中南大学

Web应用开发技术

实验报告

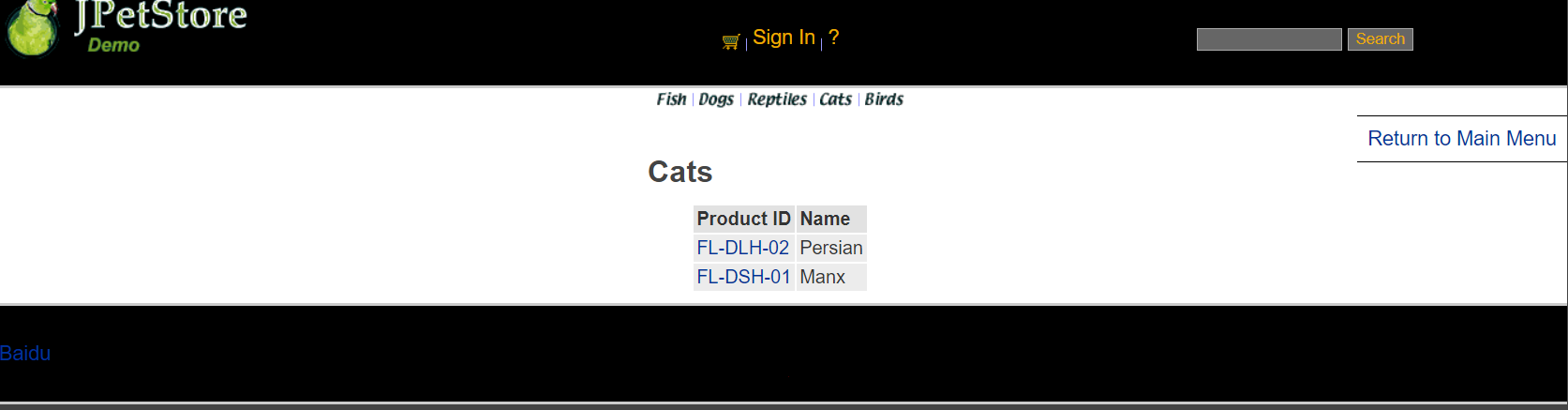
|  |  |
| --- | --- |
| 姓 名： | 周航宇，王鹤淞 |
| 学 号： | 8209190323，8209190314 |
| 班 级： | 软件工程1903 |
| 指导老师： | 宋 铁 |
| 完成时间： | 2021年1月30日 |
| 分 数： |  |
| 批阅日期： |  |

**实验一 基于MVC用JSP/Servlet实现JPetStore**

**一、基本任务**

1、**商品展示模块，分为大类Category，小类Product和具体商品Item的展示和搜索模块，功能。**

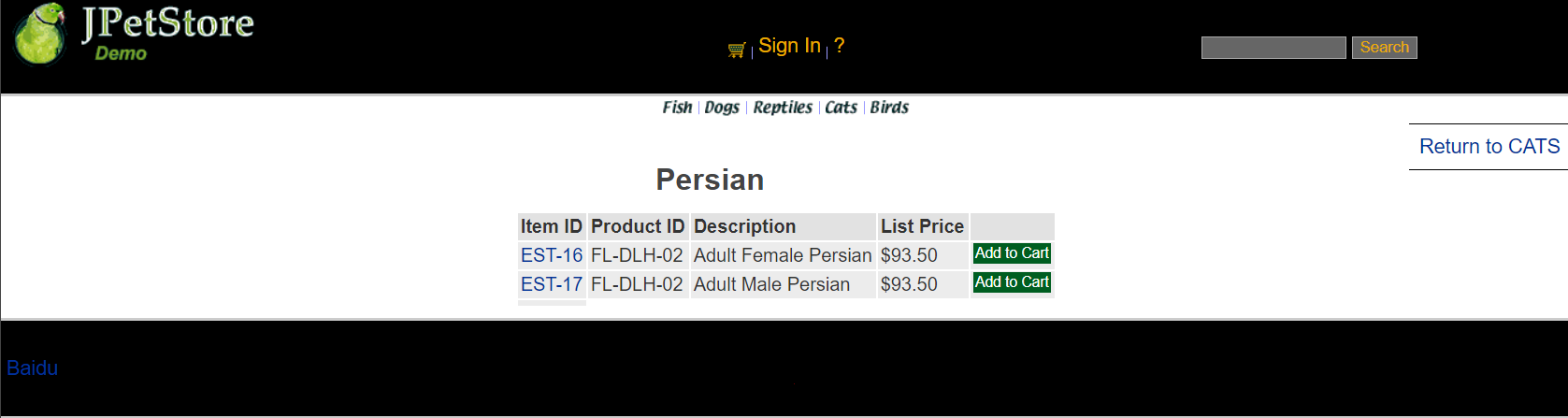
在实现大类Category时，先实现Category.jsp对大类进行具体的展示。jsp中的c:foreach语法可以实现大类的展示，通过ID选择不同的Product。在domain层中的Category.java中设计了大类相关的函数。在CategoryDAO中定义相关的数据库函数名称，在CategoryDAOImpl中对函数数据库的操作进行实现。最后通过ViewCategoryServlet实现页面之间信息的传递和jsp页面的跳转。



ViewCategoryServlet操作为：

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");*//获取参数categoryId，而categoryId在* CatalogService service = new CatalogService();  
 Category category = service.getCategory(categoryId);  
 List<Product> productList = service.getProductListByCategory(categoryId);  
  
 *//保存数据* HttpSession session = request.getSession();  
 session.setAttribute("category", category);  
 session.setAttribute("productList", productList);  
  
 *//HttpSession session = request.getSession();* Account account = (Account)session.getAttribute("account");  
  
 if(account != null){  
 HttpServletRequest httpRequest= request;  
 String strBackUrl = "http://" + request.getServerName() + ":" + request.getServerPort()  
 + httpRequest.getContextPath() + httpRequest.getServletPath() + "?" + (httpRequest.getQueryString());  
  
 LogService logService = new LogService();  
 String logInfo = logService.logInfo(" ") + strBackUrl + " 跳转到商品种类 " + category;  
 logService.insertLogInfo(account.getUsername(), logInfo);  
 }  
  
 *//跳转页面* request.getRequestDispatcher(*VIEW\_CATEGORY*).forward(request, response);  
 }

在实现小类Product时，先设计Product.jsp作为小类的展示界面，通过ProductID展示不同的Product界面，通过ItemID来获得具体商品的信息。利用ProductDAOImpl来实现从数据库中获取该小类的信息，最后通过ViewProductServlet来实现界面交互。



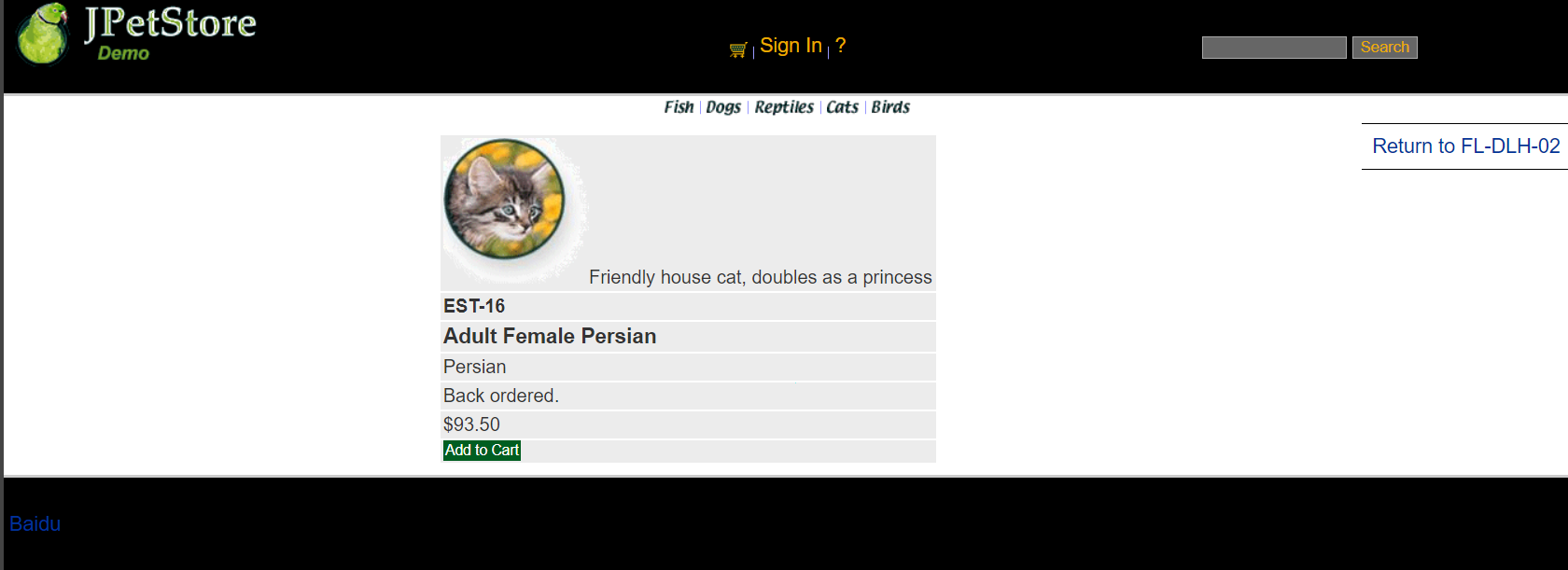
ViewProductServlet为：

public class ViewProductServlet extends HttpServlet {  
 private static final String *VIEW\_PRODUCT* = "/WEB-INF/jsp/catalog/Product.jsp";  
 private String productId;  
  
 protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {  
 doGet(request, response);  
 }  
  
 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);  
  
 *//HttpSession session = request.getSession();* Account account = (Account)session.getAttribute("account");  
  
 if(account != null){  
 HttpServletRequest httpRequest= request;  
 String strBackUrl = "http://" + request.getServerName() + ":" + request.getServerPort()  
 + httpRequest.getContextPath() + httpRequest.getServletPath() + "?" + (httpRequest.getQueryString());  
  
 LogService logService = new LogService();  
 String logInfo = logService.logInfo(" ") + strBackUrl + " 查看产品 " + product;  
 logService.insertLogInfo(account.getUsername(), logInfo);  
 }  
  
 request.getRequestDispatcher(*VIEW\_PRODUCT*).forward(request, response);  
 }  
}

在实现具体商品Item类时，通过ItemID获得当前选择商品的jsp界面，代码如下：

<tr>  
 <td>${sessionScope.product.description}</td>  
</tr>  
<tr>  
 <td><b> ${sessionScope.item.itemId} </b></td>  
</tr>  
<tr>  
 <td><b><font size="4">  
 ${sessionScope.item.attribute1}  
 ${sessionScope.item.attribute2}  
 ${sessionScope.item.attribute3}  
 ${sessionScope.item.attribute4}  
 ${sessionScope.item.attribute5}  
 ${sessionScope.product.name}  
 </font></b></td>  
</tr>

之后完成相关DAO和DAOImpl以及相关的Servlet。

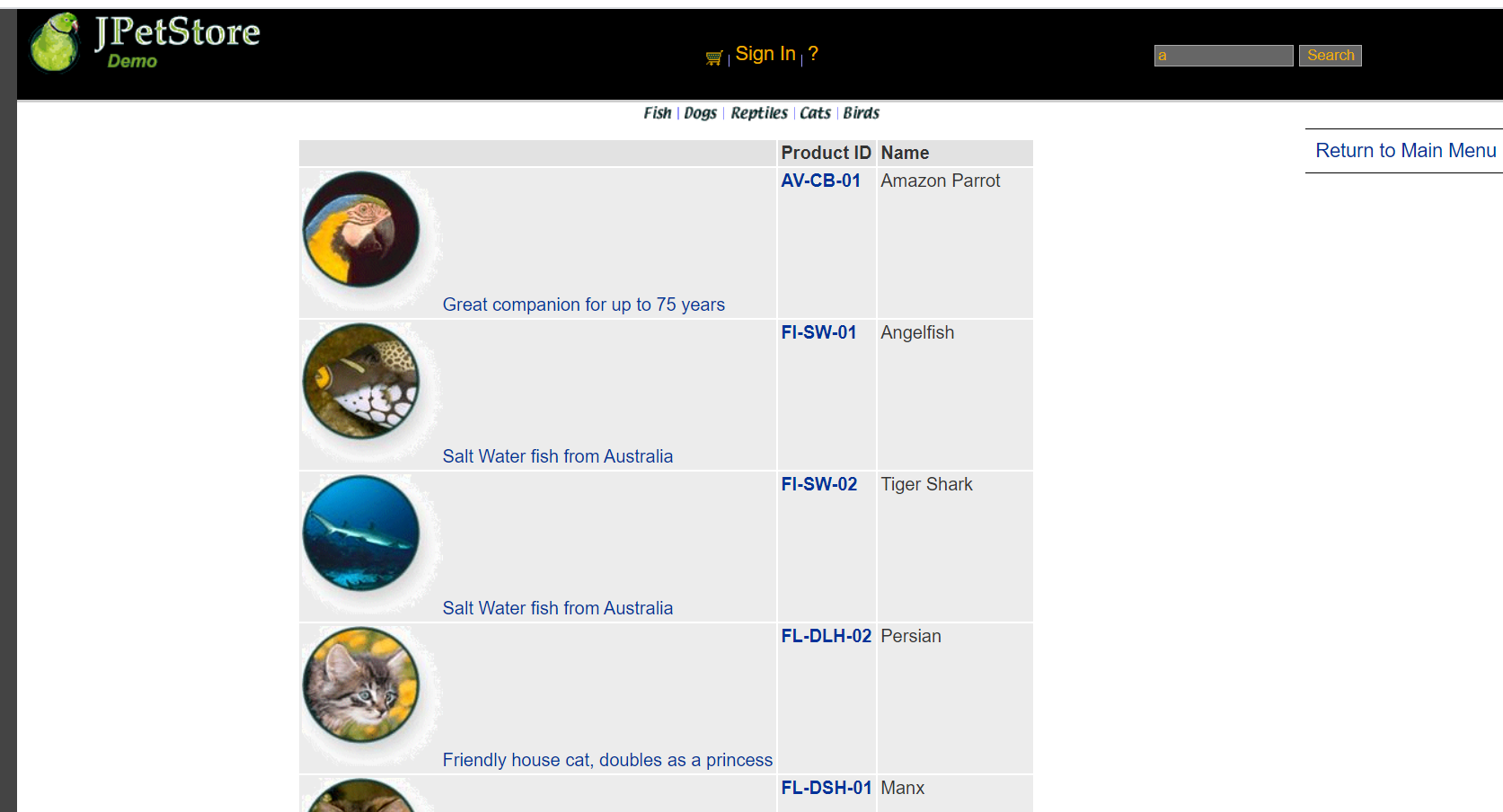


ViewItemServlet如下：

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);  
  
