中南大学

Web应用开发技术

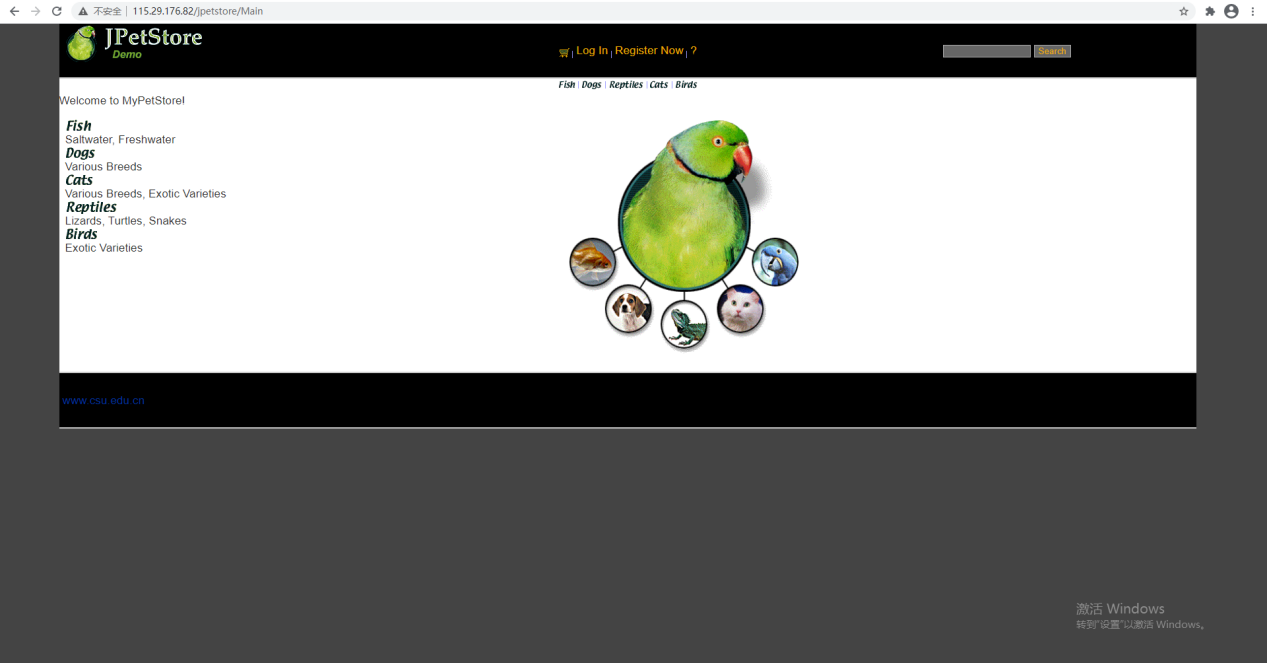
实验报告

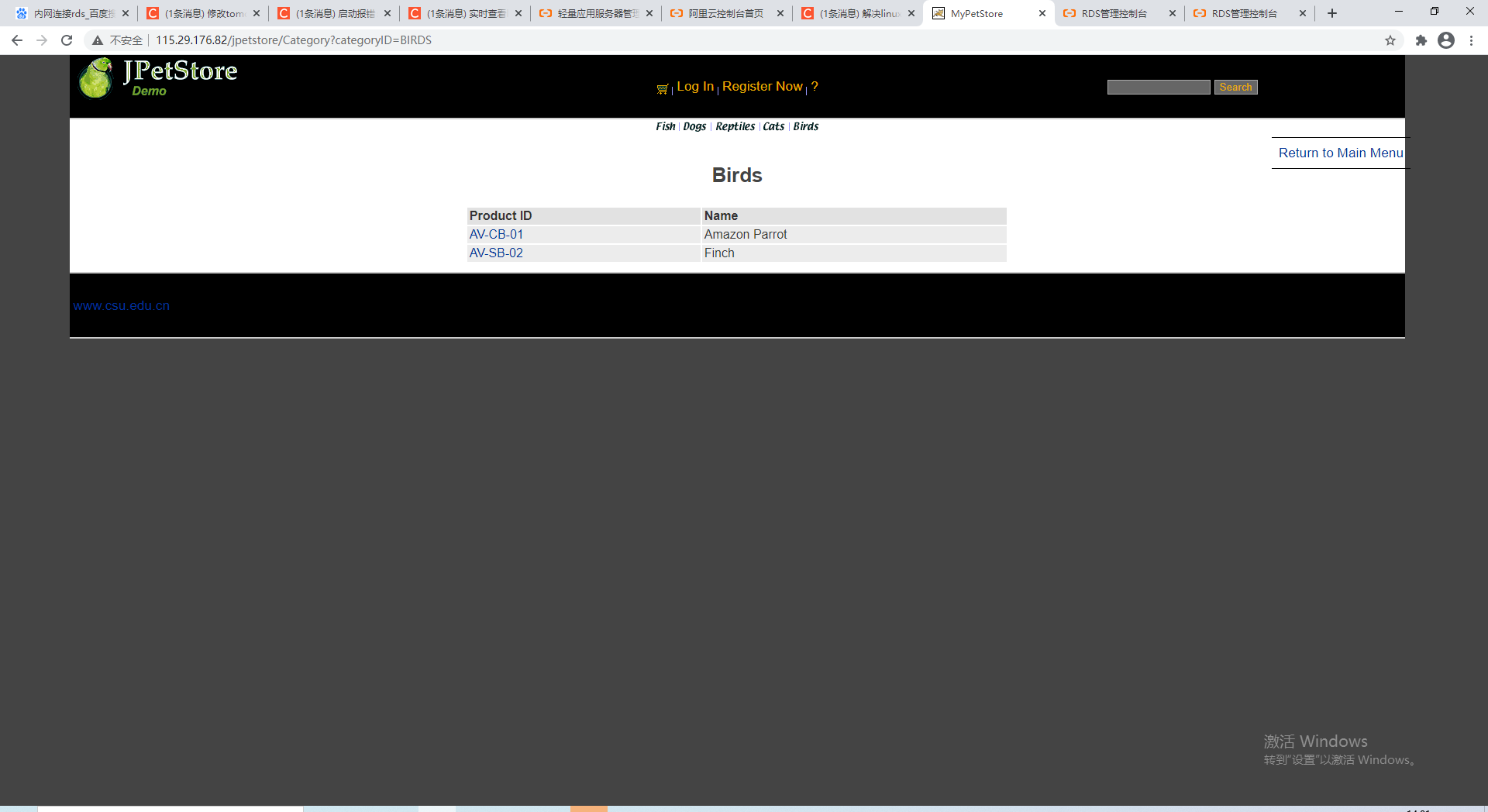
|  |  |
| --- | --- |
| 姓 名： | 陶行云 |
| 学 号： | 8204190405 |
| 班 级： | 软件工程1906 |
| 指导老师： | 宋 铁 |
| 完成时间： | 2022 1.12 |
| 分 数： |  |
| 批阅日期： |  |

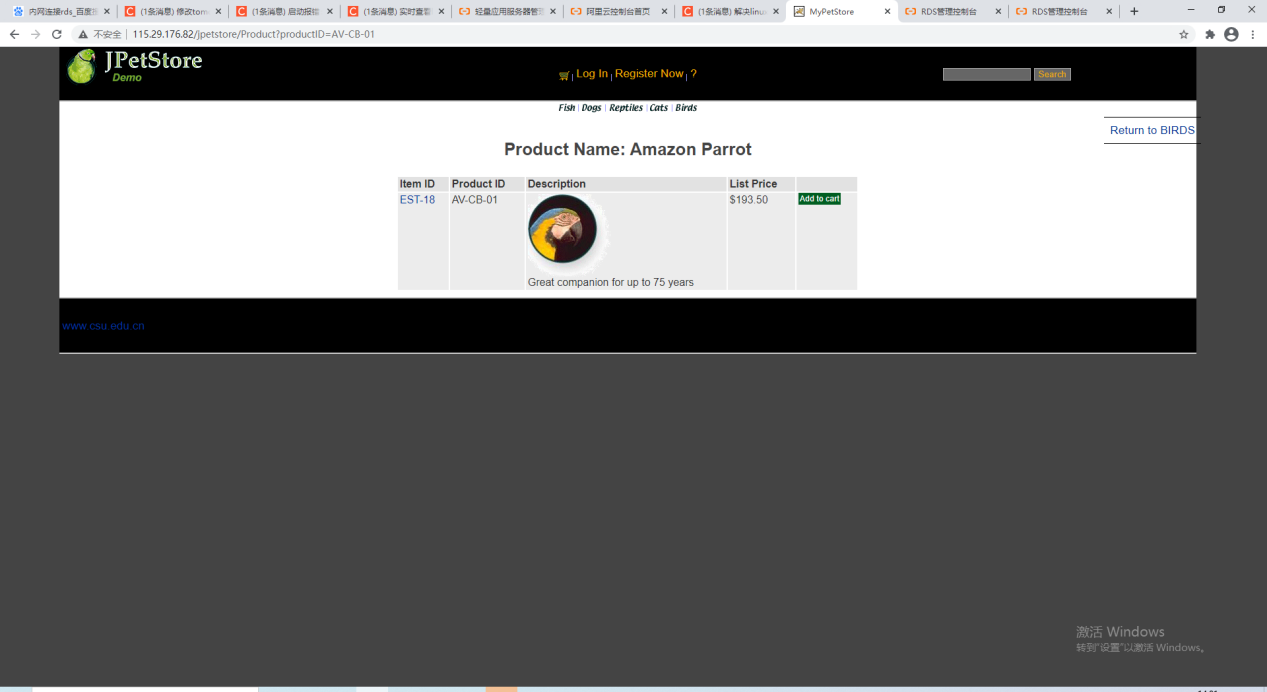
**实验一 基于MVC用JSP/Servlet实现JPetStore**

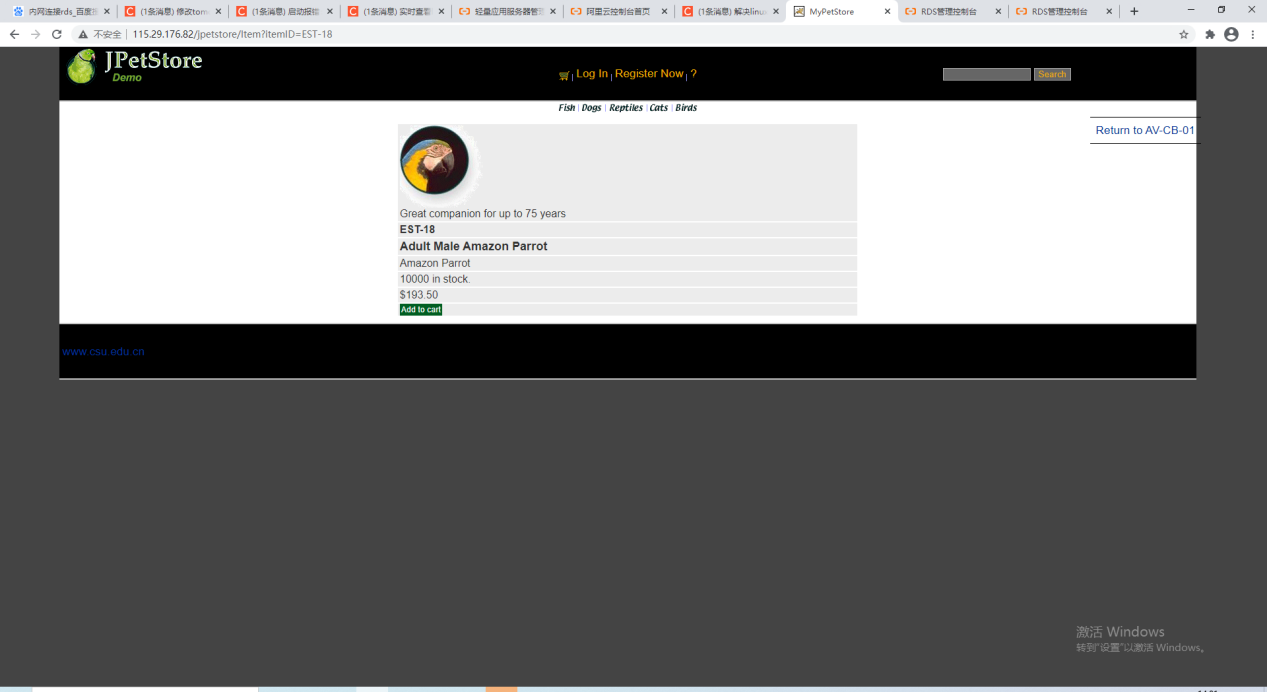
1. **实验内容——基本任务。**
2. 商品展示业务模块，包括大类Category、小类Product和具体商品Item的展示和搜索功能。

1.1 效果展示截图：









1.2 核心代码

**//ItemServlet.java  
@WebServlet(name = ServletNameConstant.ITEM\_SERVLET\_NAME, urlPatterns = {ServletUrlConstant.ITEM\_SERVLET\_URL})  
public class ItemServlet extends HttpServlet {  
 private final CatalogService catalogService = new CatalogService();  
 public static final String ITEM\_ID\_TO\_SHOW = "itemID";  
  
 @Override  
 protected void doGet(HttpServletRequest req, HttpServletResponse resp) throws ServletException, IOException {  
 String itemId = req.getParameter(ITEM\_ID\_TO\_SHOW);  
 Item item = catalogService.getItem(itemId);  
 req.getSession().setAttribute(SessionAttributeKeyConstant.ITEM\_TO\_SHOW, item);  
 req.getRequestDispatcher(WebPageUrlConstant.ITEM\_PAGE\_URL).forward(req, resp);  
 }  
  
 @Override  
 protected void doPost(HttpServletRequest req, HttpServletResponse resp) throws ServletException, IOException {  
 doGet(req, resp);  
 }  
}**

**//CatelogService.java**

**public class CatalogService {  
 private final CategoryDAO categoryDAO;  
 private final ProductDAO productDAO;  
 private final ItemDAO itemDAO;  
  
