

本科实验报告

Web程序设计

课程名称： web程序设计 姓 名： 史惠文

学院： 计算机学院

专业： 软件工程

学号： 8209190513

指导老师： 宋铁

小组成员： 史惠文

完成时间： 2021.1.26

分数：

2021年2月8日星期一

**实验一 基于MVC用JSP/Servlet实现JPetStore**

实验学时： 4

每组人数： 2-3

实验类型： 2 （1：基础性 2：综合性 3：设计性 4：研究性）

实验要求： 1 （1：必修 2：选修 3：其它）

实验类别： 3 （1：基础 2：专业基础 3：专业 4：其它）

**一、实验目的**

掌握使用JSP/Servlet技术进行Web应用网站的开发。熟练使用IntelliJ IDEA工具进行Web应用项目的构建和开发，掌握JSP和Servlet相关技术的运用，包括JSP语法、EL表达式语言、JSTL标准标签库、Servlet常用接口和类、JDBC等技术，加深对MVC开发模式的理解和应用。

**二、实验内容**

基于MVC开发模式用JSP/Servlet技术实现宠物商店网站JPetStore。具体内容包含基本任务和扩展任务两个部分：

**基本任务：**

1. 商品展示业务模块，包括大类Category、小类Product和具体商品Item的展示和搜索功能。
2. 用户管理业务模块，包括用户注册、登录、修改用户信息、查询用户相关订单等业务功能。
3. 订单管理模块，包括购物车管理、新增订单、地址信息等业务功能。

**扩展任务：**

1. 添加验证码功能：在用户注册和用户登录模块中添加验证码功能。
2. 购物车功能：原项目中购物车为内存购物车，即将用户相关的购物车存储在session中；在数据库中添加购物车表，将购物车数据持久化，以便用户下次登录时可查看或修改自己的购物车数据。
3. 日志功能：数据库中添加日志信息表，给项目添加日志功能，用户登录后记录用户行为，比如浏览了哪些商品、将商品添加进购物车、生成订单等。

**三、实验要求**

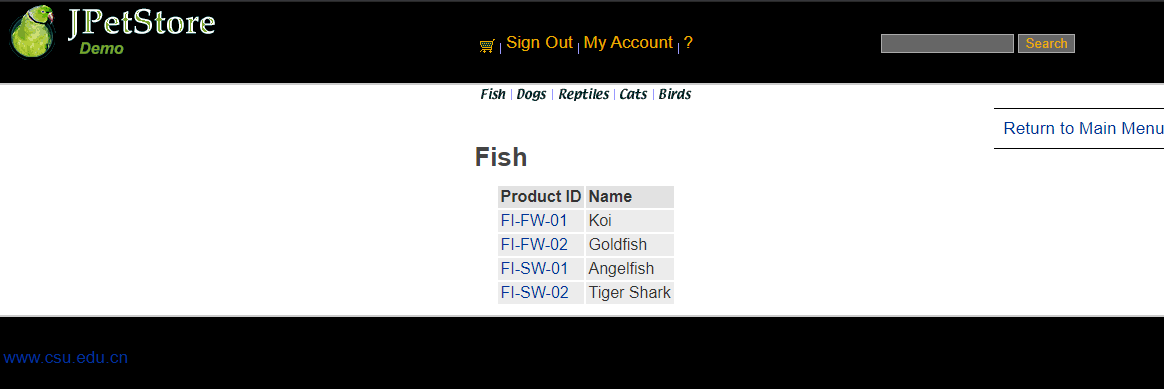
1. 使用IntelliJ IDEA、MySQL和Tomcat环境进行开发；
2. 基础业务功能需和传统宠物商店示例一样；
3. 扩展任务功能（验证码和日志）在完成任务书要求上可自由发挥。

**四、实验步骤**

1. 按传统宠物商店示例完成商品、用户、订单三个基础业务模块；
2. 完成验证码、购物车和日志三个扩展功能。
3. **主要仪器设备**

**Intellij IDEA**

1. 功能实现与关键代码
2. 商品展示业务模块，包括大类Category、小类Product和具体商品Item的展示和搜索功能。
3. 大类Category



大类Category先实现Category.jsp进行大类的具体展示，利用jsp中c:foreach语法对每个大类进行展示，具体类的ID利用a标签进行跳转到小类进行查看，接下来实现domain层的Category.java，完成大类的基础设计，接下来实现persistence中的数据库连接层，先写DAO定义所有接口，再写实现的Implement方法，对数据库进行操作；再写一个Category的Service类，可以在Servlet层中调用该类的相关方法进行进一步操作，最后完成ViewCategoryServlet对jsp页面与业务层进行连接跳转。

关键代码：

public class ViewCategoryServlet extends HttpServlet {

private static final String VIEW\_CATEGORY = "/WEB-INF/jsp/catalog/Category.jsp";

private String categoryId;

protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {

doGet(request, response);

}

protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {

categoryId = request.getParameter("categoryId");

CatalogService service = new CatalogService();

Category category = service.getCategory(categoryId);

List<Product> productList = service.getProductListByCategory(categoryId);

HttpSession session = request.getSession();

session.setAttribute("category" , category);

session.setAttribute("productList" , productList);

if(session.getAttribute("account") != null) {

DailyService dailyService = new DailyService();

String action = "添加购物车数据";

Account account = (Account) session.getAttribute("account");

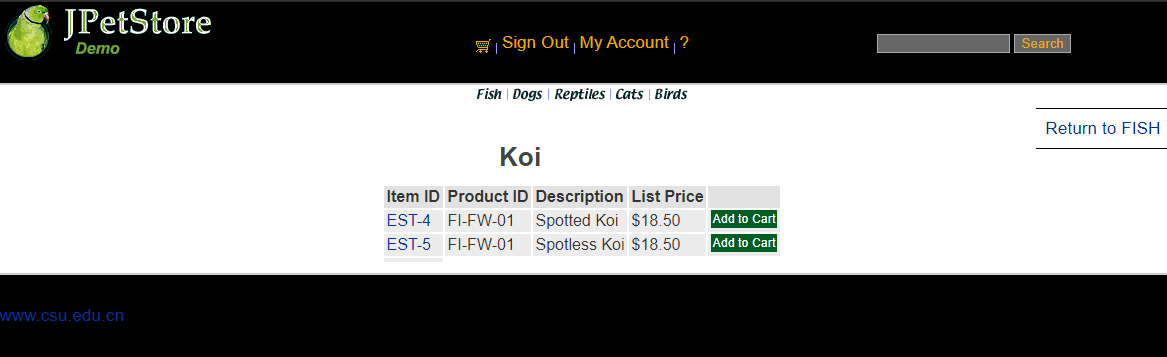
dailyService.insertlogInfo(account.getUsername(), action);

}

request.getRequestDispatcher(VIEW\_CATEGORY).forward(request,response);

}

}

1. 小类product

小类Product先完善Product.jsp，主要是通过productId对产品类进行信息展示，之后再实现Product.java，接下进行ProductDAOImpl数据库的连接进行编写，最后完成Servlet的编写。

关键代码：

public class ViewProductServlet extends HttpServlet {

private static final String VIEW\_PRODUCT = "/WEB-INF/jsp/catalog/Product.jsp";

private String productId;

protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {

productId = request.getParameter("productId");

CatalogService service = new CatalogService();

Product product = service.getProduct(productId);

List<Item> itemList = service.getItemListByProduct(productId);

HttpSession session = request.getSession();

session.setAttribute("product", product);

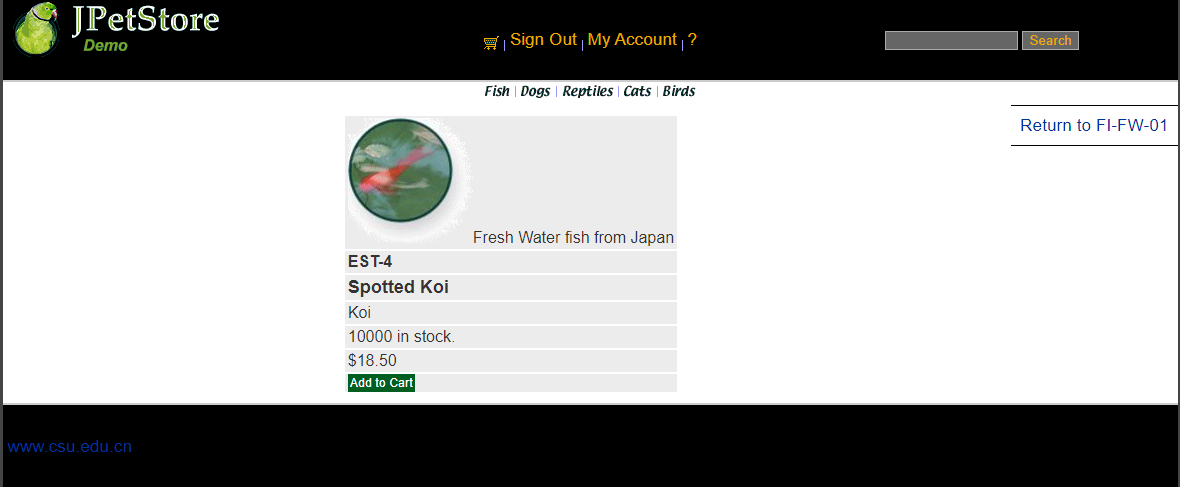
session.setAttribute("itemList", itemList);

request.getRequestDispatcher(VIEW\_PRODUCT).forward(request, response);

