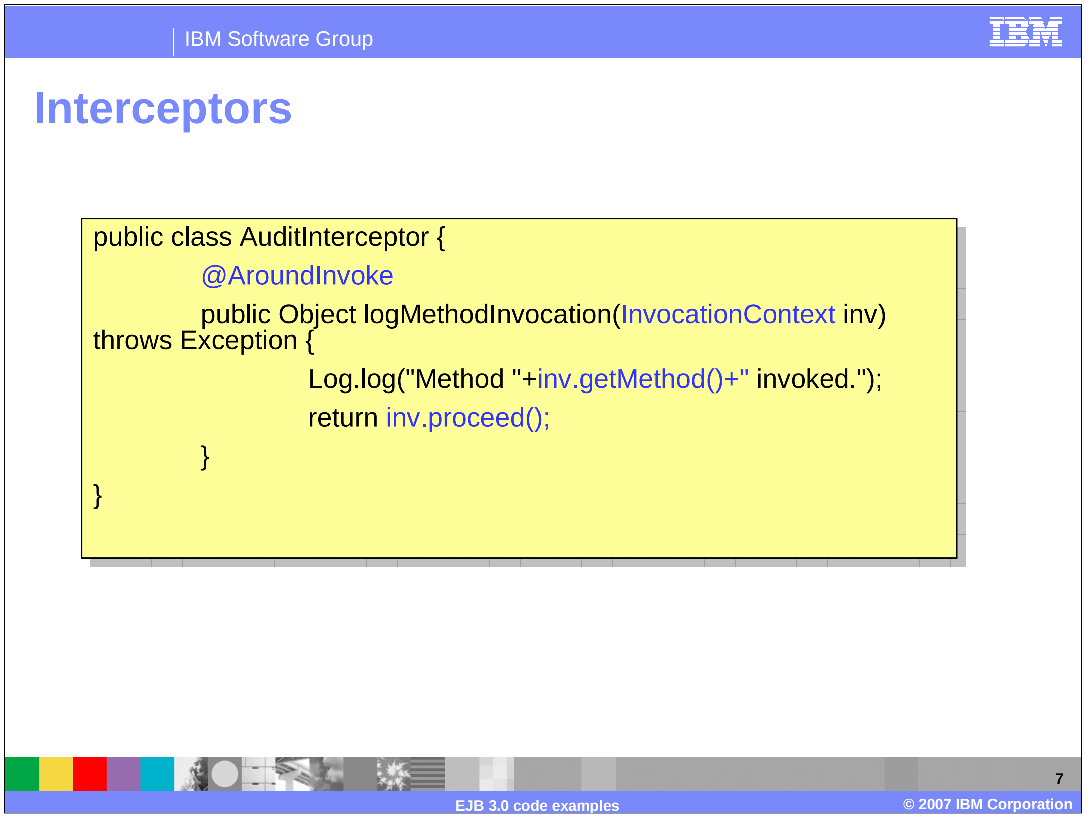
Here you see the difference between EJB 2.1 and 3.0 when it comes to EJB references. With EJB 2.1, if you want to call an EJB from the Web container or from another EJB, you need to first lookup the bean’s home interface, then use “narrow()” to cast the object to the appropriate interface type, and call “create()” to instantiate the bean. With EJB 3.0, you can use the @EJB annotation to easily inject an instance of an EJB, as shown in the example on the right. The example injects an instance of ShoppingCart named myCart, and you can then call methods on that object as you normally would.



Message-driven bean example

```
@MessageDriven(activationConfig = {
    @ActivationConfigProperty(propertyName="connectionFactoryJndiName",propertyValue="jms/counterQCF"),
    @ActivationConfigProperty(propertyName="destinationName",propertyValue="jms/counterQueue"),
    @ActivationConfigProperty(propertyName="destinationType",propertyValue="javax.jms.Queue")
                }
    )

public class EJB3CounterMDB implements MessageListener{

    @Resource MessageDrivenContext mc;

    public void onMessage(Message message) {

         Sytem.out.println(message);
}
```

This slide shows a basic EJB 3.0 message-driven bean, designated as such by the “@MessageDriven” annotation. You can use the “@ActivationConfigProperty” annotation to provide information for the bean to use, such as destination and connection factory JNDI names. You must also provide a bindings file so that the bean will be able to locate the objects that exist with these names. Bindings files are discussed in detail in the Information Center. It is not necessary to inject the MessageDrivenContext in this example, but it is shown here to demonstrate how it can be injected to give you access to information maintained by the container, such as transaction information.



