Architecture of Enterprise Applications 3
Server Component – Enterprise Bean

Haopeng Chen

***RE**liable, **IN**telligent and **S**calable Systems Group (**REINS**)*
Shanghai Jiao Tong University
Shanghai, China
http://reins.se.sjtu.edu.cn/~chenhp
e-mail: chen-hp@sjtu.edu.cn

Contents

- Enterprise JavaBean
  - Overview
  - Session Bean
  - Stateless Session Bean
  - Web Client
  - Stateful Session Bean

What is Enterprise Bean?

- An enterprise bean is
  - written in the Java programming language
  - a server-side component that encapsulates the business logic of an application.

- The **business logic** is the code that fulfills the purpose of the application.
  - For example, in an inventory control application, the enterprise beans might implement the business logic in methods called `checkInventoryLevel` and `orderProduct`.
  - By invoking these methods, clients can access the inventory services provided by the application.

Benefits of Enterprise Beans

- First,
  - because the EJB container provides system-level services to enterprise beans,
  - the bean developer can concentrate on solving business problems.
- Second,
  - because the beans rather than the clients contain the application's business logic,
  - the client developer can focus on the presentation of the client..
- Third,
  - because enterprise beans are portable components,
  - the application assembler can build new applications from existing beans.

When to Use Enterprise Beans?

- You should consider using enterprise beans if your application has any of the following requirements.
  - The application must be scalable.
  - Transactions must ensure data integrity.
  - The application will have a variety of clients.

- Types of Enterprise Beans
  - Session
    - Performs a task for a client; optionally, may implement a web service
  - Message-driven
    - Acts as a listener for a particular messaging type, such as the Java Message Service API

What Is a Session Bean?

- A session bean
  - encapsulates business logic that can be invoked programmatically by a client over local, remote, or web service client views.

- Session beans are of three types:
  - Stateful:
    - In a stateful session bean, the instance variables represent the state of a unique client/bean session. This state is often called the conversational state.
  - Stateless:
    - A stateless session bean does not maintain a conversational state with the client.
  - Singleton:
    - A singleton session bean is instantiated once per application and exists for the lifecycle of the application.