}

}

1. 具体商品item



用户点击产品后进入查看具体商品，完善Item.jsp，Copy代码模版中的Item.java，实现ItemDAOImpl的具体数据库连接操作，完成最后的Servlet连接。

关键代码

public class ViewItemServlet extends HttpServlet {

private static final String VIEW\_ITEM = "/WEB-INF/jsp/catalog/Item.jsp";

private String itemId;

protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {

doGet(request, response);

}

protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {

itemId = request.getParameter("itemId");

CatalogService service = new CatalogService();

Item item = service.getItem(itemId);

HttpSession session = request.getSession();

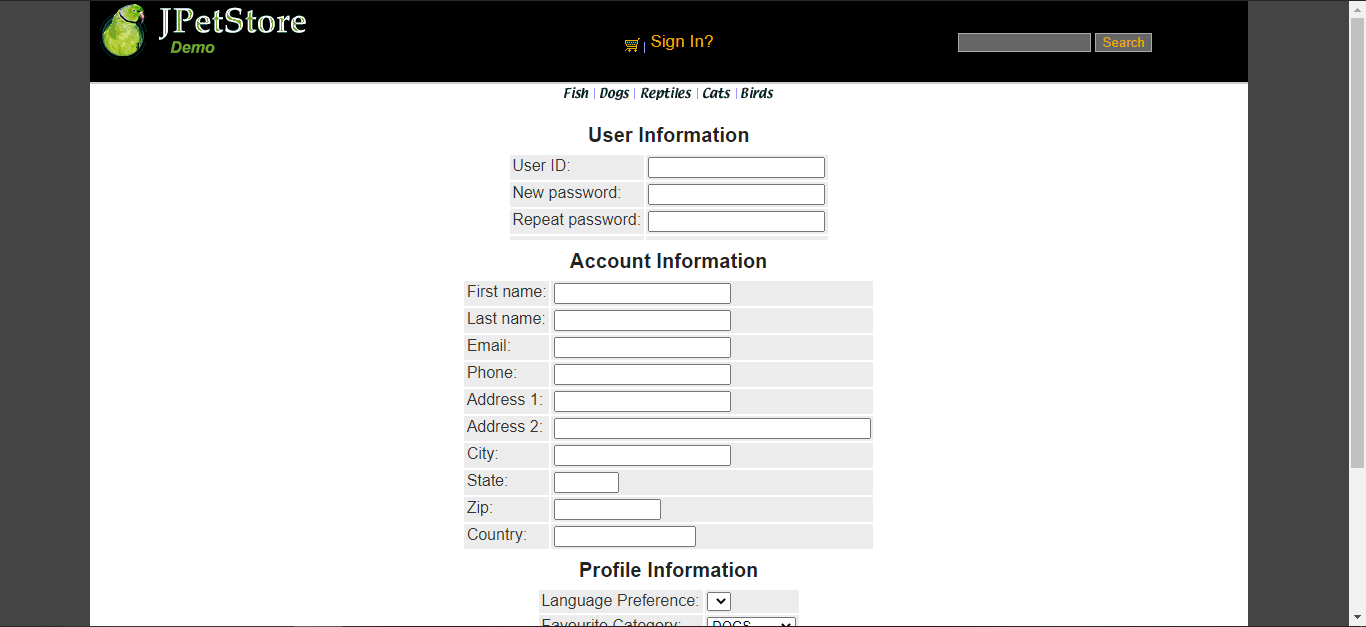
session.setAttribute("item", item);

request.getRequestDispatcher(VIEW\_ITEM).forward(request, response);

}

}

1. 用户管理业务模块，包括用户注册、登录、修改用户信息、查询用户相关订单等业务功能。

 用户注册时，需要一个全新的包含用户所有信息的输入界面即NewAccountForm.jsp，之后需要先进行设计NewAccountFormServlet，该Servlet的功能仅为实现用户点击注册后跳转用户输入新账号信息的表格，接下实现Account.java，完成Account的基础功能，之后进行AccountDAO接口定义以及完成AccountDAOImpl连接数据库的实现，接下来完成AccountService的编写，之后进行NewAccountServlet配置，获取网页界面的信息，赋值给一个新账号，再将该账号插入到数据库中。

源代码：

package org.csu.mypetstore.web.servlets;

import org.csu.mypetstore.domain.Account;

import org.csu.mypetstore.persistence.DBUtil;

import org.csu.mypetstore.service.DailyService;

import javax.servlet.ServletException;

import javax.servlet.http.HttpServlet;

import javax.servlet.http.HttpServletRequest;

import javax.servlet.http.HttpServletResponse;

import javax.servlet.http.HttpSession;

import java.io.IOException;

import java.sql.Connection;

import java.sql.PreparedStatement;

import java.sql.ResultSet;

public class UpdateServlet extends HttpServlet {

private static final String MYACCOUNT = "/WEB-INF/jsp/catalog/Main.jsp";

private final String EDIT\_FORM = "/WEB-INF/jsp/account/EditAccountForm.jsp";

private static final String UPDATE\_ACCOUNT =

"UPDATE account SET email = ? ,firstname =?,lastname = ?," +

"addr1 = ?,addr2 = ?,city = ?,state = ?,zip = ?,country = ?," +

"phone = ? ,password = ? WHERE userid = ? ";

@Override

protected void doPost(HttpServletRequest req, HttpServletResponse resp) throws ServletException, IOException {

doGet(req,resp);

}

@Override

protected void doGet(HttpServletRequest req, HttpServletResponse resp) throws ServletException, IOException {

HttpSession session = req.getSession();

String email = req.getParameter("account.email");

String firstname = req.getParameter("account.firstName");

String lastname = req.getParameter("account.lastName");

String phone = req.getParameter("account.phone");

String address1 = req.getParameter("account.address1");

String address2 = req.getParameter("account.address2");

String city = req.getParameter("account.city");

String state = req.getParameter("account.state");

String zip = req.getParameter("account.zip");

String country = req.getParameter("account.country");

String user = "a";

String password = req.getParameter("password");

String repeat = req.getParameter("repeat");

if (password.equals(repeat)) {

try {

Connection connection = DBUtil.getConnection();

PreparedStatement preparedStatement = connection.prepareStatement(UPDATE\_ACCOUNT);

preparedStatement.setString(1, email);

preparedStatement.setString(2, firstname);

preparedStatement.setString(3, lastname);

preparedStatement.setString(4, address1);

preparedStatement.setString(5, address2);

preparedStatement.setString(6, city);

preparedStatement.setString(7, state);

preparedStatement.setString(8, zip);

preparedStatement.setString(9, country);

preparedStatement.setString(10, phone);

preparedStatement.setString(11, password);

preparedStatement.setString(12, user);

// preparedStatement.setString(2,password);

int resultSet = preparedStatement.executeUpdate();

DBUtil.closeResultSet(resultSet);

DBUtil.closePreparedStatement(preparedStatement);

DBUtil.closeConnection(connection);

} catch (Exception ex) {

ex.printStackTrace();

}

DailyService dailyService = new DailyService();

String action = "fix password";

Account account = (Account) session.getAttribute("account");

dailyService.insertlogInfo(account.getUsername(),action);

session.removeAttribute("account");

req.getRequestDispatcher(MYACCOUNT).forward(req, resp);

}else {

req.setAttribute("msg","no similar");

req.getRequestDispatcher(EDIT\_FORM).forward(req,resp);

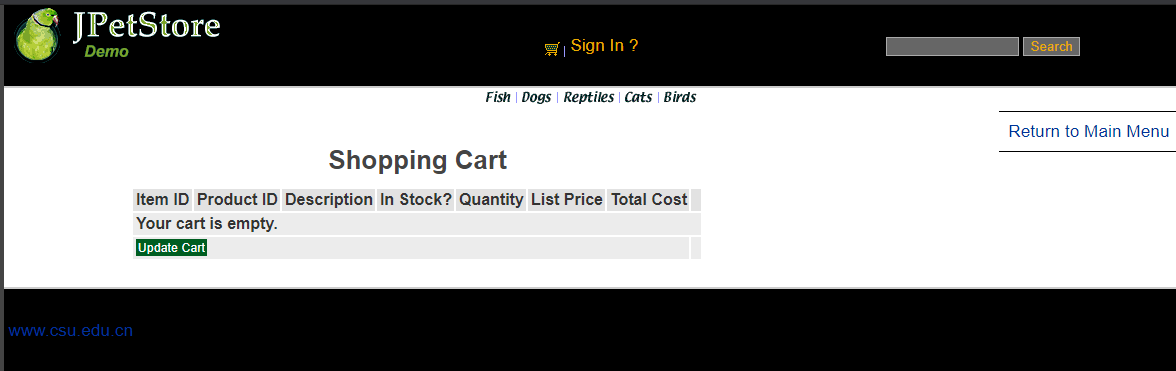
}

}

}

1. 订单管理模块，包括购物车管理、新增订单、地址信息等业务功能。

购物车管理(添加，删除)：



先实现Cart.jsp，能够在网页显示购物车栏目信息，接下来具体实现Cart.java使得用户可以创建Cart类，最后创建AddItemToCartServlet，当用户在浏览商品界面时，点击Add to Cart时可以将该商品加入到购物车中并显示相关信息。