 *//HttpSession session = request.getSession();* Account account = (Account)session.getAttribute("account");  
  
 if(account != null){  
 HttpServletRequest httpRequest= request;  
 String strBackUrl = "http://" + request.getServerName() + ":" + request.getServerPort()  
 + httpRequest.getContextPath() + httpRequest.getServletPath() + "?" + (httpRequest.getQueryString());  
  
 LogService logService = new LogService();  
 String logInfo = logService.logInfo(" ") + strBackUrl + " 查看具体商品 " + item;  
 logService.insertLogInfo(account.getUsername(), logInfo);  
 }  
  
 request.getRequestDispatcher(*VIEW\_ITEM*).forward(request, response);  
 }  
}

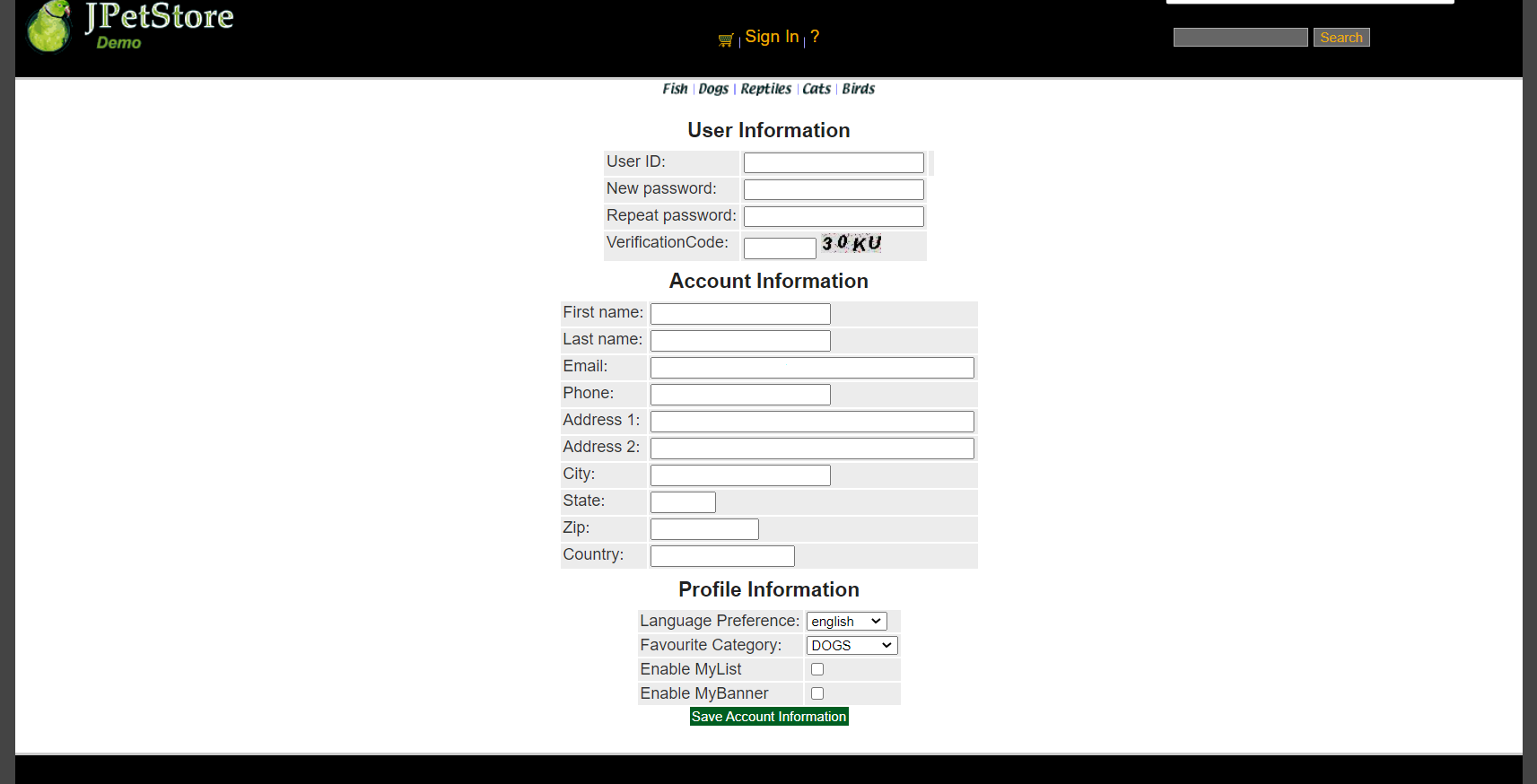
在实现搜索功能时，SearchProducts.jsp为搜索界面的展示界面。通过关键字进行查处，跳转页面，显示所有含有关键字的项目。





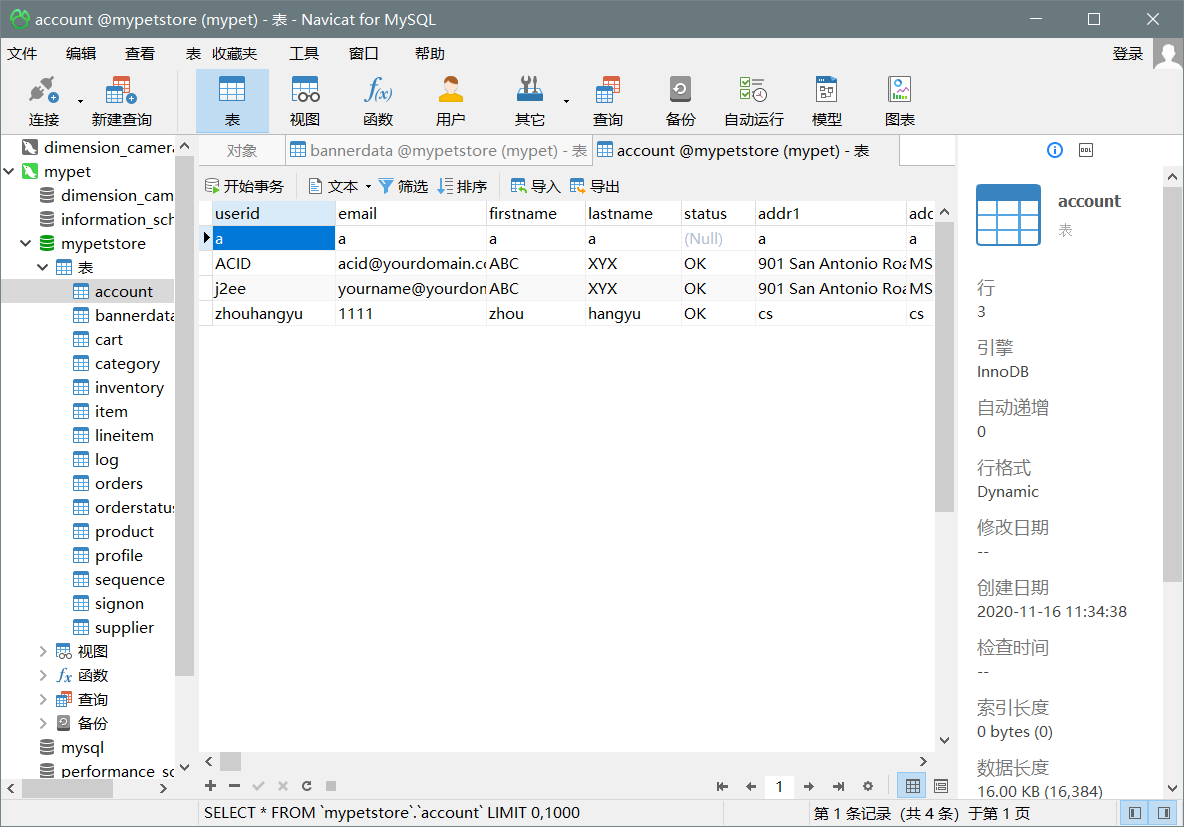
**2、用户管理业务模块，包括用户注册、登录、修改用户信息、查询用户相关订单等业务功能**

1、用户注册：



当用户注册时，需要创建一个用户注册表单界面NewAccountForm.jsp，创建界面完成后根据需求设计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，再将该账号的相关信息存入数据库中。

数据库如下所示：



注册界面有两个Servlet，分别为NewAccountServlet和NewAccountFormServlet。其中NewAccountFormServlet实现的是跳转到注册界面，NewAccountServlet实现的是将注册信息的保存和跳转主界面。

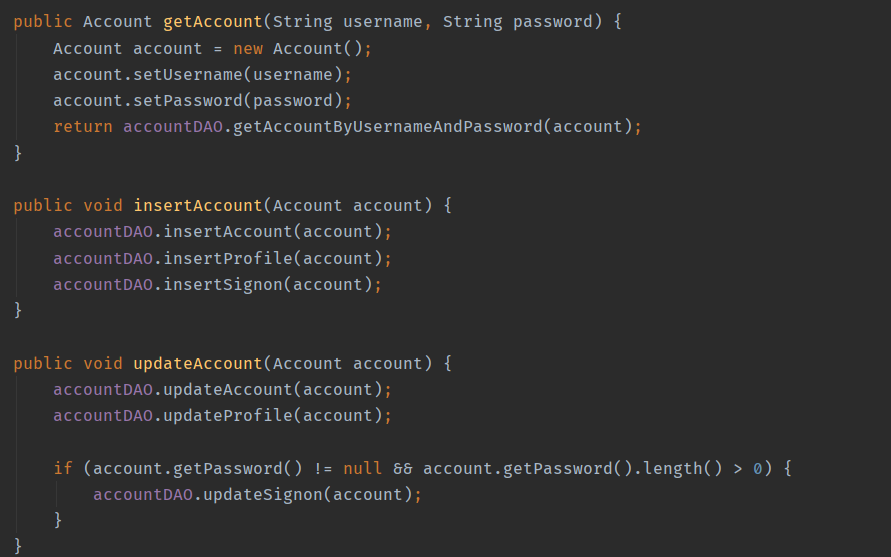
NewAccountServlet：

public class NewAccountServlet extends HttpServlet {  
 private static final String *MAIN* = "/WEB-INF/jsp/catalog/Main.jsp";  
 private static final String *NEWACCOUNTFORM* = "/WEB-INF/jsp/account/NewAccountForm.jsp";  
  
 private Account account;  
 private Account account1;  
 private 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();  
 account = (Account) session.getAttribute("account");  
 account = null;  
 session.setAttribute("account", account);  
  
 *//获得输入的验证码值* String value1=request.getParameter("vCode");  
 /\*获取图片的值\*/  
 String value2=(String)session.getAttribute("checkcode");  
 Boolean isSame = false;  
 /\*对比两个值（字母不区分大小写）\*/  
 if(value2.equalsIgnoreCase(value1)){  
 isSame = true;  
 }  
  
 String username = request.getParameter("username");  
 String password = request.getParameter("password");  
 String firstName = request.getParameter("firstName");  
 String lastName = request.getParameter("lastName");  
 String email = request.getParameter("email");  
 String phone = request.getParameter("phone");  
 String address1 = request.getParameter("address1");  
 String address2 = request.getParameter("address2");  
 String city = request.getParameter("city");  
 String state = request.getParameter("state");  
 String zip = request.getParameter("zip");  
 String country = request.getParameter("country");  
 String languagePreference = request.getParameter("languagePreference");  
 String favouriteCategoryId = request.getParameter("favouriteCategoryId");  
 String listOption = request.getParameter("listOption");  
 String bannerOption = request.getParameter("bannerOption");account1.setPassword(password);  
 account1.setFirstName(firstName);  
 account1.setLastName(lastName);  
 account1.setEmail(email);  
 account1.setPhone(phone);  
 account1.setAddress1(address1);  
 account1.setAddress2(address2);  
 account1.setCity(city);  
 account1.setState(state);  
 account1.setZip(zip);  
 account1.setCountry(country);  
 account1.setLanguagePreference(languagePreference);  
 account1.setFavouriteCategoryId(favouriteCategoryId);  
 account1.setListOption(Boolean.*parseBoolean*(listOption));  
 account1.setBannerOption(Boolean.*parseBoolean*(bannerOption));  
  
  
  
 if(isSame){  
 accountService = new AccountService();  
 accountService.insertAccount(account1);  
  
 if(account1 != null){  
 HttpServletRequest httpRequest= request;  
 String strBackUrl = "http://" + request.getServerName() + ":" + request.getServerPort()  
 + httpRequest.getContextPath() + httpRequest.getServletPath() + "?" + (httpRequest.getQueryString());  
  
 LogService logService = new LogService();  
 String logInfo = logService.logInfo(" ") + strBackUrl + " 注册新账号";  
 logService.insertLogInfo(account1.getUsername(), logInfo);  
 }  
  
 request.getRequestDispatcher(*MAIN*).forward(request, response);  
 }else{  
 session.setAttribute("messageAccount", "Invalid Verification Code.");  
  
 if(account1 != null){  
 HttpServletRequest httpRequest= request;  
 String strBackUrl = "http://" + request.getServerName() + ":" + request.getServerPort()  
 + httpRequest.getContextPath() + httpRequest.getServletPath() + "?" + (httpRequest.getQueryString());  
  
 LogService logService = new LogService();  
 String logInfo = logService.logInfo(" ") + strBackUrl + " 注册账号，验证码错误";  
 logService.insertLogInfo(account1.getUsername(), logInfo);  
 }  
  
 request.getRequestDispatcher(*NEWACCOUNTFORM*).forward(request, response);  
 }  
 }  
}