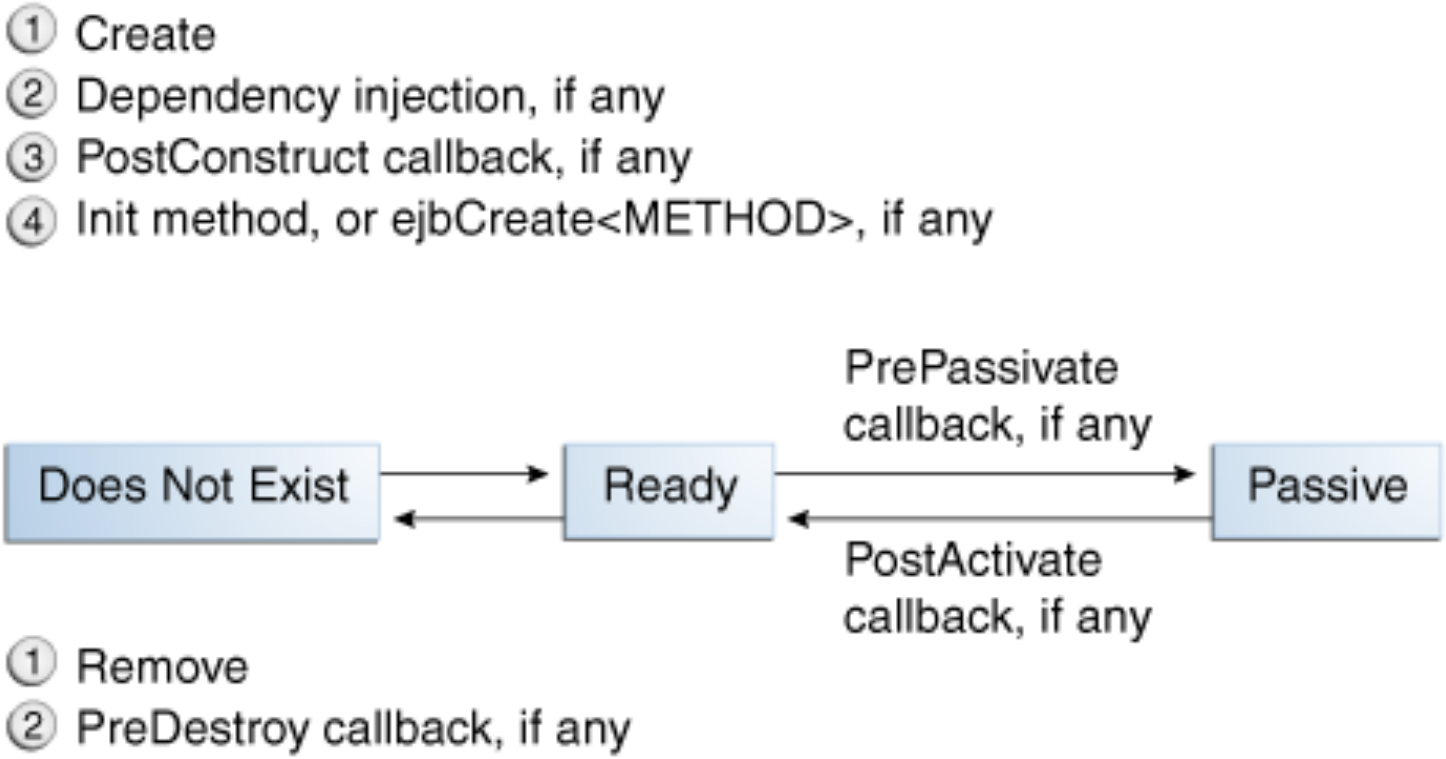
The Contents of an Enterprise Bean

- To develop an enterprise bean, you must provide the following files:
  - Enterprise bean class:
    - Implements the business methods of the enterprise bean and any lifecycle callback methods.
  - Business interfaces:
    - Define the business methods implemented by the enterprise bean class. A business interface is not required if the enterprise bean exposes a local, no-interface view.
  - Helper classes:
    - Other classes needed by the enterprise bean class, such as exception and utility classes.

- Naming Conventions for Enterprise Beans
  - Enterprise bean class
    - *name*Bean
    - For example: AccountBean
  - Business interface
    - *name*
    - For example: Account