源代码：

Cart.jsp

<%@ include file="../common/IncludeTop.jsp"%>

<div id="BackLink">

<a href="main">Return to Main Menu</a>

</div>

<div id="Catalog">

<div id="Cart">

<h2>Shopping Cart</h2>

<form action="zhuceServelet" method="post">

<table>

<tr>

<th><b>Item ID</b></th>

<th><b>Product ID</b></th>

<th><b>Description</b></th>

<th><b>In Stock?</b></th>

<th><b>Quantity</b></th>

<th><b>List Price</b></th>

<th><b>Total Cost</b></th>

<th>&nbsp;</th>

</tr>

<c:if test="${sessionScope.cart.numberOfItems == 0}">

<tr>

<td colspan="8"><b>Your cart is empty.</b></td>

</tr>

</c:if>

<c:forEach var="cartItem" items="${sessionScope.cart.cartItems}">

<tr>

<td>

<a href="viewItem?itemId=${cartItem.item.itemId}">${cartItem.item.itemId}</a>

</td>

<td>

${cartItem.item.product.productId}

</td>

<td>

${cartItem.item.attribute1} ${cartItem.item.attribute2}

${cartItem.item.attribute3} ${cartItem.item.attribute4}

${cartItem.item.attribute5} ${cartItem.item.product.name}

</td>

<td>

${cartItem.inStock}

</td>

<td>

<input type="text" name="${cartItem.item.itemId}" value="${cartItem.quantity}" />

</td>

<td>

<fmt:formatNumber value="${cartItem.item.listPrice}" pattern="$#,##0.00" />

</td>

<td>

<fmt:formatNumber value="${cartItem.total}" pattern="$#,##0.00" />

</td>

<td>

<a href="removeItemFromCart?workingItemId=${cartItem.item.itemId}" class="Button">Remove</a>

</td>

</tr>

</c:forEach>

<tr>

<td colspan="7">

<input type="submit" value="Update Cart"/>

</td>

<td>&nbsp;</td>

</tr>

</table>

</form>

<c:if test="${sessionScope.cart.numberOfItems > 0}">

<a href="zhuceServelet" class="Button">Proceed to Checkout</a>

</c:if>

</div>

<div id="Separator">&nbsp;</div>

</div>

<%@ include file="../common/IncludeBottom.jsp"%>

Viewcartgroyservlet：

package org.csu.mypetstore.web.servlets;

import org.csu.mypetstore.domain.Account;

import org.csu.mypetstore.domain.Category;

import org.csu.mypetstore.domain.Product;

import org.csu.mypetstore.service.CatalogService;

import org.csu.mypetstore.service.DailyService;

import javax.servlet.ServletException;

import javax.servlet.http.HttpServlet;

import javax.servlet.http.HttpServletRequest;

import javax.servlet.http.HttpServletResponse;

import javax.servlet.http.HttpSession;

import java.io.IOException;

import java.util.List;

/\*\*

\* Created by songtie on 2015/4/21.

\*/

public class ViewCategoryServlet extends HttpServlet {

private static final String VIEW\_CATEGORY = "/WEB-INF/jsp/catalog/Category.jsp";

private String categoryId;

protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {

doGet(request, response);

}

protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {

categoryId = request.getParameter("categoryId");

CatalogService service = new CatalogService();

Category category = service.getCategory(categoryId);

List<Product> productList = service.getProductListByCategory(categoryId);

HttpSession session = request.getSession();

session.setAttribute("category" , category);

session.setAttribute("productList" , productList);

if(session.getAttribute("account") != null) {

DailyService dailyService = new DailyService();

String action = "添加购物车数据";

Account account = (Account) session.getAttribute("account");

dailyService.insertlogInfo(account.getUsername(), action);

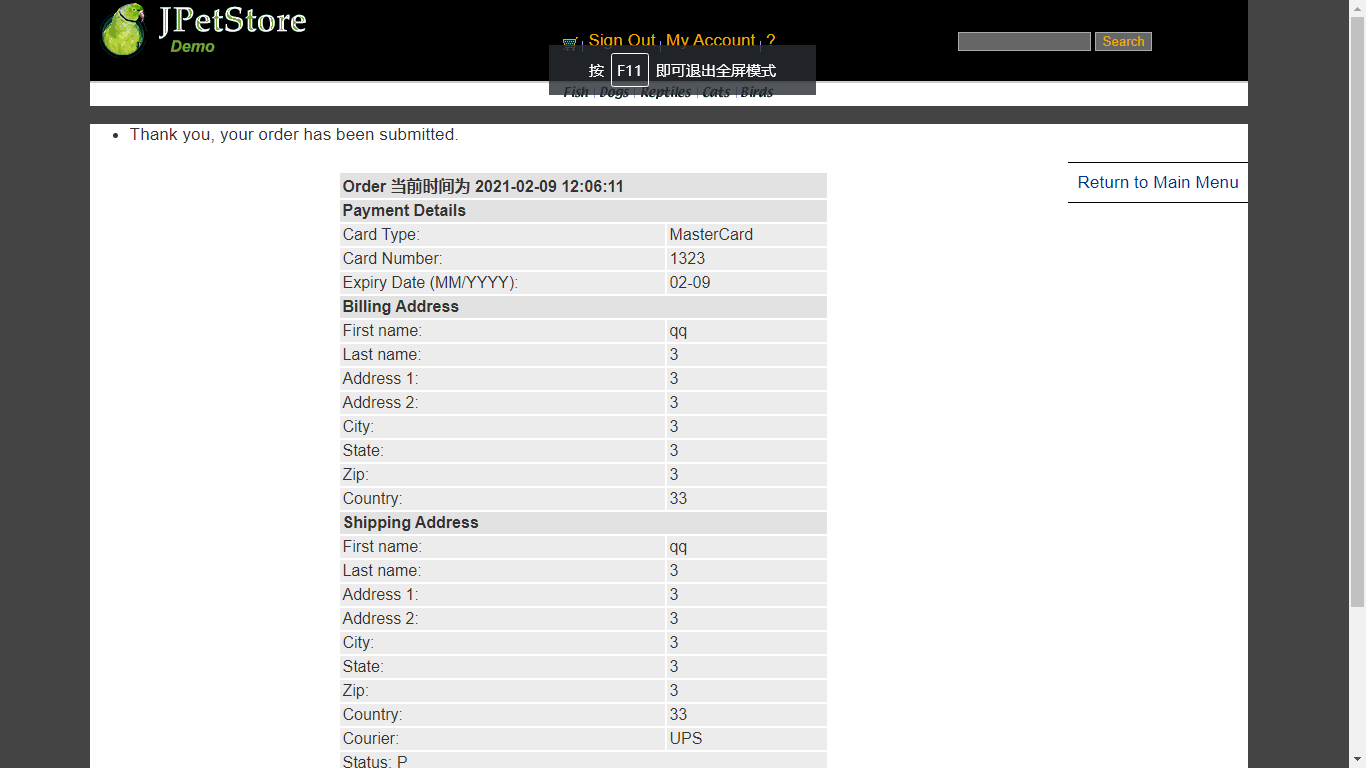
}

request.getRequestDispatcher(VIEW\_CATEGORY).forward(request,response);

}

}

订单：



先写出3个jsp，然后利用3个servlet跳转界面

源代码：

package org.csu.mypetstore.web.servlets.order;

import org.csu.mypetstore.domain.Account;

import org.csu.mypetstore.service.DailyService;

import javax.servlet.ServletException;

import javax.servlet.http.HttpServlet;

import javax.servlet.http.HttpServletRequest;

import javax.servlet.http.HttpServletResponse;

import javax.servlet.http.HttpSession;

import java.io.IOException;

public class ConfirmOrderServlet extends HttpServlet {

private static final String CONFIEM = "/WEB-INF/jsp/order/ConfirmOrder.jsp";

@Override

protected void doGet(HttpServletRequest req, HttpServletResponse resp) throws ServletException, IOException {

DailyService dailyService = new DailyService();

HttpSession session = req.getSession();

if(session.getAttribute("account") != null) {

String action = "用户继续提交订单";

Account account = (Account) session.getAttribute("account");

dailyService.insertlogInfo(account.getUsername(), action);

}

req.getRequestDispatcher(CONFIEM).forward(req,resp);

}

@Override

protected void doPost(HttpServletRequest req, HttpServletResponse resp) throws ServletException, IOException {

doGet(req,resp);

}

}

package org.csu.mypetstore.web.servlets.order;

import javax.servlet.ServletException;

import javax.servlet.http.HttpServlet;

import javax.servlet.http.HttpServletRequest;

import javax.servlet.http.HttpServletResponse;

import java.io.IOException;

public class MyOrderServlet extends HttpServlet {

private final String MY\_ORDER = "/WEB-INF/jsp/order/ListOrders.jsp";

@Override

protected void doPost(HttpServletRequest req, HttpServletResponse resp) throws ServletException, IOException {

doGet(req,resp);

}

@Override

protected void doGet(HttpServletRequest req, HttpServletResponse resp) throws ServletException, IOException {

req.getRequestDispatcher(MY\_ORDER).forward(req,resp);

}

}

package org.csu.mypetstore.web.servlets.order;

import org.csu.mypetstore.domain.Account;

import org.csu.mypetstore.service.DailyService;

import javax.servlet.ServletException;

import javax.servlet.http.HttpServlet;

import javax.servlet.http.HttpServletRequest;

import javax.servlet.http.HttpServletResponse;

import javax.servlet.http.HttpSession;

import java.io.IOException;

public class viewOrderServlet extends HttpServlet {

private final String VIEWORDER = "/WEB-INF/jsp/order/ViewOrder.jsp";

