实验十一会话**Bean**——用会话**Bean**实现用户登录及购物

车应用

一、基础实验——无状态会话**Bean**的调用

# （一）实验目的

1. 掌握 EJB 的概念；
2. 掌握 JBoss 服务器的安装与配置；
3. 掌握 JNDI 服务的发布；
4. 掌握会话 Bean 的开发步骤。

# （二）基本知识与原理

1. EJB(Enterprise JavaBean)是 sun 的 JavaEE 服务器端组件模型，定义了一个用于开发基于组件的企业多重应用程序的标准；凭借 java 跨平台的优势，用 EJB 技术部署的分布式系统可以不限于特定的平台；
2. JNDI(Java Naming and Directory Interface,Java 命名和目录接口)是一组在 Java 应用中访问命名和目录服务的 API；命名服务将名称和对象联系起来，使得可用名称来访问对象；
3. 会话 Bean(Session Bean)用于执行业务流程的逻辑，属于客户端程序在服务器上的部分逻辑延伸，每个 Session Bean 对象对应于特定的客户端，不能在多个客户端间共享。

# （三）实验内容及步骤

1. 下载**JBoss**：登录JBoss官网 <http://jbossas.jboss.org/downloads> ，选择版本7.1.1

Final ，点击并下载JBoss AS(Application Server)服务器，如图11-1所示：



图 11-1 下载并解压安装 JBoss AS 7.1.1

1. 安装运行**JBoss**：解压下载成功的压缩包，然后打开CMD命令提示符窗口，进入JBoss解压主目录下的子目录bin( 如：**C:\jboss-as-7.1.1.Final\bin** )，运行 **standlone.bat**来启动JBoss服务器( 注意：要预先正确设置**JAVA\_HOME**环境变

量 )，在JBoss服务器启动完毕之后，打开浏览器并输入地址[http://localhost:8080](http://localhost:8080/)，

如出现JBoss欢迎页面(如图11-2所示)，则表示JBoss安装并运行成功；

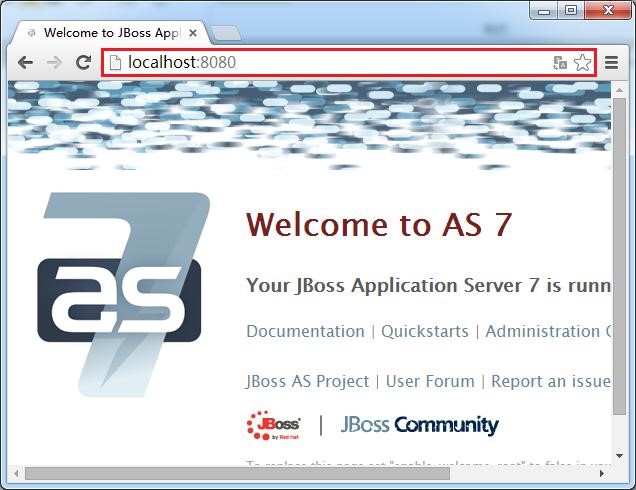


图 11-2 启动并测试访问 JBoss 服务器

1. 查看**Eclipse**的版本号：打开Eclipse，选择菜单[**Help->About Eclipse**]，在弹

出的版本信息窗口中查看Eclipse的版本号，如图11-3所示（不同Eclipse的版本号可能不同，本例中Eclipse的版本号为**Kepler**）：

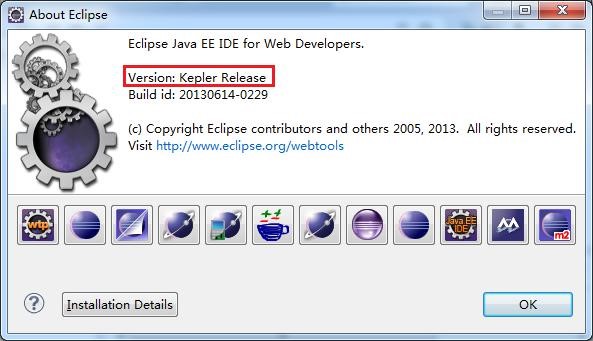


图 11-3 查看 Eclipse 的版本号

1. 在**Eclipse**中安装**JBoss Tools**开发插件：

## 1）打开Eclipse，选择菜单[Help->Eclipse Marketplace]，如图11-4所示：

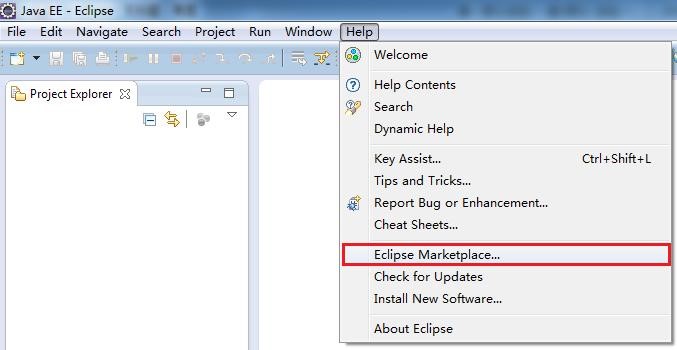


图 11-4 启动并测试访问 JBoss 服务器

1. 在弹出的窗口搜索编辑框中输入“**JBoss Tools**”，然后在结果列表中找

到“**JBoss Tools(Kepler)**”，点击按钮[**Install**]，如图11-5所示( 注意：此操作需要连接到Internet互联网，并且所选择的JBoss Tools的版本号必须和Eclipse的版本