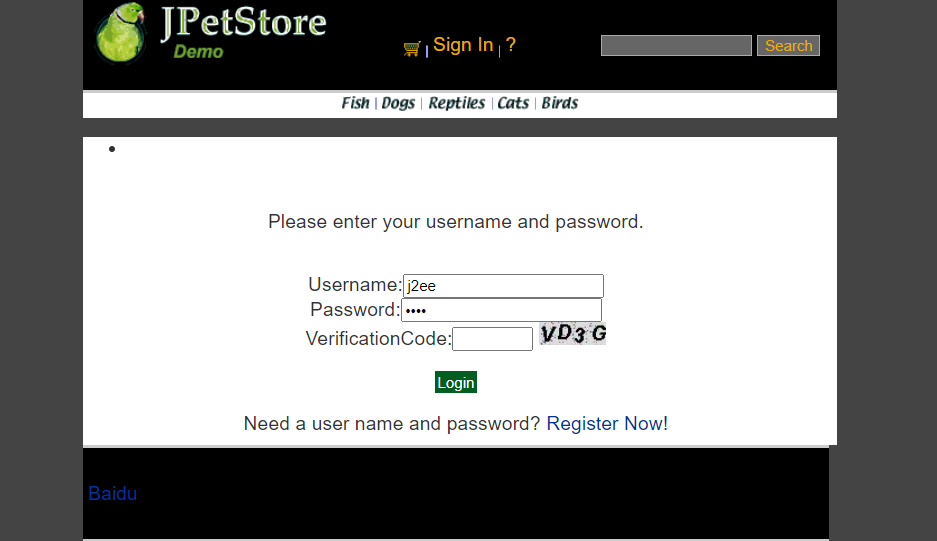
NewAccountFormServlet：

public class NewAccountFormServlet extends HttpServlet {  
  
 private static final String *NEWACCOUNTFORM* = "/WEB-INF/jsp/account/NewAccountForm.jsp";  
  
 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();  
 Account account = (Account)session.getAttribute("account");  
  
 if(account != null){  
 HttpServletRequest httpRequest= request;  
 String strBackUrl = "http://" + request.getServerName() + ":" + request.getServerPort()  
 + httpRequest.getContextPath() + httpRequest.getServletPath() + "?" + (httpRequest.getQueryString());  
  
 LogService logService = new LogService();  
 String logInfo = logService.logInfo(" ") + strBackUrl + " 跳转到注册新账号界面";  
 logService.insertLogInfo(account.getUsername(), logInfo);  
 }  
  
 request.getRequestDispatcher(*NEWACCOUNTFORM*).forward(request, response);  
 }  
}

在AccountService中需要完成的操作有获取、增加和更新用户操作。



2、用户登录



用户登录先完成SignonForm.jsp，能够使用户输入登入的账号与密码，通过SignOnFormServlet来实现点击登录按钮跳转到登录界面，SignOnServlet实现获取输入的用户名和密码并和数据库中的信息相比较，实现登录功能。

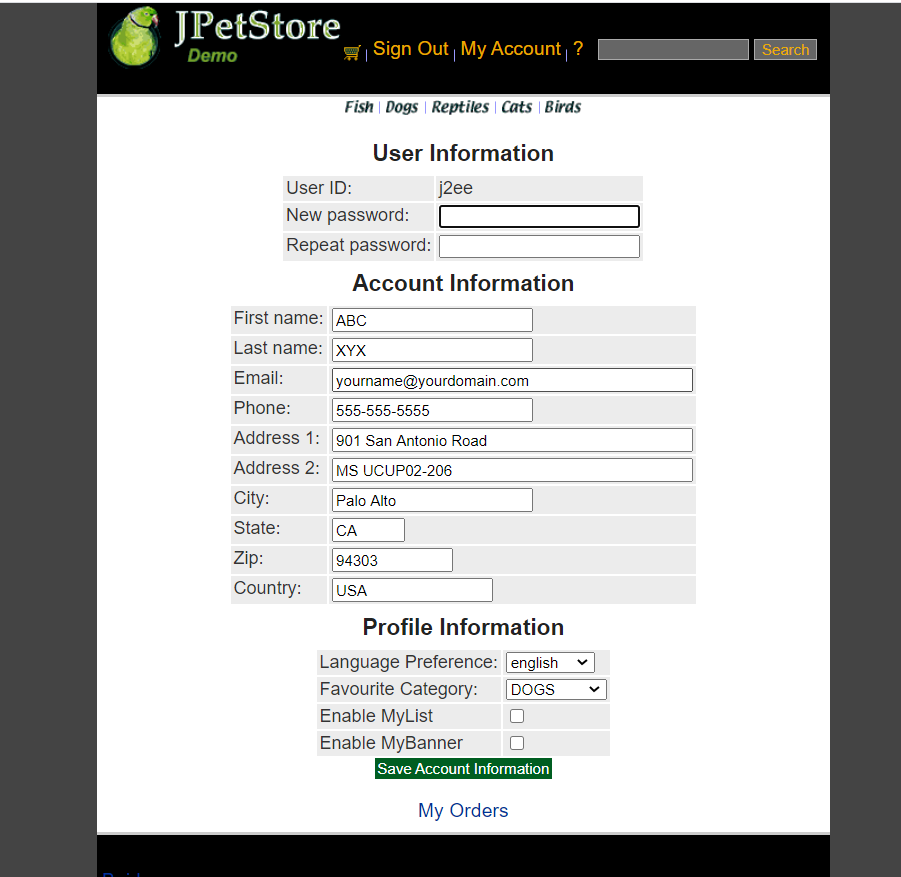
SignOnFormServlet：

public class SignOnFormServlet extends HttpServlet {  
  
 private static final String *SIGNONFORM* = "/WEB-INF/jsp/account/SignonForm.jsp";  
  
 private Account account;  
  
 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();  
 account = (Account)session.getAttribute("account");  
  
 if(account != null){  
 HttpServletRequest httpRequest= request;  
 String strBackUrl = "http://" + request.getServerName() + ":" + request.getServerPort()  
 + httpRequest.getContextPath() + httpRequest.getServletPath() + "?" + (httpRequest.getQueryString());  
  
 LogService logService = new LogService();  
 String logInfo = logService.logInfo(" ") + strBackUrl + " 用户进入登录界面";  
 logService.insertLogInfo(account.getUsername(), logInfo);  
 }  
  
 request.getRequestDispatcher(*SIGNONFORM*).forward(request, response);  
 }  
}

SignOnServlet：

public class SignOnServlet extends HttpServlet {  
 private static final String *MAIN* = "/WEB-INF/jsp/catalog/Main.jsp";  
 private static final String *SIGNONFORM* = "/WEB-INF/jsp/account/SignonForm.jsp";  
  
 private Account account;  
 private 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 {  
  
 String username = request.getParameter("username");  
 String password = request.getParameter("password");  
  
 accountService = new AccountService();  
 account = accountService.getAccount(username, password);  
  
 HttpSession session = request.getSession();  
 session.setAttribute("account", account);  
  
 if(account != null){  
 HttpServletRequest httpRequest= request;  
 String strBackUrl = "http://" + request.getServerName() + ":" + request.getServerPort()  
 + httpRequest.getContextPath() + httpRequest.getServletPath() + "?" + (httpRequest.getQueryString());  
  
 LogService logService = new LogService();  
 String logInfo = logService.logInfo(" ") + strBackUrl + " 用户登录";  
 logService.insertLogInfo(account.getUsername(), logInfo);  
 }  
  
 *//获得输入的验证码值* String value1=request.getParameter("vCode");  
 /\*获取图片的值\*/  
 String value2=(String)session.getAttribute("checkcode");  
 Boolean isSame = false;  
 /\*对比两个值（字母不区分大小写）\*/  
 if(value2.equalsIgnoreCase(value1)){  
 isSame = true;  
 }  
  
  
 if (account == null || !isSame){  
 if(!isSame){  
 session.setAttribute("messageSignOn", "Invalid Verification Code. Signon failed.");  
 }else{  
 session.setAttribute("messageSignOn", "Invalid username or password. Signon failed.");  
 }  
 session.setAttribute("account", null);  
 request.getRequestDispatcher(*SIGNONFORM*).forward(request, response);  
 }else{  
 account.setPassword(null);  
 request.getRequestDispatcher(*MAIN*).forward(request, response);  
 }  
 }  
}

3、用户修改信息：



当用户点击My Account时进入账号界面，在此界面可以修改账户信息，所以第一步需要设计一个EditAccountForm.jsp。完成修改后点击Save保存信息修改。

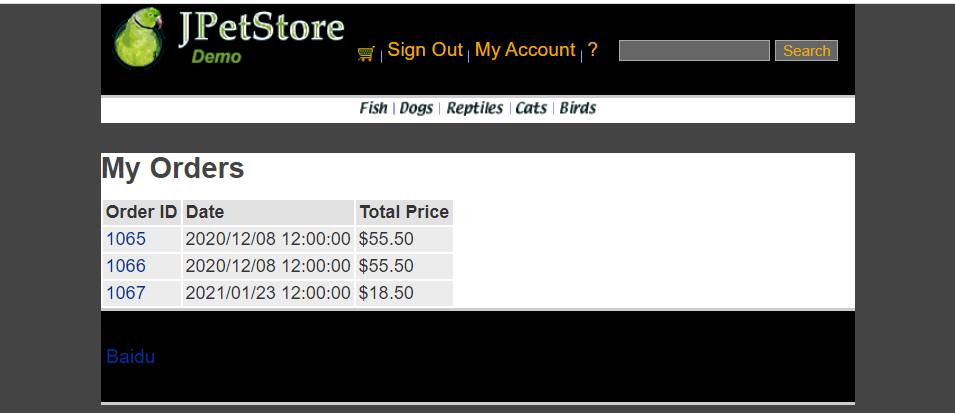
实现进入账户界面的Servlet为EditAccountServlet：

public class EditAccountServlet extends HttpServlet {  
  
 private static final String *EDITACOUNT* = "/WEB-INF/jsp/account/EditAccountForm.jsp";  
  
 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();  
 Account account = (Account)session.getAttribute("account");  
  
 if(account != null){  
 HttpServletRequest httpRequest= request;  
 String strBackUrl = "http://" + request.getServerName() + ":" + request.getServerPort()  
 + httpRequest.getContextPath() + httpRequest.getServletPath() + "?" + (httpRequest.getQueryString());  
  
 LogService logService = new LogService();  
 String logInfo = logService.logInfo(" ") + strBackUrl + " 跳转到编辑账号信息界面";  
 logService.insertLogInfo(account.getUsername(), logInfo);  
 }  
  
 request.getRequestDispatcher(*EDITACOUNT*).forward(request, response);  
 }  
}

在修改界面中保存信息的Servlet为SaveAccountServlet：

public class SaveAccountServlet extends HttpServlet {  
  
 private static final String *EDITACOUNT* = "/WEB-INF/jsp/account/EditAccountForm.jsp";  
  
 private Account account;  
 private 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();  
 account = (Account) session.getAttribute("account");  
  
 String username = account.getUsername();  
 String password = request.getParameter("password");  
 String firstName = request.getParameter("firstName");  
 String lastName = request.getParameter("lastName");  
 String email = request.getParameter("email");  
 String phone = request.getParameter("phone");  
 String address1 = request.getParameter("address1");  
 String address2 = request.getParameter("address2");  
 String city = request.getParameter("city");  
 String state = request.getParameter("state");  
 String zip = request.getParameter("zip");  
 String country = request.getParameter("country");  
 String languagePreference = request.getParameter("languagePreference");  
 String favouriteCategoryId = request.getParameter("favouriteCategoryId");  
 String listOption = request.getParameter("listOption");  
 String bannerOption = request.getParameter("bannerOption");  
  
 account.setUsername(username);  
 account.setPassword(password);  
 account.setFirstName(firstName);  
 account.setLastName(lastName);  
 account.setEmail(email);  
 account.setPhone(phone);  
 account.setAddress1(address1);  
 account.setAddress2(address2);  
 account.setCity(city);  
 account.setState(state);  
 account.setZip(zip);  
 account.setCountry(country);  
 account.setLanguagePreference(languagePreference);  
 account.setFavouriteCategoryId(favouriteCategoryId);  
 account.setListOption(Boolean.*parseBoolean*(listOption));  
 account.setBannerOption(Boolean.*parseBoolean*(bannerOption));  
  
 accountService = new AccountService();  
 accountService.updateAccount(account);  
  
 session.setAttribute("account", account);  
  
 if(account != null){  
 HttpServletRequest httpRequest= request;  
 String strBackUrl = "http://" + request.getServerName() + ":" + request.getServerPort()  
 + httpRequest.getContextPath() + httpRequest.getServletPath() + "?" + (httpRequest.getQueryString());  
  
 LogService logService = new LogService();  
 String logInfo = logService.logInfo(" ") + strBackUrl + " 账号信息更改";  
 logService.insertLogInfo(account.getUsername(), logInfo);  
 }  
  
 request.getRequestDispatcher(*EDITACOUNT*).forward(request, response);  
 }  
}

3、用户订单查询：



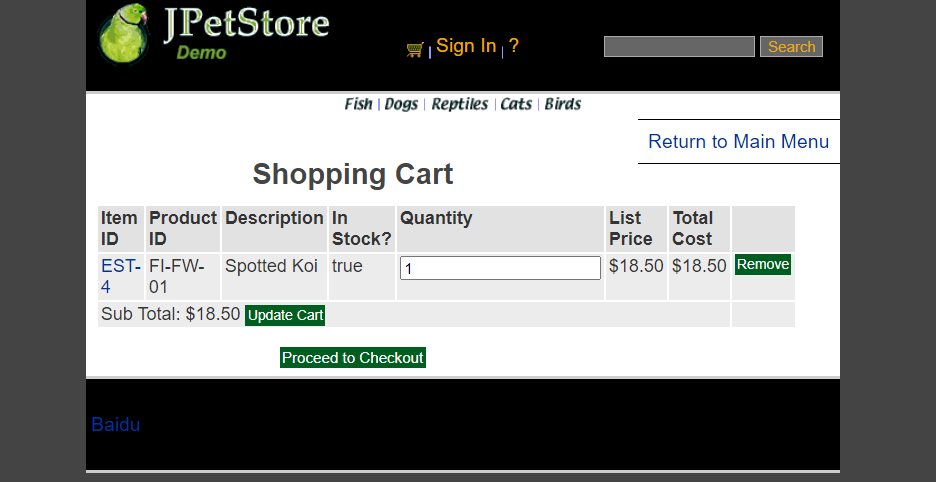
点击my order可以显示该账号的账单。首先设计ListOrders.jsp，接下来完成Order.java的基础模块。创建OrderDAO接口，设计相关的数据库操作。在OrderDAOImpl中完成数据库的具体操作。接下来实现OrderService，成功完成业务层调用。最后通过ViewListOrderServlet实现用户可以从数据库中查看到自己的订单信息。