@Override

protected void doPost(HttpServletRequest req, HttpServletResponse resp) throws ServletException, IOException {

DailyService dailyService = new DailyService();

HttpSession session = req.getSession();

if(session.getAttribute("account") != null) {

String action = "提交";

Account account = (Account) session.getAttribute("account");

dailyService.insertlogInfo(account.getUsername(), action);

}

doGet(req,resp);

}

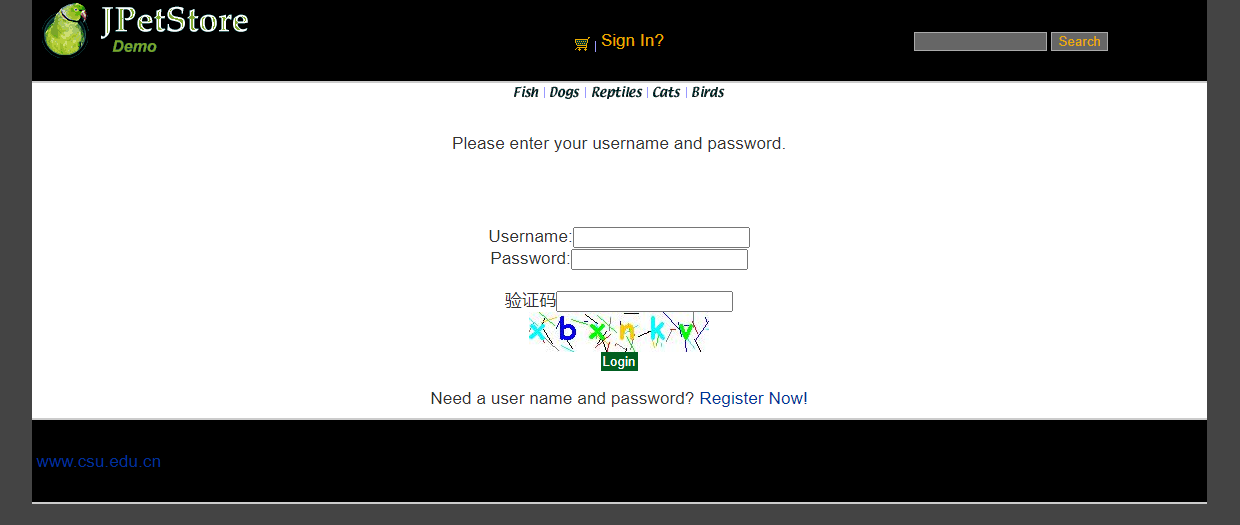
@Override

protected void doGet(HttpServletRequest req, HttpServletResponse resp) throws ServletException, IOException {

req.getRequestDispatcher(VIEWORDER).forward(req,resp);

}

}

1. **实验内容——扩展任务。**
2. 添加验证码功能：在用户注册和用户登录模块中添加验证码功能。

基本原理：

可以看到这里有一串验证码，就是先设置字符表即个字母和各个数字，接下来利用字符表产生随机验证码，同时随机产生干扰点（点数适合即可，具体没有特别要求），产生图像，画背景，在不同高度输出验证码的不同字符，设置设置浏览器不要缓存此图片，然后创建内存图像并获得图形，将图像传到客户端，验证码功能即可实现随机显示。

import java.awt.Color;

import java.awt.Font;

import java.awt.Graphics;

import java.awt.image.BufferedImage;

import java.io.IOException;

import java.io.OutputStream;

import java.util.Random;

import javax.imageio.ImageIO;

public final class GraphicHelper {

/\*\*

\* 以字符串形式返回生成的验证码，同时输出一个图片

\*

\* @param width

\* 图片的宽度

\* @param height

\* 图片的高度

\* @param imgType

\* 图片的类型

\* @param output

\* 图片的输出流(图片将输出到这个流中)

\* @return 返回所生成的验证码(字符串)

\*/

public static String create(final int width, final int height, final String imgType, OutputStream output) {

StringBuffer sb = new StringBuffer();

Random random = new Random();

BufferedImage image = new BufferedImage(width, height, BufferedImage.TYPE\_INT\_RGB);

Graphics graphic = image.getGraphics();

graphic.setColor(Color.getColor("F8F8F8"));

graphic.fillRect(0, 0, width, height);

Color[] colors = new Color[] { Color.BLUE, Color.GRAY, Color.GREEN, Color.RED, Color.BLACK, Color.ORANGE,

Color.CYAN };

// 在 "画板"上生成干扰线条 ( 50 是线条个数)

for (int i = 0; i < 50; i++) {

graphic.setColor(colors[random.nextInt(colors.length)]);

final int x = random.nextInt(width);

final int y = random.nextInt(height);

final int w = random.nextInt(20);

final int h = random.nextInt(20);

final int signA = random.nextBoolean() ? 1 : -1;

final int signB = random.nextBoolean() ? 1 : -1;

graphic.drawLine(x, y, x + w \* signA, y + h \* signB);

}

// 在 "画板"上绘制字母

graphic.setFont(new Font("Comic Sans MS", Font.BOLD, 30));

for (int i = 0; i < 6; i++) {

final int temp = random.nextInt(26) + 97;

String s = String.valueOf((char) temp);

sb.append(s);

graphic.setColor(colors[random.nextInt(colors.length)]);

graphic.drawString(s, i \* (width / 6), height - (height / 3));

}

graphic.dispose();

try {

ImageIO.write(image, imgType, output);

} catch (IOException e) {

e.printStackTrace();

}

return sb.toString();

}

}

import java.io.IOException;

import java.io.OutputStream;

import javax.servlet.ServletException;

import javax.servlet.http.HttpServlet;

import javax.servlet.http.HttpServletRequest;

import javax.servlet.http.HttpServletResponse;

import javax.servlet.http.HttpSession;

public class VerifyCodeServlet extends HttpServlet {

private static final long serialVersionUID = 3398560501558431737L;

@Override

protected void service(HttpServletRequest request, HttpServletResponse response)

throws ServletException, IOException {

// 获得 当前请求 对应的 会话对象

HttpSession session = request.getSession();

// 从请求中获得 URI ( 统一资源标识符 )

String uri = request.getRequestURI();

System.out.println("hello : " + uri);

final int width = 180; // 图片宽度

final int height = 40; // 图片高度

final String imgType = "jpeg"; // 指定图片格式 (不是指MIME类型)

final OutputStream output = response.getOutputStream(); // 获得可以向客户端返回图片的输出流

// (字节流)

// 创建验证码图片并返回图片上的字符串

String code = GraphicHelper.create(width, height, imgType, output);

System.out.println("验证码内容: " + code);

// 建立 uri 和 相应的 验证码 的关联 ( 存储到当前会话对象的属性中 )

session.setAttribute(uri, code);

System.out.println(session.getAttribute(uri));

}

}

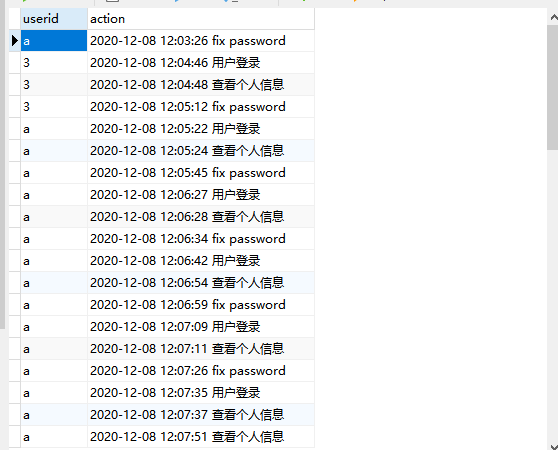
<span class="cell-5" style="text-align: center;">

<img src="paint" οnclick="myRefersh(this)">

</span>

1. 日志功能：数据库中添加日志信息表，给项目添加日志功能，用户登录后记录用户行为，比如浏览了哪些商品、将商品添加进购物车、生成订单等。

项目所完成的功能情况：

可以看到，日志加入到了数据库中，显示某个用户的具体时间和行为，可以记录的有：时间（年，月，日，时，分，秒），具体行为（登录，个人信息，修改密码，添加商品到购物车，提交订单等）

关键代码：

package org.csu.mypetstore.service;

import org.csu.mypetstore.persistence.DailyDao;

import org.csu.mypetstore.persistence.impl.DailyDaoImpl;

import java.text.SimpleDateFormat;

import java.util.Date;

public class DailyService {

private DailyDao daily;

public DailyService(){

daily = new DailyDaoImpl();

}

public void insertlogInfo(String username,String action){

action = getTime()+action;

daily.insertDaily(username,action);

}

public String getTime(){

Date data = new Date();

String time = new SimpleDateFormat("yyyy-MM-dd HH:mm:ss").format(data);

time += " ";

return time;

}

}

**三 、小组成员分工及实验总结**

1. 小组分工：

史惠文是组长，负责全部

1. 实验总结

This experiment has benefited me a lot, and I feel a lot. Before doing the experiment, you must thoroughly understand the knowledge in the textbook, because this is the basis of doing the experiment, otherwise, you will not understand it when the teacher explains it, which will make it more difficult for you to do the experiment, and waste the precious time of doing the experiment, This will make you a great waste of time and get twice the result with half the effort

When doing an experiment, you must do it yourself. You must make clear every step and every detail. After the experiment, you have to review and think. In this way, you will be impressed and remember firmly. Otherwise, you will forget everything soon after. It's better not to do it. When doing the experiment, the teacher will teach us some knowledge that is not in the textbook according to his own experience, Broaden our horizons and make us realize that in the classic MVC mode of life, m refers to business model, V refers to user interface, and C is controller. The purpose of using MVC is to separate the implementation code of M and V, so that the same program can use different forms of expression. Among them, the definition of view is relatively clear, that is, the user interface. The application is so extensive

This experiment is mainly to use an MVC model to do a small project, and finally I have done a lot of work. There are captcha function Shopping Cart module, and then there are many other functions I have implemented, and then most of them use MVC model, which makes me understand the general idea and framework of this project. It also benefits me a lot, which improves my ability of web programming, and makes some progress in the logic of the whole project.