号一致，如本例中二者版本号同为“**Kepler**”)：

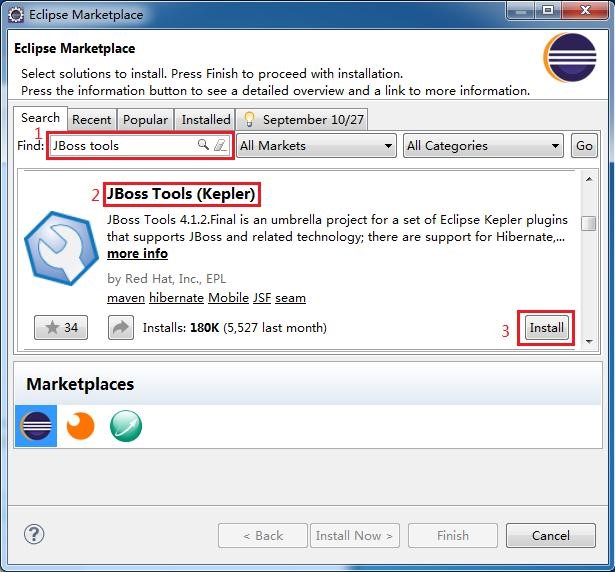


图 11-5 在 Eclipse 中安装 JBoss 开发插件

1. 在弹出的窗口中选择[**JBoss AS Tools**]，点击按钮[**Confirm**]，接着在安装许可协议窗口中选择[**I accept the terms of the license agreement**]，如图11-6所示：

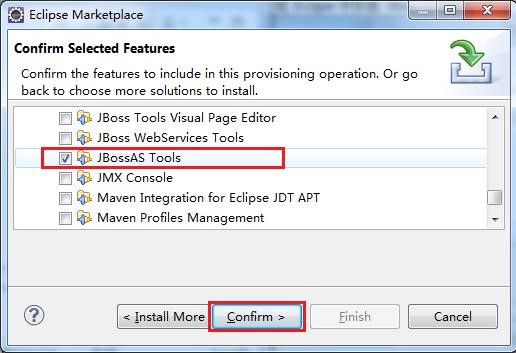


图 11-6 在 Eclipse Marketplace 中选择 JBoss AS 工具

1. 然后点击按钮[**Finish**]等待Eclipse自动下载并安装JBossAS开发插件，如图

11-6和图11-7所示：

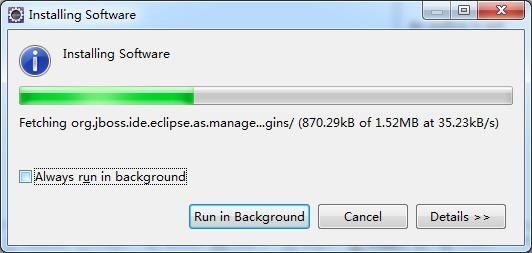


图 11-7 Eclipse 正在下载安装 JBossAS Tools 开发插件

1. 如在安装过程出现Warning警告，请点击按钮[**OK**]忽略警告，安装完毕后，

重启Eclipse以使JBossAS开发插件生效。

1. 在**Eclipse**中配置**JBoss**服务器：
   1. 打开Eclipse，选择菜单[**Windows->Preferences**]，在弹出的窗口左侧导航栏中点击选择[**Server->Runtime Environments**]，然后在右侧点击按钮[**Add**] ，

如图11-8所示：

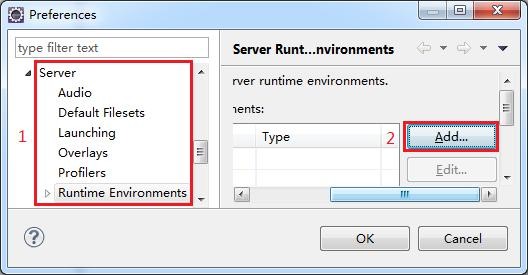


图 11-8 在 Eclipse 首选项窗口中添加新服务器

* 1. 接着在弹出的服务器类型窗口中选择[**JBoss Community->JBoss 7.1**

**Runtim**e]，选中复选框[**Create a new local server**]，最后点击按钮[Next]，如图11-9

所示：

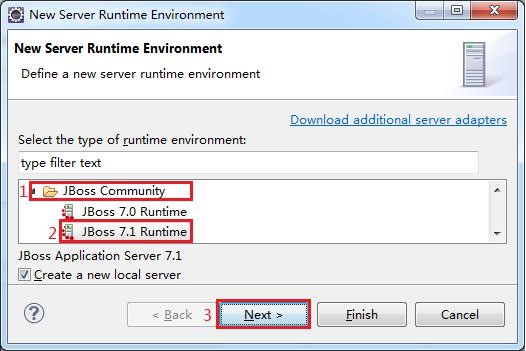


图 11-9 在 Eclipse 配置 JBoss 服务器步骤 1

* 1. 在弹出的JBoss服务器配置窗口中点击按钮[**Browse**]，选择JBoss的安装主目录（如：**C:\jboss-as-7.1.1.Final** ）和**jre**运行环境，点击按钮[**Finish**]和[**OK**]