 //private ProductMapper productMapper;  
 public CatalogService() {  
 categoryDAO = new CategoryDAOImpl();  
 productDAO = new ProductDAOImpl();  
 itemDAO = new ItemDAOImpl();  
 }  
  
  
 public List<Category> getCategoryList() {  
 return categoryDAO.getCategoryList();  
 }  
  
  
 public Category getCategory(String categoryId) {  
 return categoryDAO.getCategory(categoryId);  
 }  
  
  
 public Product getProduct(String productId) {  
 return productDAO.getProduct(productId);  
 }  
  
  
 public List<Product> getProductListByCategory(String categoryId) {  
 return productDAO.getProductList(categoryId);  
 }  
  
  
 public boolean categoryIDExists(String categoryID) {  
 return categoryDAO.getCategoryFromId(categoryID) != null;  
 }  
  
  
 public boolean productIDExists(String productID) {  
 return productDAO.getProduct(productID) != null;  
 }  
  
  
 public List<Item> getItemListByProduct(String productId) {  
 return itemDAO.getItemListByProduct(productId);  
 }  
  
  
 public Item getItem(String itemId) {  
 return itemDAO.getItem(itemId);  
 }  
  
  
 public List<Product> searchProductByKeyword(String keyword) {return productDAO.searchProductList(keyword);}  
  
  
 public boolean isItemInStock(String itemId) {  
 return itemDAO.getInventoryQuantity(itemId) > 0;  
 }  
}**

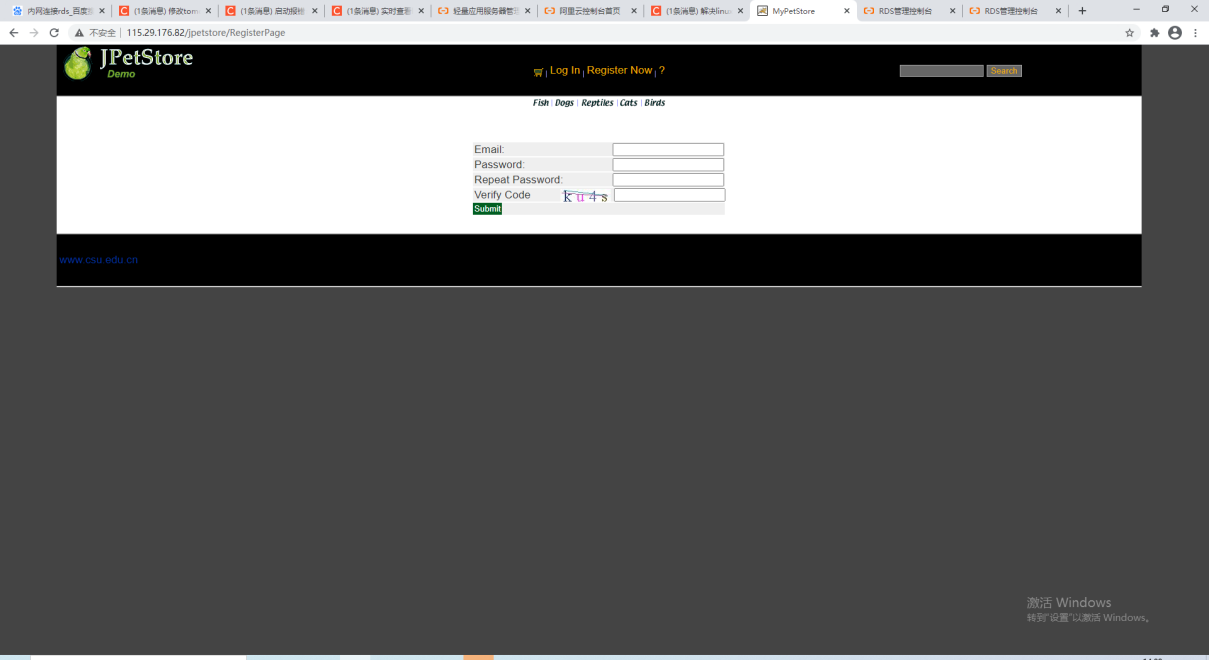
**//ItemDaoImpl.java 部分代码**

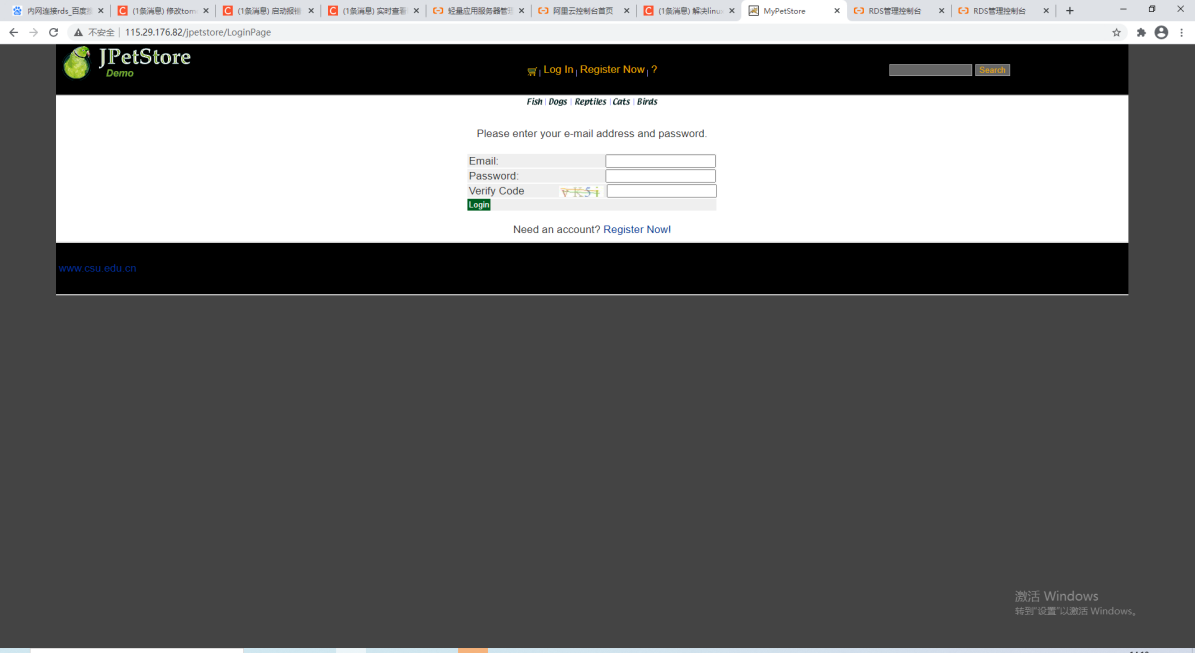
**private static final String SELECT\_ITEM\_LIST\_BY\_PRODUCTID = "SELECT\n" +  
 " I.itemid," +  
 " listprice," +  
 " unitcost," +  
 " supplier AS supplierId," +  
 " IV.qty AS \"inventory.quintity\"," +  
 " P.productid AS \"product.productId\"," +  
 " name AS \"product.name\"," +  
 " descn AS \"product.description\"," +  
 " category AS \"product.categoryId\"," +  
 " STATUS," +  
 " ATTR1 AS attribute1," +  
 " ATTR2 AS attribute2," +  
 " ATTR3 AS attribute3," +  
 " ATTR4 AS attribute4," +  
 " ATTR5 AS attribute5" +  
 " FROM item I, product P, inventory IV" +  
 " WHERE P.productid = I.productid" +  
 " AND I.productid =?" +  
 " AND I.itemid=IV.itemid;";**

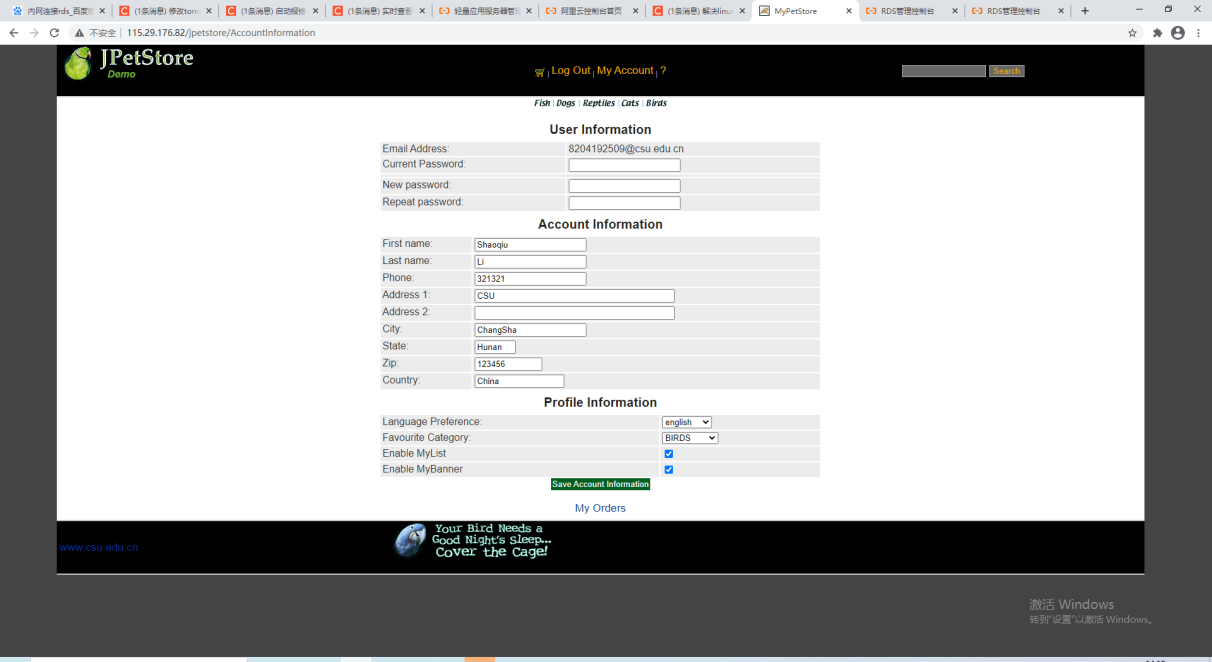
**/\*\*  
 \* @author MokuSakura  
 \*/  
@Override  
public List<Item> getItemListByProduct(String productId) {  
 List<Item> res = null;  
 try {  
 Connection connection = DBUtil.getConnection();  
 PreparedStatement preparedStatement = connection.prepareStatement(SELECT\_ITEM\_LIST\_BY\_PRODUCTID);  
 preparedStatement.setString(1, productId);  
 ResultSet resultSet = preparedStatement.executeQuery();  
 res = new ArrayList<>();  
 while (resultSet.next()) {  
 Item item = new Item();  
 setItemPropertyFromResultSet(item, resultSet);  
 res.add(item);  
 }  
 DBUtil.closeResultSet(resultSet);  
 DBUtil.closePreparedStatement(preparedStatement);  
 DBUtil.closeConnection(connection);  
 } catch (Exception e) {  
 e.printStackTrace();  
 }  
 return res;  
}**

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。

2.1效果截图：







**//AccountInformationServlet.java**

**public class ChangeAccountInformationServlet extends HttpServlet {  
 private final UserService userService = new UserService();  
 public static final String[] PARAMETER\_KEY = {  
 "newPassword", "bannerOption", "listOption", "favouriteCategoryId", "languagePreference", "phone",  
 "country", "zip", "state", "city", "address2", "address1", "firstName", "lastName", "currentPassword"};  
 public static final String BANNER\_OPTION\_SELECTED\_STRING = "on";  
 public static final String LIST\_OPTION\_SELECTED\_STRING = "on";  
  
 @Override  
 protected void doPost(HttpServletRequest req, HttpServletResponse resp) throws ServletException, IOException {  
 User user = (User) req.getSession().getAttribute(SessionAttributeKeyConstant.LOGIN\_USER);  
 int index = 0;  
 user.setPassword(req.getParameter(PARAMETER\_KEY[index++]));  
 user.setBannerOption(BANNER\_OPTION\_SELECTED\_STRING.equals(req.getParameter(PARAMETER\_KEY[index++]))); user.setListOption(LIST\_OPTION\_SELECTED\_STRING.equals(req.getParameter(PARAMETER\_KEY[index++]))); user.setFavouriteCategoryId(req.getParameter(PARAMETER\_KEY[index++]));  
user.setLanguagePreference(req.getParameter(PARAMETER\_KEY[index++]));  
 user.setPhone(req.getParameter(PARAMETER\_KEY[index++]));  
 user.setCountry(req.getParameter(PARAMETER\_KEY[index++]));  
 user.setZip(req.getParameter(PARAMETER\_KEY[index++]));  
 user.setState(req.getParameter(PARAMETER\_KEY[index++]));  
 user.setCity(req.getParameter(PARAMETER\_KEY[index++]));  
 user.setAddress2(req.getParameter(PARAMETER\_KEY[index++]));  
 user.setAddress1(req.getParameter(PARAMETER\_KEY[index++]));  
 user.setFirstName(req.getParameter(PARAMETER\_KEY[index++]));  
 user.setLastName(req.getParameter(PARAMETER\_KEY[index++]));  
 user = userService.update(user, req.getParameter(PARAMETER\_KEY[index]));  
 if (user == null) {  
 req.setAttribute(RequestAttributeKeyConstant.UPDATE\_ACCOUNT\_INFORMATION\_MESSAGE, "Illegal Password");  
 } else {  
 req.getSession().setAttribute(SessionAttributeKeyConstant.LOGIN\_USER, user); req.setAttribute(RequestAttributeKeyConstant.UPDATE\_ACCOUNT\_INFORMATION\_MESSAGE, "Update Success");  
 } req.gtRequestDispatcher(WebPageUrlConstant.ACCOUNT\_INFORMATION\_PAGE\_URL).forward(req, resp);  
 }  
  
}**

**//UserService.java部分代码**

**public User update(User user, String currentPassword) {  
 if (user.getPassword().equals("")) {  
 user.setPassword(currentPassword);  
 }  
 //else if(!RegularExpressionVerifyUtil.isLegalPassword(user.getPassword()))  
 //return null;  
 user = userDao.updateUserInfo(user);  
 if (user != null) {  
 user = userDao.getUserByEmail(user.getEmail());  
 user.setCart(cartDAO.getUserCart(user));  
 user.setPassword(null);  
 }  
 return user;  
}**

