

# 实验七 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WildFly**：登录WildFly官网 <https://www.wildfly.org/downloads/>，选择版本点击并下载WildFly服务器，如图7-1所示：

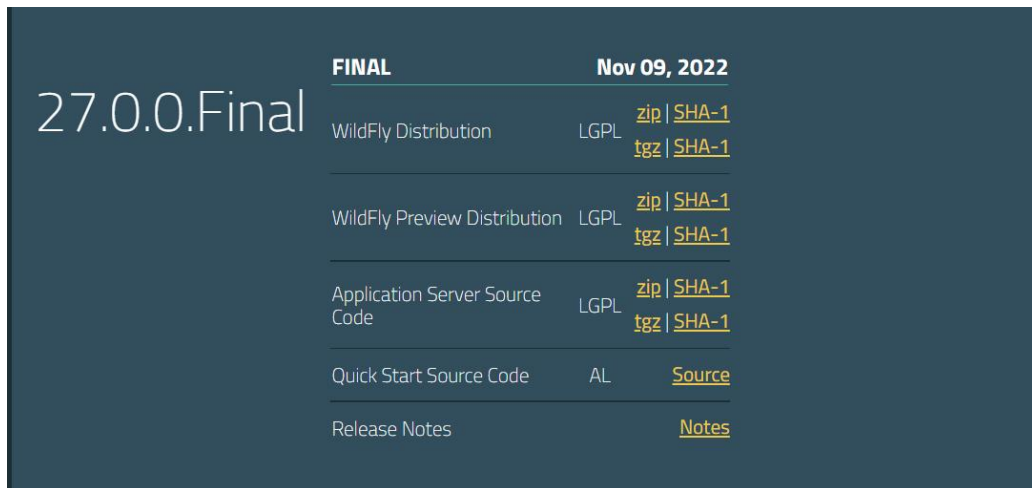


图 7-1 下载并解压安装 WildFly

2、**安装运行WildFly:** 解压下载成功的压缩包，然后打开CMD命令提示符窗口，进入解压主目录下的子目录bin，运行**standlone.bat**来启动服务器( **注意：要预先正确设置JAVA\_HOME环境变量** )，在服务器启动完毕之后，打开浏览器并输入地址<http://localhost:8080>，如出现欢迎页面(如图7-2所示)，则表示WildFly安装并运行成功；