IBM Software Group

Interceptors

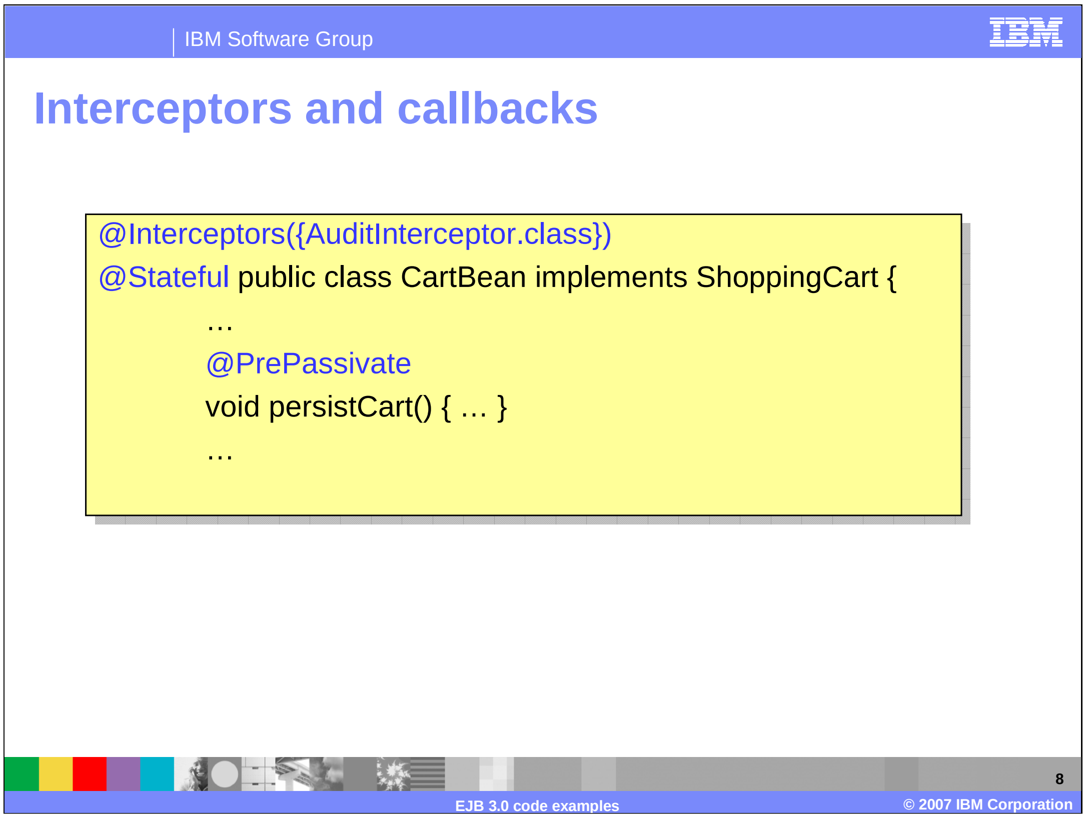
```
public class AuditInterceptor {
            @AroundInvoke
            public Object logMethodInvocation(InvocationContext inv)
throws Exception {
                        Log.log("Method "+inv.getMethod()+" invoked.");
                        return inv.proceed();
            }
}
```