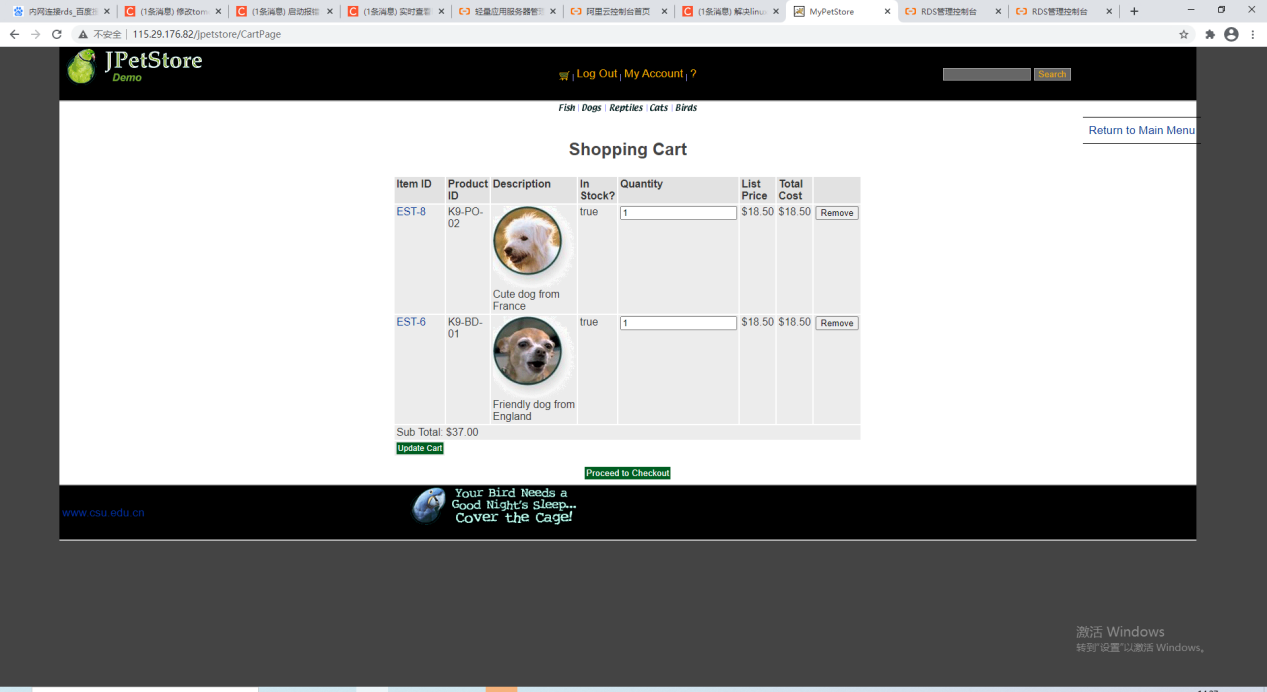
**//UserDaoImpl.java部分代码**

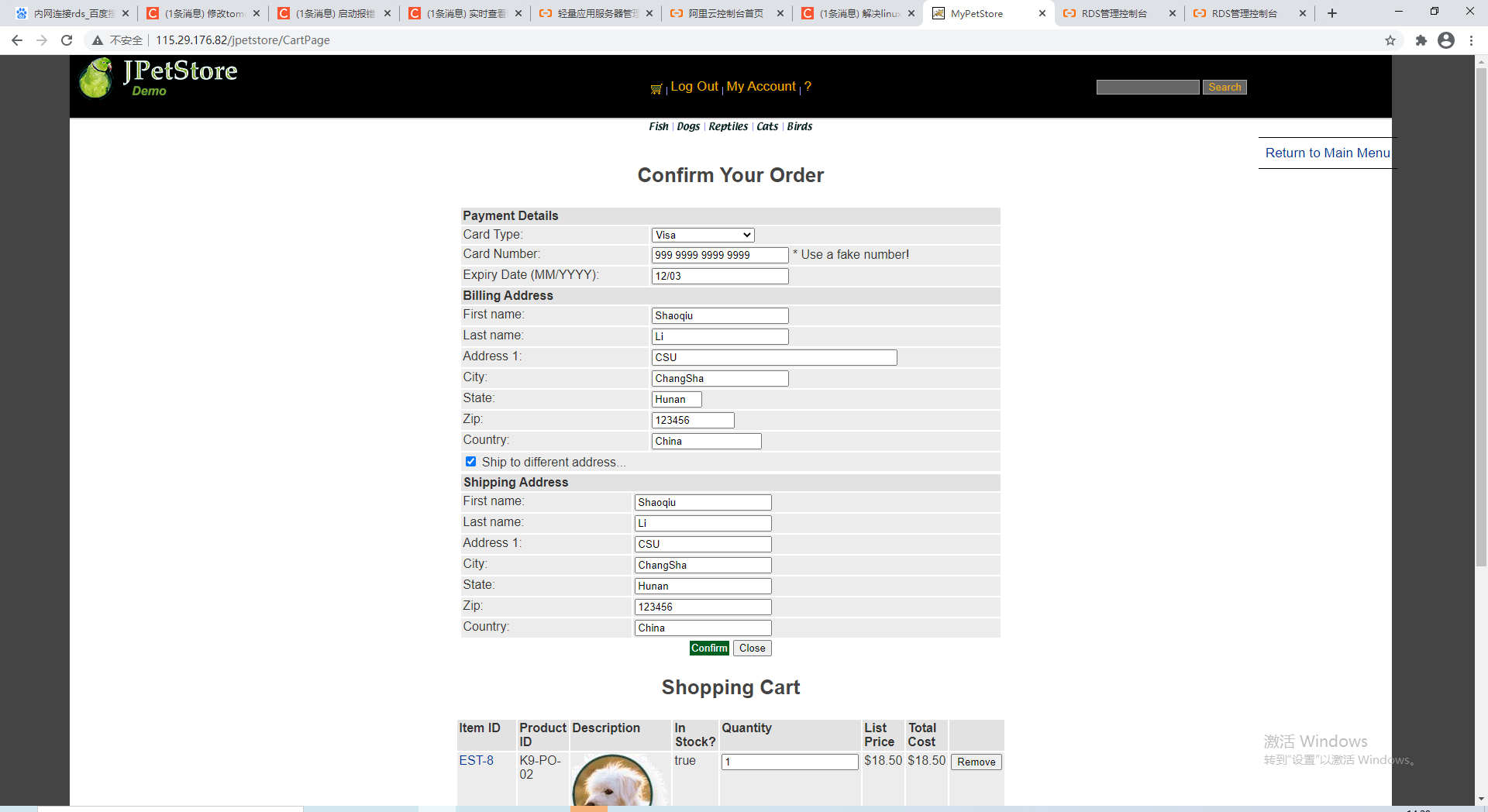
**private static final String UPDATE\_LOGINUSER = "UPDATE loginuser SET password = ? WHERE email = ?";  
private static final String UPDATE\_USERFILE = "UPDATE userfile SET " +  
 "langpref = ?,favcategory = ?,mylistopt=?,banneropt=? " +  
 "WHERE email = ?";  
private static final String UPDATE\_ACCOUNT = "UPDATE account SET" +  
 " FIRSTNAME = ?," +  
 " LASTNAME = ?," +  
 " STATUS = ?," +  
 " ADDR1 = ?," +  
 " ADDR2 = ?," +  
 " CITY = ?," +  
 " STATE = ?," +  
 " ZIP = ?," +  
 " COUNTRY = ?," +  
 " PHONE = ?" +  
 " WHERE email = ?";**

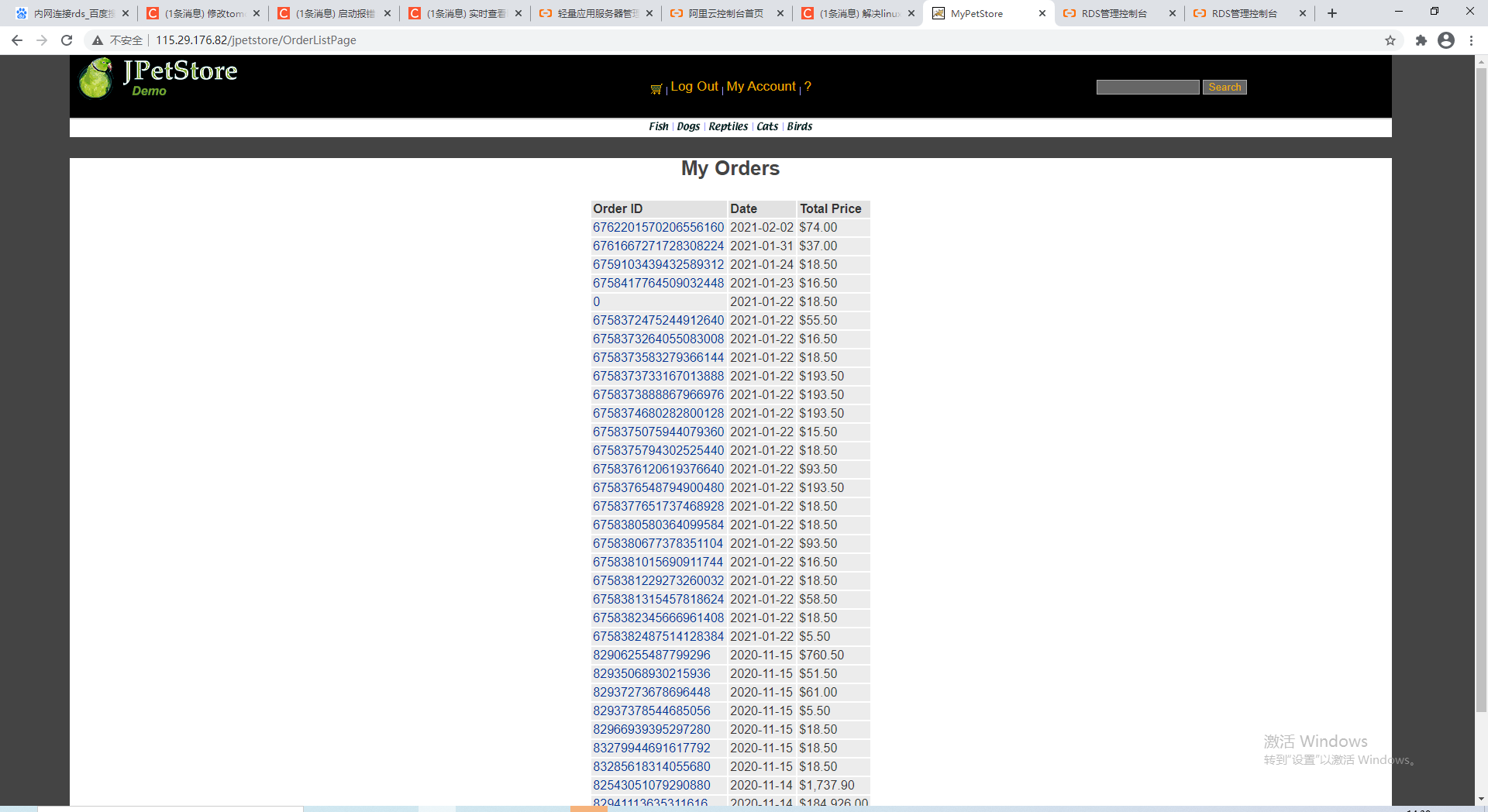
**@Override  
public User updateUserInfo(User user) {  
 User res = null;  
 try {  
 Connection connection = DBUtil.getConnection();  
 PreparedStatement loginPreparedStatement = connection.prepareStatement(UPDATE\_LOGINUSER);  
 PreparedStatement accountPreparedStatement = connection.prepareStatement(UPDATE\_ACCOUNT);  
 PreparedStatement userfilePreparedStatement = connection.prepareStatement(UPDATE\_USERFILE);  
 loginPreparedStatement.setString(1, user.getPassword());  
 loginPreparedStatement.setString(2, user.getEmail());  
 loginPreparedStatement.execute();  
 DBUtil.closePreparedStatement(loginPreparedStatement);  
 userfilePreparedStatement.setString(1, user.getLanguagePreference());  
 userfilePreparedStatement.setString(2, user.getFavouriteCategoryId());  
 userfilePreparedStatement.setString(3, user.getListOption() ? "1" : "0");  
 userfilePreparedStatement.setString(4, user.getBannerOption() ? "1" : "0");  
 userfilePreparedStatement.setString(5, user.getEmail());  
 userfilePreparedStatement.execute();  
 DBUtil.closePreparedStatement(userfilePreparedStatement);  
 int i = 1;  
 accountPreparedStatement.setString(i++, user.getFirstName());  
 accountPreparedStatement.setString(i++, user.getLastName());  
 accountPreparedStatement.setString(i++, user.getStatus());  
 accountPreparedStatement.setString(i++, user.getAddress1());  
 accountPreparedStatement.setString(i++, user.getAddress2());  
 accountPreparedStatement.setString(i++, user.getCity());  
 accountPreparedStatement.setString(i++, user.getState());  
 accountPreparedStatement.setString(i++, user.getZip());  
 accountPreparedStatement.setString(i++, user.getCountry());  
 accountPreparedStatement.setString(i++, user.getPhone());  
 accountPreparedStatement.setString(i++, user.getEmail());  
 accountPreparedStatement.execute();  
 DBUtil.closePreparedStatement(accountPreparedStatement);  
 DBUtil.closeConnection(connection);  
 res = user;  
 } catch (Exception e) {  
 e.printStackTrace();  
 }  
 return res;  
}**