图 7-2 启动并测试访问 WildFly 服务器

3、在Eclipse中安装JBoss Tools开发插件：

1) 打开Eclipse，选择菜单[Help->Eclipse Marketplace]，如图7-3所示：

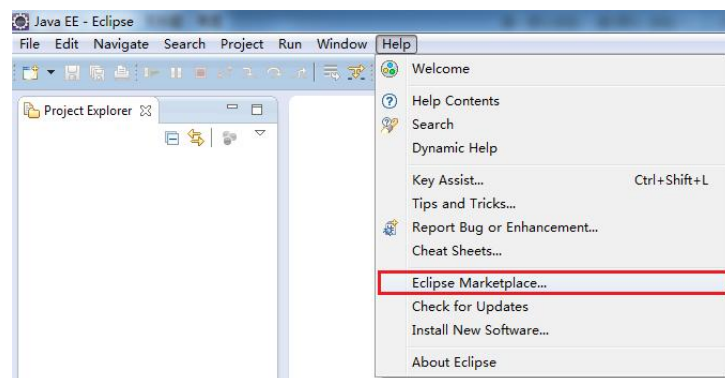


图 7-3 搜索 JBoss 开发插件

2) 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”，点击按钮[Install]，如图7-4所示：

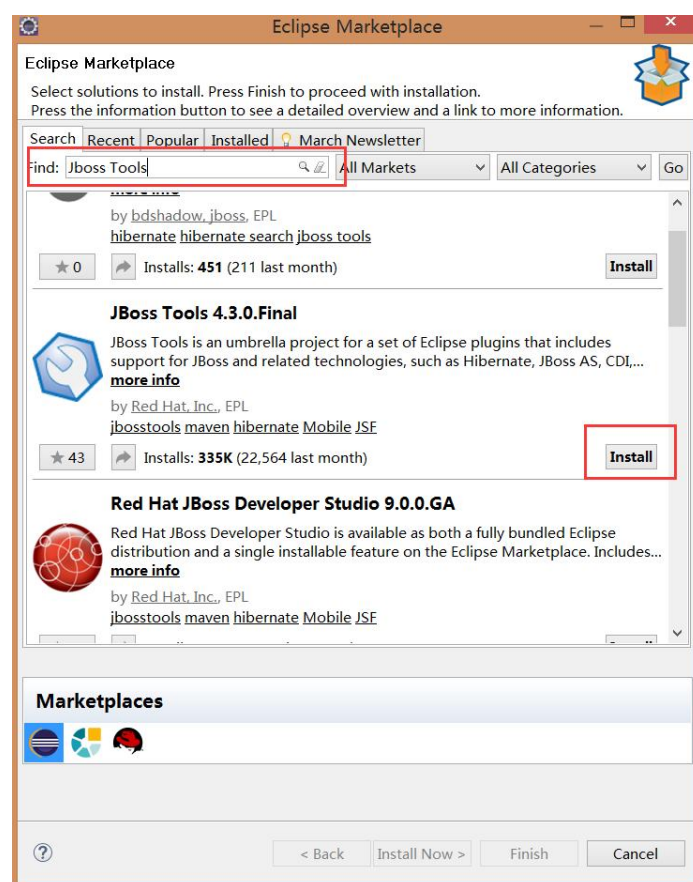


图 7-4 在 Eclipse 中安装 JBoss 开发插件

3) 在弹出的窗口中选择[JBoss AS, WildFly & EAP Server 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 agreements]，如图7-5所示：

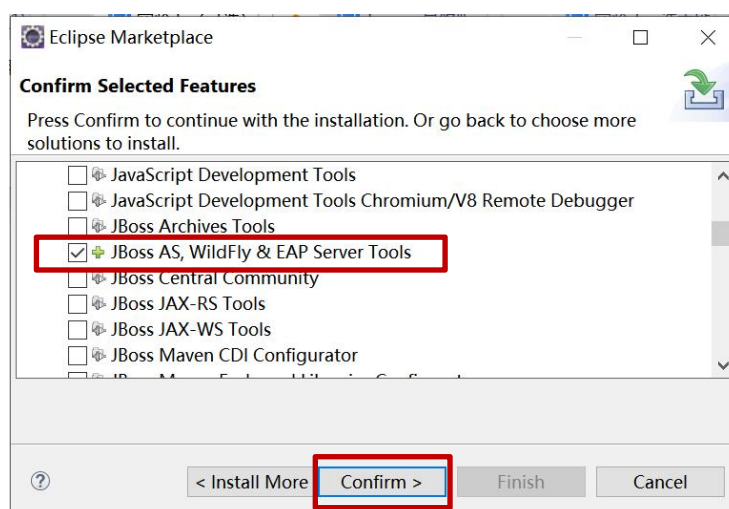


图 7-5 在 Eclipse Marketplace 中选择 JBoss AS, WildFly & EAP Server 工具

4) 如在安装过程出现Warning警告, 请点击按钮[install anyway]忽略警告, 安装完毕后, 重启Eclipse以使JBossAS开发插件生效。