**实验二 使用AJAX和jQuery改善用户体验**

实验学时： 4

每组人数： 2-3

实验类型： 2 （1：基础性 2：综合性 3：设计性 4：研究性）

实验要求： 1 （1：必修 2：选修 3：其它）

实验类别： 3 （1：基础 2：专业基础 3：专业 4：其它）

**一、实验目的**

熟悉AJAX相关技术，掌握异步交互的通信方式，熟练使用JavaScript以及CSS等客户端技术，熟练掌握XMLHttpRequest对象的使用，熟悉通过jQuery插件简化前端界面交互和AJAX代码。

**二、实验内容**

AJAX开发。在JSP/Servlet实现的JPetStore项目基础上，在表单验证、表单输入、订单更新等环节增加AJAX技术，改善项目的用户体验。以下三个任务为必须完成部分，还可自行添加其他改进用户体验的功能。

**基本任务：**

1. 账号管理模块。在新建账号和修改账号信息页面上用AJAX技术进行表单验证，如“判断用户名是否已存在”等功能。
2. 查询商品模块。在主页面右上角的查询商品功能中添加自动补全功能。
3. 购物车模块。在购物车模块中修改购买数量时，通过AJAX修改购物车信息。

**选做任务：**

1. 商品展示模块。在首页展示商品的图片上添加AJAX效果，即当鼠标移到某个商品图片上时，用悬浮窗展示该类型商品的信息。
2. 订单模块。通过选项卡或模版内容实现订单多个信息部分的录入，不需要页面跳转，在一个页面完成订单所有内容的录入。
3. 其他效果：可自行选择其他UI模版，或选择其他功能点进行用户体验的优化。

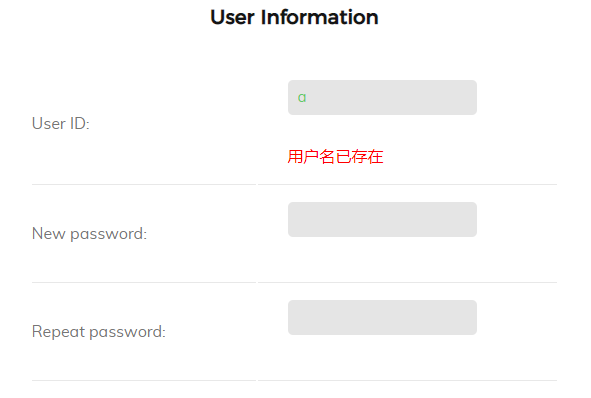
**三、实验要求**

1. 使用JavaScript+HTML+CSS进行客户端编程；
2. 熟练XMLHttpRequest对象与服务器异步交互；
3. 熟练使用AJAX技术改善Web应用的用户体验。
4. 建议使用jQuery插件进行开发，减少JavaScript代码量。

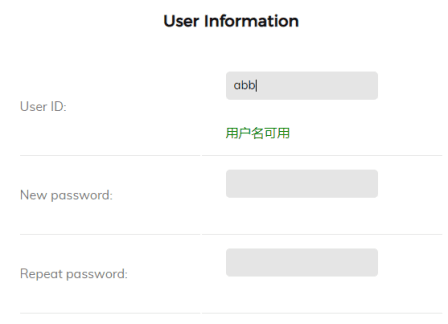
**四、实验步骤**

1. 将之前用JSP/Servlet完成的JPetStore项目部署并配置运行；
2. 用AJAX改善登录的用户体验。
3. 功能实现与核心代码
4. 账号管理模块。

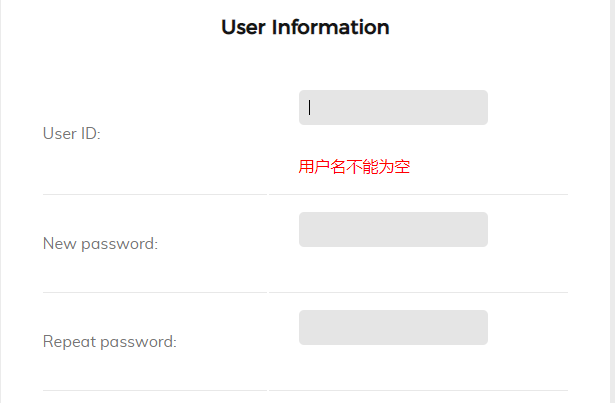
当用户名已存在时



当用户名不存在时



当用户名为空时



核心代码：

package org.csu.mypetstore.web.servlets;

import org.csu.mypetstore.domain.Account;

import org.csu.mypetstore.service.AccountService;

import javax.servlet.ServletException;

import javax.servlet.http.HttpServlet;

import javax.servlet.http.HttpServletRequest;

import javax.servlet.http.HttpServletResponse;

import java.io.IOException;

import java.io.PrintWriter;

public class UsernameIsExistServlet extends HttpServlet {

protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {

doGet(request,response);

}

protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {

String username = request.getParameter("username");

AccountService service = new AccountService();

response.setContentType("text/xml");

PrintWriter out = response.getWriter();

out.println("<response>");

if (username.isEmpty())

{

out.println("<msg>Empty</msg>");

System.out.println("Empty");

}

else if(service.usernameIsExist(username)){

out.println("<msg>Exist</msg>");

System.out.println("Exist");

}

else {

out.println("<msg>NotExist</msg>");

System.out.println("NotExist");

}

out.println("</response>");

out.flush();

out.close();

}

}

Js代码：

var xmlHttpRequest;

function createXMLHttpRequest()

{

if (window.XMLHttpRequest) //非IE浏览器

{

xmlHttpRequest = new XMLHttpRequest();

}

else if (window.ActiveObject)//IE6以上版本的IE浏览器

{

xmlHttpRequest = new ActiveObject("Msxml2.XMLHTTP");

}

else //IE6及以下版本IE浏览器

{

xmlHttpRequest = new ActiveObject("Microsoft.XMLHTTP");

}

}

function usernameIsExist() {

var username = document.registerForm.username.value;

sendRequest("usernameIsExist?username=" + username);

}

function sendRequest(url) {

createXMLHttpRequest();

xmlHttpRequest.open("GET", url, true);

xmlHttpRequest.onreadystatechange = processResponse;

xmlHttpRequest.send(null);

}

function processResponse() {

if (xmlHttpRequest.readyState == 4) {

if (xmlHttpRequest.status == 200) {

var responseInfo = xmlHttpRequest.responseXML.getElementsByTagName("msg")[0].firstChild.data;

var div1 = document.getElementById("usernameMsg");

if (responseInfo == "Empty")

{

div1.innerHTML = "<font color='red'>用户名不能为空</font>";

}

else if (responseInfo == "Exist") {

div1.innerHTML = "<font color='red'>用户名已存在</font>";

} else {

div1.innerHTML = "<font color='green'>用户名可用</font>";

}

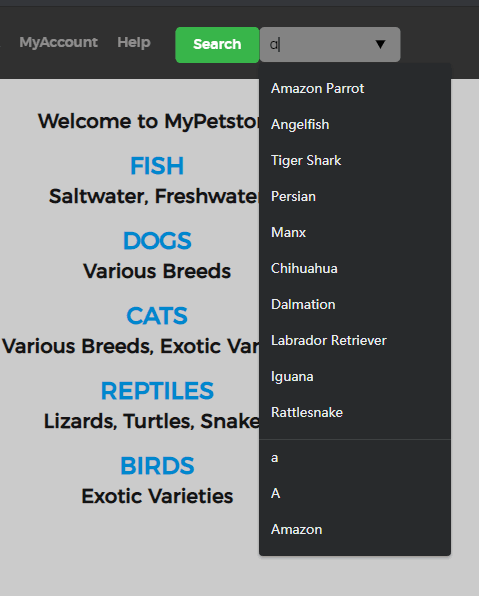
}

}

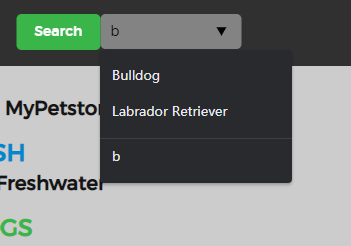
}

1. 查询商品模块。在主页面右上角的查询商品功能中添加自动补全功能。

当输入的为a时



当输入的为b时：



核心代码：

package org.csu.mypetstore.web.servlets;

import org.csu.mypetstore.domain.Product;

import org.csu.mypetstore.service.CatalogService;

import javax.servlet.ServletException;

import javax.servlet.http.HttpServlet;

import javax.servlet.http.HttpServletRequest;

import javax.servlet.http.HttpServletResponse;

import java.io.IOException;

import java.io.PrintWriter;

import java.util.ArrayList;

import java.util.List;

public class AutoComplementServlet extends HttpServlet {

CatalogService catalogService;

protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {

doGet(request,response);