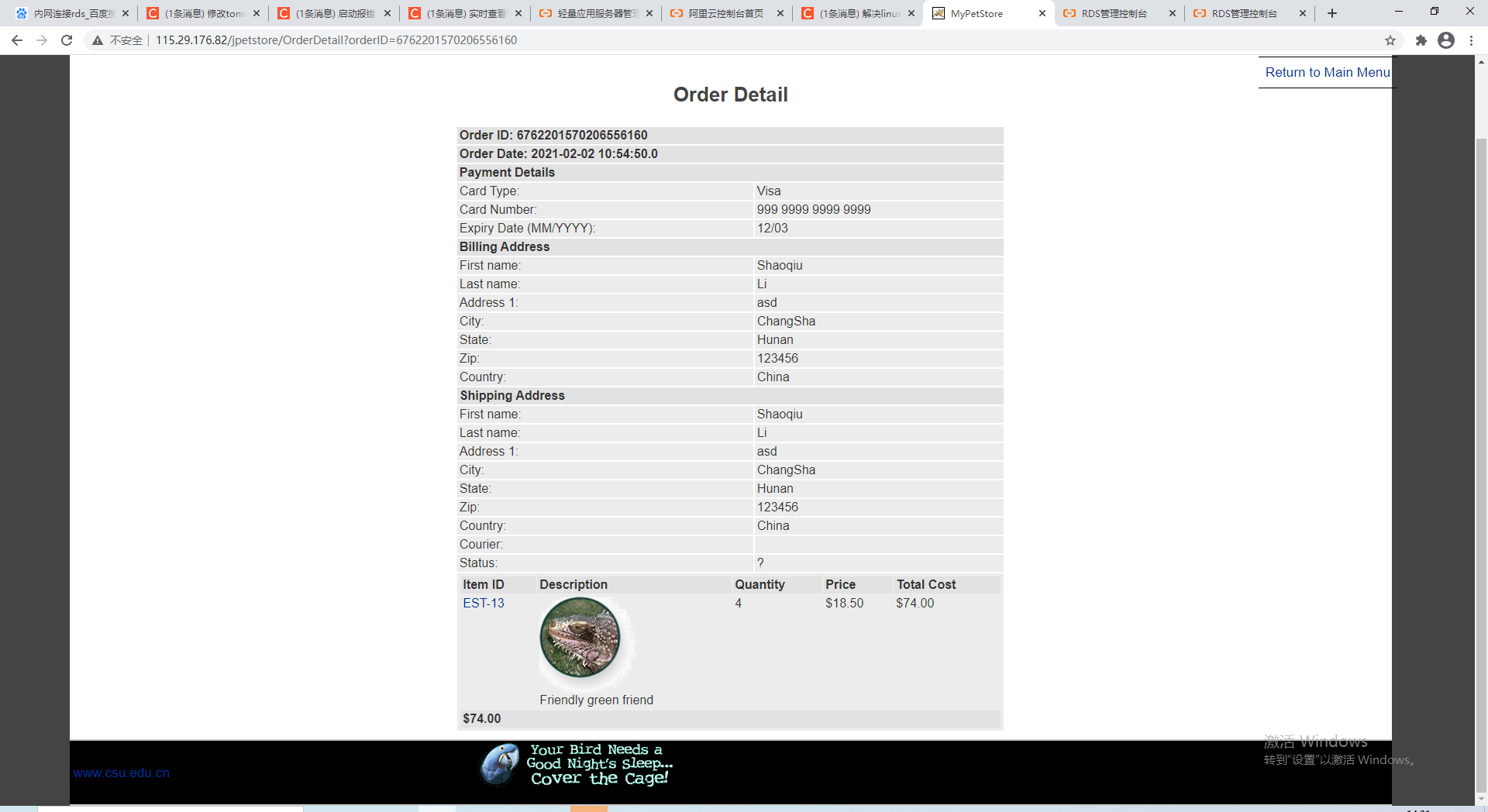
1. 订单管理模块，包括购物车管理、新增订单、地址信息等业务功能。

3.1 效果截图









**public void initOrder(Account account, Cart cart) {**

**// System.out.print("1111111111");**

**username = account.getUsername();**

**orderDate = new Date();**

**shipToFirstName = account.getFirstName();**

**//System.out.print(account.getFirstName());**

**shipToLastName = account.getLastName();**

**shipAddress1 = account.getAddress1();**

**shipAddress2 = account.getAddress2();**

**shipCity = account.getCity();**

**shipState = account.getState();**

**shipZip = account.getZip();**

**shipCountry = account.getCountry();**

**billToFirstName = account.getFirstName();**

**//System.out.print(billToFirstName);**

**billToLastName = account.getLastName();**

**billAddress1 = account.getAddress1();**

**billAddress2 = account.getAddress2();**

**billCity = account.getCity();**

**billState = account.getState();**

**billZip = account.getZip();**

**billCountry = account.getCountry();**

**totalPrice = cart.getSubTotal();**

**creditCard = "999 9999 9999 9999";**

**expiryDate = "12/03";**

**cardType = "Visa";**

**courier = "UPS";**

**locale = "CA";**

**status = "P";**

**Iterator<CartItem> i = cart.getAllCartItems();**

**while (i.hasNext()) {**

**CartItem cartItem = (CartItem) i.next();**

**addLineItem(cartItem);**

**}**

**}**

**public void addLineItem(CartItem cartItem) {**

**LineItem lineItem = new LineItem(lineItems.size() + 1, cartItem);**

**addLineItem(lineItem);**

**}**

**public void addLineItem(LineItem lineItem) {**

**lineItems.add(lineItem);**

**}**

**public boolean isShippingAddressRequired() {**

**return shippingAddressRequired;**

**}**

**public void setShippingAddressRequired(boolean shippingAddressRequired) {**

**this.shippingAddressRequired = shippingAddressRequired;**

**}**

**}**

**package org.csu.myjpetstore.domain;**

**import java.io.Serializable;**

**import java.math.BigDecimal;**

**import java.util.ArrayList;**

**import java.util.Date;**

**import java.util.Iterator;**

**import java.util.List;**

**public void initOrder(Account account, Cart cart) {**

**// System.out.print("1111111111");**

**username = account.getUsername();**

**orderDate = new Date();**

**shipToFirstName = account.getFirstName();**

**//System.out.print(account.getFirstName());**

**shipToLastName = account.getLastName();**

**shipAddress1 = account.getAddress1();**

**shipAddress2 = account.getAddress2();**

**shipCity = account.getCity();**

**shipState = account.getState();**

**shipZip = account.getZip();**

**shipCountry = account.getCountry();**

**billToFirstName = account.getFirstName();**

**//System.out.print(billToFirstName);**

**billToLastName = account.getLastName();**

**billAddress1 = account.getAddress1();**

**billAddress2 = account.getAddress2();**

**billCity = account.getCity();**

**billState = account.getState();**

**billZip = account.getZip();**

**billCountry = account.getCountry();**

**totalPrice = cart.getSubTotal();**

**creditCard = "999 9999 9999 9999";**

**expiryDate = "12/03";**

**cardType = "Visa";**

**courier = "UPS";**

**locale = "CA";**

**status = "P";**

**Iterator<CartItem> i = cart.getAllCartItems();**

**while (i.hasNext()) {**

**CartItem cartItem = (CartItem) i.next();**

**addLineItem(cartItem);**

**}**

**}**

**public void addLineItem(CartItem cartItem) {**

**LineItem lineItem = new LineItem(lineItems.size() + 1, cartItem);**

**addLineItem(lineItem);**

**}**

**public void addLineItem(LineItem lineItem) {**

**lineItems.add(lineItem);**

**}**

**public boolean isShippingAddressRequired() {**

**return shippingAddressRequired;**

**}**

**public void setShippingAddressRequired(boolean shippingAddressRequired) {**

**this.shippingAddressRequired = shippingAddressRequired;**

**}**

**}**

**package org.csu.myjpetstore.domain;**

**import java.io.Serializable;**

**import java.math.BigDecimal;**

**import java.util.ArrayList;**

**import java.util.Date;**

**import java.util.Iterator;**

**import java.util.List;**

**public void setBillZip(String billZip) {**

**this.billZip = billZip;**

**}**

**public String getBillCountry() {**

**return billCountry;**

**}**

**public void setBillCountry(String billCountry) {**

**this.billCountry = billCountry;**

**}**

**public String getCourier() {**

**return courier;**

**}**

**public void setCourier(String courier) {**

**this.courier = courier;**

**}**

**public BigDecimal getTotalPrice() {**

**return totalPrice;**

**}**

**public void setTotalPrice(BigDecimal totalPrice) {**

**this.totalPrice = totalPrice;**

**}**

**public String getBillToFirstName() {**

**return billToFirstName;**

**}**

**public void setBillToFirstName(String billToFirstName) {**

**this.billToFirstName = billToFirstName;**

**}**

**public String getBillToLastName() {**

**return billToLastName;**

**}**

**public void setBillToLastName(String billToLastName) {**

**this.billToLastName = billToLastName;**

**}**

**public String getShipToFirstName() {**

**return shipToFirstName;**

**}**

**public void setShipToFirstName(String shipFoFirstName) {**

**this.shipToFirstName = shipFoFirstName;**

**}**

**public String getShipToLastName() {**

**return shipToLastName;**

**}**

**public void setShipToLastName(String shipToLastName) {**

**this.shipToLastName = shipToLastName;**

**}**

**public String getCreditCard() {**

**return creditCard;**

**}**

**public void setCreditCard(String creditCard) {**

**this.creditCard = creditCard;**

**}**

**public String getExpiryDate() {**

**return expiryDate;**

**}**

**public void setExpiryDate(String expiryDate) {**

**this.expiryDate = expiryDate;**

**}**

**public String getCardType() {**

**return cardType;**

**}**

**public void setCardType(String cardType) {**

**this.cardType = cardType;**

**}**

**public String getLocale() {**

**return locale;**

**}**

**public void setLocale(String locale) {**

**this.locale = locale;**

**}**

**public String getStatus() {**

**return status;**

**}**

**public void setStatus(String status) {**

**this.status = status;**

**}**

**public void setLineItems(List<LineItem> lineItems) {**

**this.lineItems = lineItems;**

**}**

**public List<LineItem> getLineItems() {**

**return lineItems;**

**}**

**public void initOrder(Account account, Cart cart) {**

**// System.out.print("1111111111");**

**username = account.getUsername();**

**orderDate = new Date();**

**shipToFirstName = account.getFirstName();**

**//System.out.print(account.getFirstName());**

**shipToLastName = account.getLastName();**

**shipAddress1 = account.getAddress1();**

**shipAddress2 = account.getAddress2();**

**shipCity = account.getCity();**

**shipState = account.getState();**

**shipZip = account.getZip();**

**shipCountry = account.getCountry();**

**billToFirstName = account.getFirstName();**

**//System.out.print(billToFirstName);**

**billToLastName = account.getLastName();**

**billAddress1 = account.getAddress1();**

**billAddress2 = account.getAddress2();**

**billCity = account.getCity();**

**billState = account.getState();**

**billZip = account.getZip();**

**billCountry = account.getCountry();**

**totalPrice = cart.getSubTotal();**

**creditCard = "999 9999 9999 9999";**

**expiryDate = "12/03";**

**cardType = "Visa";**

**courier = "UPS";**

**locale = "CA";**

**status = "P";**

**Iterator<CartItem> i = cart.getAllCartItems();**

**while (i.hasNext()) {**

**CartItem cartItem = (CartItem) i.next();**

**addLineItem(cartItem);**

**}**

**}**

**public void addLineItem(CartItem cartItem) {**

**LineItem lineItem = new LineItem(lineItems.size() + 1, cartItem);**

**addLineItem(lineItem);**

**}**

**public void addLineItem(LineItem lineItem) {**

**lineItems.add(lineItem);**

**}**

**public boolean isShippingAddressRequired() {**

**return shippingAddressRequired;**

**}**

**public void setShippingAddressRequired(boolean shippingAddressRequired) {**

**this.shippingAddressRequired = shippingAddressRequired;**

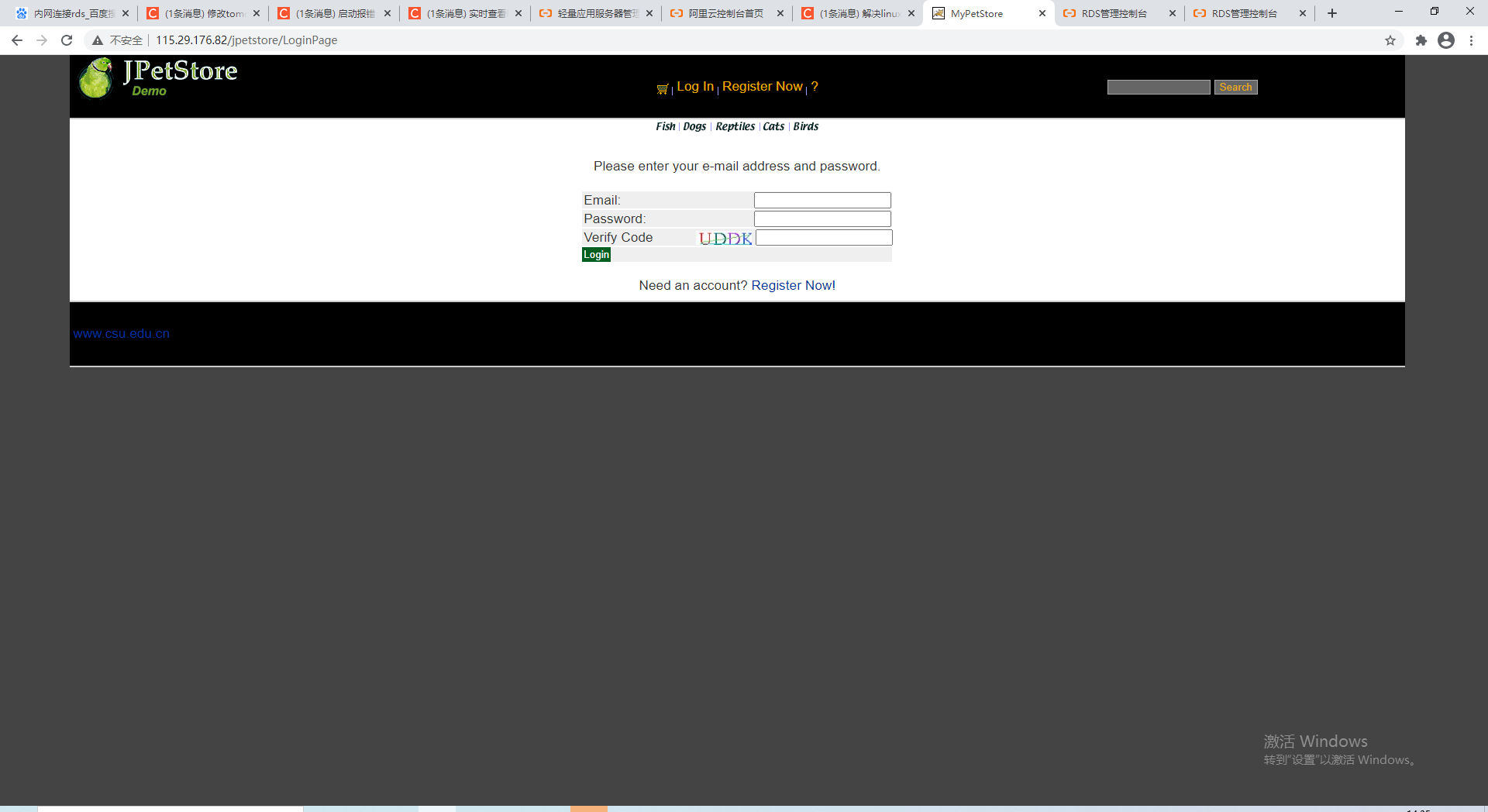