#### 4、在Eclipse中配置WildFly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7-6所示:

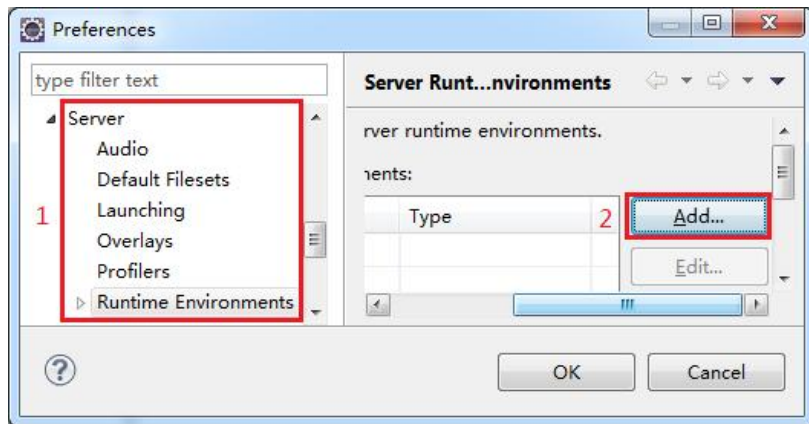


图 7-6 在 Eclipse 首选项窗口中添加新服务器

2) 接着在弹出的服务器类型窗口中选择相应版本的WildFly Application Server, 选中复选框[Create a new local server], 最后点击按钮[Next];

3) 在弹出的JBoss服务器配置窗口中点击按钮[Browse], 选择JBoss的安装主目录和jre运行环境, 点击按钮[Finish]和[OK]完成WildFly服务器配置;

#### 5、编写EJB工程（名称为ejb-project1）:

1) 打开Eclipse, 选择菜单[File->New->EJB Project], 如图7-7所示:

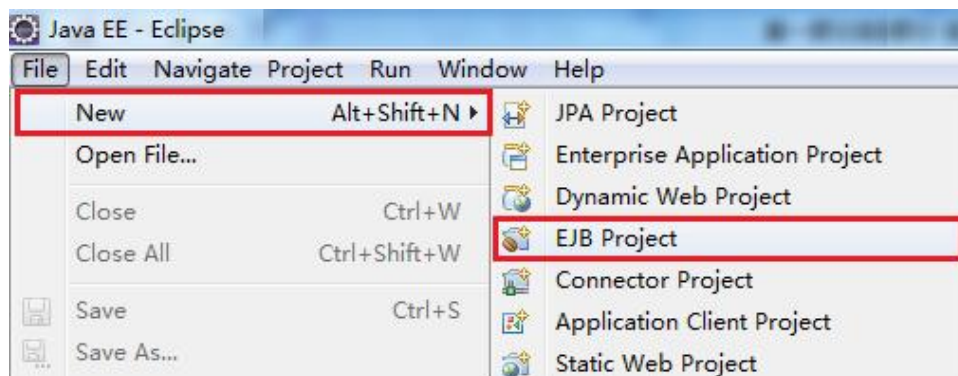


图 7-7 创建 EJB 工程之步骤 1

2) 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WildFly, EJB模块版本号选择[3.0], 然后点击[Finish]完成工程的创建;

3) 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]；

4) 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7-8所示：

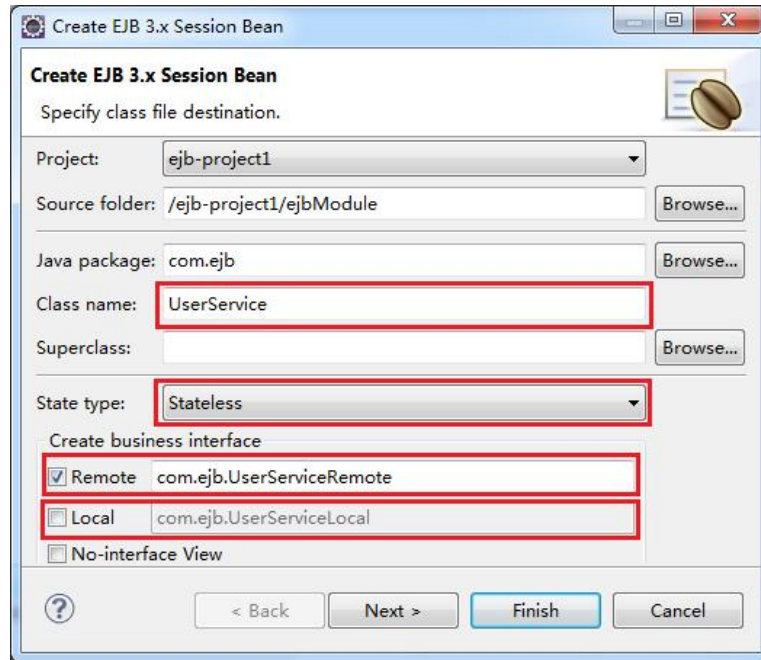


图 7-8 填写会话 Bean 信息

5) 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);
}
```

6) 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;
    }
}
}