7

EJB 3.0 code examples

© 2007 IBM Corporation

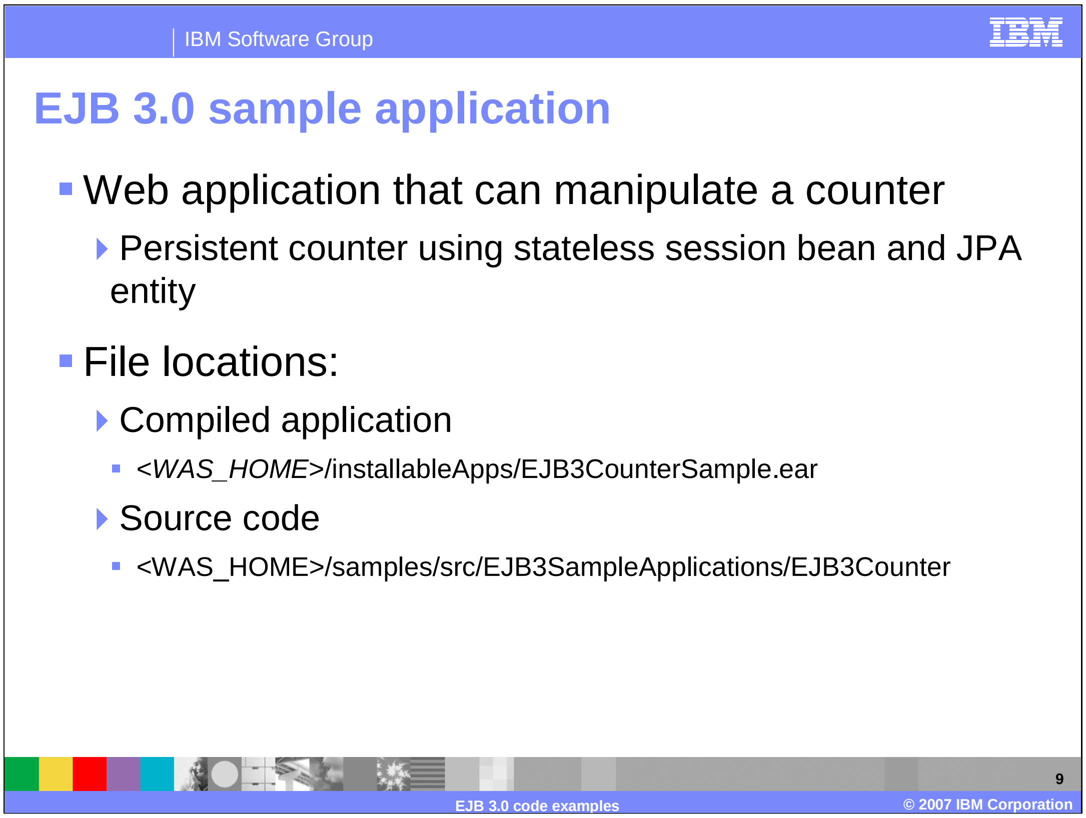
Interceptors give you the ability to do pre- and post-processing, similar to what you can do with a servlet. An interceptor can be defined within the class or in a separate class, and must take an InvocationContext as an argument. A complete list of available interceptors can be found in the EJB 3.0 specification. This example uses a separate class to define an interceptor that logs the name of each method being called.



Interceptors and callbacks

```
@Interceptors({AuditInterceptor.class})
@Stateful public class CartBean implements ShoppingCart {
        …
        @PrePassivate
        void persistCart() { … }
        …
```

This snippet shows an example of how you specify that a class should use an interceptor. The “@Interceptors” annotation can take a list of interceptor classes as arguments. There is also an example here of how to use callback annotations. In this case, the “@PrePassivate” annotation says that the “persistCart()” method should be called immediately before the bean is passivated.