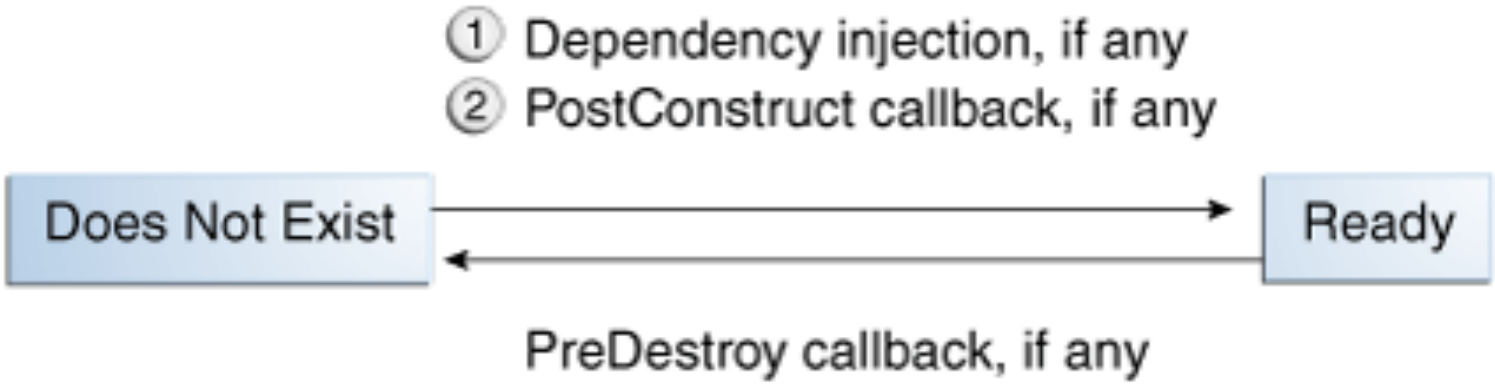
The Lifecycles of Enterprise Beans

- The Lifecycle of a Stateful Session Bean



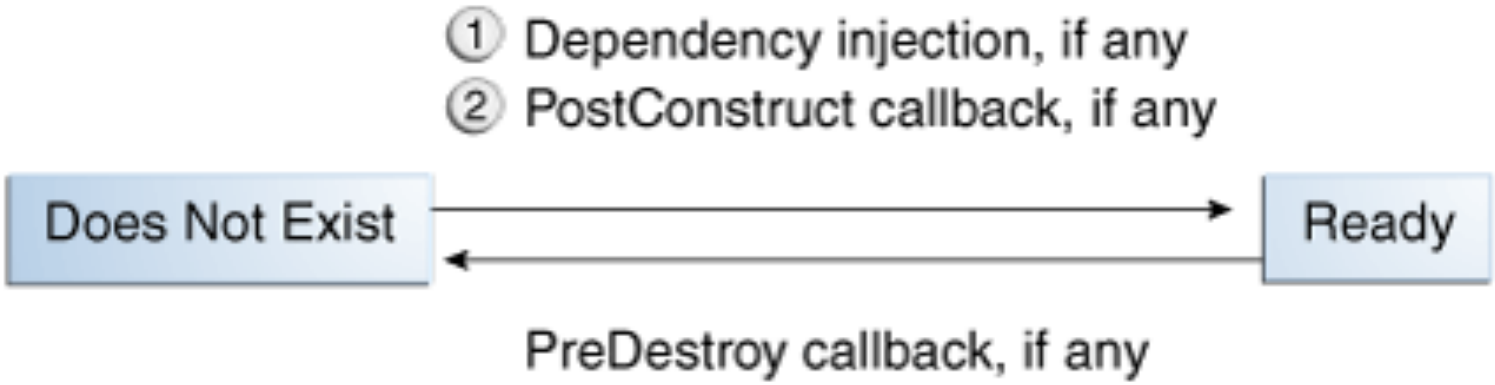
The Lifecycles of Enterprise Beans

- The Lifecycle of a Stateless Session Bean



The Lifecycles of Enterprise Beans

- The Lifecycle of a Singleton Session Bean



To develop an enterprise bean

- A stateless session bean: `HelloWorldBean`

- Interface of `HelloWorldBean`

HelloWorld.java

```
package HelloWorld;

import javax.ejb.Remote;

@Remote
public interface HelloWorld {
    public String hello();
}
```