```

7) 将**ejb-project1**工程部署到**WildFly**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WildFly主目录下的standalone\deployments子目录为部署地址[**Destination**];

8) 在Eclipse中启动**WildFly**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[**WildFly \*.\* Runtime Server**], 点击该选项卡右侧的图标启动服务器;

9) 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**;

## 6、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** 导出成 **jar** 包并本工程的[**Java Build Path-Libraries**]中, 并添加类库 **jboss-client.jar** (WildFly 主目录下的 **bin\client**) 到本工程的[**Java Build Path-Libraries**]中;

3) 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 beanName = "UserService";
    final String viewClassName = UserServiceRemote.class.getName();
    final String namespace = "ejb:" + appName + "/" + moduleName
        + "/" + 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();
    }
}
}

```

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

```

remote.connectionprovider.create.options.org.xnio.Options.SSL_ENABL
ED=false
remote.connections=default
remote.connection.default.host=localhost
remote.connection.default.port = 8080
remote.connection.default.connect.options.org.xnio.Options.SASL_POL
ICY_NOANONYMOUS=false

```

5) 启动服务器，然后运行LoginClient.java，观察输出结果；

6) 修改登录用户名和密码，观察输出结果；

7) 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；
- (4) 碰到的问题及思考；
- (5) 实验收获及总结。

2、上交程序源代码，代码中应有相关注释。

## 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** 导出成 **jar** 包并添加到本工程的[Java Build Path-Libraries]中；
- 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 7-9 所示：



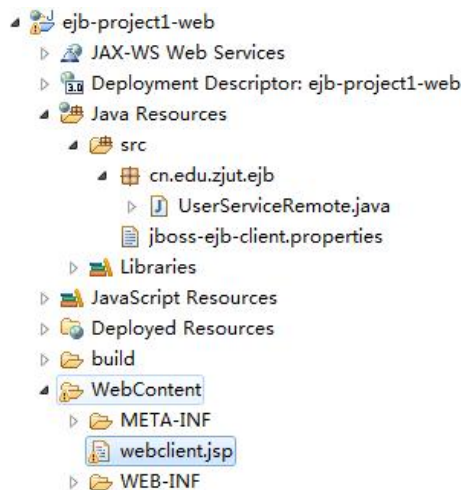


图 7-9 ejb-project1-web 工程结构

5) 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 beanName = "UserService";
        final String viewClassName = UserServiceRemote.class.getName();
        final String namespace = "ejb:" + appName + "/" + moduleName
            + "/" + 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();
    }
%>
```