ViewListOrderServlet为：

public class ViewListOrderServlet extends HttpServlet {  
 private static final String *VIEWLISTORDER* = "/WEB-INF/jsp/order/ListOrders.jsp";  
  
 private String username;  
 private OrderService orderService;  
 private List<Order> orderList = new ArrayList<Order>();  
  
 protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {  
 doGet(request, response);  
 }  
  
 protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {  
 username = request.getParameter("username");  
 orderService = new OrderService();  
 orderList = orderService.getOrdersByUsername(username);  
  
 HttpSession session = request.getSession();  
 session.setAttribute("orderList", orderList);  
  
 *//HttpSession session = request.getSession();* Account account = (Account)session.getAttribute("account");  
  
 if(account != null){  
 HttpServletRequest httpRequest= request;  
 String strBackUrl = "http://" + request.getServerName() + ":" + request.getServerPort()  
 + httpRequest.getContextPath() + httpRequest.getServletPath() + "?" + (httpRequest.getQueryString());  
  
 LogService logService = new LogService();  
 String logInfo = logService.logInfo(" ") + strBackUrl + " 查看订单 " + orderList;  
 logService.insertLogInfo(account.getUsername(), logInfo);  
 }  
  
 request.getRequestDispatcher(*VIEWLISTORDER*).forward(request, response);  
 }  
}

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**

1、添加购物车：



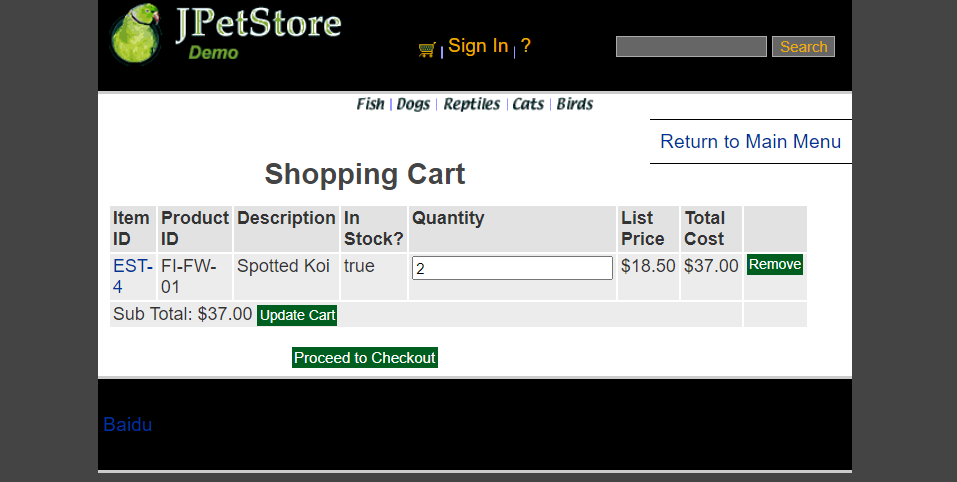
先设计Cart.jsp，用于网页显示购物车栏目信息，接下来具体实现Cart.java使得用户可以创建Cart类。通过AddItemToCartServlet来实现用户将商品添加到购物车。

AddItemToCartServlet：

为了实现后续操作，这里对购物车的操作已改为通过数据库中获取数据而不是从session中获取。

public class AddItemToCartServlet extends HttpServlet {  
 *//Servlet的功能即负责中转  
 //1.处理完请求后的跳转页面* private static final String *VIEW\_CART* = "/WEB-INF/jsp/cart/Cart.jsp";  
  
 *//2.定义处理该请求所需要的数据* private String workingItemId;  
 private Cart cart;*//购物车* private CartItem cartItem = new CartItem();  
  
 *//3.是否需要调用业务逻辑层* 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 {  
 workingItemId = request.getParameter("workingItemId");  
  
 Account account;  
 *//从对话中，获取购物车* HttpSession session = request.getSession();  
 cart = (Cart)session.getAttribute("cart");  
  
 account = (Account)session.getAttribute("account");  
  
 if(cart == null){  
 *//第一次使用购物车* cart = new Cart();  
 }  
  
 CartItemDAOImpl cartItemDAO = new CartItemDAOImpl();  
 if(cart.containsItemId(workingItemId)){  
 *//已有该物品，数量加一* cart.incrementQuantityByItemId(workingItemId);  
  
 if(account != null){  
 HttpServletRequest httpRequest= (HttpServletRequest) request;  
 String strBackUrl = "http://" + request.getServerName() + ":" + request.getServerPort()  
 + httpRequest.getContextPath() + httpRequest.getServletPath() + "?" + (httpRequest.getQueryString());  
  
 LogService logService = new LogService();  
 Item item = catalogService.getItem(workingItemId);  
 String logInfo = logService.logInfo(" ") + strBackUrl + " " + item + "数量+1 ";  
 logService.insertLogInfo(account.getUsername(), logInfo);  
 }  
 }else{  
  
 catalogService = new CatalogService();  
 boolean isInStock = catalogService.isItemInStock(workingItemId);  
 Item item = catalogService.getItem(workingItemId);  
 cart.addItem(item, isInStock);  
  
 cartItem.itemId=workingItemId;  
  
 *//明天记得改product Id* cartItem.productId=item.getProduct().getProductId();  
  
 cartItem.description = item.getAttribute1();  
 cartItem.instock = isInStock?1:0;  
  
 cartItem.quantity = 1;  
  
 cartItem.price = item.getListPrice().floatValue();  
  
 cartItem.total1 = item.getListPrice().floatValue();  
  
 cartItemDAO.insertCart(cartItem);  
  
  
 session.setAttribute("cart", cart);  
  
 if(account != null){  
 HttpServletRequest httpRequest= request;  
 String strBackUrl = "http://" + request.getServerName() + ":" + request.getServerPort()  
 + httpRequest.getContextPath() + httpRequest.getServletPath() + "?" + (httpRequest.getQueryString());  
  
 LogService logService = new LogService();  
 String logInfo = logService.logInfo(" ") + strBackUrl + " 添加物品 " + item + " 到购物车";  
 logService.insertLogInfo(account.getUsername(), logInfo);  
 }  
  
  
 request.getRequestDispatcher(*VIEW\_CART*).forward(request, response);  
 }  
 }  
}

2、更新购物车中的商品：



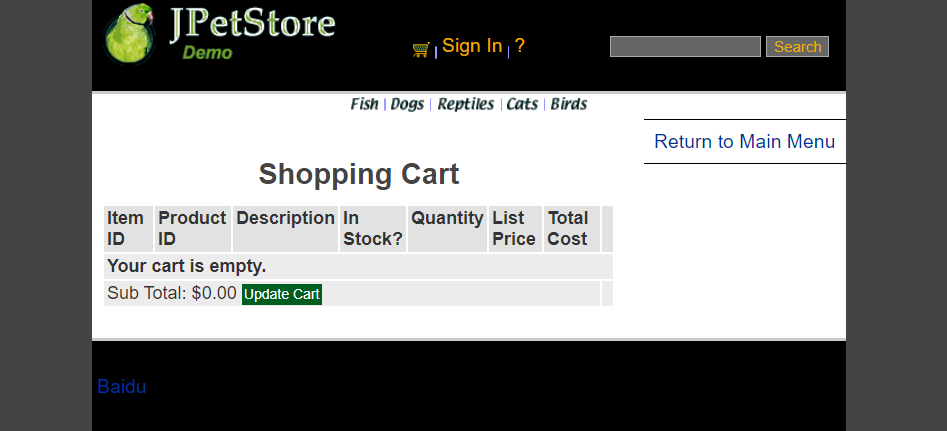
当修改Quantity的值时，点击Update，可以通过利用UpdateCartQuantitiesServlet实现信息更新以及页面更新，传递商品的数量信息获取后进行相关操作。

由于后续操作，UpdateCartQuantitiesServlet已经改为改变会持续化保持在数据库中，不再从session中获取

UpdateCartQuantitiesServlet：

public class UpdateCartQuantitiesServlet extends HttpServlet {  
  
 private static final String *VIEW\_CART* = "/WEB-INF/jsp/cart/Cart.jsp";  
  
 private String workingItemId;  
 private Cart cart;  
 private CartItem cartItem = new CartItem();  
  
 protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {  
 doGet(request, response);  
 }  
  
 protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {  
 workingItemId = request.getParameter("workingItemId");  
 CatalogService catalogService = new CatalogService();  
 CartItemDAOImpl cartItemDAO = new CartItemDAOImpl();  
  
 *//从对话中，获取购物车* HttpSession session = request.getSession();  
 cart = (Cart)session.getAttribute("cart");  
  
 Iterator<CartItem> cartItemIterator = cart.getAllCartItems();  
  
 while (cartItemIterator.hasNext()){  
 *//产品数量增加* CartItem cartItem = (CartItem)cartItemIterator.next();  
 String itemId = cartItem.getItem().getItemId();  
  
 try {  
 int quantity = Integer.*parseInt*((String) request.getParameter(itemId));  
 cart.setQuantityByItemId(itemId, quantity);  
  
 float total=cartItem.getItem().getListPrice().floatValue()\*quantity;  
 if (quantity < 1) {  
 cartItemIterator.remove();  
 cartItemDAO.removeItemById(itemId);  
 }  
 cartItemDAO.update(itemId,quantity,total);  
 } catch (Exception e) {  
 *//ignore parse exceptions on purpose* e.printStackTrace();  
 }  
  
 *//CartItem cartItem = cartItemIterator.next();  
 //cartItem.incrementQuantity();* }  
  
 session.setAttribute("cart", cart);  
  
 *//HttpSession session = request.getSession();* Account account = (Account)session.getAttribute("account");  
  
 if(account != null){  
 HttpServletRequest httpRequest= request;  
 String strBackUrl = "http://" + request.getServerName() + ":" + request.getServerPort()  
 + httpRequest.getContextPath() + httpRequest.getServletPath() + "?" + (httpRequest.getQueryString());  
  
 LogService logService = new LogService();  
 String logInfo = logService.logInfo(" ") + strBackUrl + " 更新购物车商品数量";  
 logService.insertLogInfo(account.getUsername(), logInfo);  
 }  
  
 request.getRequestDispatcher(*VIEW\_CART*).forward(request, response);  
 }  
}

3、从购物车中移除数据：



点击remove按钮获取购物车中的信息，之后进入RemoveItemFromCartServlet进行删除操作，并更新该购物车中的信息，如果误操作，则提示该误操作信息。

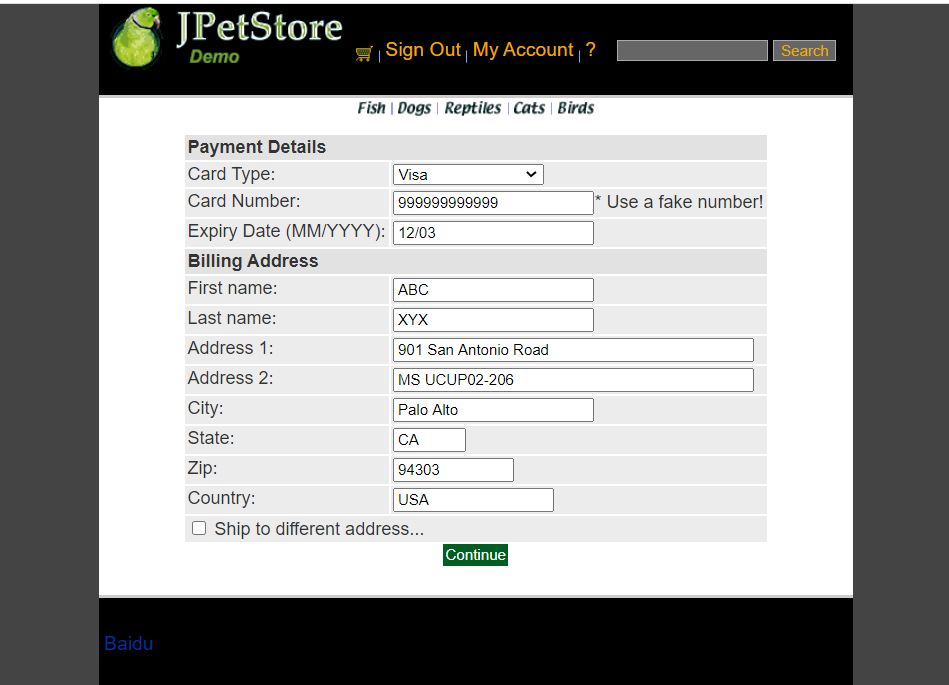
由于后续操作，RemoveItemFromCartServlet已经改为改变会持续化保持在数据库中，不再从session中获取