}

protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {

String keyword = request.getParameter("keyword");

catalogService = new CatalogService();

List<Product> productList = catalogService.searchProductList(keyword);

System.out.println("productList" + productList);

response.setContentType("text/xml");

PrintWriter out = response.getWriter();

out.println("<products>");

for (Product product:productList) {

out.println("<product>");

out.println("<name>"+product.getName()+"</name>");

out.println("</product>");

System.out.println(product.getName());

}

out.println("</products>");

out.flush();

out.close();

}

}

Js代码：

var xmlHttpRequest;

function createXMLHttpRequest()

{

if (window.XMLHttpRequest) //非IE浏览器

{

xmlHttpRequest = new XMLHttpRequest();

}

else if (window.ActiveObject)//IE6以上版本的IE浏览器

{

xmlHttpRequest = new ActiveObject("Msxml2.XMLHTTP");

}

else //IE6及以下版本IE浏览器

{

xmlHttpRequest = new ActiveObject("Microsoft.XMLHTTP");

}

}

function completeItemList() {

var keyword = document.getElementById("keyword").value;

sendRequest("autoComplement?keyword=" + keyword);

}

function sendRequest(url) {

createXMLHttpRequest();

xmlHttpRequest.open("GET", url, true);

xmlHttpRequest.onreadystatechange = processResponse;

xmlHttpRequest.send(null);

}

function processResponse() {

if (xmlHttpRequest.readyState == 4) {

if (xmlHttpRequest.status == 200) {

var responseList = xmlHttpRequest.responseXML.getElementsByTagName("name");

var productList = document.getElementById("products");

productList.innerHTML = "";

var option = null;

for(var i = 0 ; i < responseList.length; i++){

option = document.createElement("option");

option.appendChild(document.createTextNode(responseList[i].firstChild.nodeValue));

option.setAttribute("value",responseList[i].firstChild.nodeValue);

productList.appendChild(option);

}

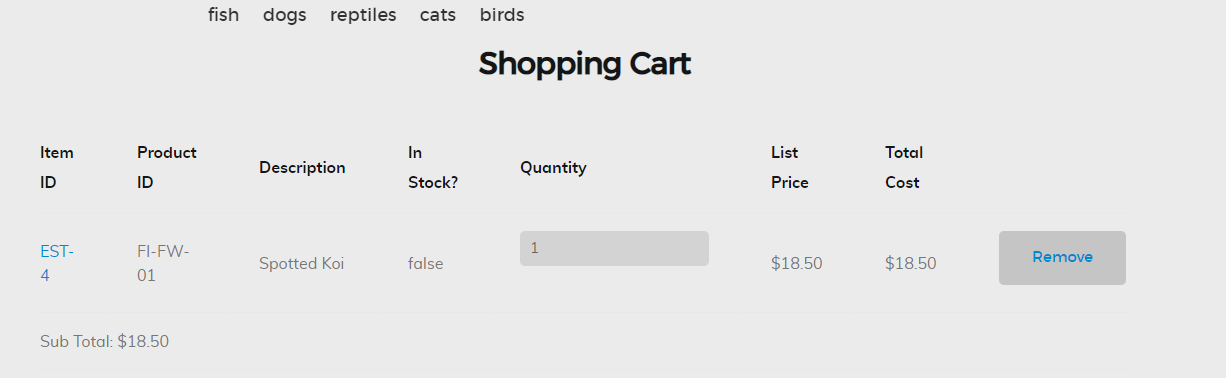
}

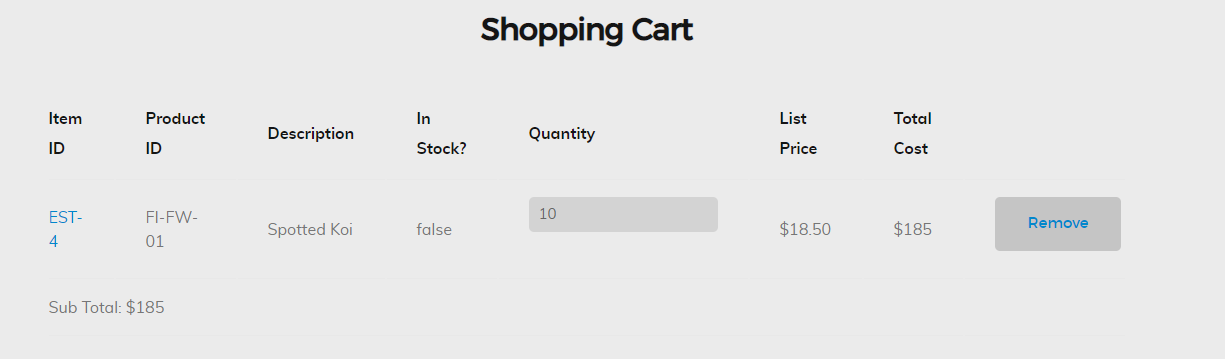
}

}

1. 购物车模块。在购物车模块中修改购买数量时，通过AJAX修改购物车信息。

当有货物加入到购物车时，可以看到商品的总价为18.5

当商品的数量为10时，可以看到商品的总价变成了185



核心代码：

package org.csu.mypetstore.web.servlets;

import org.csu.mypetstore.domain.Account;

import org.csu.mypetstore.domain.Cart;

import org.csu.mypetstore.service.CatalogService;

import javax.servlet.ServletException;

import javax.servlet.http.HttpServlet;

import javax.servlet.http.HttpServletRequest;

import javax.servlet.http.HttpServletResponse;

import javax.servlet.http.HttpSession;

import java.io.IOException;

import java.io.PrintWriter;

public class AutoRefreshCartServlet extends HttpServlet {

int quantity;

String itemId;

Account account;

Cart cart;

private CatalogService catalogService;

protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {

doGet(request,response);

}

protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {

HttpSession session = request.getSession();

cart = (Cart) session.getAttribute("cart");

account = (Account) session.getAttribute("account");

itemId = request.getParameter("itemId");

quantity = Integer.parseInt(request.getParameter("quantity"));

cart.setQuantityByItemId(itemId,quantity);

catalogService = new CatalogService();

catalogService.updateCart(cart,account);

session.setAttribute("cart",cart);

response.setContentType("text/xml");

PrintWriter out = response.getWriter();

out.println("<response>");

out.println("<msg>Update successfully</msg>");

out.println("<SubTotal>" + cart.getSubTotal() + "</SubTotal>");

out.println("</response>");

out.flush();

out.close();

}

}

Js代码：

var xmlHttpRequest;

var getListPrice;

var quantity;

var getItemId;

function createXMLHttpRequest()

{

if (window.XMLHttpRequest) //非IE浏览器

{

xmlHttpRequest = new XMLHttpRequest();

}

else if (window.ActiveObject)//IE6以上版本的IE浏览器

{

xmlHttpRequest = new ActiveObject("Msxml2.XMLHTTP");

}

else //IE6及以下版本IE浏览器

{

xmlHttpRequest = new ActiveObject("Microsoft.XMLHTTP");

}

}

function refreshCart(itemId,listPrice) {

quantity = parseInt(document.getElementById("quantityOf" + itemId).value);

var total = document.getElementById("totalOf" + itemId);

total.innerHTML ="";

if (quantity > 0)

{

getListPrice = parseFloat(listPrice);

getItemId = itemId.toString();

sendRequest("autoRefreshCart?itemId=" + itemId +"&quantity=" + quantity);

}

else alert("Please enter a correct quantity!");

}

function sendRequest(url) {

createXMLHttpRequest();

xmlHttpRequest.open("GET", url, true);

xmlHttpRequest.onreadystatechange = processResponse;

xmlHttpRequest.send(null);

}

function processResponse() {

if (xmlHttpRequest.readyState == 4) {

if (xmlHttpRequest.status == 200) {

var msg = xmlHttpRequest.responseXML.getElementsByTagName("msg")[0].firstChild.data;

var getSubtotal = xmlHttpRequest.responseXML.getElementsByTagName("SubTotal")[0].firstChild.data;

if (msg == "Update successfully") {

var total = document.getElementById("totalOf" + getItemId);

total.innerHTML = toMoney(getListPrice \* quantity);

var SubTotal = document.getElementById("SubTotal");

SubTotal.innerHTML = "Sub Total: " + toMoney(parseFloat(getSubtotal));

}

}

}

}

function toMoney(num) {

num = num.toFixed(2);

num = parseFloat(num);