To develop an enterprise bean

- A stateless session bean: `HelloWorldBean`

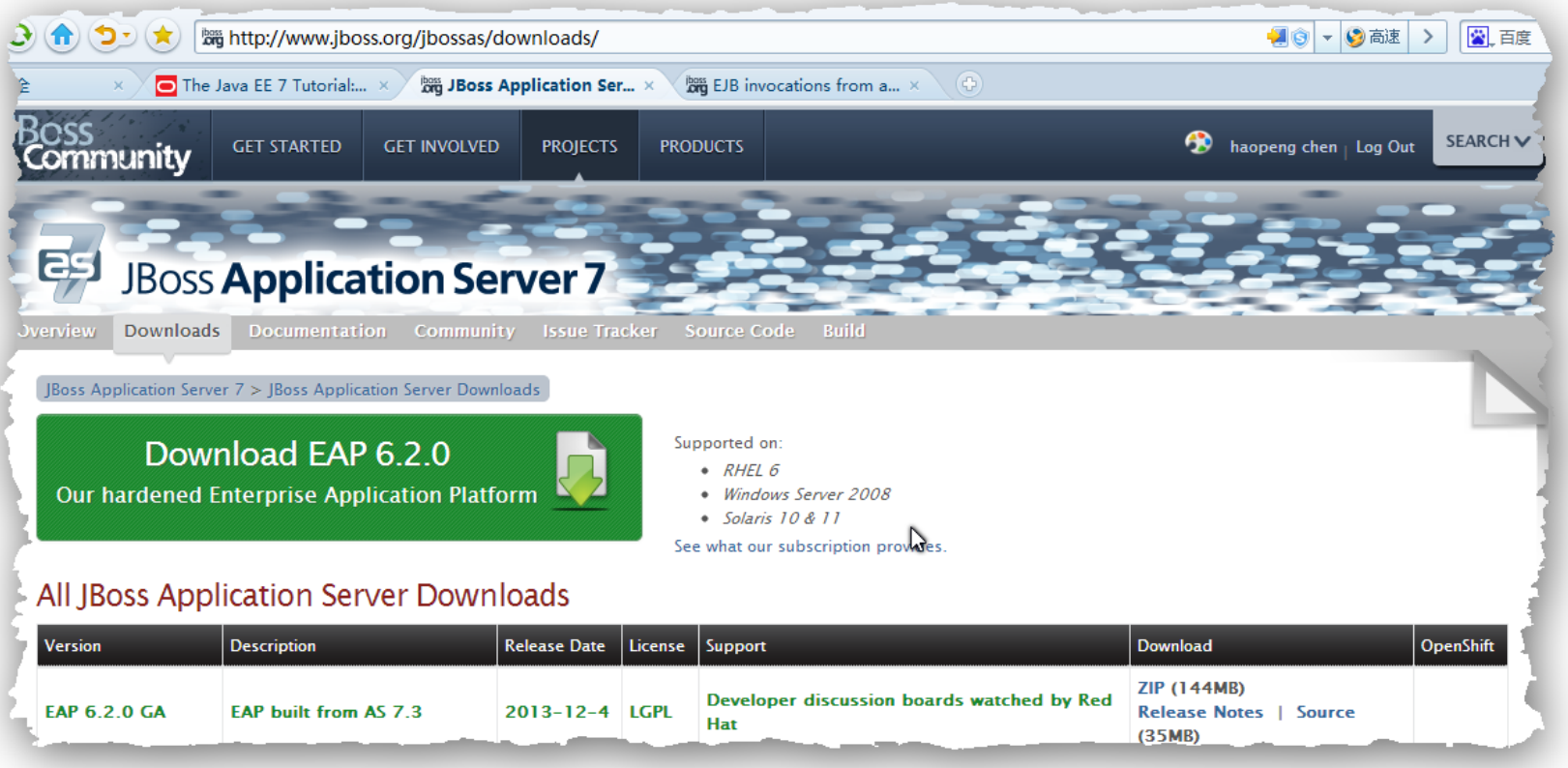
- Implementation class of `HelloWorldBean`

  HelloWorldBean.java

```
package HelloWorld;

import javax.ejb.Stateless;

@Stateless
public class HelloWorldBean implements HelloWorld {
    public HelloWorldBean() {}
    public String hello(){
          System.out.println("Hello World!!");
          return "Hello World!";
    }
}
```

To develop an enterprise bean

- Deploy `HelloWorldBean` into JBoss EAP 6.2



To develop an enterprise bean

EJB Project

Create an EJB Project and add it to a new or existing Enterprise Application.

Project name: HelloWorld_

Project location

☑ Use default location

Location: E:\Projects\JavaEE\HelloWorld_ Browse...

Target runtime

JBoss EAP 6.1+ Runtime New Runtime...

EJB module version

3.1

Configuration

Default Configuration for JBoss EAP 6.1+ Runtime Modify...

A good starting point for working with JBoss EAP 6.1+ Runtime runtime. Additional facets can later be installed to add new functionality to the project.

EAR membership

☐ Add project to an EAR

EAR project name: EAR New Project...

Working sets

☐ Add project to working sets

Working sets: Select...

< Back Next > Finish Cancel

- HelloWorld
  - Deployment Descriptor: HelloWorld
  - JAX-WS Web Services
  - ejbModule
    - HelloWorld
      - HelloWorld.java
      - HelloWorldBean.java
    - META-INF
  - JRE System Library [jdk1.7.0_51]
  - JBoss Enterprise Application Platform 6.1+
  - EAR Libraries
  - build

Markers | Properties | Servers | Data Source Explorer

- JBoss EAP 6.1+ Runtime Server [Started, Synchronized]
- Tomcat v7.0 Server at localhost [Stopped]

To develop an enterprise bean

- Three JNDI namespaces are used for portable JNDI lookups:
  - The `java:global` JNDI namespace is the portable way of finding remote enterprise beans using JNDI lookups.