完成JBoss服务器配置，如图11-10和图11-11所示：

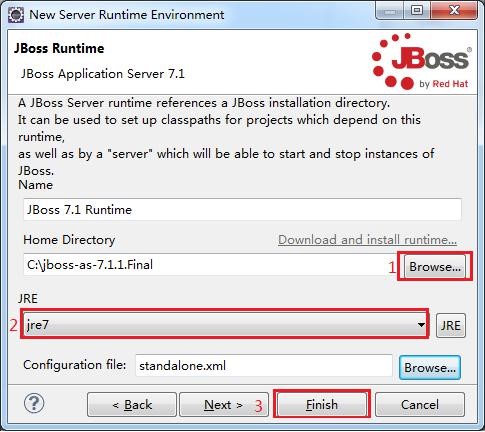


图 11-10 在Eclipse中配置JBoss服务器步骤2

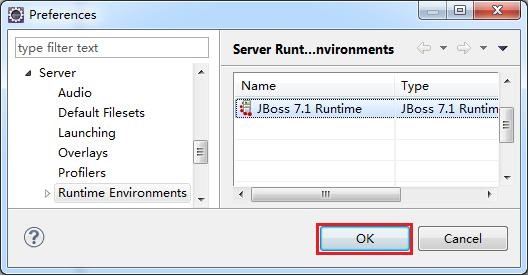


图 11-11 完成 JBoss 服务器配置

1. 编写**EJB**工程（名称为**ejb-project1**）：

## 1）打开Eclipse，选择菜单[File->New->EJB Project]，如图11-12所示：

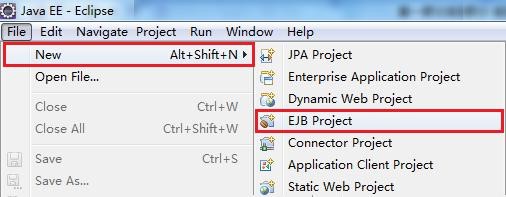


图 11-12 创建 EJB 工程之步骤 1

1. 在弹出的窗口中填写工程名称为**ejb-project1**，运行服务器选择上述步骤所配置的[**JBoss7.1 Runtime]**，EJB模块版本号选择[**3.0**]，然后点击[**Finish**]完成

工程的创建，如图11-13所示：

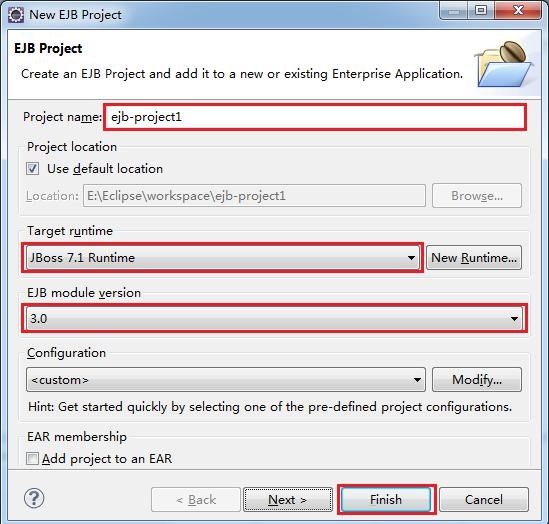


图 11-13 创建 EJB 工程之步骤 2

1. 在工程中添加一个**cn.edu.zjut.ejb**包：右键单击ejb-project1工程，在弹出的菜单中选择[**New->Package**]，填写包名称**cn.edu.zjut.ejb**，点击按钮[**Finish**]；
2. 创建**UserService**会话**Bean**：右键单击**cn.edu.zjut.ejb**包，在弹出的菜单中选择[**New->Session Bean**]；然后在弹出的窗口中输入类名称为[**UserService]**，状态类型为[**Stateless**]，选中复选框[**Remote**]，去除复选框[**Local**]，最后点击按钮

[**Finish**]，如图11-14所示：

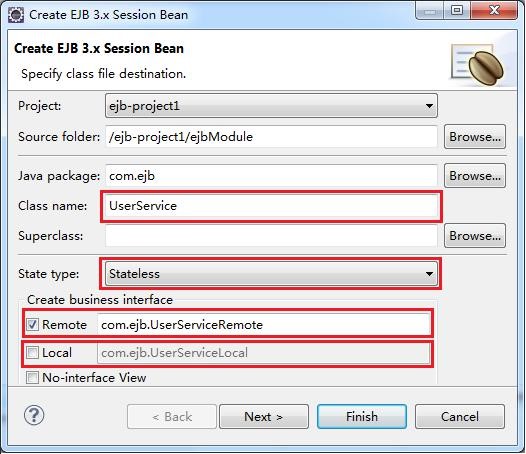


图 11-14 填写会话 Bean 信息

1. 双击打开**UserServiceRemote.java**，在其中添加一个抽象方法**login(String username, String password)**，用于登录验证，完整代码如下：

|  |
| --- |
| package cn.edu.zjut.ejb; import javax.ejb.Remote;  @Remote public interface UserServiceRemote { public boolean login(String username, String password); } |