RemoveItemFromCartServlet：

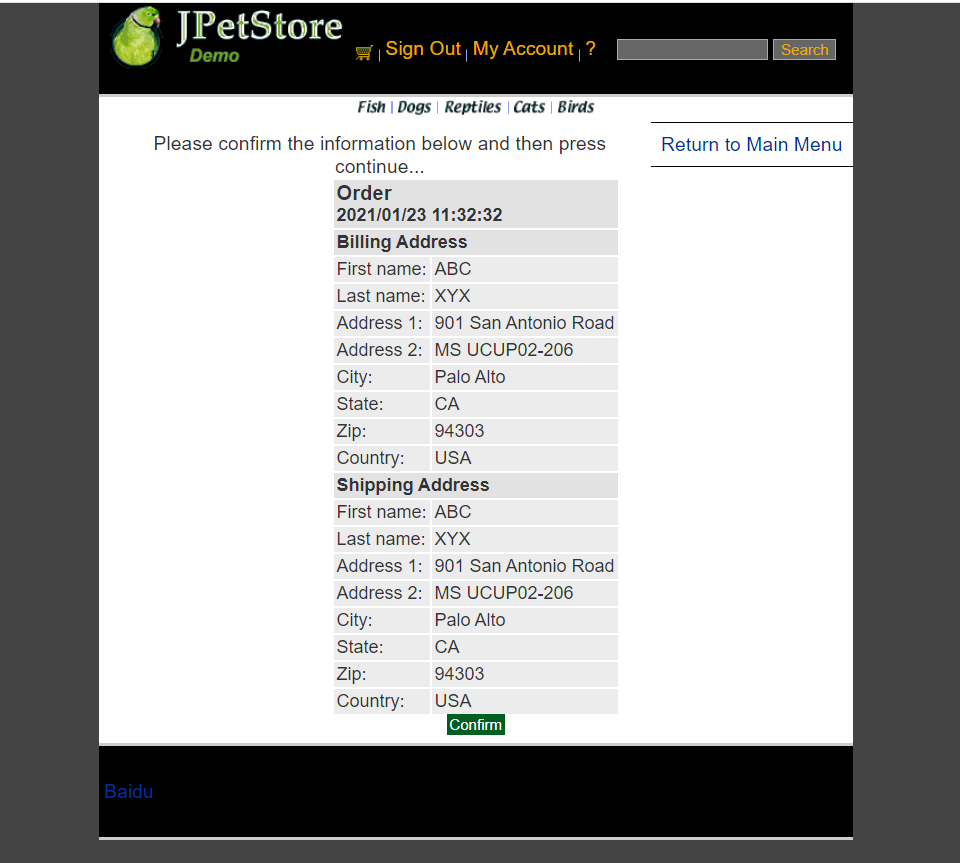
public class RemoveItemFromCartServlet extends HttpServlet {  
  
 private static final String *VIEW\_CART* = "/WEB-INF/jsp/cart/Cart.jsp";  
 private static final String *ERROR* = "/WEB-INF/jsp/common/Error.jsp";  
  
 private String workingItemId;  
 private Cart cart;  
 private CartItem cartItem = new CartItem();  
  
 protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {  
 doGet(request, response);  
 }  
  
 protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {  
 CartItemDAOImpl cartItemDAO = new CartItemDAOImpl();  
 *//获取jsp中workingItemId的值* workingItemId = request.getParameter("workingItemId");  
  
 HttpSession session = request.getSession();  
 cart = (Cart)session.getAttribute("cart");  
  
 Item item = cart.removeItemById(workingItemId);  
  
 if(item == null) {  
 session.setAttribute("message", "Attempted to remove null CartItem from Cart.");  
  
 Account account = (Account)session.getAttribute("account");  
  
 if(account != null){  
 HttpServletRequest httpRequest= request;  
 String strBackUrl = "http://" + request.getServerName() + ":" + request.getServerPort()  
 + httpRequest.getContextPath() + httpRequest.getServletPath() + "?" + (httpRequest.getQueryString());  
  
 LogService logService = new LogService();  
 String logInfo = logService.logInfo(" ") + strBackUrl + " 物品为空，不能移除";  
 logService.insertLogInfo(account.getUsername(), logInfo);  
 }  
  
 request.getRequestDispatcher(*ERROR*).forward(request, response);  
 }else{  
  
 Account account = (Account)session.getAttribute("account");  
  
 if(account != null){  
 HttpServletRequest httpRequest= request;  
 String strBackUrl = "http://" + request.getServerName() + ":" + request.getServerPort()  
 + httpRequest.getContextPath() + httpRequest.getServletPath() + "?" + (httpRequest.getQueryString());  
  
 LogService logService = new LogService();  
 String logInfo = logService.logInfo(" ") + strBackUrl + " " + item + " 已从购物车中移除";  
 logService.insertLogInfo(account.getUsername(), logInfo);  
 }  
  
 cartItemDAO.removeItemById(item.getItemId());  
  
 request.getRequestDispatcher(*VIEW\_CART*).forward(request, response);  
 }  
 }  
}

4、新增订单：

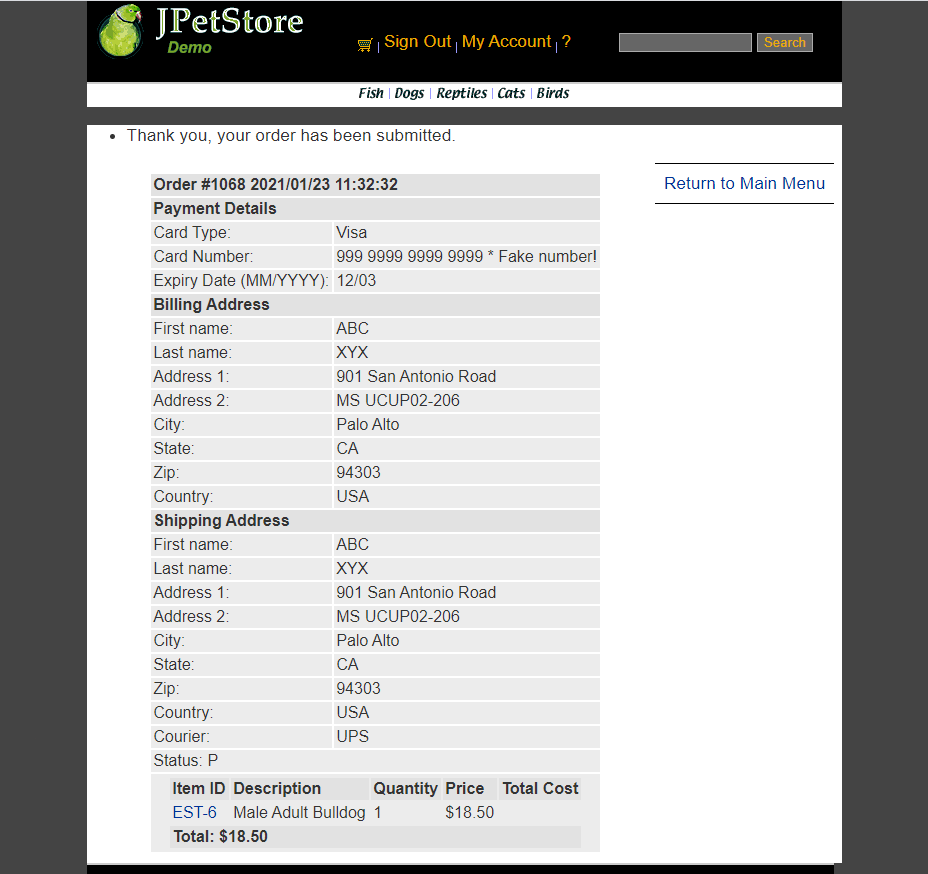
NewOrderForm.jsp



ConfirmOrder.jsp



ViewOrder.jsp



生成订单界面，包括修改地址信息时共有三个页面，第一个页面是购买方的支付与寄件地址信息确认，即NewOrderForm.jsp，第二个为确认后是发货人地址与收货人地址信息确认的界面ConfirmOrder.jsp，最后为曲儿之后即可生成最终订单ViewOrder.jsp。

NewOrderFormServlet：

public class NewOrderFormServlet extends HttpServlet {  
 private static final String *NEW\_ORDER* = "/WEB-INF/jsp/order/NewOrderForm.jsp";  
 private static final String *SIGNONFORM* = "/WEB-INF/jsp/account/SignonForm.jsp";  
 private static final String *ERROR* = "/WEB-INF/jsp/common/Error.jsp";  
  
 private Account account;  
 private Order order;  
 private Cart cart;  
  
 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();  
 account = (Account)session.getAttribute("account");  
 cart = (Cart)session.getAttribute("cart");  
  
 if (account == null){  
 session.setAttribute("message", "You must sign on before attempting to check out. Please sign on and try checking out again.");  
 request.getRequestDispatcher(*SIGNONFORM*).forward(request, response);  
 } else if(cart != null){  
 order = new Order();  
 order.initOrder(account, cart);  
 session.setAttribute("order", order);  
  
 Account account = (Account)session.getAttribute("account");  
  
 if(account != null){  
 HttpServletRequest httpRequest= request;  
 String strBackUrl = "http://" + request.getServerName() + ":" + request.getServerPort()  
 + httpRequest.getContextPath() + httpRequest.getServletPath() + "?" + (httpRequest.getQueryString());  
  
 LogService logService = new LogService();  
 String logInfo = logService.logInfo(" ") + strBackUrl + " 跳转到新订单页面";  
 logService.insertLogInfo(account.getUsername(), logInfo);  
 }  
  
 request.getRequestDispatcher(*NEW\_ORDER*).forward(request, response);  
 }else{  
 session.setAttribute("message", "An order could not be created because a cart could not be found.");  
  
 Account account = (Account)session.getAttribute("account");  
  
 if(account != null){  
 HttpServletRequest httpRequest= request;  
 String strBackUrl = "http://" + request.getServerName() + ":" + request.getServerPort()  
 + httpRequest.getContextPath() + httpRequest.getServletPath() + "?" + (httpRequest.getQueryString());  
  
 LogService logService = new LogService();  
 String logInfo = logService.logInfo(" ") + strBackUrl + " 生成订单页面信息错误";  
 logService.insertLogInfo(account.getUsername(), logInfo);  
 }  
  
 request.getRequestDispatcher(*ERROR*).forward(request, response);  
 }  
 }  
}

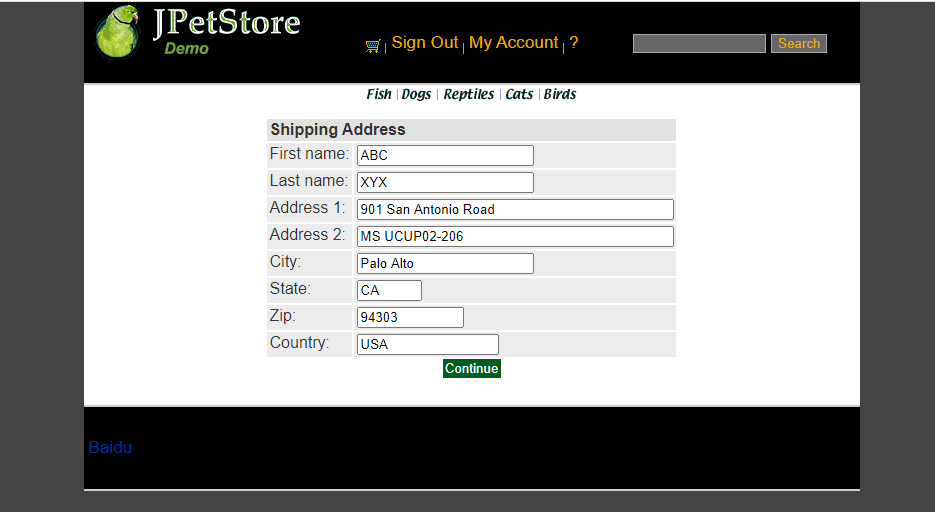
ConfirmOrderFormServlet：

public class ConfirmOrderFormServlet extends HttpServlet {  
 private static final String *CONFIRM\_ORDER\_FORM* = "/WEB-INF/jsp/order/ConfirmOrder.jsp";  
 private static final String *SHIPPINGFORM* = "/WEB-INF/jsp/order/ShippingForm.jsp";  
  
 private String shippingAddressRequired;  
 private Order order;  
 private OrderService orderService;  
  
 protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {  
 doGet(request, response);  
 }  
  
 protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {  
 shippingAddressRequired = request.getParameter("shippingAddressRequired");  
 order = new Order();  
  
 HttpSession session = request.getSession();  
 order = (Order)session.getAttribute("order");  
 Account account = (Account)session.getAttribute("account");  
  
 if (shippingAddressRequired == null){  
 if(account != null){  
 HttpServletRequest httpRequest= request;  
 String strBackUrl = "http://" + request.getServerName() + ":" + request.getServerPort()  
 + httpRequest.getContextPath() + httpRequest.getServletPath() + "?" + (httpRequest.getQueryString());  
  
 LogService logService = new LogService();  
 String logInfo = logService.logInfo(" ") + strBackUrl + " 确认生成订单 ";  
 logService.insertLogInfo(account.getUsername(), logInfo);  
 }  
 CatalogService catalogService=new CatalogService();  
 CartItemDAOImpl cartItemDAO = new CartItemDAOImpl();  
 List<CartItem> cartList= cartItemDAO.get();  
 for(int i=0;i<cartList.size();i++){  
 cartItemDAO.removeItemById(cartList.get(i).getItemId());  
 }  
  
 request.getRequestDispatcher(*CONFIRM\_ORDER\_FORM*).forward(request, response);  
 }  
 else{  
 shippingAddressRequired = null;  
  
 if(account != null){  
 HttpServletRequest httpRequest= request;  
 String strBackUrl = "http://" + request.getServerName() + ":" + request.getServerPort()  
 + httpRequest.getContextPath() + httpRequest.getServletPath() + "?" + (httpRequest.getQueryString());  
  
 LogService logService = new LogService();  
 String logInfo = logService.logInfo(" ") + strBackUrl + " 更改收货地址";  
 logService.insertLogInfo(account.getUsername(), logInfo);  
 }  
  
 request.getRequestDispatcher(*SHIPPINGFORM*).forward(request, response);  
 }  
  
  
  
 }  
}

ViewOrderServlet：

public class ViewOrderServlet extends HttpServlet {  
 private static final String *VIEWORDER* = "/WEB-INF/jsp/order/ViewOrder.jsp";  
 private static final String *ERROR* = "/WEB-INF/jsp/common/Error.jsp";  
  
 private Order order;  
 private OrderService orderService;  
 private Cart cart;  
  
 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();  
 order = (Order) session.getAttribute("order");  
 cart = (Cart) session.getAttribute("cart");  
  
 if (order != null) {  
 orderService = new OrderService();  
 orderService.insertOrder(order);  
 session.setAttribute("order", order);  
 *//清空购物车* cart = null;  
 session.setAttribute("cart", cart);  
  
 session.setAttribute("message", "Thank you, your order has been submitted.");  
  