    ```
    java:global[/application name]/module name/enterprise bean name
    [/interface name ]
    ```

  - The `java:module` namespace is used to look up local enterprise beans within the same module.

    ```
    java:module/enterprise bean name/[interface name]
    ```

  - The `java:app` namespace is used to look up local enterprise beans packaged within the same application.

    ```
    java:app[/module name]/enterprise bean name [/interface name]
    ```

To develop an enterprise bean

Markers | Properties | Servers | Data Source Explorer | Snippets | Console

```
JBoss EAP 6.1+ Runtime Server [JBoss Application Server Startup Configuration] C:\Program Files\Java\jdk1.7.0_51\bin\javaw.exe (2014年2月26日 下午3:12:13)
15:12:17,992 INFO  [org.jboss.as.webservices] (ServerService Thread Pool -- 46) JBAS015537: 激活WebServices Extension
15:12:18,037 INFO  [org.jboss.as.connector.subsystems.datasources] (ServerService Thread Pool -- 25) JBAS010403: 部署兼容JDBC ?动  ?class org.h2.D
15:12:19,035 INFO  [org.jboss.as.server.deployment.scanner] (MSC service thread 1-3) JBAS015012: 为目录FileSystemDeploymentService 启动的E:\jboss-e
15:12:19,047 INFO  [org.jboss.as.server.deployment] (MSC service thread 1-3) JBAS015876: 开始"HelloWorld.jar" 的部署(runtime-name: "HelloWorld.ja
15:12:19,399 INFO  [org.jboss.ws.common.management] (MSC service thread 1-1) JBWS022052: Starting JBoss Web Services - Stack CXF Server 4.2.3.F
15:12:19,404 INFO  [org.jboss.as.ejb3.deployment.processors.EjbJndiBindingsDeploymentUnitProcessor] (MSC service thread 1-3) JNDI bindings for

        java:global/HelloWorld/HelloWorldBean!HelloWorld.HelloWorld
        java:app/HelloWorld/HelloWorldBean!HelloWorld.HelloWorld
        java:module/HelloWorldBean!HelloWorld.HelloWorld
        java:jboss/exported/HelloWorld/HelloWorldBean!HelloWorld.HelloWorld
        java:global/HelloWorld/HelloWorldBean
        java:app/HelloWorld/HelloWorldBean
        java:module/HelloWorldBean
```

To develop an enterprise bean

- Develop a client of `HelloWorldBean`

- Instantiate an `InitialContext`

```
final Hashtable jndiProperties = new Hashtable();
jndiProperties.put(Context.URL_PKG_PREFIXES,
            "org.jboss.ejb.client.naming");
final Context context = new InitialContext(jndiProperties);
```

- Create `jboss-ejb-client.properties`

```
endpoint.name=client-endpoint
remote.connectionprovider.create.options.org.xnio.Options.
     SSL_ENABLED=false
remote.connections=default
remote.connection.default.host=localhost
remote.connection.default.port = 4447
remote.connection.default.connect.options.org.xnio.Options.
     SASL_POLICY_NOANONYMOUS=false
remote.connection.default.username=YOURNAME
remote.connection.default.password=YOURPWD
```

To develop an enterprise bean

- Lookup `HelloWorldBean`

```
HelloWorld hello = (HelloWorld) context.lookup
 ("ejb:/HelloWorld//HelloWorldBean!HelloWorld.HelloWorld");
```