IBM Software Group

EJB 3.0 sample application

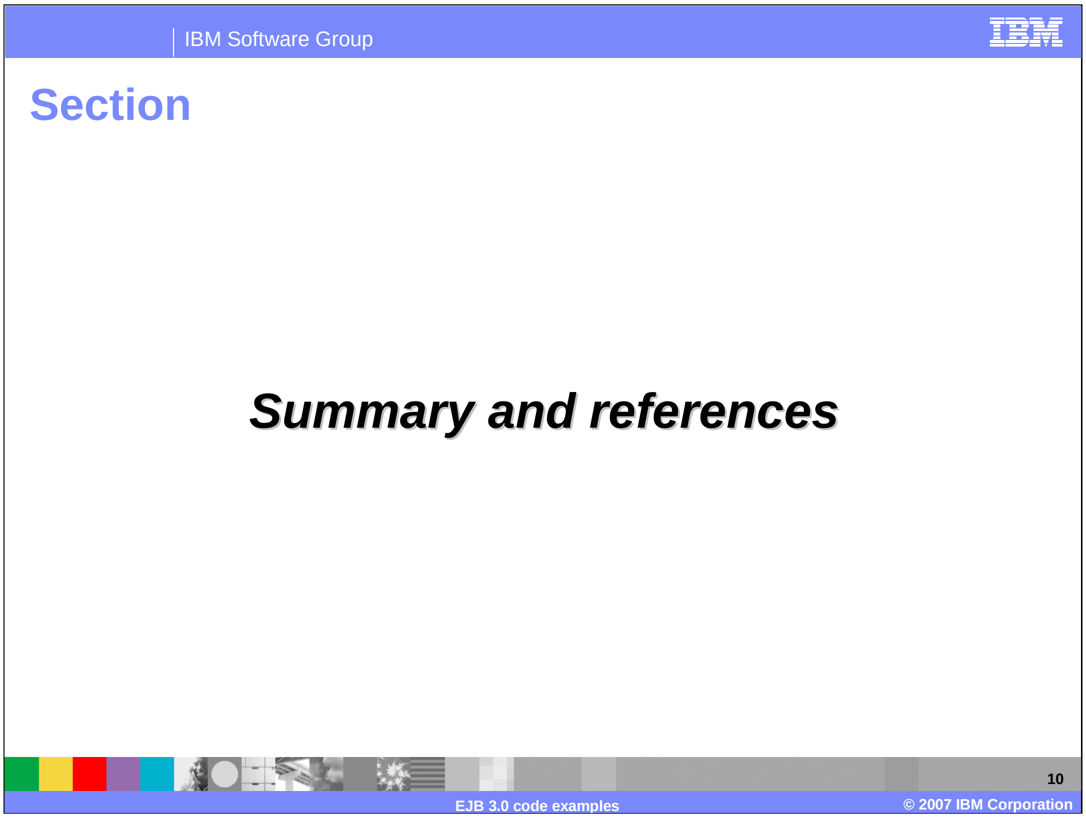
- Web application that can manipulate a counter
  - Persistent counter using stateless session bean and JPA entity
- File locations:
  - Compiled application
    - <*WAS_HOME*>/installableApps/EJB3CounterSample.ear
  - Source code
    - <WAS_HOME>/samples/src/EJB3SampleApplications/EJB3Counter

9

EJB 3.0 code examples

© 2007 IBM Corporation

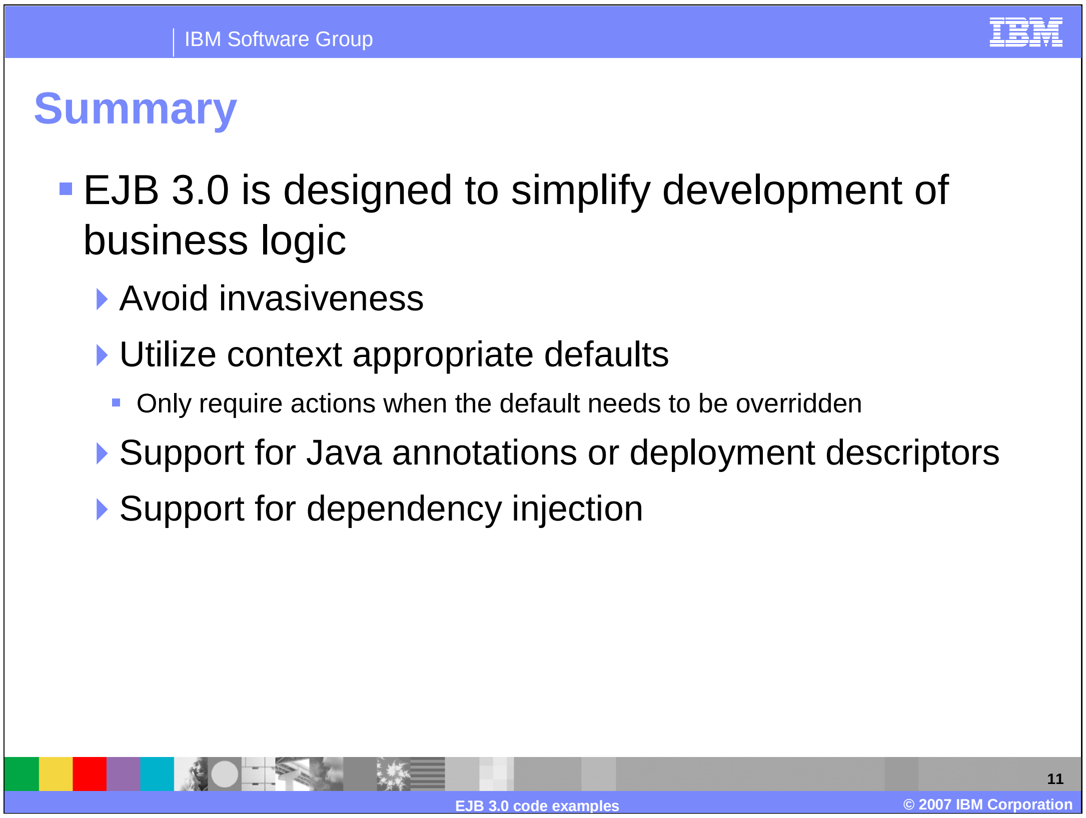
If you have installed the Feature Pack for EJB 3.0, you can find a sample EJB 3.0 application in the “installableApps” directory. The source code for the application can be found in the “samples” directory. The sample application comes with an Ant build.xml file to show how EJB 3.0 applications can be built using Ant. You can also download a lab exercise that will walk you through the process of configuring the Application Server Toolkit to support development of EJB 3.0 applications using this sample.