1. 双击打开**UserService.java**，实现**UserServiceRemote**接口的**login(String username, String password)**方法，完整代码如下：

|  |
| --- |
| **package cn.edu.zjut.ejb; import javax.ejb.Stateless;**  **@Stateless public class UserService implements UserServiceRemote { public UserService() { } public boolean login(String username, String password){**  **if(username.equals("zjut")&&password.equals("zjut")){ return true;** |
| **}else return false;**  **}**  **}** |

1. 将**ejb-project1**工程部署到**JBoss**服务器：右键单击ejb-project1工程，在弹出的菜单中选择[**Export->EJB JAR file**]，在弹出的窗口中点击按钮[**Browse**]，并在目录浏览窗口中选择JBoss主目录下的standalone\deployments子目录为部署地址[**Destination]**( 如：**C:\jboss-as-7.1.1.Final\standalone\deployments** )，如图11-15

所示：

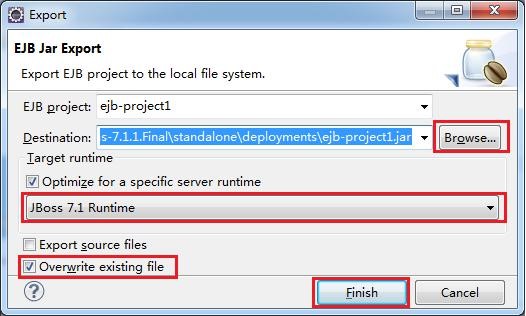


图 11-15 将 ejb-project1 工程部署到 JBoss 服务器

1. 在**Eclipse**中启动**JBoss**服务器：选择Eclipse右下区域的[**Servers**]选项卡，在其中可以看到已经部署的[**JBoss 7.1 Runtime Server**]，点击该选项卡右侧的图标启动JBoss服务器，如图11-16所示：

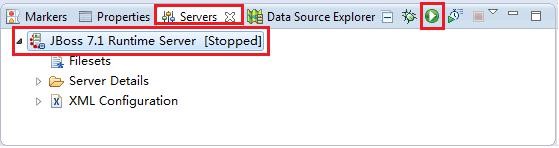


图 11-16 在 Eclipse 中启动 JBoss 服务器

1. 服务器启动完成后，选择Eclipse右下区域的[**Consoles**] 选项卡中查看到 ejb-project1工程已经部署到服务器的信息，在其中可以看到**UserService**服务发布

地址为**module/UserService**，如图11-17所示：

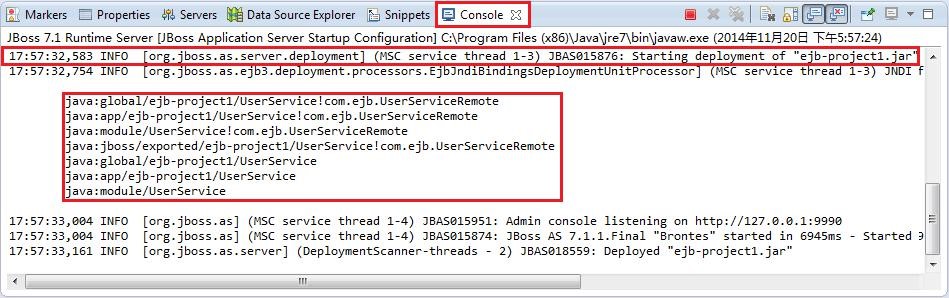


图 11-17 ejb-project1 工程在 JBoss 服务器上部署成功的信息 7、编写客户端工程（名称为**ejb-project1-client**）：

1. 新建一个名称为 **ejb-project1-client** 的[**Java Project**]工程，
2. 在工程的 src 目录下新建一个名称为 **cn.edu.zjut.ejb** 的包，将 EJB 工程 **ejb-project1** 中的接口程序 **UserServiceRemote.java** 拷贝到该包下，并添加类库

**jboss-client.jar** 到本工程的[**Java Build Path-Libraries]**中；

1. 在 src 目录下新建一个名称为 **cn.edu.zjut.ejb.client** 的包，在该包下新建一个名称为 LoginClient.java 的客户端程序，具体代码如下：

|  |
| --- |
| package cn.edu.zjut.ejb.client;  **import javax.naming.Context; import javax.naming.InitialContext; import javax.naming.NamingException;**  import java.security.Security; import java.util.Hashtable; import cn.edu.zjut.ejb.UserServiceRemote;  **public class LoginClient** { private static UserServiceRemote lookupRemoteStatelessEjbBean()  throws NamingException { final Hashtable jndiProperties = new Hashtable();  jndiProperties.put(Context.URL\_PKG\_PREFIXES,  "org.jboss.ejb.client.naming"); final Context context = new InitialContext(jndiProperties); final String appName = ""; final String moduleName = "ejb-project1"; final String distinctName = ""; **final String beanName = "UserService";** finalStringviewClassName=UserServiceRemote.class.getName(); final String namespace = "ejb:" + appName + "/" + moduleName  + "/" + distinctName + "/" + beanName + "!" + viewClassName; return (UserServiceRemote) context.lookup(namespace); } |
| **public static void main(String[] args) {** // TODO Auto-generated method stub try{  **UserServiceRemote usBean = lookupRemoteStatelessEjbBean();** System.out.println(usBean);  boolean b1 = usBean.login("zjut","zjut"); System.out.println(b1);  }catch(NamingException e){  e.printStackTrace();  }  **}**  } |