 *//HttpSession session = request.getSession();* Account account = (Account)session.getAttribute("account");  
  
 if(account != null){  
 HttpServletRequest httpRequest= request;  
 String strBackUrl = "http://" + request.getServerName() + ":" + request.getServerPort()  
 + httpRequest.getContextPath() + httpRequest.getServletPath() + "?" + (httpRequest.getQueryString());  
  
 LogService logService = new LogService();  
 String logInfo = logService.logInfo(" ") + strBackUrl + " 查看订单 " + order;  
 logService.insertLogInfo(account.getUsername(), logInfo);  
 }  
  
 request.getRequestDispatcher(*VIEWORDER*).forward(request, response);  
 } else {  
 session.setAttribute("message", "An error occurred processing your order (order was null).");  
 request.getRequestDispatcher(*ERROR*).forward(request, response);  
 }  
 }  
}

5、修改地址：



生成订单的第一个界面时有个更改地址的选择框，选择之后点击Continue即可利用ShippingFormServlet跳转到更改收件人地址的界面ShippingForm.jsp,接下里可以进行收货人地址信息更改，再利用ShippingAdressServlet进行数据更新存入数据库接下来使用

ShippingAdressServlet：

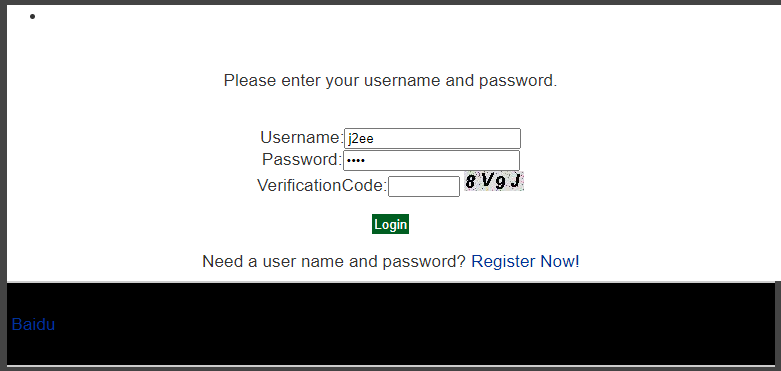
public class ShippingAddressServlet extends HttpServlet {  
 private static final String *CONFIRM\_ORDER\_FORM* = "/WEB-INF/jsp/order/ConfirmOrder.jsp";  
  
 private Order order;  
 private String shipToFirstName;  
 private String shipToLastName;  
 private String shipAddress1;  
 private String shipAddress2;  
 private String shipCity;  
 private String shipState;  
 private String shipZip;  
 private String shipCountry;  
  
 protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {  
 doGet(request, response);  
 }  
  
 protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {  
 shipToFirstName = request.getParameter("shipToFirstName");  
 shipToLastName = request.getParameter("shipToLastName");  
 shipAddress1 = request.getParameter("shipAddress1");  
 shipAddress2 = request.getParameter("shipAddress2");  
 shipCity = request.getParameter("shipCity");  
 shipState = request.getParameter("shipState");  
 shipZip = request.getParameter("shipZip");  
 shipCountry = request.getParameter("shipCountry");  
  
 HttpSession session = request.getSession();  
 order = (Order)session.getAttribute("order");  
  
 order.setShipToFirstName(shipToFirstName);  
 order.setShipToLastName(shipToLastName);  
 order.setShipAddress1(shipAddress1);  
 order.setShipAddress2(shipAddress2);  
 order.setShipCity(shipCity);  
 order.setShipState(shipState);  
 order.setShipZip(shipZip);  
 order.setCourier(shipCountry);  
  
 session.setAttribute("order", order);  
  
 Account account = (Account)session.getAttribute("account");  
  
 if(account != null){  
 HttpServletRequest httpRequest= request;  
 String strBackUrl = "http://" + request.getServerName() + ":" + request.getServerPort()  
 + httpRequest.getContextPath() + httpRequest.getServletPath() + "?" + (httpRequest.getQueryString());  
  
 LogService logService = new LogService();  
 String logInfo = logService.logInfo(" ") + strBackUrl + " 修改订单邮寄地址";  
 logService.insertLogInfo(account.getUsername(), logInfo);  
 }  
  
 request.getRequestDispatcher(*CONFIRM\_ORDER\_FORM*).forward(request, response);  
 }  
}

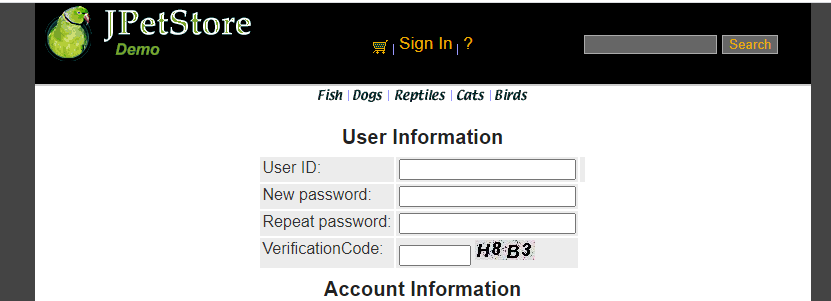
ShippingFormServlet：

public class ShippingFormServlet extends HttpServlet {  
 private static final String *SHIPPINGFORM* = "/WEB-INF/jsp/order/ShippingForm.jsp";  
 private static final String *SIGNONFORM* = "/WEB-INF/jsp/account/SignonForm.jsp";  
  
 private Account account;  
  
 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();  
 account = (Account)session.getAttribute("account");  
  
 if (account == null){  
 request.getRequestDispatcher(*SIGNONFORM*).forward(request, response);  
 } else{  
 *//HttpSession session = request.getSession();* Account account = (Account)session.getAttribute("account");  
  
 if(account != null){  
 HttpServletRequest httpRequest= request;  
 String strBackUrl = "http://" + request.getServerName() + ":" + request.getServerPort()  
 + httpRequest.getContextPath() + httpRequest.getServletPath() + "?" + (httpRequest.getQueryString());  
  
 LogService logService = new LogService();  
 String logInfo = logService.logInfo(" ") + strBackUrl + " 跳转到修改地址界面";  
 logService.insertLogInfo(account.getUsername(), logInfo);  
 }  
  
 request.getRequestDispatcher(*SHIPPINGFORM*).forward(request, response);  
 }  
 }  
}

**二、拓展任务**

1、**添加验证码功能，在用户注册和登录模块中添加验证码**





实现的原理为：设置字符表即26个字母和各个数字，接下来利用字符表产生随机验证码，同时随机产生若干个干扰点，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。

在VerificationCode.java实现随机生成数字和字母组成图片的功能，在SigninForm.jsp 和NewAccountForm.jsp页面中添加实现显示验证码和用户输入验证码的功能，再用SignOnServlet和NewAccountServlet分别获取图片和用户输入的数据，判断两个数据是否一致，之后再进行登录功能与注册功能的实现。

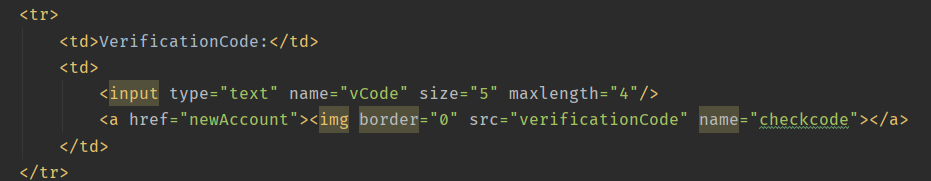
VerificationCode.java：

public class VerificationCode extends HttpServlet {  
 public VerificationCode() {  
 super();  
 }  
  
  
 public void destroy() {  
 super.destroy();  
 }  
  
 public void doGet(HttpServletRequest request, HttpServletResponse response)  
 throws ServletException, IOException {  
  
 doPost(request, response);  
 }  
  
  
 /\*实现的核心代码\*/  
 public void doPost(HttpServletRequest request, HttpServletResponse response)  
 throws ServletException, IOException {  
  
 response.setContentType("image/jpeg");  
 HttpSession session=request.getSession();  
 int width=60;  
 int height=20;  
  
 *//设置浏览器不要缓存此图片* response.setHeader("Pragma", "No-cache");  
 response.setHeader("Cache-Control", "no-cache");  
 response.setDateHeader("Expires", 0);  
  
 *//创建内存图像并获得图形上下文* BufferedImage image=new BufferedImage(width, height,BufferedImage.*TYPE\_INT\_RGB*);  
 Graphics g=image.getGraphics();  
  
 /\*  
 \* 产生随机验证码  
 \* 定义验证码的字符表  
 \*/  
 String chars="0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZ";  
 char[] rands=new char[4];  
 for(int i=0;i<4;i++){  
 int rand=(int) (Math.*random*() \*36);  
 rands[i]=chars.charAt(rand);  
 }  
  
 /\*  
 \* 产生图像  
 \* 画背景  
 \*/  
 g.setColor(new Color(0xDCDCDC));  
 g.fillRect(0, 0, width, height);  
  
 /\*  
 \* 随机产生120个干扰点  
 \*/  
  
 for(int i=0;i<120;i++){  
 int x=(int)(Math.*random*()\*width);  
 int y=(int)(Math.*random*()\*height);  
 int red=(int)(Math.*random*()\*255);  
 int green=(int)(Math.*random*()\*255);  
 int blue=(int)(Math.*random*()\*255);  
 g.setColor(new Color(red,green,blue));  
 g.drawOval(x, y, 1, 0);  
 }  
 g.setColor(Color.*BLACK*);  
 g.setFont(new Font(null, Font.*ITALIC*|Font.*BOLD*,18));  
  
 *//在不同高度输出验证码的不同字符* g.drawString(""+rands[0], 1, 17);  
 g.drawString(""+rands[1], 16, 15);  
 g.drawString(""+rands[2], 31, 18);  
 g.drawString(""+rands[3], 46, 16);  
 g.dispose();  
  
 *//将图像传到客户端* ServletOutputStream sos=response.getOutputStream();  
 ByteArrayOutputStream baos=new ByteArrayOutputStream();  
 ImageIO.*write*(image, "JPEG", baos);  
 byte[] buffer=baos.toByteArray();  
 response.setContentLength(buffer.length);  
 sos.write(buffer);  
 baos.close();  
 sos.close();  
  
 session.setAttribute("checkcode", new String(rands));  
 }  
  
  
 public void init() throws ServletException {  
 *// Put your code here* }  
}

在SignonForm.jsp中：



在NewAccountForm.jsp中：

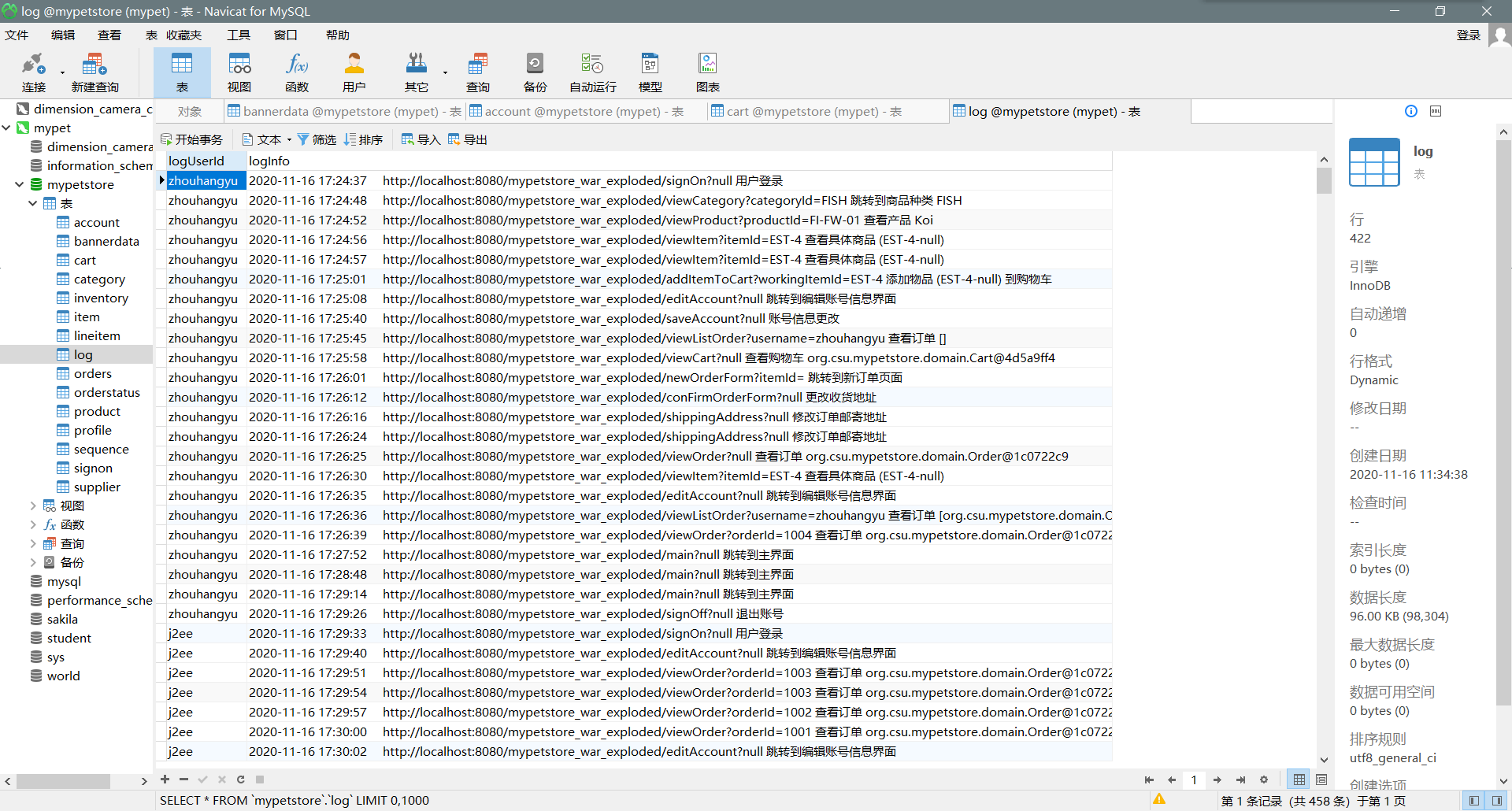


在SignOnServlet和NewAccountServlet中设置相应的获取验证码的操作：



2、**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。**

当用户每进行一次操作，就会有相应的信息被记录下来并保存到数据库当中



利用Log.java和HTTPServletRequest的各种方法获取当前页面跳转时的用户账号、跳转时间、当前操作页面的网址、具体操作页面的名字、操作的商品信息，再加上自定义的具体操作说明，传输给LogService的相关方法即可将其插入到数据库中记录下来。