6) 将工程部署到 WildFly 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；

7) 改变**ejb-project1**工程中的用户登录逻辑，然后重新部署到服务器上，再次观察webclient.jsp的运行结果。

## 2、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 7-10 所示：

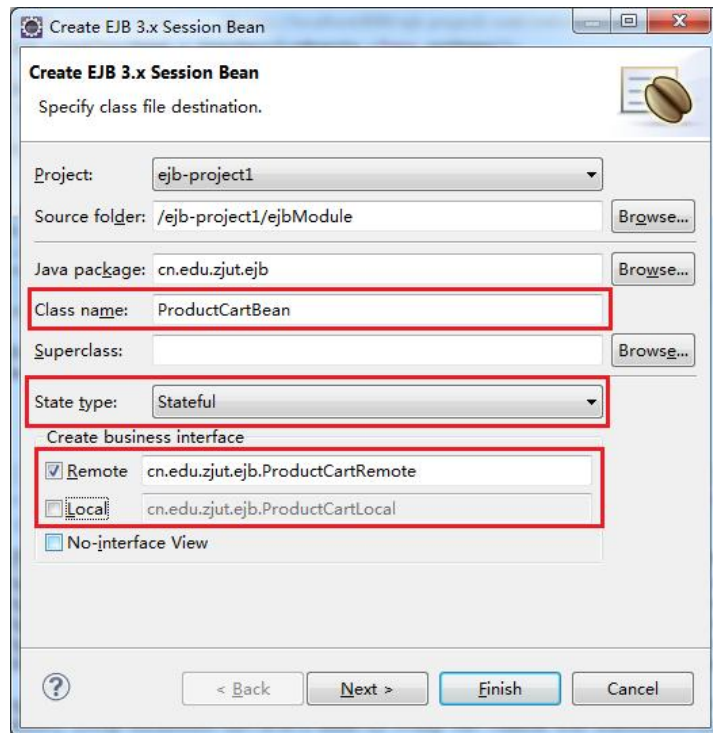


图 7-10 编写有状态会话 Bean

2) 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();
}
```

3) 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;}
}

```

4) 将新修改的有状态会话 Bean 部署到服务器。

### 3、修改ejb-project1-web工程:

1) 将 EJB 工程 **ejb-project1** 中的接口程序 **ProductCartRemote.java** 导出成 **jar 包**并添加到本工程的[**Java Build Path-Libraries**]中;

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 beanName = "ProductCartBean";
        final String viewClassName = ProductCartRemote.class.getName();
        final String namespace = "ejb:" + appName + "/" + moduleName
            + "/" + 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>

```

3) 将工程部署到 WildFly 服务器上，然后打开浏览器输入网址：<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；
- (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。

### 三、扩展实验——实体 Bean 的开发

#### （一）实验目的

- 1、掌握实体 Bean 的概念和 JPA 规范。
- 2、掌握实体 Bean 的开发步骤。
- 3、掌握通过实体管理器来执行数据库更新的方法。
- 4、掌握实体 Bean 的监听和回调。
- 5、掌握使用 JPQL 查询语言来执行数据库实体查询。

#### （二）基本知识与原理

- 1、Entity Bean（实体 Bean）是持久数据组件，代表存储在外部介质中的持久（Persistence）对象或者已有的企业应用系统资源。一个 Entity Bean 可以代表数据库中的一行记录，多个客户端应用能够以共享方式访问表示该数据库记录的 Entity Bean。
- 2、JPA（Java Persistence API，Java 持久化接口）是指通过注解或 XML 来描述对象到关系表的映射，并将运行期的实体对象持久化到数据库中。通过使用 JPA，开发人员不再局限于私有供应商提供的特有 API，除非该功能是供应商特有。
- 3、JPQL（Java Persistence Query Language，Java 持久化查询语言）是一种可移植的查询语言，旨在以面向对象表达式语言的表达式，将 SQL 语法和简单查询语义绑定在一起，使用这种语言编写的查询是可移植的，可以被编译成所有主流数据库服务器上的 SQL。JPQL 是完全面向对象的，具备继承、多态和关联等特性。

#### （三）实验内容及步骤

##### 1、下载并安装MySQL:

- 1) 输入<http://dev.mysql.com/downloads/mysql/5.5.html#downloads>，选择适合自己系统的MySQL版本并点击按钮[Download]。
- 2) 在弹出的页面输入账号和密码（如果没有可以立即注册一个），继续完成下载并安装。
- 3) 安装MySQL完毕后，按照系统会提示完成实例配置（注意记住为root用

户设置的密码)。

4) 为了便于操作MySQL数据库, 建议安装配套的MySQL workbench版本。

## 2、创建数据库EIS:

1) 使用root用户在mysql中创建一个名称为**EIS**的数据库;

2) 在EIS数据库中创建一个名称为**userlist**的表, 具体的创建语表句如下:

字段名称	类型	中文名
userid	Int	用户ID, 自动增长
username	Varchar(50)	用户名
userpwd	Varchar(50)	密码
age	Int	年龄