or

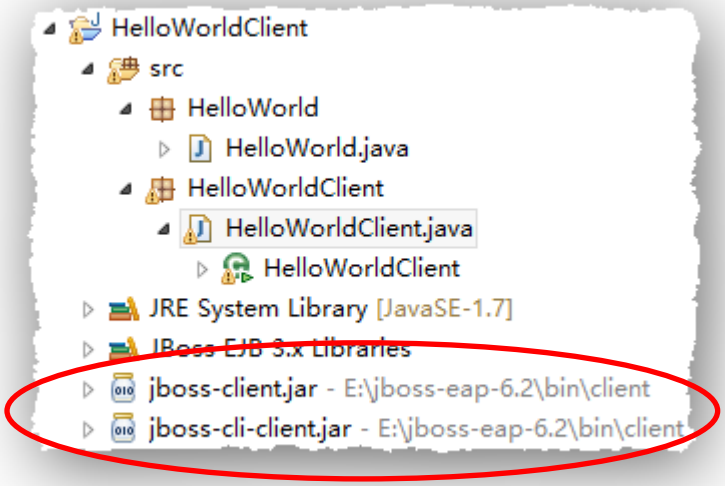
```
final String appName = "";
final String moduleName = "HelloWorld";
final String distinctName = "";
final String beanName = "HelloWorldBean";
final String viewClassName = "HelloWorld.HelloWorld";
HelloWorld hello = (HelloWorld) context.lookup
    ("ejb:" + appName + "/" + moduleName + "/" +
     distinctName + "/" + beanName + "!" + viewClassName);
```

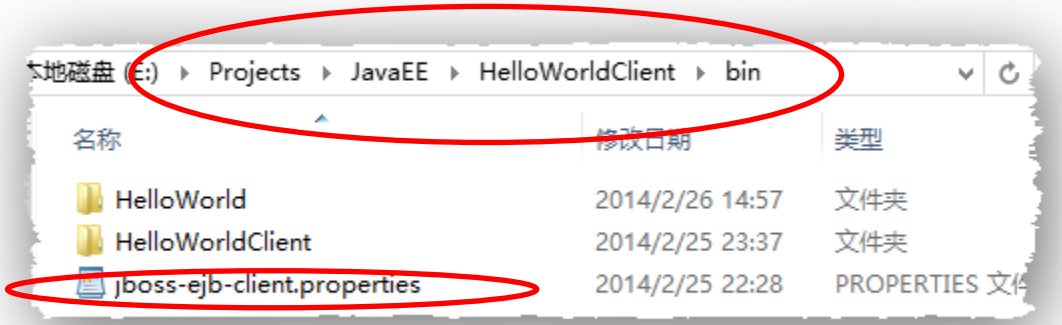
To develop an enterprise bean

- Client of HelloWorldClient
- HelloWorldClient.java

```
package HelloWorldClient;
import java.util.Hashtable;
import javax.naming.Context;
import javax.naming.InitialContext;
import HelloWorld.HelloWorld;

public class HelloWorldClient {
    public static void main(String[] args) {
      try{
            final Hashtable jndiProperties = new Hashtable();
            jndiProperties.put(Context.URL_PKG_PREFIXES,
                "org.jboss.ejb.client.naming");
            final Context context = new InitialContext(jndiProperties);

             HelloWorld hello = (HelloWorld) context.lookup
               ("ejb:/HelloWorld//HelloWorldBean!HelloWorld.HelloWorld");
            System.out.println(hello.hello());
      }catch(Exception e){
         e.printStackTrace();
      }

}
```

To develop an enterprise bean





An more useful enterprise bean

- Create a dynamic web project ConverterWeb
- ConverterBean.java

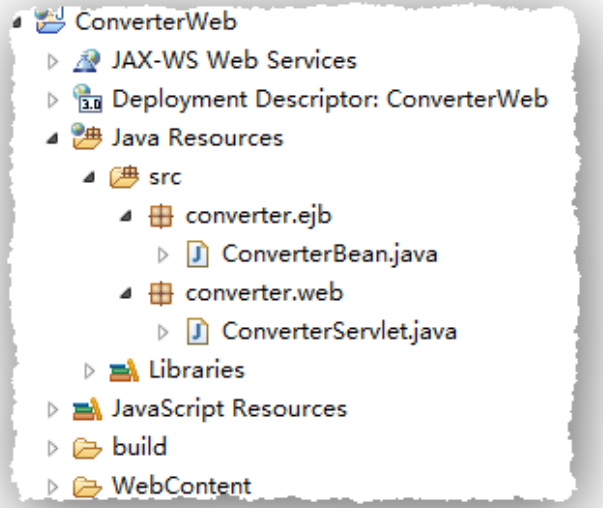
```
package converter.ejb;
import java.math.BigDecimal;
import javax.ejb.Stateless

@Stateless
public class ConverterBean {
    private BigDecimal yenRate = new BigDecimal("79.3916");
    private BigDecimal euroRate = new BigDecimal("0.0100169");
    public BigDecimal dollarToYen(BigDecimal dollars) {
        BigDecimal result = dollars.multiply(yenRate);
        return result.setScale(2, BigDecimal.ROUND_UP);
    }
    public BigDecimal yenToEuro(BigDecimal yen) {
       BigDecimal result = yen.multiply(euroRate);
       return result.setScale(2, BigDecimal.ROUND_UP);
    }
}
```