**}**

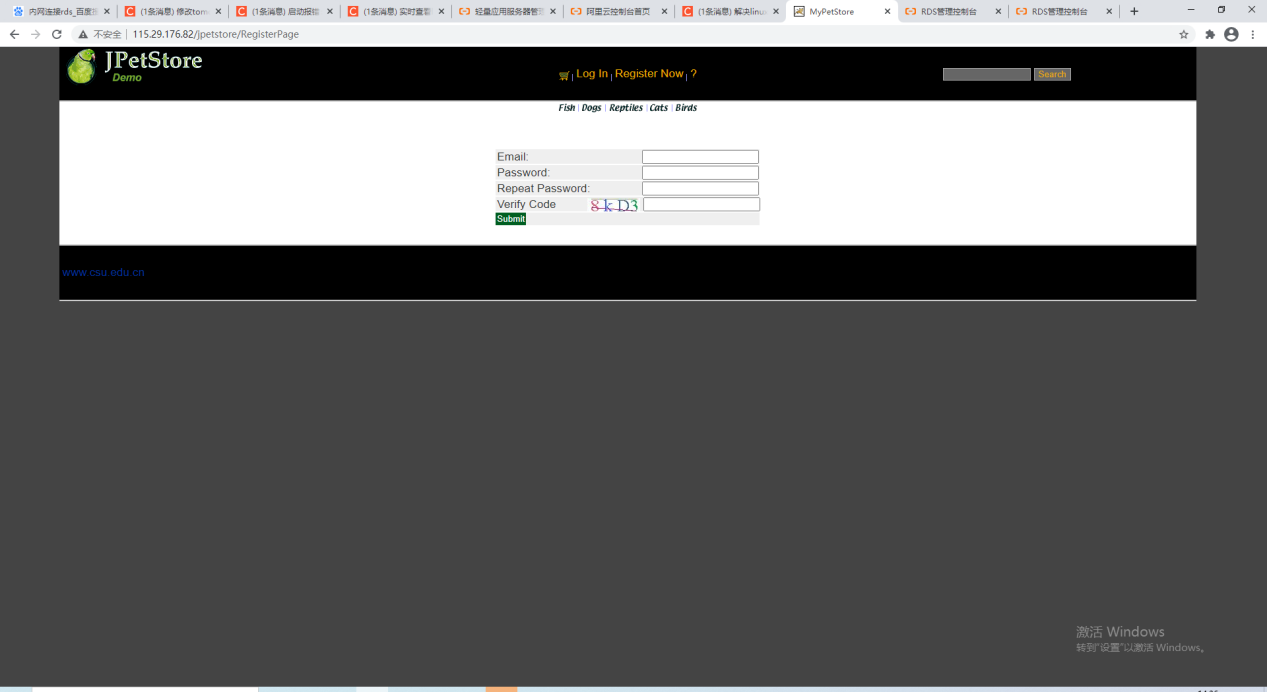
**}**

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。

我计划采用四位验证码来实现判别是否为机器人的验证功能。这个功能的实现原理具体如下：在jsp页面被打开的时候，验证码图片的src会调用RandomCodeServlet用来生成四个随机字符和验证码的总体图片，在点击登录之后需要进行判断用户输入的验证码是否和生成的一致，在这里使用RandomUtils类来完成这个功能，判断相同则完成登录功能。

1.1 效果截图





1.2核心代码

1.2.1 返回验证码图片的VerifyCodeServlet.java

*/\*\*  
 \* @author MokuSakura  
 \*/***@WebServlet(name = ServletNameConstant.VERIFY\_CODE\_SERVLET\_NAME, urlPatterns = {ServletUrlConstant.VERIFY\_CODE\_SERVLET\_URL})  
public class VerifyCodeServlet extends HttpServlet {  
 private static final String CONTENT\_TYPE = "image/jpeg";  
 private static final String HEADER\_PRAGMA = "Pragma";  
 private static final String HEADER\_CACHE\_CONTROL = "Cache-Control";  
 private static final String STRING\_NO\_CACHE = "No-cache";  
 private static final int INT\_NO\_CACHE = 0;  
 private static final int WIDTH = 80;  
 private static final int HEIGHT = 20;  
 private static final String HEADER\_EXPIRES = "Expires";  
 private static final String FORMAT\_NAME = "JPEG";  
  
 @Override  
 protected void doGet(HttpServletRequest req, HttpServletResponse resp) throws ServletException, IOException {  
 // response with image  
 resp.setContentType(CONTENT\_TYPE);  
 // no cache  
 resp.setHeader(HEADER\_PRAGMA, STRING\_NO\_CACHE);  
 resp.setHeader(HEADER\_CACHE\_CONTROL, STRING\_NO\_CACHE);  
 resp.setDateHeader(HEADER\_EXPIRES, INT\_NO\_CACHE);  
  
 OutputStream outputStream = resp.getOutputStream();  
 BufferedImage bufferedImage = new BufferedImage(WIDTH, HEIGHT, BufferedImage.TYPE\_INT\_RGB);  
 String verifyCode = VerifyCodeGenerator.GenerateVerifyCode(bufferedImage);  
 req.getSession().setAttribute(SessionAttributeKeyConstant.CORRECT\_VERIFY\_CODE, verifyCode);  
 ImageIO.write(bufferedImage, FORMAT\_NAME, outputStream);  
 }  
  
 @Override  
 protected void doPost(HttpServletRequest req, HttpServletResponse resp)  
 throws ServletException, IOException {  
 doGet(req, resp);  
 }  
  
}**

**1.2.2随机生成验证码和图片的VerifyCodeGenerator.java**

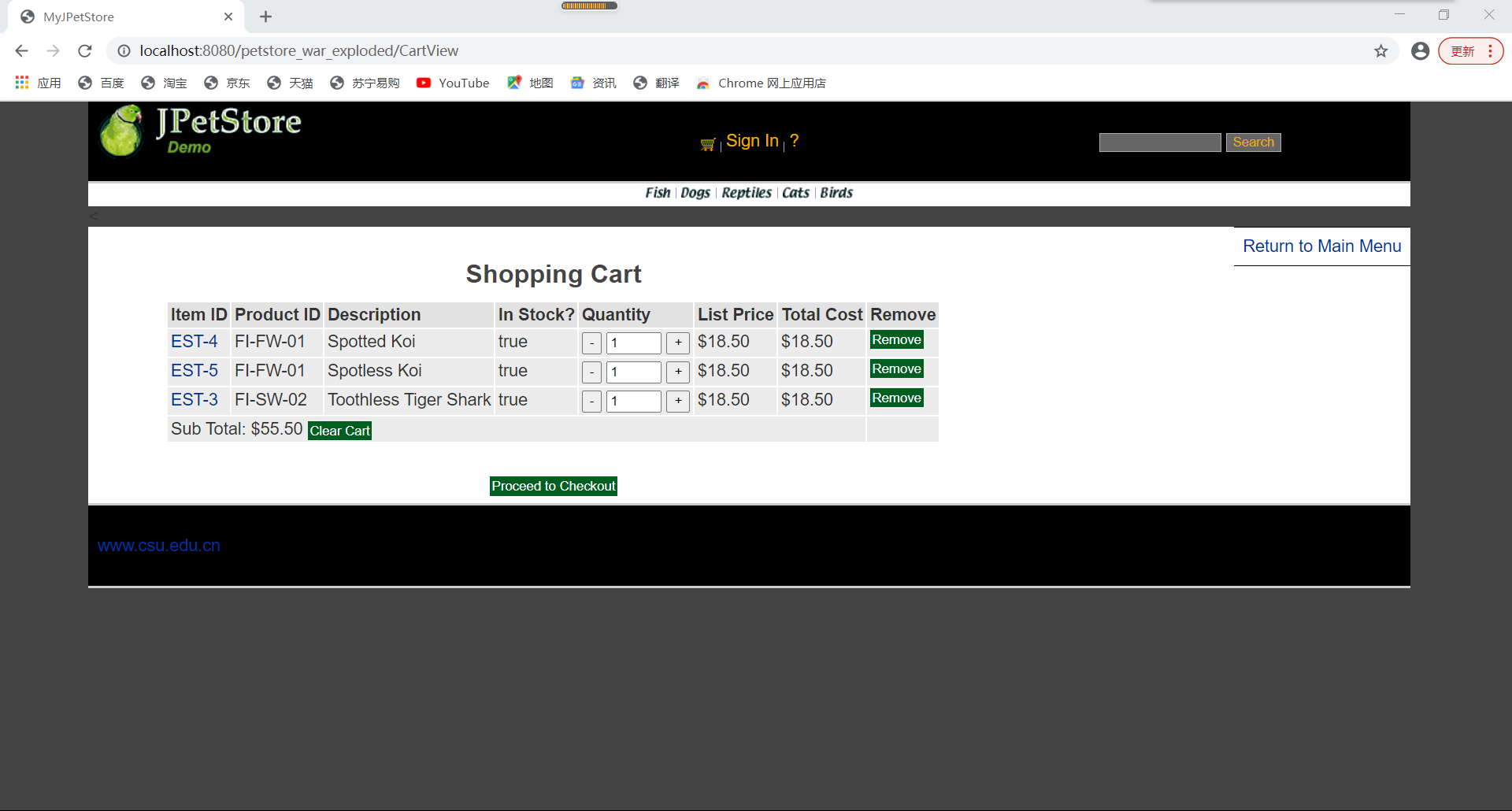
**public class VerifyCodeGenerator {  
 private static final int WIDTH = 80;  
 private static final int HEIGHT = 20;  
 private static final String STRING = "abcdefghjkmnopqrstuvwxyzABCDEFGHJKLMNPQRSTUVWXYZ23456789";  
 private static final int LENGTH = STRING.length();  
  
 public static String GenerateVerifyCode(BufferedImage bufferedImage) {  
 Graphics2D graphics2D = bufferedImage.createGraphics();  
 graphics2D.fillRect(0, 0, WIDTH, HEIGHT);  
 StringBuilder res = new StringBuilder();  
 Random random = new Random();  
 graphics2D.setFont(new Font("Times New Roman", Font.PLAIN, 26));  
 //Random String  
 for (int i = 0; i < 4; i++) {  
 char rand = STRING.charAt(random.nextInt(LENGTH));  
 res.append(rand);  
 graphics2D.setColor(new Color(random.nextInt(255), random.nextInt(255), random.nextInt(255)));  
 graphics2D.drawString(String.valueOf(rand), 20 \* i + 5, 20);  
 }  
 //Line to interfere  
 graphics2D.setColor(new Color(random.nextInt(255), random.nextInt(255), random.nextInt(255)));  
 graphics2D.drawLine(random.nextInt(10), random.nextInt(HEIGHT), random.nextInt(10) + 70,  
 random.nextInt(HEIGHT));  
 graphics2D.setColor(new Color(random.nextInt(255), random.nextInt(255), random.nextInt(255)));  
 graphics2D.drawLine(random.nextInt(10), random.nextInt(HEIGHT), random.nextInt(10) + 70,  
 random.nextInt(HEIGHT));  
 graphics2D.dispose();  
 return res.toString();  
 }  
}**

**1.2.3用于验证的接口VerifyServlet.java**

**/\*\*  
 \* @author MokuSakura  
 \*/  
@WebServlet(name = ServletNameConstant.VERIFY\_SERVLET\_NAME, urlPatterns = {ServletUrlConstant.VERIFY\_SERVLET\_URL})  
public class VerifyServlet extends HttpServlet {  
 private static final String RESPONSE\_TYPE = "text/plain";  
 private static final String PARAMETER\_KEY\_VERIFY\_CODE = "verifyCode";  
  
 @Override  
 protected void doPost(HttpServletRequest req, HttpServletResponse resp) throws ServletException, IOException {  
 resp.setContentType(RESPONSE\_TYPE);  
 PrintWriter printWriter = resp.getWriter();  
 String verifyCode = req.getParameter(PARAMETER\_KEY\_VERIFY\_CODE);  
 String correctVerifyCode = req.getSession().getAttribute(  
 SessionAttributeKeyConstant.CORRECT\_VERIFY\_CODE).toString();  
 if (verifyCode != null && verifyCode.equalsIgnoreCase(correctVerifyCode)) {  
 printWriter.print("true");  
 } else {  
 printWriter.print("false");  
 }  
 printWriter.flush();  
 printWriter.close();  
 }  
}**

1. 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。

2.1 效果截图



2.2 核心代码

**用于持久化读写的CartDaoImpl.java**

**public class CartDAOImpl implements CartDAO {  
 private static final ItemDAO ITEM\_DAO = new ItemDAOImpl();  
 private static final String ADD\_ITEM\_TO\_CART = "INSERT INTO cart (email,itemid,quantity) VALUES(?,?,?)";  
  
 private static final String UPDATE\_QUANTITY = "UPDATE cart SET quantity=? WHERE email=? AND itemid=?";  
  
 private static final String DELETE\_ITEM = "DELETE FROM cart WHERE email=? AND itemid=?";  
  
 private static final String DELETE\_ALL\_ITEM\_FROM\_CART = "DELETE FROM cart WHERE email=?";  
  
 private static final String SELECT\_USER\_CART = "SELECT \* FROM cart WHERE email=?";  
  
 @Override  
 public int addItemToCart(User user, Item item) {  
 int res = 0;  
 try {  
 Connection connection = DBUtil.getConnection();  
 PreparedStatement preparedStatement = connection.prepareStatement(ADD\_ITEM\_TO\_CART);  
 preparedStatement.setString(1, user.getEmail());  
 preparedStatement.setString(2, item.getItemId());  
 preparedStatement.setInt(3, 1);  
 res = preparedStatement.executeUpdate();  
 DBUtil.closePreparedStatement(preparedStatement);  
 DBUtil.closeConnection(connection);  
 } catch (Exception e) {  
  
 MyLogger.getInstance().log(e.getMessage(), MyLogger.Level.ERROR);  
 e.printStackTrace();  
 }  
 return res;  
 }  
  
 @Override  
 public int updateQuantity(User user, Item item, int quantity) {  
 int res = 0;  
 try {  
 Connection connection = DBUtil.getConnection();  
 PreparedStatement preparedStatement = connection.prepareStatement(UPDATE\_QUANTITY);  
 preparedStatement.setInt(1, quantity);  
 preparedStatement.setString(2, user.getEmail());  
 preparedStatement.setString(3, item.getItemId());  
 res = preparedStatement.executeUpdate();  
 DBUtil.closePreparedStatement(preparedStatement);  
 DBUtil.closeConnection(connection);  
 } catch (Exception e) {  
 MyLogger.getInstance().log(e.getMessage(), MyLogger.Level.ERROR);  
 e.printStackTrace();  
 }  
  
 return res;  
 }  
  
 @Override  
 public int deleteItem(User user, Item item) {  
 int res = 0;  
 try {  
 Connection connection = DBUtil.getConnection();  
 PreparedStatement preparedStatement = connection.prepareStatement(DELETE\_ITEM);  
 preparedStatement.setString(1, user.getEmail());  
 preparedStatement.setString(2, item.getItemId());  
 res = preparedStatement.executeUpdate();  
 DBUtil.closePreparedStatement(preparedStatement);  
 DBUtil.closeConnection(connection);  
 } catch (Exception e) {  
 MyLogger.getInstance().log(e.getMessage(), MyLogger.Level.ERROR);  
 e.printStackTrace();  
 }  
 return res;  
 }  
  