```
CREATE TABLE `userlist` (  
  `userid` int(11) AUTO_INCREMENT,  
  `age` int(11) DEFAULT NULL,  
  `username` varchar(50) DEFAULT NULL,  
  `userpwd` varchar(50) DEFAULT NULL,  
  PRIMARY KEY (`userid`)  
) ENGINE=InnoDB DEFAULT CHARSET=gb2312;
```

3) 为**userlist**表中添加10条数据。

## 3、在WildFly中配置数据源:

1) 在[%JBOSS\_HOME%\modules\com]目录下创建mysql\main目录, 将

Mysql驱动程序拷贝到此目录下, 并创建**module.xml**文件, 具体内容如下:

```
<?xml version="1.0" encoding="UTF-8"?>  
<module xmlns="urn:jboss:module:1.1" name="com.mysql">  
  <resources>  
    <resource-root path="mysql-connector-java-5.1.26-bin.jar"/>  
  </resources>  
  <dependencies>  
    <module name="javax.api"/>  
    <module name="javax.transaction.api"/>  
    <module name="javax.servlet.api" optional="true"/>  
  </dependencies>  
</module>
```

2) 修改[%JBOSS\_HOME%\standalone\configuration]目录下的XML配置

文件**standalone.xml**中的<datasources>元素, 修改完毕后的元素内容如下

(加粗字体为新增内容):

```
<datasources>
    <datasource
jndi-name="java:jboss/datasources/ExampleDS" pool-name="ExampleDS"
enabled="true" use-java-context="true">
<connection-url>jdbc:h2:mem:test;DB_CLOSE_DELAY=-1</connection-url>
    <driver>h2</driver>
    <security>
        <user-name>sa</user-name>
        <password>sa</password>
    </security>
</datasource>

    <datasource jndi-name="java:/MySQLDS"
pool-name="MySQLDS" enabled="true" use-java-context="true">
<connection-url>jdbc:mysql://localhost:3306/eis</connection-url>
    <driver>mysql</driver>
    <pool>
        <min-pool-size>20</min-pool-size>
        <max-pool-size>20</max-pool-size>
        <prefill>true</prefill>
    </pool>
    <security>
        <user-name>root</user-name>
        <password>root</password>
    </security>
</datasource>
<drivers>
    <driver name="h2" module="com.h2database.h2">

<xa-datasource-class>org.h2.jdbcx.JdbcDataSource</xa-datasource-class>

    </driver>
    <driver name="mysql" module="com.mysql">

<driver-class>com.mysql.jdbc.Driver</driver-class>

<xa-datasource-class>com.mysql.jdbc.jdbc2.optional.MysqlXADataSource</xa-datasource-class>
    </driver>
</drivers>
</datasources>
```

- 3) 进入[%JBOSS\_HOME%\bin]目录运行**add-user.bat**文件, 新建一个用户名为**appuser**, 密码为**apppwd**的普通用户 (Application User), 及一个



用户名为**admin**，密码为**manager**的管理用户（**Management User**）。

- 4) 运行[%JBOSS\_HOME%\bin] 目下的**standalone.bat**启动服务器。
- 5) 打开浏览器，输入网址: <http://127.0.0.1:9990/console/App.html#ds-metrics>，在弹出的登录窗口中输入用户名: **admin**，密码: **manager**，进入后在左侧导航栏查看菜单[**Subsystems->Metrics->Datasources**]，看到如图7-11所示的**MySQLDS**，则表示数据源配置成功；或者在控制台看到如下输出信息也表示数据源配置成功：

```
08:55:07,193 INFO [org.jboss.as.connector.subsystems.datasources]
(MSC service thread 1-1) JBAS010400: Bound data source [java:/MySQLDS]
```

图7-11 在管理后台查看数据源

#### 4、创建EJB工程：

- 1) 新建一个名称为**ejb-project2**的[EJB Project]工程，在其[Java Build Path->Libraries] 添加一个 External JAR 库文件 **hibernate-entitymanager-\*.\*.Final.jar**，该文件所在位置为 [%JBOSS\_HOME%\modules\org\hibernate\main]。
- 2) 在**ejbModule\META-INF**下创建一个名称为**persistence.xml**的持久化配置文件，具体内容如下：