在每次页面跳转之前先获取当前对话中用户是否已登录，如果账号不为空，那么进行获取日志记录操作，并插入到数据库中，之后正常跳转。

如果用户未登录则不记录该操作。

Log.java:

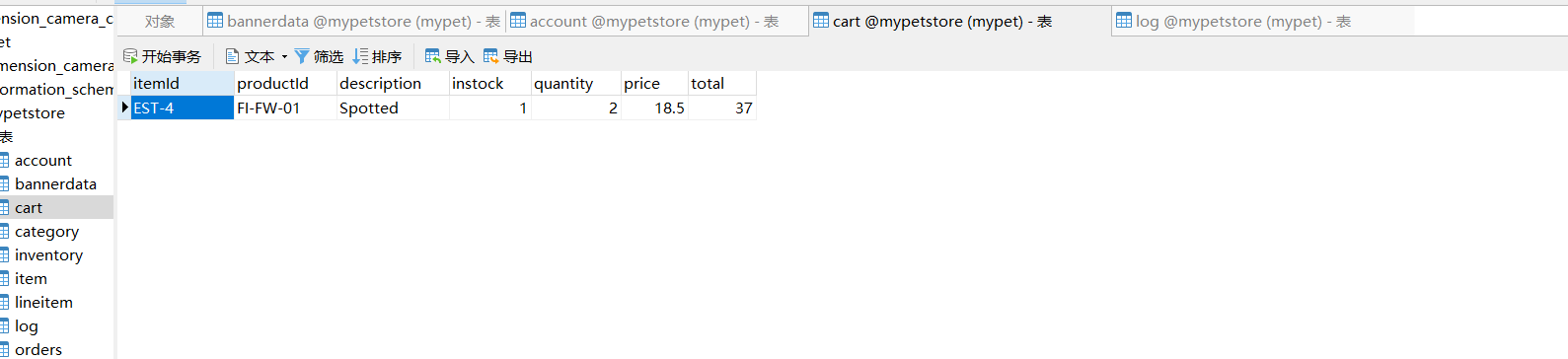
public class Log {  
 public final static int *OUT\_CONSOLE*=1;  
 public final static int *OUT\_FILE*=2;  
 public final static int *OUT\_BOTH*=3;  
 public final static int *DEBUG\_LEVEL*=0;  
 public final static int *LOG\_LEVEL*=3;  
 public final static int *INFO\_LEVEL*=6;  
 public final static int *WARN\_LEVEL*=9;  
 public final static int *ERROR\_LEVEL*=12;  
 public final static int *FATAL\_LEVEL*=15;  
 protected static int *level*=*DEBUG\_LEVEL*;*// 输出的级别开关，高于指定级别的输出* protected static int *out*=*OUT\_CONSOLE*; *//输出模式，输出到控制台，文件或者都输出* public static String *logdir*="log"; *// 保存日志文件的目录* protected static BufferedWriter *bw* =null;  
 protected static String *currentDate* = "";  
 protected static Date *data*=null;  
 protected static StringBuilder *bb*=new StringBuilder();  
 protected static String *marsk*=" [ERROE] ";  
 */\*\*  
 \* 日志函数，不受 level 限制，始终打印，最高级别  
 \** ***@param*** *s  
 \*/* synchronized public static String log(Object ...s) {  
 *bb*.delete(0, *bb*.length());  
 *bb*.append(*time*());  
 *bb*.append(*marsk*);  
 *// 获取调用的位置  
 //StackTraceElement[] stack = new Exception().getStackTrace();  
 //bb.append(stack[2].getClassName().replaceAll("\\$\n.$", ""));  
 bb*.append(" ");  
  
 for(Object str : s) {  
 *bb*.append(str);  
 *bb*.append(" ");  
 }  
 *//out(bb.toString());* return *bb*.toString();  
 }  
  
 public static String logInfomation(Object ...s){  
 *marsk* = " ";  
 return *log*(s);  
 }  
  
  
 protected static BufferedWriter getWriter() {  
 if(*currentDate*.equals(new SimpleDateFormat("dd").format(*data*))) {  
 return *bw*;  
 }  
 File fi=new File(*logdir* +File.*separator*+new SimpleDateFormat("yyyy年").format(*data*)  
 +File.*separator*+new SimpleDateFormat("MM月").format(*data*)  
 +File.*separator*+new SimpleDateFormat("dd").format(*data*)+"日.log");  
 try {  
 if(!fi.exists()) {  
 if(!fi.getParentFile().exists())  
 fi.getParentFile().mkdirs();  
 fi.createNewFile();  
 }  
 FileOutputStream fiou = new FileOutputStream(fi,true);  
 OutputStreamWriter opw = new OutputStreamWriter(fiou, "UTF-8");  
 *bw*= new BufferedWriter(opw);  
 }catch(Exception e) {  
  
 }  
 return *bw*;  
 }  
 protected static String time() {  
 *data* = new Date();  
 return new SimpleDateFormat("yyyy-MM-dd HH:mm:ss").format(*data*);  
 }  
  
   
 public static void init(int out\_put\_level,int out2where,String log\_dir) {  
 *level* = out\_put\_level;  
 *out* = out2where;  
 *logdir* = log\_dir;  
 }  
  
 public static void main(String[] args) {  
  
 }  
}

这里用跳转到商品种类界面为例子，举例日志中写入功能：



3、**购物车持久化功能，使购物车的信息可以持久化，下次打开服务器仍然会保持：**

为了实现这个功能，购物车的内容需要把从session中获取改为从数据库中获取，为此在数据库中要设置一个Cart表存放信息



为了方便数据库的操作，设置一个CartItemDAO接口来设定CartItem相关数据库操作，并在CartItemDAOImpl中进行实现。同时将Servlet中从session中获取的部分改为从数据库中获取。

同时为了逻辑化，我们小组还设置了在购买完成后自动进行Cart删除操作，删除Cart中的所有元素。

CartItemDAOImpl：

public class CartItemDAOImpl implements CartItemDAO {  
 private static final String *insertCartString* = "INSERT INTO CART (ITEMID, PRODUCTID, DESCRIPTION, INSTOCK, QUANTITY, PRICE, TOTAL) VALUES(?, ?, ?, ?, ?, ?, ?)";  
 private static final String *removeString*="delete from cart where itemId = ?";  
 private static final String *setQuantity*="UPDATE cart SET quantity = ?, total = ? WHERE itemId = ?";  
 private static final String *getCart*="select \* from cart";  
 private final String SELECT = "select \* from cart where itemId = ?";  
  
 private CatalogService catalogService;  
  
 public void insertCart(CartItem cartItem){  
 try{  
 Connection connection = DBUtil.*getConnection*();  
 PreparedStatement preparedStatement = connection.prepareStatement(*insertCartString*);  
 preparedStatement.setString(1, cartItem.getItemId() );  
 preparedStatement.setString(2, cartItem.getProductId());  
 preparedStatement.setString(3, cartItem.getDescription());  
 preparedStatement.setInt(4, cartItem.getInstock());  
 preparedStatement.setInt(5, cartItem.getQuantity());  
 preparedStatement.setFloat(6, cartItem.getPrice());  
 preparedStatement.setFloat(7, cartItem.getTotal1());  
  
 preparedStatement.executeUpdate();  
 DBUtil.*closePreparedStatment*(preparedStatement);  
 DBUtil.*closeConnection*(connection);  
 }catch (Exception e){  
 e.printStackTrace();  
 }  
 }  
  
 public List<CartItem> get(){  
 List<CartItem> cartList=new ArrayList<CartItem>();  
 try{  
 Connection connection = DBUtil.*getConnection*();  
 PreparedStatement preparedStatement = connection.prepareStatement(*getCart*);  
 ResultSet resultSet = preparedStatement.executeQuery();  
 while(resultSet.next()){  
 CartItem cartItem =new CartItem();  
  
 cartItem.setItemId(resultSet.getString(1));  
 cartItem.setProductId(resultSet.getString(2));  
 cartItem.setDescription(resultSet.getString(3));  
 cartItem.setInstock(resultSet.getInt(4));  
 cartItem.setQuantity(resultSet.getInt(5));  
 cartItem.setPrice(resultSet.getFloat(6));  
 cartItem.setTotal(resultSet.getFloat(7));  
  
 cartList.add(cartItem);  
 }  
 DBUtil.*closeResultSet*(resultSet);  
 DBUtil.*closePreparedStatment*(preparedStatement);  
 DBUtil.*closeConnection*(connection);  
 }catch (Exception e){  
 e.printStackTrace();  
 }  
 return cartList;  
 }  
  
 public void removeItemById(String itemId){  
 try{  
 Connection connection=DBUtil.*getConnection*();  
 PreparedStatement preparedStatement=connection.prepareStatement(*removeString*);  
 System.*out*.println(itemId);  
 preparedStatement.setString(1,itemId);  
 preparedStatement.execute();  
 DBUtil.*closePreparedStatment*(preparedStatement);  
 DBUtil.*closeConnection*(connection);  
  
 }catch (Exception e){  
 e.printStackTrace();  
 }  
 }  
  
 public void update(String itemId,int quantity,float total){  
 try{  
 Connection connection=DBUtil.*getConnection*();  
 PreparedStatement preparedStatement=connection.prepareStatement(*setQuantity*);  
 preparedStatement.setInt(1,quantity);  
 preparedStatement.setFloat(2,total);  
 preparedStatement.setString(3,itemId);  
 preparedStatement.execute();  
  
 DBUtil.*closePreparedStatment*(preparedStatement);  
 DBUtil.*closeConnection*(connection);  
  
 }catch (Exception e){  
 e.printStackTrace();  
 }  
 }  
}

向购物车中加入商品的操作修改：

CartItemDAOImpl cartItemDAO = new CartItemDAOImpl();  
if(cart.containsItemId(workingItemId)){  
 *//已有该物品，数量加一* cart.incrementQuantityByItemId(workingItemId);  
  
 if(account != null){  
 HttpServletRequest httpRequest= (HttpServletRequest) request;  
 String strBackUrl = "http://" + request.getServerName() + ":" + request.getServerPort()  
 + httpRequest.getContextPath() + httpRequest.getServletPath() + "?" + (httpRequest.getQueryString());  
  
 LogService logService = new LogService();  
 Item item = catalogService.getItem(workingItemId);  
 String logInfo = logService.logInfo(" ") + strBackUrl + " " + item + "数量+1 ";  
 logService.insertLogInfo(account.getUsername(), logInfo);  
 }  
}else{  
  
 catalogService = new CatalogService();  
 boolean isInStock = catalogService.isItemInStock(workingItemId);  
 Item item = catalogService.getItem(workingItemId);  
 cart.addItem(item, isInStock);  
  
 cartItem.itemId=workingItemId;  
  
 *//明天记得改product Id* cartItem.productId=item.getProduct().getProductId();  
  
 cartItem.description = item.getAttribute1();  
 cartItem.instock = isInStock?1:0;  
  
 cartItem.quantity = 1;  
  
 cartItem.price = item.getListPrice().floatValue();  
  
 cartItem.total1 = item.getListPrice().floatValue();  
  
 cartItemDAO.insertCart(cartItem);  
  
  
 session.setAttribute("cart", cart);

更新购物车操作：

while (cartItemIterator.hasNext()){  
 *//产品数量增加* CartItem cartItem = (CartItem)cartItemIterator.next();  
 String itemId = cartItem.getItem().getItemId();  
  
 try {  
 int quantity = Integer.*parseInt*((String) request.getParameter(itemId));  
 cart.setQuantityByItemId(itemId, quantity);  
  
 float total=cartItem.getItem().getListPrice().floatValue()\*quantity;  
 if (quantity < 1) {  
 cartItemIterator.remove();  
 cartItemDAO.removeItemById(itemId);  
 }  
 cartItemDAO.update(itemId,quantity,total);  
 } catch (Exception e) {  
 *//ignore parse exceptions on purpose* e.printStackTrace();  
 }  
  
 *//CartItem cartItem = cartItemIterator.next();  
 //cartItem.incrementQuantity();*}  
  
session.setAttribute("cart", cart);  
  
*//HttpSession session = request.getSession();*Account account = (Account)session.getAttribute("account");

**三、小组分工和实验总结**

**周航宇为组长，王鹤淞为组员**

**王鹤淞：**

1、商品展示界面相关Servlet，包括大类小类和具体商品

2、日志功能

3、订单类Servlet包括，订单生成查询

4、商品和订单的数据库操作，包括DAO和Impl

5、购物车功能实现，包括加入，删除和更新

**周航宇：**

1、商品搜索界面相关Servlet。

2、载入界面的重写，使用了js，css和html技术

3、购物车的功能更新，使购物车实现持久化

4、验证码功能，实现登录验证码和注册验证码

5、账号操作相关的Servlet，包括创建账号，登录账户等以及相关数据库操作

6、实验报告的编写

**实验总结：**

这次实验的难度对于目前的我们来说还是比较高的，它涉及到了Java Servlet、JSP、HTML与CSS以及JDBC等方面。在刚开始接手项目时，还是感觉比较难以下手的。好在老师为我们准备了几个视频让我们可以对项目有一个初步的认知。

在项目中我们遇到的第一个难点是对MVC架构此前没有清楚的认知，这就使得我们开始项目时比较感觉手足无措。MVC架构要求将模型视图和控制器三者分开，这是我们此前的项目并没有操作过的。但是当我们根据老师的视频了解了MVC之后，我们发现MVC的使用可以大大方便了项目的管理，每个不同的方面分门别类的放好，这样也便于后续通过git的多人项目同时进行合作。