Section

Summary and references

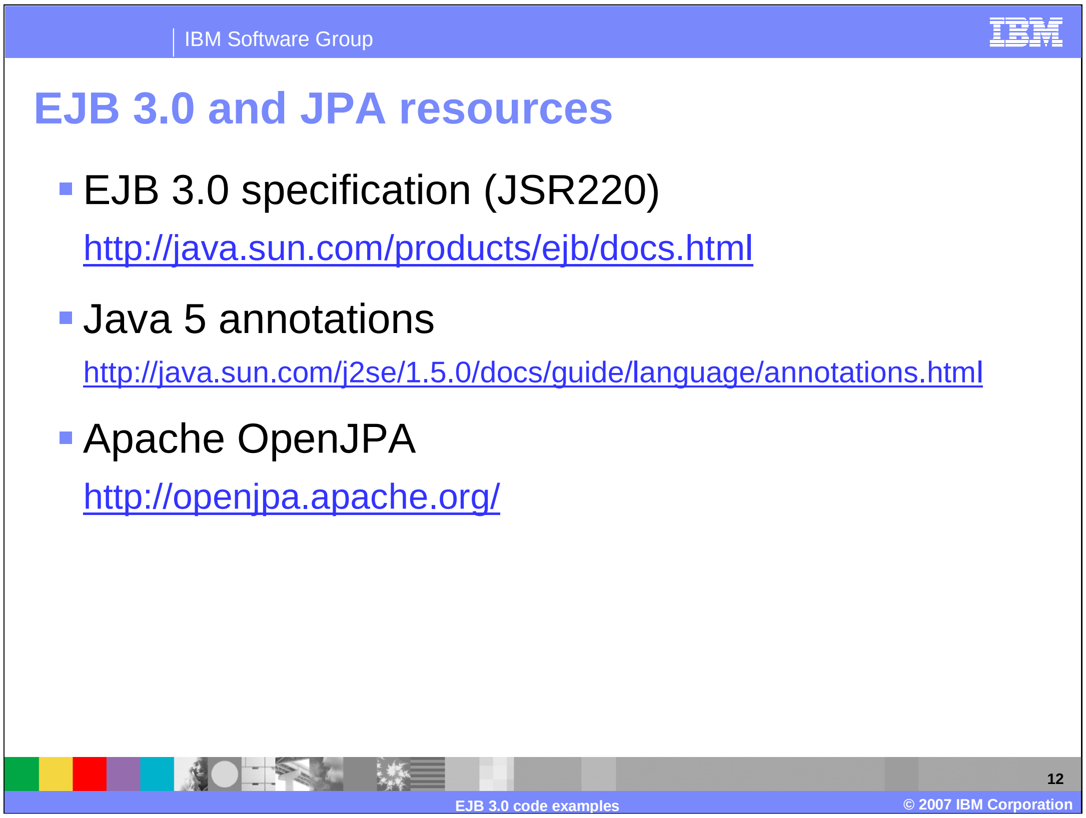
This section will summarize the presentation.



Summary

- EJB 3.0 is designed to simplify development of business logic
  - Avoid invasiveness
  - Utilize context appropriate defaults
    - Only require actions when the default needs to be overridden
  - Support for Java annotations or deployment descriptors
  - Support for dependency injection

The EJB 3.0 specification is designed to make it faster and easier to develop your business logic. By introducing Java annotations for EJBs, dependency injection support, removing boilerplate code, and making use of intelligent defaults, EJB 3.0 provides a framework that enables anyone with Java skills to develop enterprise beans. The code snippets in this presentation have given basic examples of how create interfaces, session beans, and message-driven beans, and also how to use interceptors and callback methods. The samples also show how the POJO-based EJB 3.0 framework is simpler than EJB 2.1.