1. 在src目录下新建jboss-ejb-client.properties文件，具体代码如下：

endpoint.name=client-endpoint

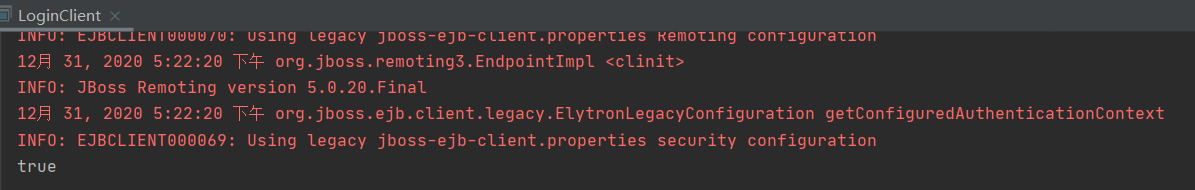
remote.connectionprovider.create.options.org.xnio.Options.SSL\_ENABL ED=false remote.connections=default remote.connection.default.host=127.0.0.1 remote.connection.default.port = 4447

remote.connection.default.connect.options.org.xnio.Options.SASL\_POL

ICY\_NOANONYMOUS=false

1. 启动JBoss7.1.1Final服务器，然后运行LoginClient.java，观察输出结果；
2. 修改登录用户名和密码，观察输出结果；
3. 修改ejb-project1工程的用户登录逻辑，重新部署，运行LoginClient.java，

观察输出结果。



# （四）实验要求

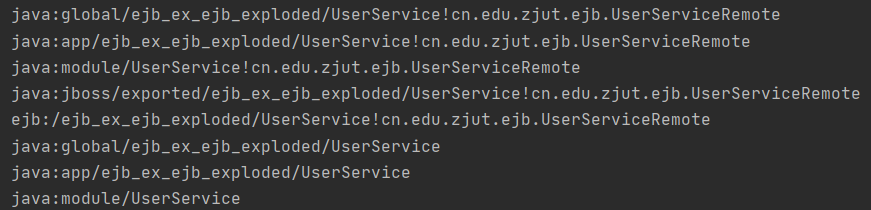
1、填写并上交实验报告，报告中应包括：

1. 运行结果截图；
2. 修改后的关键代码，及相应的运行结果或报错信息；
3. 根据实验过程，总结客户端程序调用EJB的过程；

 设置JNDI服务工厂以及JNDI服务地址系统属性，查找Home接口，从Home接口调用Create方法创建Remote接口，通过Remote接口调用其业务方法。

1. 碰到的问题及思考；
2. 实验收获及总结。
3. 上交程序源代码，代码中应有相关注释。

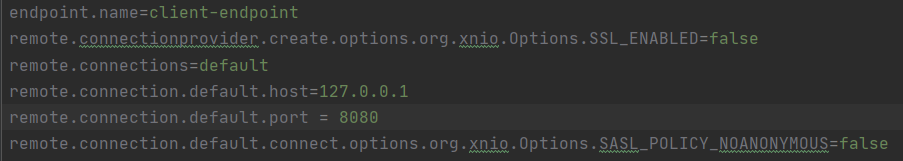
日志信息



需与



对应上



端口号改成8080

二、提高实验——有状态会话**Bean**的调用

# （一）实验目的

1. 掌握有状态会话 Bean 的编写与调用；
2. 掌握在 Web 工程中调用有状态会话和无状态会话 Bean。

# （二）基本知识与原理

1. 有状态会话 Bean 和无状态会话 Bean 都实现了 javax.ejb.SessionBean 接口，有状态会话 Bean 可以在多次访问之间保存特定客户的信息，而无状态会话 Bean 不会在客户多次访问之间保存信息；
2. 无状态会话 Bean 在对象实例池中创建和维护，提供给众多用户共同使用；有状态会话 Bean 在一个生命周期内只服务于一个用户。

# （三）实验内容及步骤

1. 编写**Web**客户端工程（名称为**ejb-project1-web**）：
   * 1. 新建一个名称为 **ejb-project1-web** 的[**Dynamic Web Project**]工程；
     2. 在 **src** 目录下新建一个名称为 **cn.edu.zjut.ejb** 的包，将 EJB 工程 **ejb-project1** 中的接口程序 **UserServiceRemote.java** 拷贝到该包下；
     3. 在 **src** 目录下新建 jboss-ejb-client.properties 文件（代码略），并添加类库 **jboss-client.jar** 到本工程的[**Java Build Path-Libraries]**中；
     4. 在 WebContent 目录下新建一个名称为 **webclient.jsp** 的 JSP 页面，并在其中的 WEB-INF 目录下新建 web.xml 文件，整个工程目录如图 11-18 所示：