An more useful enterprise bean

- Create a dynamic web project ConverterWeb
- ConverterServlet.java(snippets)

```
@WebServlet
public class ConverterServlet extends HttpServlet {
 @EJB
 ConverterBean converter;
 ...
}

try {
 String amount = request.getParameter("amount");
 if (amount != null && amount.length()> 0) {
     BigDecimal d = new BigDecimal(amount);
     BigDecimal yenAmount = converter.dollarToYen(d);
     BigDecimal euroAmount = converter.yenToEuro(yenAmount);
 ...
 }
 ...
}
```

An more useful enterprise bean



Stateful Session Bean

- A shopping cart: `CartBean`
- `Cart.java`

```
package cart.ejb;

import java.util.List;
import cart.util.BookException;
import javax.ejb.Remote;

@Remote
public interface Cart {
    public void initialize(String person) throws BookException;

    public void initialize(String person, String id) throws BookException;

    public void addBook(String title);

    public void removeBook(String title) throws BookException;

    public List<String> getContents();

    public void remove();
}
```

Stateful Session Bean

- A shopping cart: `CartBean`

- CartBean.java

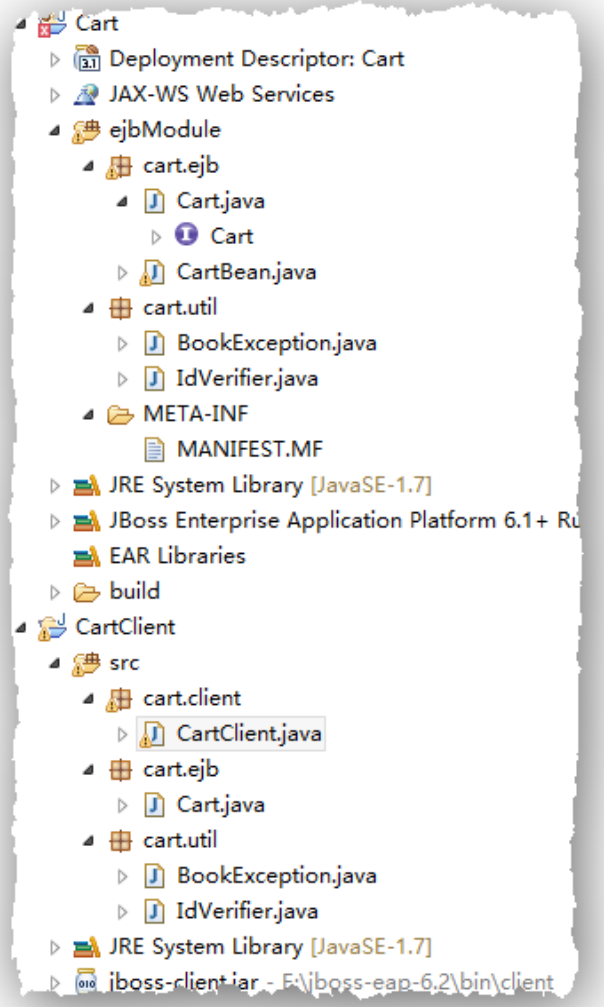
```
package cart.ejb;

@Stateful
public class CartBean implements Cart, Serializable {
    List<String> contents;
    String customerId;
    String customerName;

    @Override
    public void initialize(String person) throws BookException {
        if (person == null) {
            throw new BookException("Null person not allowed.");
        } else {
            customerName = person;
        }

        customerId = "0";
        contents = new ArrayList<>();
    }
    ……
    @Remove()
    @Override
    public void remove() {
        contents = null;
    }
}
```