EJB 3.0 and JPA resources

- EJB 3.0 specification (JSR220)

  http://java.sun.com/products/ejb/docs.html

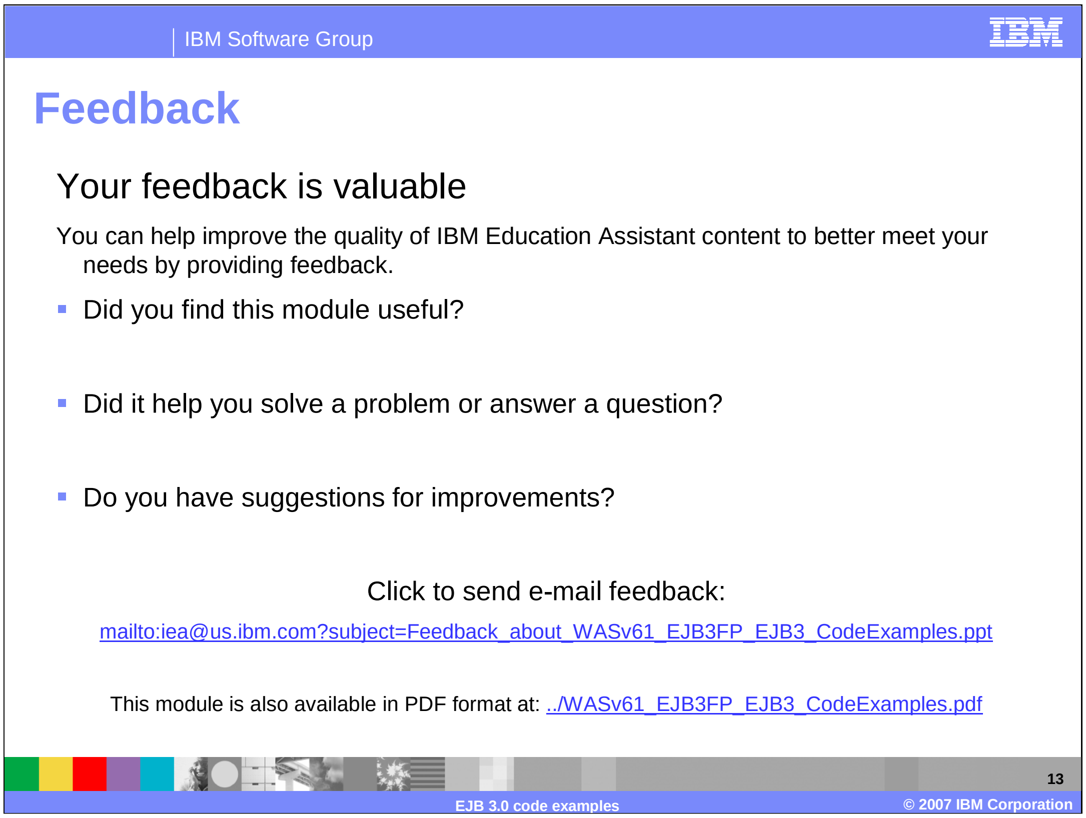
- Java 5 annotations

  http://java.sun.com/j2se/1.5.0/docs/guide/language/annotations.html

- Apache OpenJPA

  http://openjpa.apache.org/

This slide lists some resources where you can learn more about EJB 3.0 and JPA. You should also consult the Feature Pack for EJB 3.0 Information Center for more information.



Feedback

Your feedback is valuable

You can help improve the quality of IBM Education Assistant content to better meet your needs by providing feedback.

- Did you find this module useful?
- Did it help you solve a problem or answer a question?
- Do you have suggestions for improvements?

Click to send e-mail feedback:

mailto:iea@us.ibm.com?subject=Feedback_about_WASv61_EJB3FP_EJB3_CodeExamples.ppt

This module is also available in PDF format at: ../WASv61_EJB3FP_EJB3_CodeExamples.pdf

You can help improve the quality of IBM Education Assistant content by providing feedback.

Trademarks, copyrights, and disclaimers