 @Override  
 public int deleteCart(User user) {  
 int res = 0;  
 try {  
 Connection connection = DBUtil.getConnection();  
 PreparedStatement preparedStatement = connection.prepareStatement(DELETE\_ALL\_ITEM\_FROM\_CART);  
 preparedStatement.setString(1, user.getEmail());  
 res = preparedStatement.executeUpdate();  
 DBUtil.closePreparedStatement(preparedStatement);  
 DBUtil.closeConnection(connection);  
 } catch (Exception e) {  
 e.printStackTrace();  
 }  
 return res;  
 }  
  
 @Override  
 public Cart getUserCart(User user) {  
 Cart userCart = null;  
 try {  
 Connection connection = DBUtil.getConnection();  
 PreparedStatement preparedStatement = connection.prepareStatement(SELECT\_USER\_CART);  
 preparedStatement.setString(1, user.getEmail());  
 ResultSet resultSet = preparedStatement.executeQuery();  
 userCart = new Cart();  
 while (resultSet.next()) {  
 CartItem cartItem = new CartItem();  
 String itemId = resultSet.getString(2);  
 int quantity = resultSet.getInt(3);  
 Item item = ITEM\_DAO.getItem(itemId);  
 cartItem.setItem(item);  
 cartItem.setQuantity(quantity);  
 cartItem.setInStock(item.getQuantity() > 0);  
 userCart.getItemCartItemMap().put(itemId, cartItem);  
 //cartItem.calculateTotal();  
 //userCart.getItemCartItemMap().put(itemID, cartItem);  
 }  
 //userCart.calculateTotal();  
 DBUtil.closeResultSet(resultSet);  
 DBUtil.closePreparedStatement(preparedStatement);  
 DBUtil.closeConnection(connection);  
 } catch (Exception e) {  
 e.printStackTrace();  
 }  
 return userCart;  
 }  
}**

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。

首先需要实现service中的有关日志的方法同时创建一个daily的domain类，这样在每个需要记录的操作发生的时候就会调用daily的有关方法。在调用ProductViewServlet时我选择新建了一个DailyService类的对象并且调用相关的方法然后来实现浏览商品的功能。同时新建一个Daily类的对象Daily daily = new Daily();设置daily的相关属性参数后将该对象插入到数据库中dailyService.insertDaily(daily);该insert方法会调用DailyDAO中的insertDaily(Daily daily)方法。类型为browse；

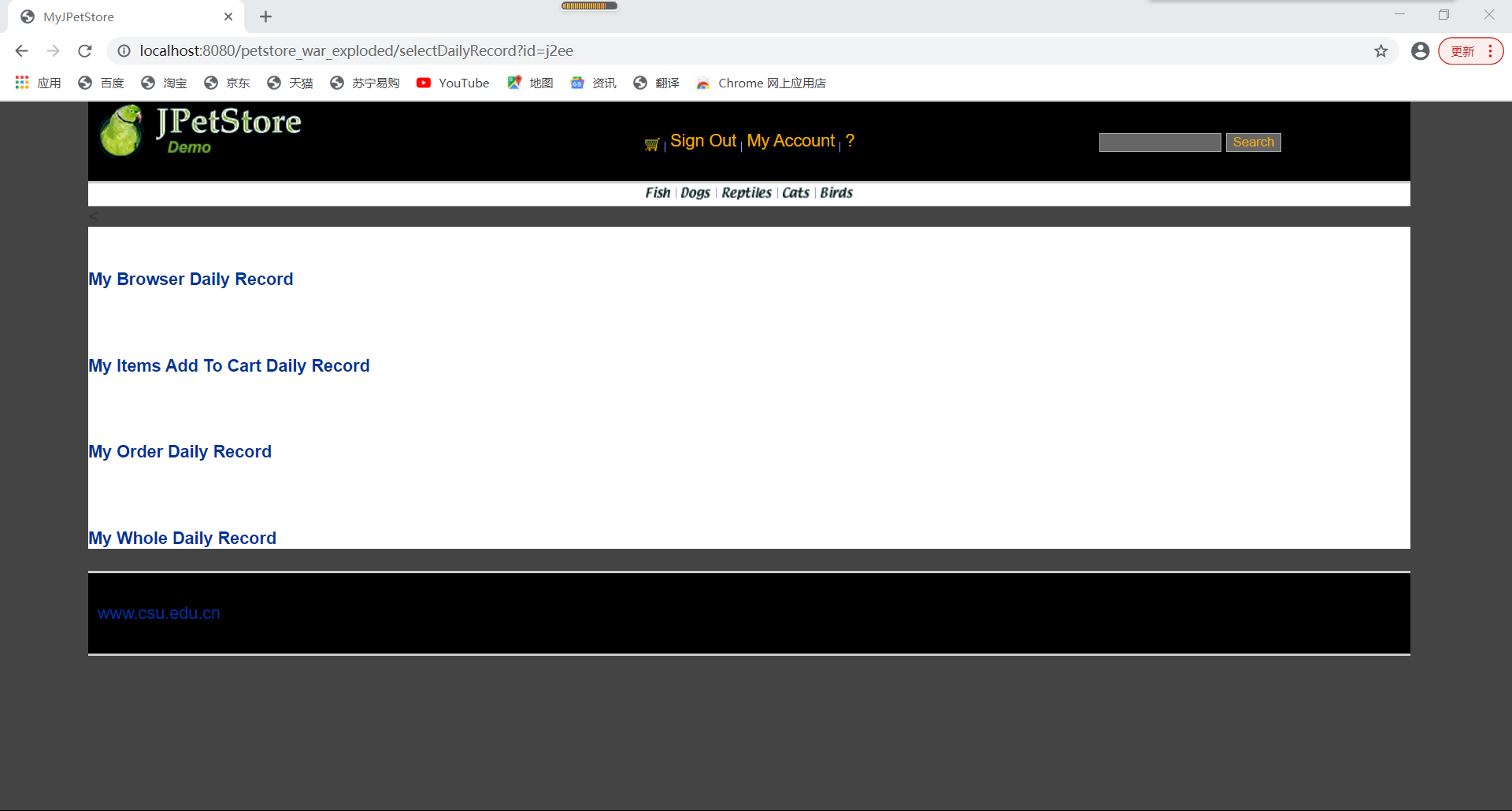
接着需要完成的就是将商品加入到购物车中的功能，在调用CartViewServlet时新建一个DailyService类的对象并调用相关的方法private DailyService dailyService = new DailyService()以此来将商品加入到购物车中。并且新建一个Daily类的对象Daily daily = new Daily();设置daily的相关属性参数后将该对象插入到数据库中dailyService.insertDaily(daily);该insert方法会调用DailyDAO中的insertDaily(Daily daily)方法。类型为cart；这样就完成了将数据插入数据库中。

订单结算是在调用ViewOrderServlet时,新建一个DailyService类的对象并调用相关的方法private DailyService dailyService = new DailyService();将数据插入数据库中的操作与上面类似，新建一个Daily类的对象Daily daily = new Daily();设置daily的相关属性参数后将该对象插入到数据库中dailyService.insertDaily(daily);该insert方法会调用DailyDAO中的insertDaily(Daily daily)方法。类型为cart；

最后我需要实现查看日志内容的功能，首先我经过SelectDailyRecordServlet会跳转到选择的界面SelectDaily.jsp来选择要查看的日志类型以及内容。然后可以点击不同的部分来调用相应的方法最后将生成符合条件的日志条目显示出来。

3.1实现效果：

只展示部分日志：



3.2 核心代码：

只展示部分日志代码：

**package org.csu.myjpetstore.service;**

**import org.csu.myjpetstore.domain.Daily;**

**import org.csu.myjpetstore.persistence.DailyDAO;**

**import org.csu.myjpetstore.persistence.impl.DailyDAOImpl;**

**import java.util.List;**

**public class DailyService {**

**private DailyDAO dailyDAO;**

**public DailyService(){**

**dailyDAO = new DailyDAOImpl();**

**}**

**public List<Daily> getDailyByUserId(String userId){**

**return dailyDAO.getDailyByUserId(userId);**

**}**

**public List<Daily> getDailyByUserIdAndKind(String userId,String kind){**

**return dailyDAO.getDailyByUserIdAndKind(userId,kind);**

**}**

**public void insertDaily(Daily daily){**

**dailyDAO.insertDaily(daily);**

**}**

**}**

**package org.csu.myjpetstore.web.Daily;**

**import org.csu.myjpetstore.domain.Account;**

**import org.csu.myjpetstore.domain.Daily;**

**import org.csu.myjpetstore.service.AccountService;**

**import org.csu.myjpetstore.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.ArrayList;**

**import java.util.List;**

**public class ShowDailyRecordServlet extends HttpServlet {**

**private static final String showDailyRecordString = "/WEB-INF/jsp/daily/ShowDaily.jsp";**

**private DailyService dailyService = new DailyService();**

**private static List<Daily> browseList = new ArrayList<>();**

**private static List<Daily> cartList = new ArrayList<>();**

**private static List<Daily> orderList = new ArrayList<>();**

**private static List<Daily> list = new ArrayList<>();**

**AccountService accountService;**

**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();**

**String kind = request.getParameter("kind");**

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

**String userId;**

**if(account.getUsername()!=null) {**

**userId = account.getUsername();**

**}**

**else {**

**userId = request.getParameter("id");**

**}**

**System.out.print(userId);**

**System.out.println(kind);**

**if(kind.equals("browse")){**

**browseList = dailyService.getDailyByUserIdAndKind(userId,kind);**

**session.setAttribute("topic","My Browse Daily Record");**

**session.setAttribute("list",browseList);**

**}**

**else if(kind.equals("cart")){**

**cartList = dailyService.getDailyByUserIdAndKind(userId,kind);**

**session.setAttribute("topic","My Items Add to Cart Daily Record");**

**session.setAttribute("list",cartList);**

**}**

**else if(kind.equals("order")){**

**orderList = dailyService.getDailyByUserIdAndKind(userId,kind);**

**session.setAttribute("topic","My Order Daily Record");**

**session.setAttribute("list",orderList);**

**}**

**else if(kind.equals("all")){**

**list = dailyService.getDailyByUserId(userId);**

**session.setAttribute("topic","My Whole Daily Record");**

**session.setAttribute("list",list);**

**}**

**else{**

**session.setAttribute("selectMessage","your choose is wrong");**

**}**

**request.getRequestDispatcher(showDailyRecordString).forward(request,response);**

**}**

**}**

1. **小组成员分工及实验总结**
2. 小组分工：

均由一人完成

1. 实验总结

由于我是大二没选这门课，我是大三的才选的，所以这些对我来说还是挺简单的。但是这种最基础的javaweb写起来还是挺烦的。在这个实验中我将知识与实践相结合，在实际的运用中更好地吸收了的课堂所学。

首先就是如何配置好环境，因为是首次编写web项目，所以我的环境配置问题就显得略微复杂，不仅要配置好tomcat，导入包，还要使用相关的包来获取与数据库的连接。还有就是在第一次打开浏览器的时候，一定会遇到404找不到界面和500空指针异常的种种问题，根据网页提示的错误信息在代码中找到错误并且完成修改。

**实验二 使用AJAX和jQuery改善用户体验**

1. **实验内容——基本任务。**
2. 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。
3. 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。
4. 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。

**//以上三个业务模块，用运行界面截图和核心代码展示功能完成情况。**

1. **实验内容——选做任务。**
2. 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。
3. 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。
4. 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。

**//以上为选做任务，没有完成填无**

**//完成了选做任务或其他特效的，除用界面截图和核心代码展示功能完成情况外，还需说明基本原理、实现方案和是否使用框架等。**

1. **小组成员分工及实验总结**
2. 小组分工：

均一人完成

1. 实验总结

**实验二 使用AJAX和jQuery改善用户体验**

1. **实验内容——基本任务。**
2. 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。
3. 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。
4. 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。

**//以上三个业务模块，用运行界面截图和核心代码展示功能完成情况。**

**1.账号管理模块：**

**核心代码：**

**NewAccountForm.html关键部分:**

**<**script type="text/javascript"**>**  
 $**(**"#username**")**.blur**(**function **() {** if **(**$**(**"#username**")**.val**()** == ""**) {** $**(**"#newAccountMessage**")**.text**(**"username can't be blank!"**)**;  
 $**(**"#submit**")**.attr**(**"name", "button"**)**;  
 return;  
 **}** $.ajax**({** url: "http://localhost:8080/hasOne/" + $**(**"#username**")**.val**()**,  
 type: "get",  
 dataType: "json",  
 data: **{}**,  
 success:  
 function **(***data***) {** if **(***data*.code == 200**) {** $**(**"#newAccountMessage**")**.text**(**$**(**"#username**")**.val**()** + " already exists!"**)**;  
 $**(**"#submit**")**.attr**(**"name", "button"**)**;  
 **}** else **{** $**(**"#newAccountMessage**")**.text**(**"please enter the password!"**)**;  
 **}  
 }  
 })**;  
 **})**;  
 $**(**"#password**")**.blur**(**function **() {** if **(**$**(**"#password**")**.val**()** == ""**) {** $**(**"#newAccountMessage**")**.text**(**"password can't be blank!"**)**;  
 $**(**"#submit**")**.attr**(**"name", "button"**)**;  
 **}** else **{** $**(**"#newAccountMessage**")**.text**(**"please repeat password!"**)**;  
 $**(**"#submit**")**.attr**(**"name", "button"**)**;  
 **}  
 })**;  
 $**(**"#repeatedPassword**")**.blur**(**function **() {** if **(**$**(**"#password**")**.val**()** != $**(**"#repeatedPassword**")**.val**()) {** $**(**"#newAccountMessage**")**.text**(**"passwords do not match!"**)**;  
 $**(**"#submit**")**.attr**(**"name", "button"**)**;  
 **}** else **{** $**(**"#newAccountMessage**")**.text**(**""**)**;  
 $**(**"#submit**")**.attr**(**"name", "submit"**)**;  
 **}  
  
  
 })**;  
   
**</**script**>**

**UserNameIsExistServlet.java部分：**

*@PutMapping***(**"/register"**)**public Map**<**String, Object**>** signin**(***@RequestBody* Signon *signon***) {** Map**<**String, Object**>** map = new HashMap**<>()**;  
 try **{** signonService.register**(***signon***)**;  
 map.put**(**"code", 200**)**;  
 map.put**(**"message", "ok"**)**;  
 map.put**(**"signinAccount", *signon***)**;  
 **}** catch **(**AlreadyRegisteredException *annotation***) {** map.put**(**"code", 501**)**;  
 map.put**(**"message", "no"**)**;  
 map.put**(**"wrongmessage", *signon*.getUsername**()** + " already registered"**)**;  
 **}** finally **{** return map;  
 **}  
}**