```
<?xml version="1.0" encoding="UTF-8" ?>
<persistence xmlns="http://java.sun.com/xml/ns/persistence"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://java.sun.com/xml/ns/persistence
    http://java.sun.com/xml/ns/persistence/persistence_1_0.xsd"
  version="1.0">
  <persistence-unit name="userpu">
    <provider>org.hibernate.ejb.HibernatePersistence</provider>
    <jta-data-source>java:/MySQLDS</jta-data-source>
    <class>cn.edu.zjut.ejb.User</class>
    <properties>
      <property name="hibernate.connection.autocommit"
value="false" />
      <property name="hibernate.dialect"
value="org.hibernate.dialect.MySQL5Dialect" />
      <property name="hibernate.show_sql" value="true" />
      <property name="hibernate.format_sql" value="true" />
      <property name="hibernate.hbm2ddl.auto" value="update" />
    </properties>
```

```
</persistence-unit>
</persistence>
```

3) 在 **ejbModule** 下创建 **cn.edu.zjut.ejb** 包，并在其中新建一个名称为 **User.java** 的实体Bean，具体内容如下：

```
package cn.edu.zjut.ejb;

import java.io.Serializable;
import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
import javax.persistence.Table;

@Entity
@Table(name="userlist")

public class User implements Serializable {
    @Id
    @GeneratedValue(strategy=GenerationType.AUTO)
    private Integer userid;

    @Column(name="username", length=50)
    private String username;

    @Column(name="userpwd", length=50)
    private String userpwd;

    @Column(name="age")
    private Integer age;

    public Integer getUserid() {
        return userid;
    }
    public void setUserid(Integer userid) {
        this.userid = userid;
    }

    public String getUsername() {
        return username;
    }
    public void setUsername(String username) {
        this.username = username;
    }
}
```

```

    }

    public String getUserpwd() {
        return userpwd;
    }

    public void setUserpwd(String userpwd) {
        this.userpwd = userpwd;
    }

    public Integer getAge() {
        return age;
    }

    public void setAge(Integer age) {
        this.age = age;
    }
}

```

- 4) 在**ejbModule**下创建**cn.edu.zjut.ejb.dao**包，并在其中新建名称分别为**UserDAO.java**和**UserDAORemote.java**的无状态会话Bean，具体内容分别如下：

```

//UserDAORemote.java
package cn.edu.zjut.ejb.dao;

import java.util.List;
import javax.ejb.Remote;
import cn.edu.zjut.ejb.User;

@Remote
public interface UserDAORemote {
    public List<User> select(String sql);
    public boolean insert(User user);
}

```

```

//UserDAO.java
package cn.edu.zjut.ejb.dao;

import java.util.List;

import javax.ejb.Stateless;
import javax.persistence.*;

import cn.edu.zjut.ejb.*;

```

```

@Stateless
public class UserDao implements UserDaoRemote {
    @PersistenceContext (type=PersistenceContextType.EXTENDED,
unitName="userpu")
    EntityManager em;
    public UserDao() {
    }
    public boolean insert(User user){
        try{
            em.persist(user);
        }catch(Exception e){
            e.printStackTrace();
            return false;
        }
        return true;
    }
    public List<User> select(String sql){
        List< User> userlist=null;
        try{
            Query q=em.createQuery(sql);
            userlist=(List<User>)q.getResultList();
        }catch(Exception e){
            e.printStackTrace();
            return userlist;
        }
        return userlist;
    }
}
}

```

5) 整个工程目录结构如图7-12所示。

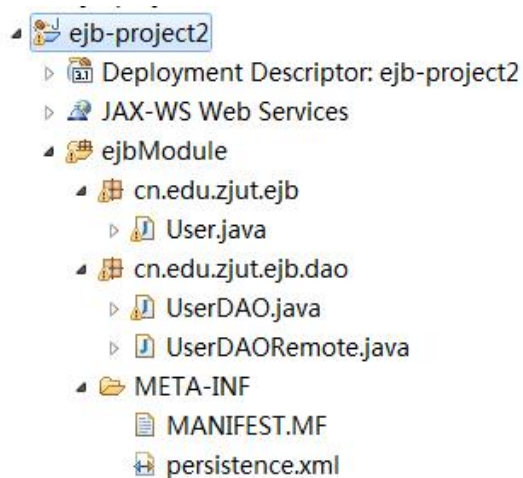


图7-12 ejb-project2工程目录结构

- 6) 右键单击**ejb-project2**工程，选择菜单[**Export->EJB Jar File**]将工程部署到服务器。

## 5、创建EJB客户端工程：

- 1) 新建一个名称为 **ejb-project2-client** 的 [Java Project] 工程，在 **ejb-project2-client**的[Java Build Path->Libraries]添加External JAR库文件 **hibernate-jpa-2.0-api-1.0.1.Final.jar**，该文件所在位置为 [%JBOSS\_HOME% \modules\javax\persistence\api\main]；添加External JAR库文件 **jboss-client.jar**，该文件所在位置为 [%JBOSS\_HOME% \bin\client]。
- 2) 在src目录下新建一个名称为**cn.edu.zjut.ejb**的包，将工程**ejb-project2**的 **User.java**拷贝到其中。
- 3) 在src目录下新建一个名称为**cn.edu.zjut.ejb.dao**的包，将工程**ejb-project2**的 **UserDAORemote.java**拷贝到其中。
- 4) 在src目录下新建一个名称为**jboss-ejb-client.properties**的文件，具体内容如下：