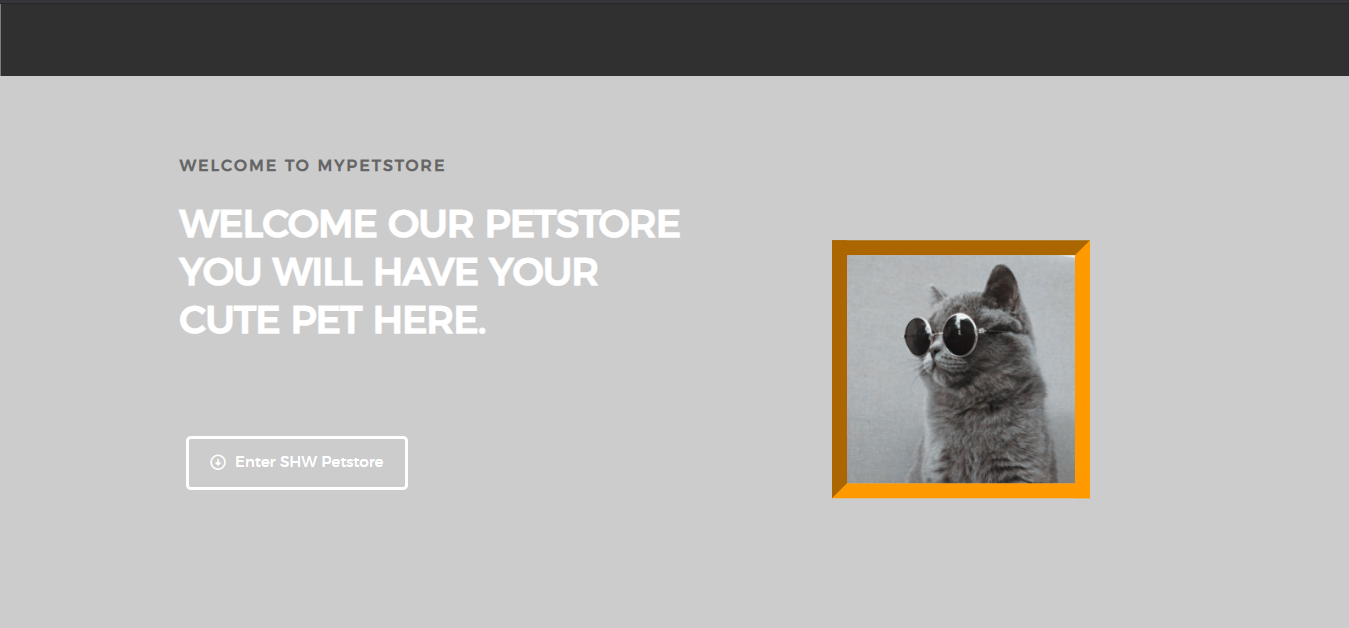
num = '$' + num.toLocaleString();

return num;