**运行效果：**



**2.查询商品模块：**

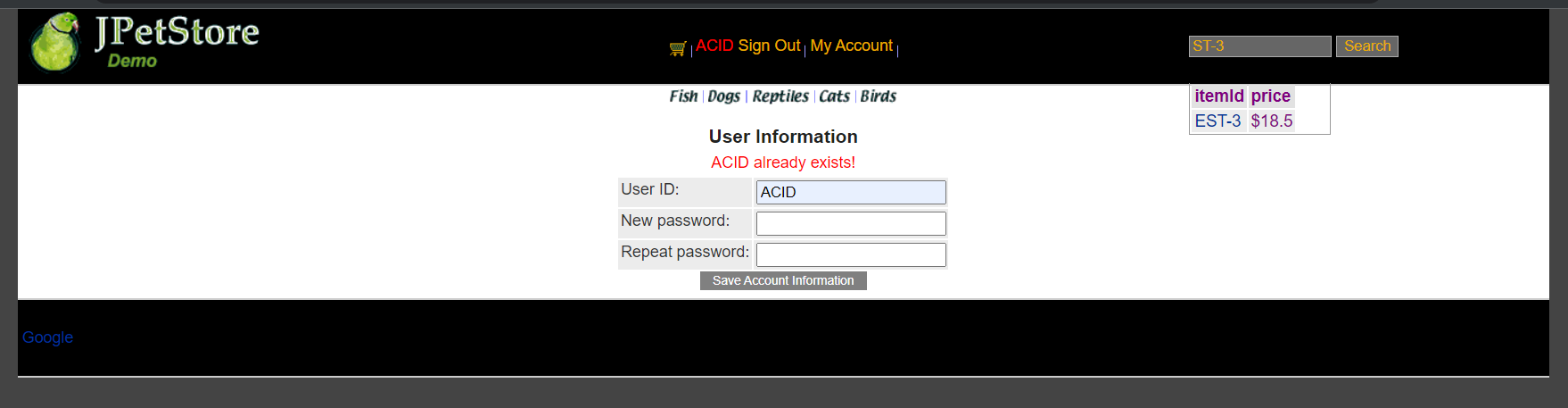
$**(**"#searchtext**")**.blur**(**function **() {** $.ajax**({** url: "http://localhost:8080/item/search/" + $**(**"#searchtext**")**.val**()**,  
 type: "get",  
 dataType: "json",  
 data: **{}**,  
 success:  
 function **(***data***) {** if **(***data*.code == 200 && *data*.category.length != 0**) {** var showText = `**<**tr class="showtable"**>  
 <**th style=" color: purple; height: 15px"**>**itemId**</**th**>  
 <**th style=" color: purple; height: 15px"**>**price**</**th**>  
 </**tr**>**`;  
 for **(**var i = 0; i < *data*.category.length; i++**) {** showText +=  
 `**<**tr class="showtable"**>  
 <**td style=" color: purple; height: 15px"**><**a href="javascript:void(0)" class="tips" id="`+  
 *data*.category**[**i**]**.itemId +`"**>**`  
 + *data*.category**[**i**]**.itemId +  
 `**</**a**></**td**>  
 <**td style=" color: purple; height: 15px"**>**$` + *data*.category**[**i**]**.listPrice + `**</**td**>  
 </**tr**>**`;  
 **}** $**(**"#show**")**.append**(**showText**)**;  
 $**(**".show**")**.show**()**;  
 **}  
  
 }  
 })**;  
**})**;  
$**(**"#searchtext**")**.focus**(**function **() {** $**(**"#show**")**.html**(**""**)**;  
 $**(**".show**")**.hide**()**;  
**})**;  
$**(**document**)**.on**(**"click",".tips",function **(***e***) {** $**(**"#searchtext**")**.val**(***e*.target.id**)**;  
 $**(**"#show**")**.html**(**""**)**;  
 $**(**".show**")**.hide**()**;  
**})**;  
$**(**"#searchbutton**")**.click**(**function **(***e***) {** *e*.preventDefault**()**;  
 if **(**$**(**"#searchtext**")**.val**()** == ""**) {** return;  
 **}** $.ajax**({** url: "http://localhost:8080/item/searchOne/" + $**(**"#searchtext**")**.val**()**,  
 type: "get",  
 dataType: "json",  
 data: **{}**,  
 success:  
 function **(***data***) {** if **(***data*.code!=200**){** alert**(**"no such category!"**)**;  
 **}**else**{** window.location="http://localhost:63342/mypetstore2/static/catalog/Product.html";  
 **}  
 }  
 })**;  
**})**;

*@GetMapping***(**"/search/{name}"**)**public Map**<**String, Object**>** search**(***@PathVariable***(**"name"**)** String *name***) {** Map**<**String, Object**>** map = new HashMap**<>()**;  
 map.put**(**"code", 200**)**;  
 map.put**(**"message", "ok"**)**;  
 map.put**(**"category", itemService.search**(***name***))**;  
 return map;  
**}**

**实现截图：**

**当输入ST-2时：**

**当输入ST-3时：**



**当搜索后：**

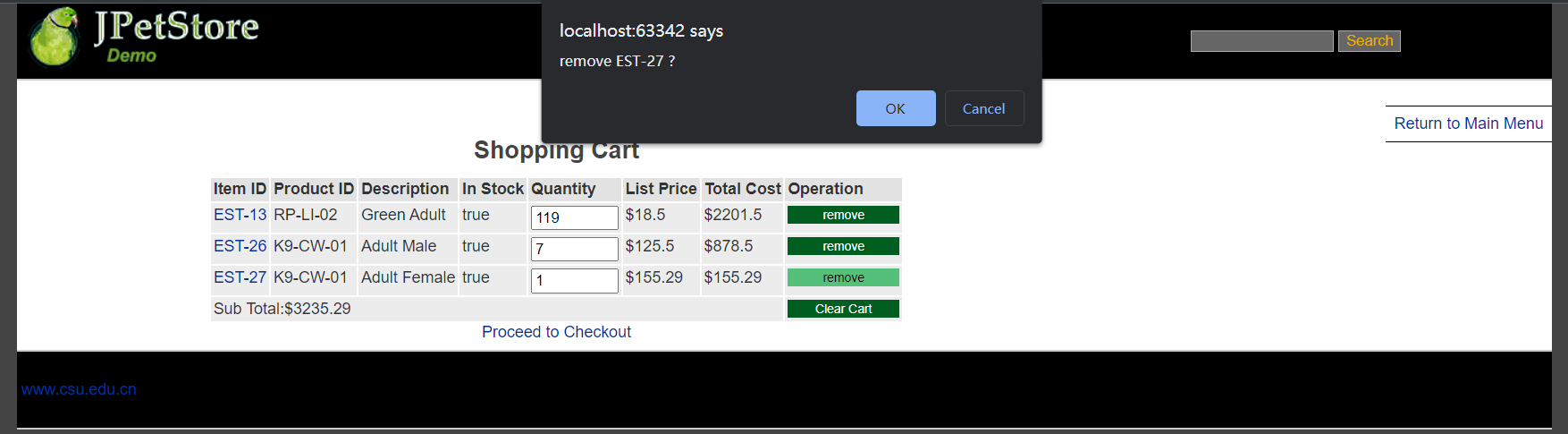
**还有下拉列表框鼠标移入移出的效果不好截图，上课时已经展示。**

**3.购物车模块：**

$.ajax**({** url: "http://localhost:8080/cart/" + $.trim**(**$**(**"#signin**")**.text**())**,  
 type: "get",  
 dataType: "json",  
 data: **{}**,  
 success:  
 function **(***data***) {** if **(***data*.code == 200**) {** var tr = "";  
 if **(***data*.itemInCart.length == 0**) {** tr += `**<**tr**><**td colspan="8"**><**b**>**Your cart is empty.**</**b**></**td**></**tr**>**`;  
 **}** else **{** for **(**var i = 0; i < *data*.itemInCart.length; i++**) {** tr += `  
 **<**tr**>  
 <**td**><**a href="javascript:void(0)"**>**` +  
 *data*.itemInCart**[**i**]**.itemid  
 + `**</**a**></**td**>  
 <**td**>**` +  
 *data*.itemInCart**[**i**]**.productId  
 + `**</**td**>  
 <**td**>**` +  
 *data*.itemInCart**[**i**]**.description  
 + `**</**td**>  
 <**td**>**true**</**td**>  
 <**td**><**input type="number" class="itemNumber" min="1" max="2000" size="3" name="" value="` +  
 *data*.itemInCart**[**i**]**.quantity  
 + `"**/>  
 </**td**>  
 <**td name="` +  
 *data*.itemInCart**[**i**]**.listprice  
 + `"**>**$` +  
 *data*.itemInCart**[**i**]**.listprice  
 + `**</**td**>  
 <**td name="` +  
 *data*.itemInCart**[**i**]**.totalcost  
 + `"**>**$` +  
 *data*.itemInCart**[**i**]**.totalcost  
 + `**</**td**>  
 <**td**><**input class="Button" name="removeButton" type="submit" value="remove" style="width: 100px;"**/></**td**>  
 </**tr**>**`;  
 **}  
 }** $**(**"#totalCount**")**.text**(**"$" + *data*.totalCount**)**;  
 $**(**"#cartHead**")**.after**(**tr**)**;  
 **}  
 }  
 })**;  
**}**$**(**document**)**.on**(**"change", ".itemNumber", function **(***e***) {** $.ajax**({** url: "http://localhost:8080/cart/" + $.trim**(**$**(**"#signin**")**.text**())**,  
 type: "put",  
 dataType: "json",  
 contentType: "application/json;charset=UTF-8",  
 data: JSON.stringify**({** "username": $**(**"#signin**")**.text**()**,  
 "itemid": $**(**this**)**.parent**()**.parent**()**.find**(**"td:nth-child**(**1**)")**.text**()**,  
 "instock": $**(**this**)**.val**()**,  
 "listprice": $**(**this**)**.parent**()**.parent**()**.find**(**"td:nth-child**(**6**)")**.attr**(**"name"**)**,  
 "totalcost": $**(**this**)**.parent**()**.parent**()**.find**(**"td:nth-child**(**6**)")**.attr**(**"name"**)**\* $**(**this**)**.val**()  
 })**,  
 success:  
 function **(***data***) {** ajax2**()**;  
 **}  
  
 })**;  
**})**;

*@PutMapping***(**"/{username}"**)**public Map**<**String, Object**>** update**(***@RequestBody* CartItem *cartItem***) {** Map**<**String, Object**>** map = new HashMap**<**String, Object**>()**;  
 if **(**cartService.updateItem**(***cartItem***)** == 1**) {** map.put**(**"code", 200**)**;  
 map.put**(**"message", "ok"**)**;  
 **}**else **{** map.put**(**"code", 501**)**;  
 map.put**(**"message", "no"**)**;  
 map.put**(**"wrongmessage", *cartItem*.getItemid**()** + " updates failed"**)**;  
 **}** return map;  
**}***@Transactional  
@PostMapping***(**"/{username}"**)**public Map**<**String, Object**>** add**(***@RequestBody* CartItem *cartItem*,*@PathVariable***(**"username"**)**String *username***) {** Map**<**String, Object**>** map = new HashMap**<**String, Object**>()**;  
 List**<**CartItem**>** cartItems = new ArrayList**<>()**;  
 cartItems.addAll**(**cartService.viewCart**(***username***))**;  
 boolean flag=false;  
 if **(**cartItems.size**()**!=0**){** for **(**CartItem item : cartItems**) {** if **(**Objects.**equals(** *cartItem*.getItemid**()**, item.getItemid**())){** CartItem newCartItem=item;  
 newCartItem.setInstock**(**item.getInstock**()**+1**)**;  
 newCartItem.setTotalcost**(**newCartItem.getListprice**()**.multiply**(**new BigDecimal**(**newCartItem.getInstock**())))**;  
 cartService.updateItem**(**newCartItem**)**;  
 flag=true;  
 map.put**(**"code",201**)**;  
 map.put**(**"message", "ok"**)**;  
 break;  
 **}  
 }  
 }** if **(**!flag**) {** if **(**cartService.addItem**(***cartItem***)** == 1**) {** map.put**(**"code", 200**)**;  
 map.put**(**"message", "ok"**)**;  
 **}**else **{** map.put**(**"code", 501**)**;  
 map.put**(**"message", "no"**)**;  
 map.put**(**"wrongmessage", *cartItem*.getItemid**()** + " adds failed"**)**;  
 **}  
 }** return map;  
**}**

**效果展示部分（控制台有信息输出）：**



1. **实验内容——选做任务。**
2. 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。
3. 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。
4. 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。