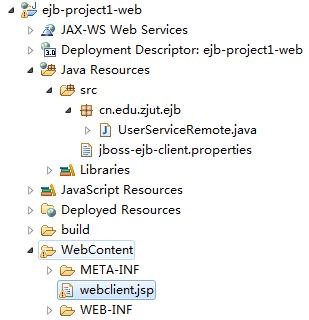


图 11-18 ejb-project1-web 工程结构

* + 1. Webclient.jsp 页面的具体代码如下：

|  |
| --- |
| <%@ page language="java" import="java.util.\*" pageEncoding="UTF-8"%>  <%@ page import="javax.naming.\*, java.util.Properties"%>  <%@ page import="cn.edu.zjut.ejb.\*"%>  <% try{ final Hashtable jndiProperties = new Hashtable();  jndiProperties.put(Context.URL\_PKG\_PREFIXES,  "org.jboss.ejb.client.naming"); final Context context = new InitialContext(jndiProperties); final String appName = ""; final String moduleName = "ejb-project1"; final String distinctName = ""; final String beanName = "UserService"; finalStringviewClassName=UserServiceRemote.class.getName(); final String namespace = "ejb:" + appName + "/" + moduleName  + "/" + distinctName + "/" + beanName + "!" + viewClassName;  UserServiceRemote usBean =(UserServiceRemote) context.lookup(namespace);  System.out.println(usBean); if(usBean.login("zjut","zjut")) out.println("login ok!");  else out.println("login failed!");  }catch(NamingException e){  e.printStackTrace();  }  %> |

* + 1. 将工程部署到 JBoss 服务器上，然后打开浏览器输入网址： <http://localhost:8080/ejb-project1-web/webclient.jsp>，查看运行结果；
    2. 改变**ejb-project1**工程中的用户登录逻辑，然后重新部署到JBoss服务器

上，再次观察webclient.jsp的运行结果。



1. 编写有状态会话**Bean**：
   1. 修改 **ejb-project1** 工程，在 **cn.edu.zjut.ejb** 包中新建一个名称为

**ProductCartBean** 的有状态会话 Bean，选择[**State Type**]为[**Stateful**]，如图 11-18

所示：

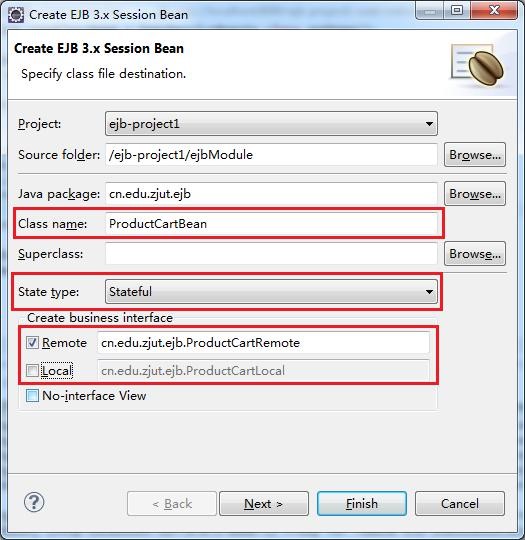


图 11-18 编写有状态会话 Bean

* 1. 编写 **ProductCartRemote.java** 代码如下：

|  |
| --- |
| package cn.edu.zjut.ejb; import java.util.ArrayList; import javax.ejb.Remote;  @Remote public interface ProductCartRemote { public void **addProduct**(String productName, int price); public ArrayList<String> **listProducts**(); public int **totalPrice**();  } |

* 1. 编写 **ProductCartBean.java** 代码如下：

|  |
| --- |
| package cn.edu.zjut.ejb; import java.util.ArrayList; import javax.ejb.Stateful;  @Stateful  @Remote(ProductCartRemote.class) public class ProductCartBean implements ProductCartRemote { public ProductCartBean() {  // TODO Auto-generated constructor stub  } private ArrayList<String> cartList = new ArrayList<String>(); private int totalPrice=0; public ArrayList **listProducts**(){ |
| return this.cartList;  } public void **addProduct**(String name, int price){ this.cartList.add(name); totalPrice+=price;  } public int **totalPrice**(){return totalPrice;}  } |

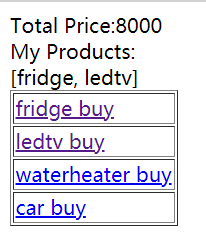
* 1. 将新修改的有状态会话 Bean 部署到 JBoss 服务器。

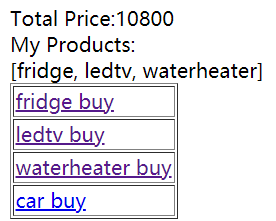
1. 修改**ejb-project1-web**工程：
   * 1. 将 EJB 工程 **ejb-project1** 中的接口程序 **ProductCartRemote.java** 拷贝到该包下。
     2. 在 WebContent 目录下新建一个名称为 **myCart.jsp** 的 JSP 页面，具体代码如下：