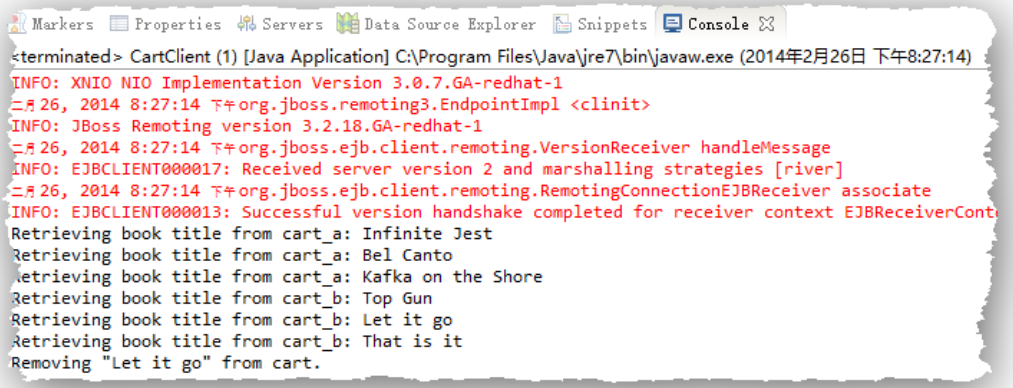
Stateful Session Bean

- A shopping cart client
- CartClient.java

```
package cart.ejb;

public class CartClient {
    public static void main(String[] args) {
        CartClient client = new CartClient(args);
        client.doTest();
    }

    public void doTest() {
        try {
              Cart cart = (Cart) context.lookup("ejb:" + appName + "/" +
                                moduleName + "/" + distinctName + "/" + beanName
                                + "!" + viewClassName + "?stateful");
              Cart cart_b = (Cart) context.lookup("ejb:" + appName + "/" +
                                moduleName + "/" + distinctName + "/" + beanName
                                + "!" + viewClassName + "?stateful");
              ............
        }
    }
}
```

Stateful Session Bean





Singleton Session Bean

- A counter

- CounterBean.java

```
package javaeetutorial.counter.ejb;

import javax.ejb.Singleton;

@Singleton
public class CounterBean {
    private int hits = 1;

    public int getHits() {
        return hits++;
    }
}
```

Singleton Session Bean

- A counter
- Counter.java

```
@Named
@ConversationScoped
public class Count implements Serializable {
    @EJB
    private CounterBean counterBean;

    private int hitCount;

    public Count() {
        this.hitCount = 0;
    }

    public int getHitCount() {
        hitCount = counterBean.getHits();
        return hitCount;
    }

    public void setHitCount(int newHits) {
        this.hitCount = newHits;
    }
}
```

Singleton Session Bean

- A counter
- index.xhtml

```
<?xml version='1.0' encoding='UTF-8' ?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html lang="en"
      xmlns="http://www.w3.org/1999/xhtml"
      xmlns:ui="http://java.sun.com/jsf/facelets">
    <body>
        This text above will not be displayed.
        <ui:composition template="/template.xhtml">
            This text will not be displayed.
            <ui:define name="title">
                This page has been accessed #{count.hitCount} time(s).
            </ui:define>
            This text will also not be displayed.
            <ui:define name="body">
                Hooray!
            </ui:define>
            This text will not be displayed.
        </ui:composition>
        This text below will also not be displayed.
    </body>
</html>
```

Singleton Session Bean

Requirements of Project

- To build your business logics with:
  - At least one stateful server and component, such as shopping cart
  - At least one stateless server and component, such as login

References

- Core Java (volume II) 9th edition
  - http://horstmann.com/corejava.html
- The Java EE 7 Tutorial
  - http://docs.oracle.com/javaee/7/tutorial/doc/javaeetutorial7.pdf

Thank You!