```
endpoint.name=client-endpoint
remote.connectionprovider.create.options.org.xnio.Options.SSL_ENABLED=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_POLICY_NOANONYMOUS=false

remote.connection.default.username=appuser
remote.connection.default.password=apppwd
```

- 5) 在src目录下新建一个名称为**cn.edu.zjut.ejb.client**的包，新建一个文件 **UserDAOTest.java**，具体内容如下：

```
package cn.edu.zjut.ejb.client;

import java.util.Hashtable;
import java.util.List;

import javax.naming.Context;
```

```

import javax.naming.InitialContext;
import javax.naming.NamingException;

import cn.edu.zjut.ejb.dao.UserDAORemote;
import cn.edu.zjut.ejb.User;

public class UserDAOTest {
private static UserDAORemote 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-project2";
    final String beanName = "UserDAO";
    final String viewClassName = UserDAORemote.class.getName();
    final String namespace = "ejb:" + appName + "/" + moduleName
        + "/" + beanName + "!" + viewClassName;
    System.out.println(namespace);
    return (UserDAORemote) context.lookup(namespace);
}

public static void main(String[] args) {
    try{
        UserDAORemote userdao = lookupRemoteStatelessEjbBean();
        System.out.println(userdao);
        List<User> userlist = userdao.select("select u from User u");
        System.out.println("userlist="+userlist);
        User u=(User)userlist.get(0);
        System.out.println("username="+u.getUsername());
        System.out.println("age="+u.getAge());
    }catch(NamingException e){
        e.printStackTrace();
    }
}
}

```

6) 整个**ejb-project2-client**工程目录结构如图7-13所示。

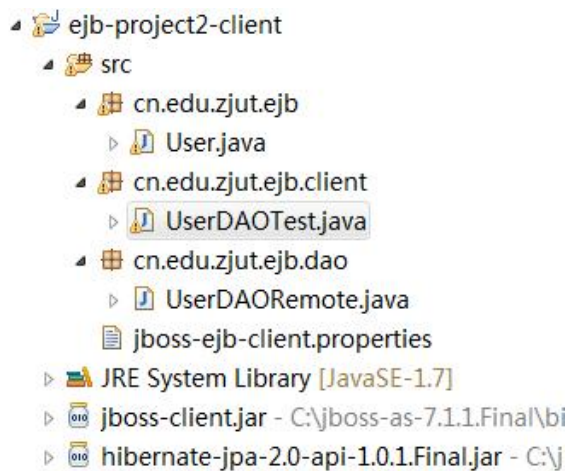


图7-13 ejb-project2-client工程目录结构

7) 启动WildFly服务器，运行**UserDAOTest.java**文件，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) 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，总结实体Bean的具体开发过程；
- (4) 碰到的问题及思考；
- (5) 实验收获及总结。

上交程序源代码，代码中应有相关注释。