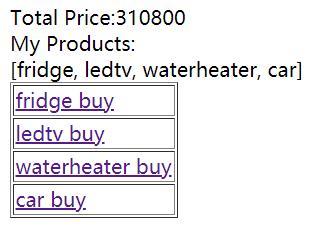
|  |
| --- |
| <%@ page language="java" import="java.util.\*" pageEncoding="UTF-8"%>  <%@ page import="javax.naming.\*, java.util.Properties"%>  <%@ page import="cn.edu.zjut.ejb.\*"%>  <% try{ final Hashtable jndiProperties = new Hashtable();  jndiProperties.put(Context.URL\_PKG\_PREFIXES,  "org.jboss.ejb.client.naming"); final Context context = new InitialContext(jndiProperties); final String appName = ""; final String moduleName = "ejb-project1"; final String distinctName = ""; final String beanName = "ProductCartBean"; finalStringviewClassName=ProductCartRemote.class.getName(); final String namespace = "ejb:" + appName + "/" + moduleName  + "/" + distinctName + "/" + beanName + "!" + viewClassName + **"?stateful"**;  ProductCartRemote cart = null; cart=(ProductCartRemote)session.getAttribute("cart"); if(cart == null){ cart = (ProductCartRemote) context.lookup(namespace);  session.setAttribute("cart",cart);  }else{  String productName=request.getParameter("product"); String sPrice=request.getParameter("price"); int price=0; if(sPrice!=null) price=Integer.parseInt(sPrice); cart.addProduct(productName, price); |
| List<String> myProducts = cart.listProducts(); out.println("Total Price:"+cart.totalPrice()+"<br>"); out.println("My Products:<br>"+myProducts); }  }catch(NamingException e){  e.printStackTrace();  }  %>  <table border=1>  <tr><td><a href="myCart.jsp?product=fridge&price=3000">fridge buy</a></td></tr>  <tr><td><a href="myCart.jsp?product=ledtv&price=5000">ledtv buy</a></td></tr>  <tr><td><a href="myCart.jsp?product=waterheater&price=2800">waterheater buy</a></td></tr>  <tr><td><a href="myCart.jsp?product=car&price=300000">car buy</a></td></tr>  </table> |

* + 1. 将工程部署到 JBoss 服务器上，然后打开浏览器输入网址： [http://localhost:8080/ejb-project1-web/myCart.jsp，](http://localhost:8080/ejb-project1-web/myCart.jsp)点击页面中的按钮“购买”并查看运行结果。









# （四）实验要求

1、填写并上交实验报告，报告中应包括：

1. 实验基本思路；

通过Web利用jsp脚本远程访问productcartnean的方法

1. 实验关键代码，及相应的运行结果及截图，或相应的报错信息；
2. 碰到的问题及思考；
3. 实验收获及总结。
4. 上交程序源代码，代码中应有相关注释。

三、扩展实验——控制会话**Bean**的生命周期

# （一）实验目的

1. 掌握会话 Bean 生命周期的概念；
2. 掌握 EJB 的关于生命周期的 6 个注解符：@postConstruct，@PreDestroy， @PrePassivate，@PostActivate，@Init 和@Remove；

# （二）基本知识与原理

1. 会话 Bean 实例的生命周期一般较短，包括 Bean 对象的创建、初始化、运行、钝化、激活、销毁；
2. 无状态会话 Bean 对应生命周期管理的注解符包括@postConstruct 和

@PreDestroy；

1. 有状态会话 Bean 对应生命周期管理的注解符包括@postConstruct，

@PreDestroy，@PrePassive，@PostActivate，@Init 和@Remove；

1. 各注解符含义如下：

|  |  |
| --- | --- |
| @postConstruct | 当 bean 对象完实例化后，标注了这个注解符的方法会立即被调用，每个 bean 类只能定义一个@PostConstruct 方法。 |
| @PreDestroy | 标注了这个注解符的方法会在容器销毁一个无用或者过期的 bean 实例之前被调用。 |
| @ PrePassivate | 当一个有状态的 bean 实例空闲时间过长，就会发生钝化(passivate)。标注了该注解符的方法会在被钝化之前被调用，bean 实例被钝化后，在一段时间内，如果仍然没有用户对 bean 实例进行操作，容器将会从硬盘中删除它，这个注释符用于有状态会话 bean。 |
| @PostActivate | 当客户端再次使用已经被钝化的有状态 bean 时，EJB 容器会重新实例化一个 Bean 实例，并从硬盘中将之前的状态恢复。标注了这个注解符的方法会在激活完成时被调用。这个注释符只适合于有状态会话 bean。 |
| @Init | 这个注释符指定了又状态 bean 初始化的方法，他区别于 @PostConstruct 注释在于：多个@Init 注释方法可以同时存在于有状态 session bean 中，但每个 bean 实例只会有一个@Init 注释方法被调用@PostConstruct 在@Init 之后调用 |
| @Remove | 当客户端调用标注了@Remove 注解符的方法时，容器将在方法执行结束后把 bean 实例删除。 |

# （三）实验内容及步骤

1. 修改**ejb-project1**工程，新建一个名称为**cn.edu.zjut.ejb.lifecycle**的包；
2. 在**cn.edu.zjut.ejb.lifecycle**包下新建一个名称为LifeCycle的有状态会话Bean，如图11-20所示。