}

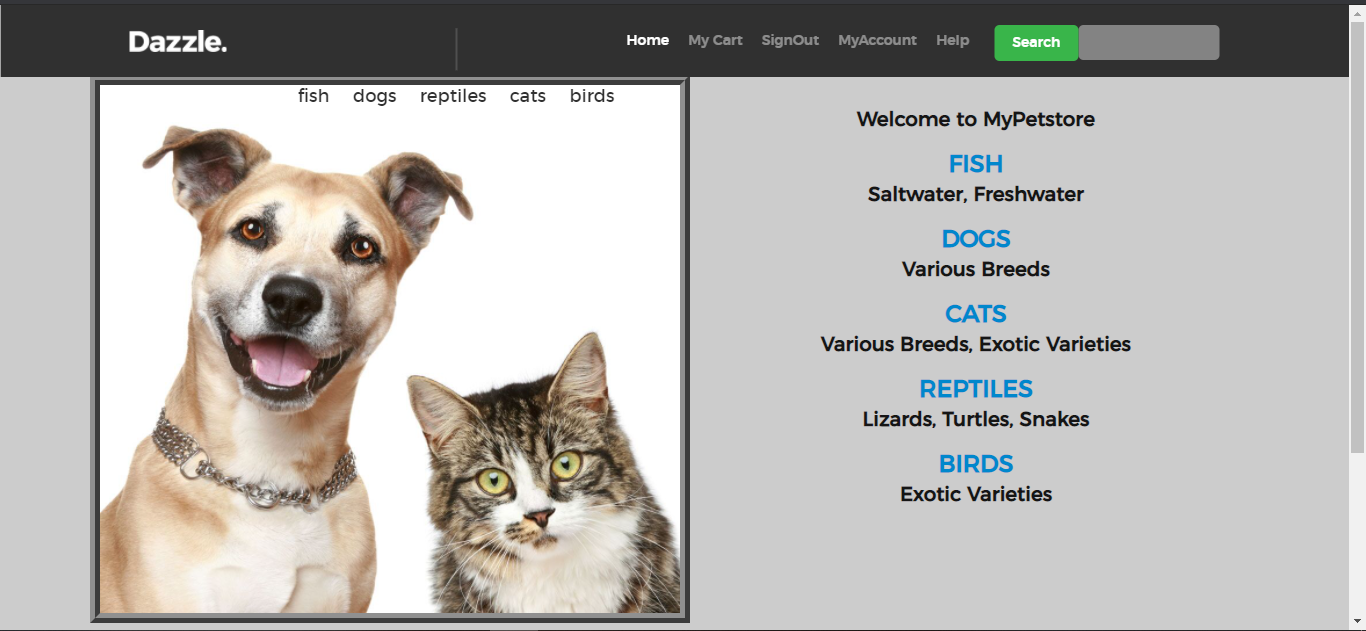
1. ui界面的设计

当刚进入界面时可以看到如下的界面

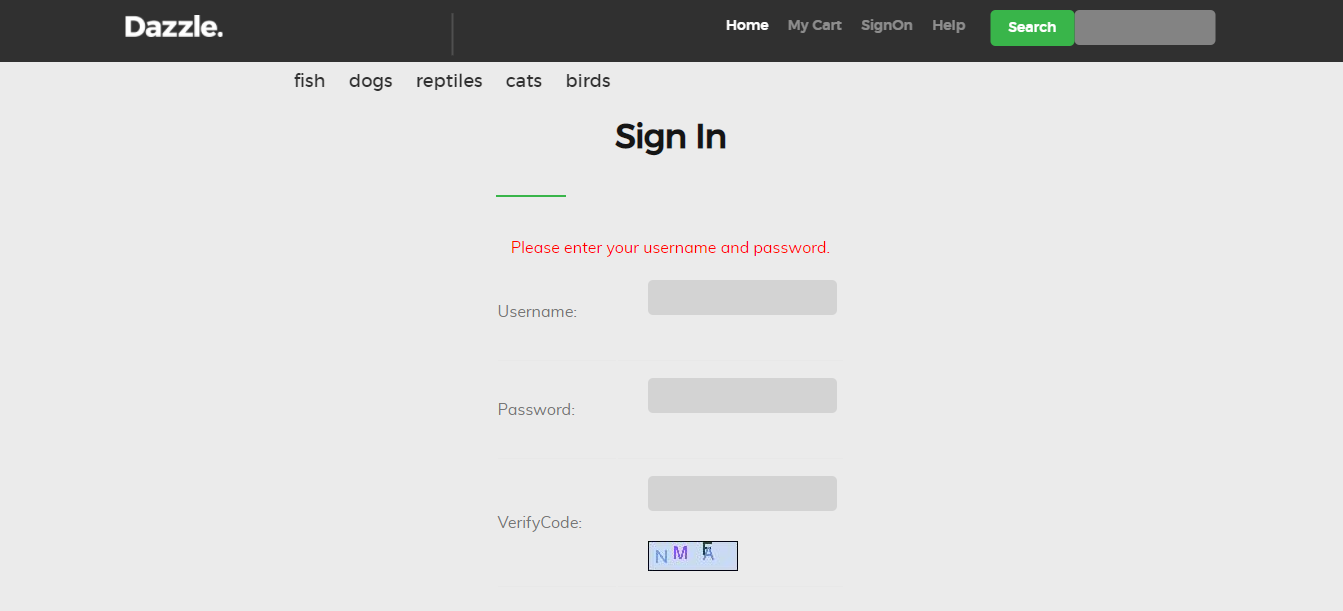
****

**（刚进入界面可以看到的界面）**

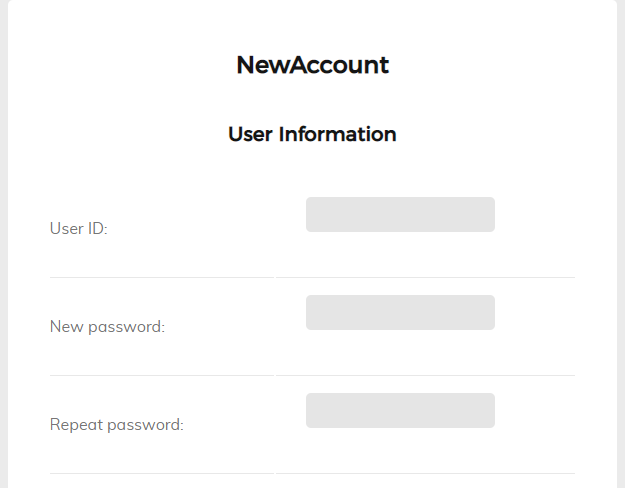
**进入商店后看到的界面：**

**（主界面）**

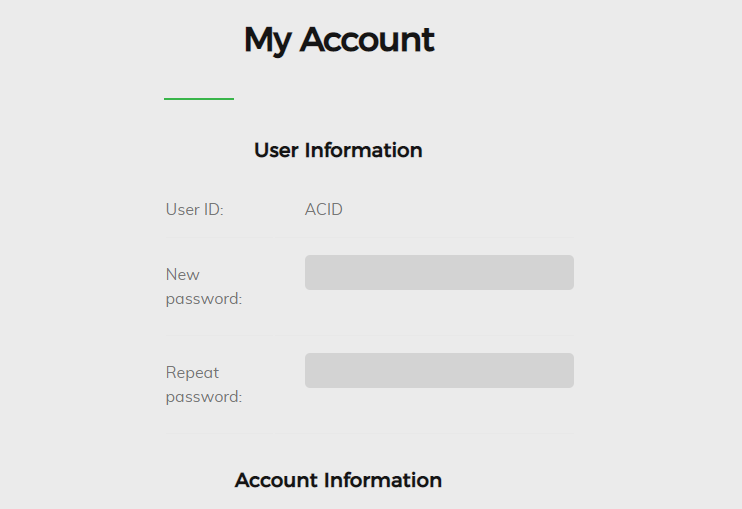
**登陆界面：**

**（登陆界面）**

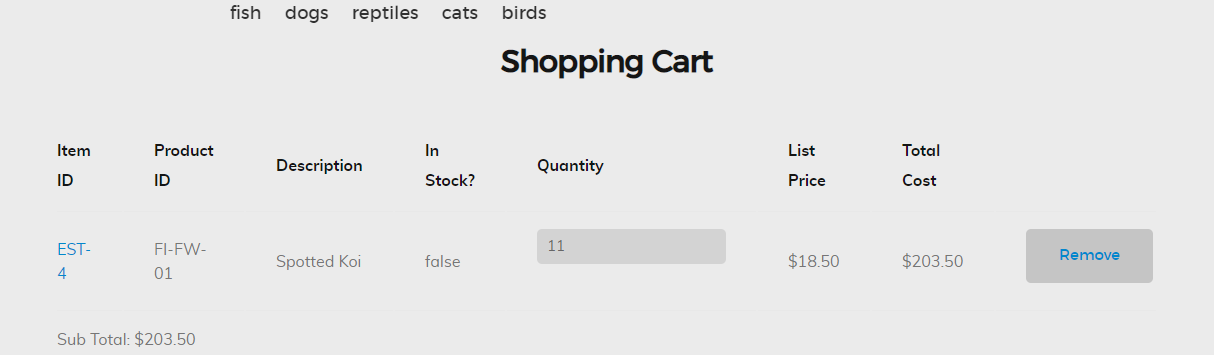
**注册界面：**

 **（注册界面的一部分）**

**账户信息**

 **（账户信息的一部分）**

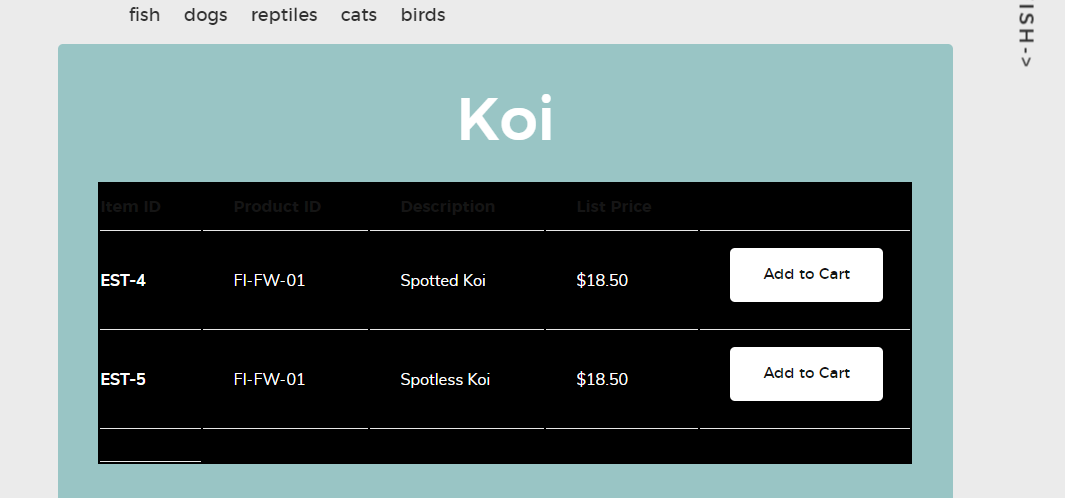
**购物车界面：**



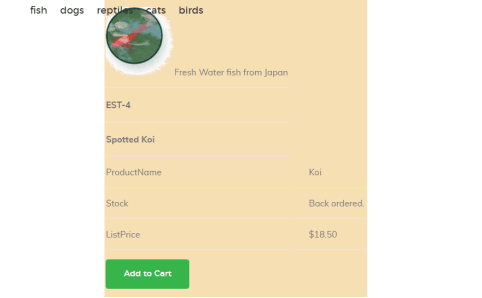
**（购物车模块）**

**商品界面**

****当点击鱼时：

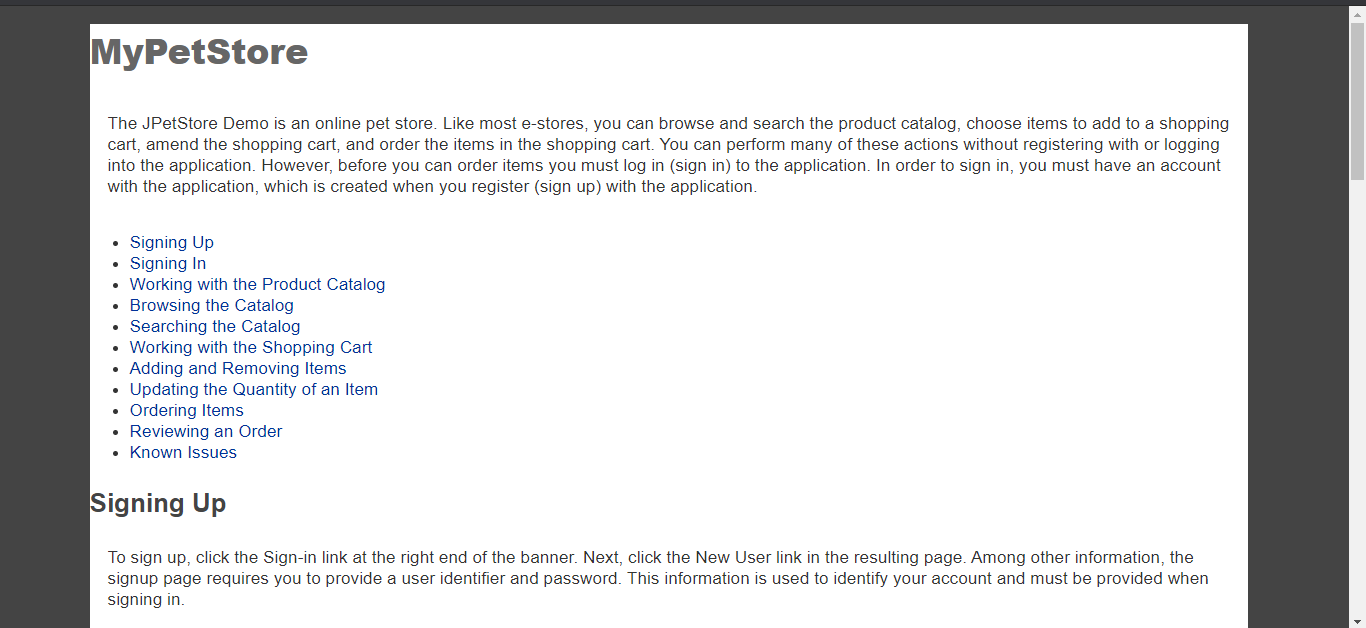
 当点击koi时

当点击est-4时



1. **Help文档**

**作用：**使用户更容易的去使用我们的宠物商店，比如注册，登录，加入购物车等是一个很有用的模块



package org.csu.mypetstore.web.servlets;

import javax.servlet.ServletException;

import javax.servlet.http.HttpServlet;

import javax.servlet.http.HttpServletRequest;

import javax.servlet.http.HttpServletResponse;

import java.io.IOException;

public class HelpServlet extends HttpServlet {

private static final String Help = "/WEB-INF/jsp/common/Help.jsp";

protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {

doGet(request,response);

}

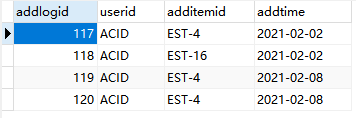
protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {

request.getRequestDispatcher(Help).forward(request,response);

}

}

1. 后台购物数据，方便购物数据的统计



**三、小组成员分工及实验总结**

1. 小组分工：

史惠文是组长，负责全部

1. 实验总结

After this test technology experiment, I personally got a lot of harvest, on the one hand, deepened my understanding of the textbook theory, on the other hand, also improved the experimental operation ability. Now I have summed up the following experience.

This experiment is different from our previous experiments, because I think I really finished it myself this time. So I think this experiment is the most valuable and profound. That is, the process of the experiment is all done by our students themselves. In this way, we must understand the principle of the experiment. Here, I deeply realize the guiding role of philosophical theory in practice: to understand the principle of experiment, and realize that the operation ability of experiment can be improved only by doing it by oneself, thinking by oneself, and consulting others by oneself.

In the process of experiment, we should cultivate our ability to analyze and solve problems independently. The premise of developing this ability is your attitude towards each experiment. If you are very casual in this aspect of the experiment, holding and waiting for the teacher to teach you how to do it, and copying the students' reports, although your grades will be very high, it is not good for your future work. For example, when I make UI interface, I first go to the network to find the template, and then innovate step by step, which is a good exercise for me.

This time I also benefited a lot. First of all, I used Ajax technology and jQuery technology, and then further implemented the function. There are account management module, commodity query module, shopping cart module, UI module and help module. So that my use of the framework has further improved. What’s more, JS, HTML and CSS have been further improved.