**//以上为选做任务，没有完成填无**

**//完成了选做任务或其他特效的，除用界面截图和核心代码展示功能完成情况外，还需说明基本原理、实现方案和是否使用框架等。**

1. **商品展示模块：**

**基本原理：使用jQuery实现对服务器上资源的异步访问，实时更新。**

**实现方案：**

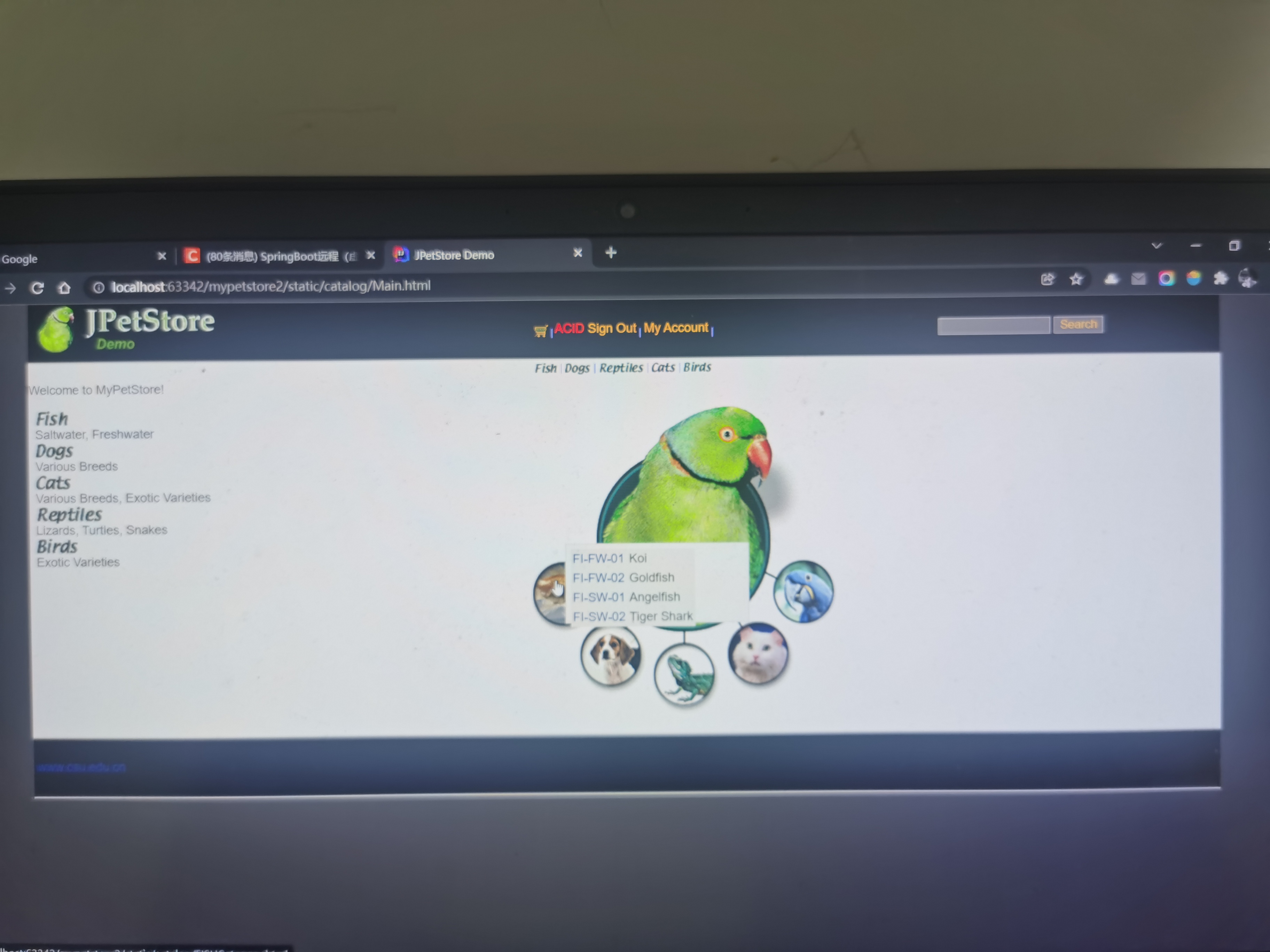
整个页面中，有且只有一个悬浮窗。该悬浮窗在一般情况下隐藏，在鼠标移至某一物品图片上后，从服务器获得所需内容后，获取鼠标位置，在鼠标位置的右边显示出来。

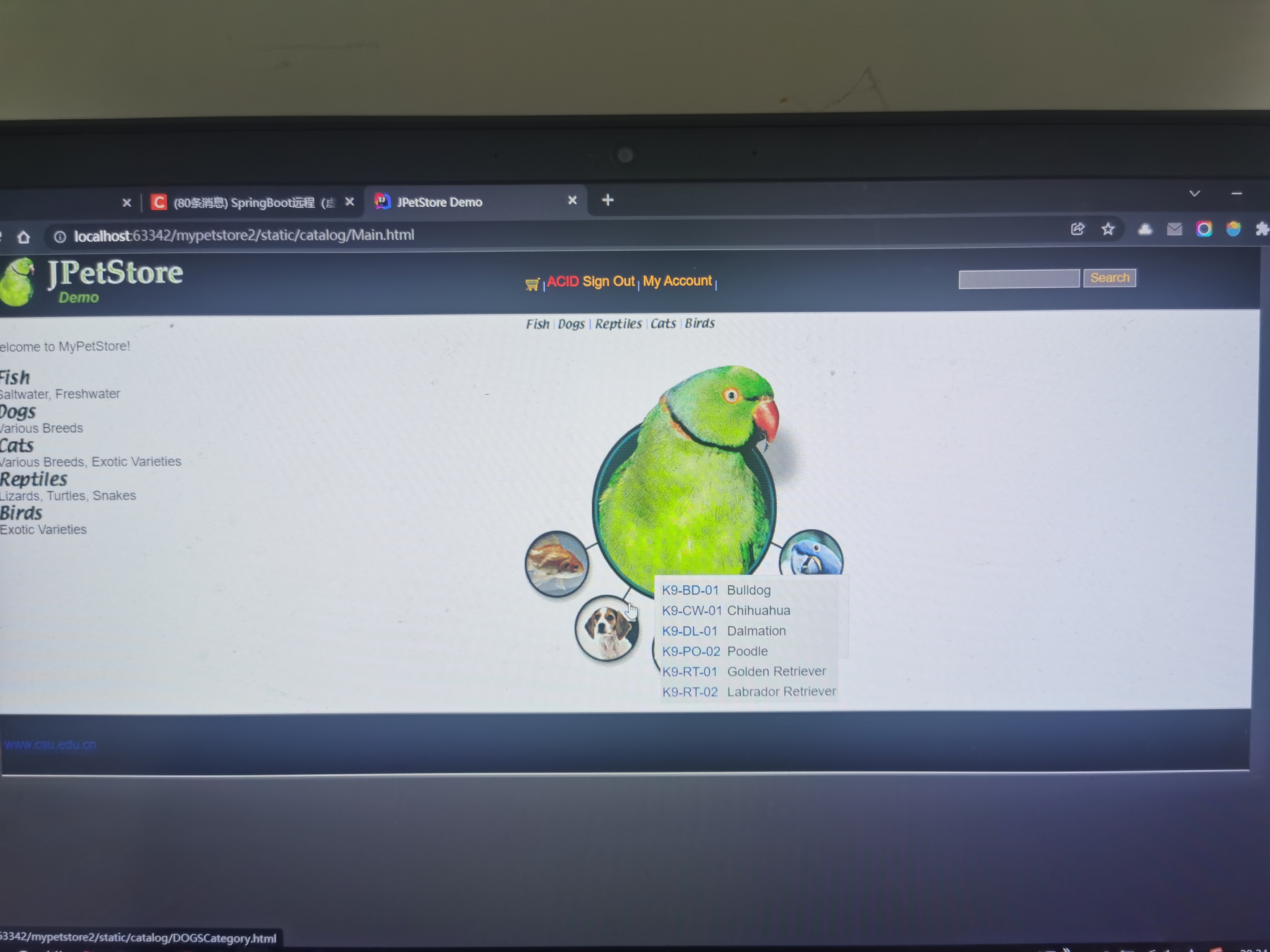
**核心代码：**

$**(**"area"**)**.mouseover**(**function **(***e***) {** // alert($(this).attr("alt"));  
 var mousePos = mousePosition**(***e***)**;  
 var xOffset = 20;  
 var yOffset = 25;  
 $**(**"#tooltip**")**.css**(**"display","block"**)**.css**(**"position","absolute"**)**.css**(**"top",**(**mousePos.y - yOffset**)** + "px"**)**.css**(**"left",**(**mousePos.x + xOffset**)** + "px"**)**;  
  
 $.ajax**({** url: "http://localhost:8080/item/view/"+$**(**this**)**.attr**(**"alt"**)**,  
 type: "get",  
 dataType: "json",  
 data: **{}**,  
 success: function **(***data***) {** var tr = "";  
 for **(**var i = 0; i < *data*.category.length; i++**) {** tr += `  
 **<**tr**>  
 <**td**><**a class="gotosearch" href="javascript:void(0)" name="` +  
 *data*.category**[**i**]**.productId  
 + `"**>**` + *data*.category**[**i**]**.productId + `**</**a**></**td**>  
 <**td**>**` + *data*.category**[**i**]**.name + `**</**td**>  
 </**tr**>**`;  
 **}** $**(**"#tooltip**")**.append**(**tr**)**;  
  
 **}  
 })**;  
 **})**;  
 $**(**"area"**)**.mouseleave**(**function **(***e***) {** $**(**"#tooltip**")**.empty**()**;  
 $**(**"#tooltip**")**.css**(**"display","none"**)**;  
 **})**;  
 **})**;  
 function mousePosition**(***ev***){** *ev* = *ev* || window.event;  
 if**(***ev*.pageX || *ev*.pageY**){** return **{**x:*ev*.pageX, y:*ev*.pageY**}**;  
 **}** return **{** x:*ev*.clientX + document.body.scrollLeft - document.body.clientLeft,  
 y:*ev*.clientY + document.body.scrollTop - document.body.clientTop  
 **}**;  
 **}  
  
  
 </**script**>  
  
 <**style type="text/css"**>** .tip**{**width:200px;height:80px;display:none;background-color:#f6f7f7;color:#333333;line-height:18px;border:1px solid #e1e3e2;padding:5px;**}  
 </**style**>  
</**head**>**

**效果展示：**

*@GetMapping***(**"/view/{category}"**)**public Map**<**String, Object**>** viewOneCategory**(***@PathVariable***(**"category"**)** String *category***) {** Map**<**String, Object**>** map = new HashMap**<**String, Object**>()**;  
 map.put**(**"code", 200**)**;  
 map.put**(**"message", "ok"**)**;  
 map.put**(**"category", productService.productList**(***category***))**;  
 return map;  
**}**

****

****

1. **订单模块：**

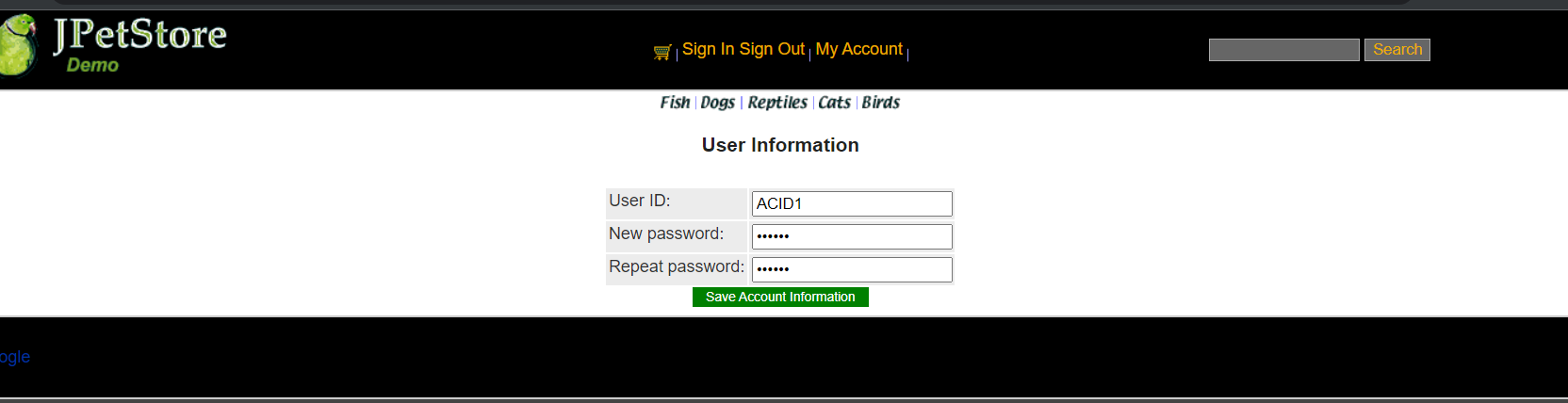
**无**

1. **其他效果：没有对界面进行大的改动，只是尽自己所能优化了一些细节，让用户可以拥有更好的体验**

**例如对注册界面的优化：**

**当输入用户名不可用时冻结下面的input标签**

**当两次输入的密码不一致时提示错误信息同时冻结下面的input标签**



**但用户已经登陆之后，在所有页面显示用户名，并禁用signin，无用户登录，禁用signout**

**在加入购物车时，如果用户第一次加入该产品和多次加入该产品提示不同**



1. **小组成员分工及实验总结**
2. 小组分工：

一人完成

1. 实验总结：

2.1遇到的问题及解决方法：

1. 在登录，注册功能中，需要进行提示信息。为了美观，表格需保持居中，而提示信息在表格右边出现。而出现提示信息后表格会向左平移。

解决方法：因提示信息占据了表格所在div的大小，进而导致居中点向右发生偏移。所以对于提示信息采用脱离文档流的绝对布局。

1. 搜索框部分，虽然功能的完成很容易，但展示效果的处理效果一直不理想。

解决方法：不使用浏览器自带的提示框，而采用自己写的提示框，并添加CSS。通过添加移入移出效果进行界面的优化。

（3）购物车部分，因为购物车物品的行数是不断变化的，采用常规操作产生错误。

解决方法：通过$(this).parent().find(...)来选取，不会出错。

（4）悬浮窗的显示位置

function mousePosition**(***ev***){** *ev* = *ev* || window.event;  
 if**(***ev*.pageX || *ev*.pageY**){** return **{**x:*ev*.pageX, y:*ev*.pageY**}**;  
 **}** return **{** x:*ev*.clientX + document.body.scrollLeft - document.body.clientLeft,  
 y:*ev*.clientY + document.body.scrollTop - document.body.clientTop  
 **}**;  
**}**

获取鼠标位置，直接在鼠标的右边显示

2.2心得体会：

通过实验2，由于第一次试验的代码缺失，再加上个人认为实验一的项目不行，正好我刚学了springboot和redis，于是我重写了一个前后端分离的项目。在此次试验中，我感受到了springboot的方便，也感受到了redis的好用，遇事不决，直接塞redis。同时我感受到了ajax带来的用户体验的直线提升。我感受到了ajax强大的功能，通过异步请求能够极大地提高用户的体验效果，打开常用的网站，使用ajax的例子比比皆是，如视频的滑动小框显示视频内容等等，多观察平时使用的网站，完成试验后我对异步访问有了更深入的了解，同时能完成一些常见的功能，体会到其实网页并不神奇，但是麻雀虽小，五脏俱全，即使是一个简单的动态表单验证，也具备Ajax异步访问的五个基本步骤，在编程时也需要十分细心，同时通过这些实验让我一步步深刻理解了前后端分离的优势和前后端在web开发中各自的作用和地位。

还有jquery十分好用。因为此次试验不允许使用前端框架，至少我是这么认为的，于是在前端我也只用了jquery这个工具包，而且我对前端技术也不是很熟悉，此次试验，我有80%的时间花在了前端上，因为之前写项目也没有这么多的写过前端。

虽然这是我补选的大二的课，但是我觉得我的时间并没有浪费，因为，这是我第一个前后端完全分离的个人项目，一个人从头写到尾的项目，由于不熟悉，我花了大量时间，有30个小时。但是我对前后端分离的写法和好处有了深刻了解。



还有一点我想感慨一下，因为我大二转专业来的，一年前，我跟老师说要退课，因为我什么也不会，但是一年后的今天，我却自己完成了一个项目，虽说谈不上强，但是也是一种肉眼可见的进步。我刚开始编程基础很差，但是通过这一年来的跟着宋铁老师的学习，跟着b站上各种视频的讲解联系，收获了很多，不仅仅巩固了web开发的知识，更多的是收获了编程的乐趣。再次感谢老师！