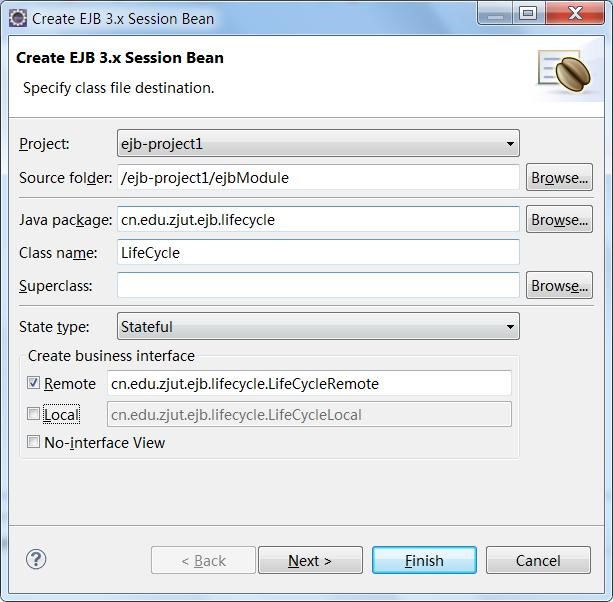


图11-20 新建一个有状态会话Bean

1. 为LifeCycle类添加一个接口方法hello及6个注释符的方法，具体代码如下：

|  |
| --- |
| **//LifeCycleRemote.java** package cn.edu.zjut.ejb.lifecycle; import javax.ejb.Remote;  @Remote public interface LifeCycleRemote { public String hello(String name); public void removeSession();  } |

**//LifeCycle.java** package cn.edu.zjut.ejb.lifecycle;

|  |
| --- |
| import javax.annotation.PostConstruct; import javax.annotation.PreDestroy; import javax.ejb.Init; import javax.ejb.PostActivate; import javax.ejb.PrePassivate; import javax.ejb.Remove; import javax.ejb.Stateful;  @Stateful public class LifeCycle implements LifeCycleRemote { public LifeCycle() {  // TODO Auto-generated constructor stub  } public String hello(String name){  try {  Thread.sleep(1000\*3);  } catch(InterruptedException e) {  e.printStackTrace(); } return "hello"+name+"!";  }  @Init  public void initialize() {  System.out.println("@Init事件触发");  }  @PostConstruct public void construct() {  System.out.println("@PostConstruct事件触发"); }  @PreDestroy public void exit() {  System.out.println("@PreDestroy事件触发");  }  @PostActivate public void activate() {  System.out.println("@PostActivate事件触发");  }  @PrePassivate public void prePassivate() { |
| System.out.println("@prePassivate事件触发");  }  @Remove  public void removeSession() {  System.out.println("@remove事件触发"); }  } |

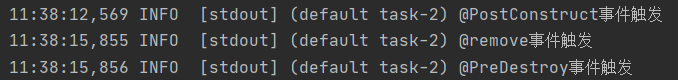
1. 将**ejb-project1**工程部署到JBoss服务器。
2. 修改客户端工程**ejb-project1-client**：
   1. 在工程中新建一个名称为**cn.edu.zjut.ejb.lifecycle**的包；
   2. 将EJB工程**ejb-project1**中的接口程序**LifeCycle.java**拷贝到本工程的 **cn.edu.zjut.ejb.lifecycle**包下；
   3. 在**cn.edu.zjut.ejb.client**包下新建一个名称为LifeCycleClient.java的客户端

程序，具体代码如下：

|  |
| --- |
| package cn.edu.zjut.ejb.client;  import javax.naming.Context; import javax.naming.InitialContext; import javax.naming.NamingException;  import java.security.Security; import java.util.Hashtable; import cn.edu.zjut.ejb.lifecycle.\*;  public class LifeCycleClient { private static LifeCycleRemote lookupRemoteStatefulEjbBean()  throws NamingException {  final Hashtable jndiProperties = new Hashtable();  jndiProperties.put(Context.URL\_PKG\_PREFIXES,  "org.jboss.ejb.client.naming");  final Context context = new InitialContext(jndiProperties); final String appName = ""; final String moduleName = "ejb-project1"; final String distinctName = ""; final String beanName = LifeCycle; |
| final String viewClassName = LifeCycleRemote.class.getName();  final String namespace = "ejb:" + appName + "/" + moduleName  + "/" + distinctName + "/" + beanName + "!" + viewClassName+  "?stateful";  System.out.println("namespace="+namespace); return (LifeCycleRemote) context.lookup(namespace);  }  public static void main(String[] args) { // TODO Auto-generated method stub try{  LifeCycleRemote lc = lookupRemoteStatefulEjbBean(); lc.hello("zjut"); lc.removeSession();  }catch(NamingException e){  e.printStackTrace();  }  }  } |

1. 运行LifeCycleClient.java程序，查看结果。

服务端



客户端将返回结果打印出来





# （四）实验要求

1、填写并上交实验报告，报告中应包括：

1. 实验基本思路；
2. 实验关键代码，及相应的运行结果及截图，或相应的报错信息；
3. 碰到的问题及思考；
4. 实验收获及总结。

2、上交程序源代码，代码中应有相关注释。