我们遇到的第二个难点是Servlet之间的连接，因为我在大一期间的SSD1不够理想，所以在Servlet这部分时相当于是从头开始了学习。应该这次的项目界面很多，功能复杂，每一个页面之间的跳转，每一个按钮的按下都要一个Servlet来进行操作，这就要求在设计Servlet之前，首先应当考虑到这个Servlet要实现的功能以及form表单的action形式。最后在小组的共同完成下完成了这些Servlet。

第三个难点是建立一个可持续化的购物车。最开始我们尝试将数据保存到数据库中时是保存到了Cart类中，但是这样我们发现保存的数据完全是乱的，数据库中根本无法存入数据。后来再经过了讨论之后，我们发现虽然这些东西是存放在Cart中，但是实际保存的是数据库中的内容，于是我们新建了一个CartItem类，这个类储存了存放在数据库中数据的相关信息。随后我们遇到了第二个难点，JDBC的连接，因为这方面了解的不多，所以在连接数据库的时候也碰到了很多困难。在编写Remove操作的Servlet时，我们没有进行execute操作就之间关闭了数据库连接，使得我们研究了半天都没有实现物件的删除，最后还是在其他同学的帮助下发现并改正了错误。

总体来说这次实验中难点还是有不少的，但是最后在同学的帮助和在网络上查阅资料后还是基本上解决掉了。经过这次实验我学到最重要的就是MVC架构的思想，这对于我以后的项目开发和管理都是有十分重要的意义的。

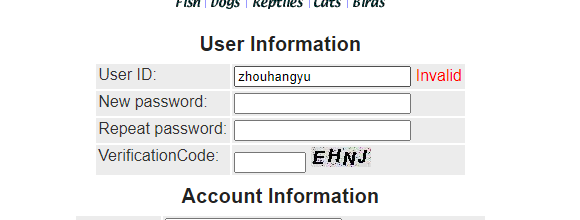
**实验二 使用AJAX和jQuery改善用户体验**

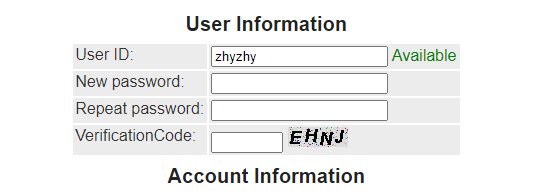
**一、基本任务**

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。**

在实现判断句账户时，最开始我使用的是java script技术，这样操作的好处是不需要导入其他的插件，但是操作十分繁琐。之后在学习了老师的视频之后，我才用了jQuery方法，通过导入jQuery.js来使用jQuery的语法，操作更加简单。

首先是设置一个UsernameExistServlet，获取传递来的用户名并和数据库中的用户名比较，如果重复返回Exist信息，不重复返回Not Exist信息。之后在表单的JSP中插入script脚本，通过jQuery语法获得信息并通过信息显示相关信息。





UsernameExistServlet：

public class UsernameExistServlet extends HttpServlet {  
 @Override  
 protected void doGet(HttpServletRequest req, HttpServletResponse resp) throws ServletException, IOException {  
 String username = req.getParameter("username");  
 AccountService accountService = new AccountService();  
 Account result = accountService.getAccount(username);  
  
 resp.setContentType("text/json");  
 PrintWriter out = resp.getWriter();  
  
  
 Result result1 = new Result();  
  
 if(result==null){*//用户名可用* result1.setCode(0);  
 result1.setMsg("Not Exist");  
*// out.print("Not Exist");* }  
 else{*//不可用* result1.setCode(1);  
 result1.setMsg("Exist");  
*// out.print("Exist");* }  
  
 String str = JSON.*toJSONString*(result1);  
 out.print(str);  
 out.flush();  
 out.close();  
 }  
}

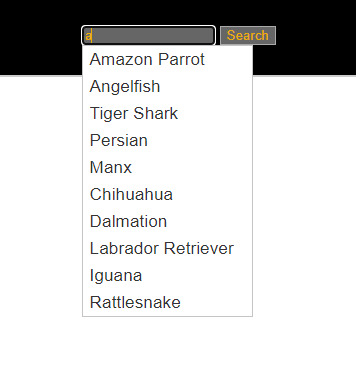
JSP中的相关script：

<script>  
 $(function () {  
 $('#username').on('blur',function () {  
 $.ajax({  
 type :"GET",  
 url :"usernameExist?username="+this.value,  
 success :function (data) {  
  
 *// console.log(data);  
 // console.log(data.code);* if(data.msg==='Exist'){  
 $('#usernameTips').attr("class","errormsg").text('Invalid');  
 }else if(data.msg==='Not Exist'){  
 $('#usernameTips').attr("class","okmsg").text('Available');  
 }  
 }  
 });  
 });  
 });  
  
</script>

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。**

为了达到自动补全的目的，首先要设置一个AutoCompleteServlet。在servlet中首先获得用户在搜索框中输入的关键字，之后通过关键字调用产品类，生产一个产品列表并保存在一个List中。之后创建一个JSON的数组，将List中的产品的名字保存在JSON数组中并转化为字符串输出。

之后在Top.jsp中导入jQuery.js和jQueryUI，通过jQuery语法从数据库中寻找相关信息，之后通过jQueryUI的autocomplete插件来显示自动补全的信息。



AutoCompleteServlet：

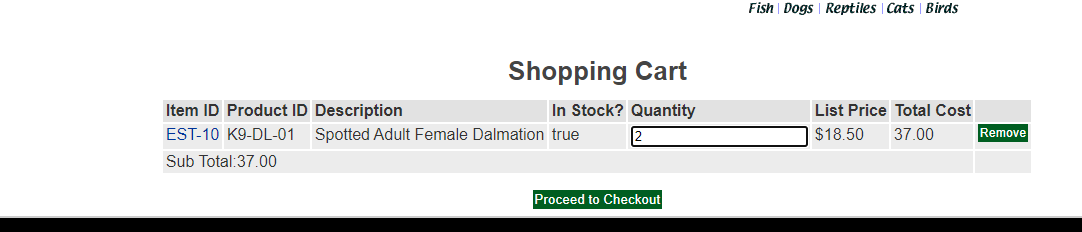
public class AutoCompleteServlet extends HttpServlet {  
 @Override  
 protected void doGet(HttpServletRequest req, HttpServletResponse resp) throws ServletException, IOException {  
 String keyword = req.getParameter("keyword");*//获取关键字* CatalogService catalogService = new CatalogService();  
 List<Product>productList = catalogService.searchProductList(keyword);resp.setContentType("application/json;charset=uft-8");  
 JSONArray searchArray = new JSONArray();  
 for(int i=0;i<productList.size();i++){  
 JSONObject searchObject = new JSONObject();  
 searchObject.put("name",productList.get(i).getName());*//产品名字* searchArray.add(searchObject);  
 }  
  
 String searchString = searchArray.toJSONString();PrintWriter out = resp.getWriter();  
 out.print(searchString);  
  
 out.flush();  
 out.close();  
 }  
}

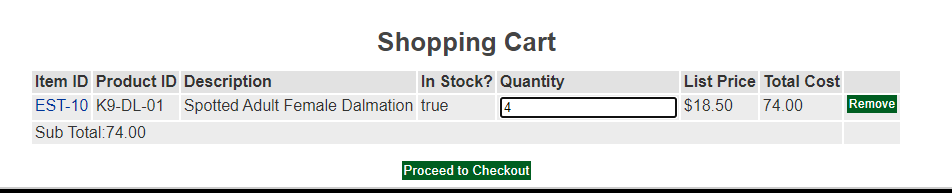
Top.jsp中的相关script：

<script>  
 $().ready(function () {  
 var $keyword = $("#keyword");  
 $keyword.on("keyup",function () {  
 var text = $keyword.val();  
 ***console***.log("输入框为"+$keyword.val());  
  
 if(text==''){*//如果输入框为空不输出  
 // console.log("搜索框为空")* }else{*//如果不为空则从数据库中寻找结果* $.ajax({  
 type:"GET",  
 url:"autoComplete",  
 data:{"keyword":$keyword.val()},  
 dataType:"json",  
 success:function (data) {  
 if(typeof (data)=='string'){  
 var stringObj = function (data) {  
 return eval("("+data+")");  
 }  
 alert(stringObj);  
 *// console.log(stringObj);* }  
 var searchResult = new ***Array***();  
 for(var i=0;i<data.length;i++){  
 searchResult[i]=data[i].name;  
 *// console.log(searchResult[i]);* }  
 $keyword.autocomplete({  
 source:searchResult  
 });  
 }  
 })  
 }  
 });  
 });  
</script>

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。**

在实现购物车模块时，抛弃了原始的按下按钮改变，所以需要一个新的Servlet，这个Servlet为UpdateCartJSServlet。在js中改变商品的数量值，再调用Servlet将更改的相关信息传输回来，再通过js中对字符串的处理显示在jsp页面上。在jsp中设置一个onkeyup函数，从而实现当数字改变时无需操作即可自动改变总结。





UpdateCartJSServlet:

public class UpdateCartJSServlet extends HttpServlet {  
  
 *//private static final String VIEW\_CART = "/WEB-INF/jsp/cart/Cart.jsp";  
  
 //private String workingItemId;* private Cart cart;  
  
 protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {  
  
 }  
  
 protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {  
 *//workingItemId = request.getParameter("workingItemId");  
  
 //从对话中，获取购物车* HttpSession session = request.getSession();  
 cart = (Cart)session.getAttribute("cart");  
  
 Iterator<CartItem> cartItemIterator = cart.getAllCartItems();  
  
 while (cartItemIterator.hasNext()) {  
 *//产品数量增加* CartItem cartItem = (CartItem) cartItemIterator.next();  
 String itemId = cartItem.getItem().getItemId();  
 CartItemDAOImpl cartItemDAO = new CartItemDAOImpl();  
  
 try {  
 int quantity = Integer.*parseInt*((String) request.getParameter("quantity"));  
 cart.setQuantityByItemId(itemId, quantity);  
 if (quantity < 1) {  
 cartItemIterator.remove();  
 }  
 float total=cartItem.getItem().getListPrice().floatValue()\*quantity;  
 cartItemDAO.update(itemId,quantity,total);  
 } catch (Exception e) {  
 e.printStackTrace();  
 }  
 }  
 session.setAttribute("cart", cart);  
  
  
 Cart cart2 = (Cart)session.getAttribute("cart");  
 Iterator<CartItem> cartItemIterator2 = cart2.getAllCartItems();  
 String quantityAll = "";  
 while (cartItemIterator2.hasNext()) {  
 *//产品数量增加* CartItem cartItem2 = cartItemIterator2.next();  
 int quantity2 = cartItem2.getQuantity();  
 quantityAll += quantity2 + "," + cartItem2.getTotal() + "," + cart2.getSubTotal();  
  
 }  
  
 response.setContentType("text/xml");  
 PrintWriter out = response.getWriter();  
 out.write(quantityAll);  
 }  
}

updateCart.js:

function updateCart() {  
 *//var username = document.NewAccountForm.username.value;  
 //var workingItemId = document.getElementById('workingItemId').value;  
 //sendRequest("updateCartQuantities?username=" + username);* var quantity = ***document***.getElementById("quantity").value;  
 sendRequest("updateCartJSServlet?quantity="+ quantity);  
 *//sendRequest("updateCartQuantities");*}  
  
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 resp = ***xmlHttpRequest***.responseText;  
 var array = resp.split(",");  
 var quantity = ***document***.getElementById("quantity");  
 var total = ***document***.getElementById("total");  
 var subtotal = ***document***.getElementById("subtotal");  
  
 quantity.innerText = array[0];  
 total.innerText = array[1];  
 subtotal.innerText = array[2];  
 }  
 }  
}

cartChange.js:

function remove() {  
 var el = ***document***.getElementById("remove");  
 var el2 = el.parentNode.parentNode;  
 el.parentNode.parentNode.remove();  
 *//var el3 = document.getElementById("lastTR");* var el3 = ***document***.getElementById("lastTR");  
 el3.before(el2);  
}  
 *//this.parentNode.parentNode.parentNode.removeChild(this.parentNode.parentNode);*$(***document***).ready(function(){  
 *//jQuery调用* $('#remove').on('click',function () {  
 $(this.parentNode.parentNode).remove();  
 });  
  
 $("tr").css("background-color", "red");  
});

Cart.jsp中的相关改变：

<td>  
 <input type="text" id="quantity" onkeyup="updateCart();" name="${cartItem.item.itemId}" value="${cartItem.quantity}">  
 <div id="cartMsg"></div>  
 <script type="text/javascript" src="${pageContext.request.contextPath }/js/updateCart.js"></script>  
</td>

**二、拓展任务**

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**

为了实现显示悬浮窗的功能，首先设置一个现象商品信息的servlet：CategoryShowJsServlet，通过categoryid来获取相关的简介以及其中的小类。在jsp中，当鼠标移动到某一个图片时，通过js获取他的ID，之后通过servlet获取对应类中商品的种类和名称。同时传会js，通过设置的悬浮窗属性进行显示。当鼠标移开时，js设置相应悬浮窗隐藏

CategoryShowJsServlet：

public class CategoryShowJsServlet extends HttpServlet {  
 protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {  
 String categoryId = request.getParameter("categoryId");  
 CatalogService catalogService = new CatalogService();  
  
 List<Product> productList = catalogService.getProductListByCategory(categoryId);  
 String resp = "";  
 int i = 0;  
 while(i < productList.size()){  
 Product product = productList.get(i);  
 resp += product.getProductId() + " " + product.getName() + "\n";  
 i++;  
 }  
  
 response.setContentType("text/xml");  
 PrintWriter out = response.getWriter();  
 out.write(resp);  
 }  
}

mouseEvent.js:

function showInform(categoryId) {  
 ***console***.log(categoryId);  
 ***xmlHttpRequest***.open("GET","categoryShowJsServlet?categoryId="+categoryId,true);  
 ***xmlHttpRequest***.onreadystatechange = processResponse;  
 ***xmlHttpRequest***.send(null);  
}  
  
  
function processResponse() {  
 if (***xmlHttpRequest***.readyState == 4) {  
 if (***xmlHttpRequest***.status == 200) {  
 var resp = ***xmlHttpRequest***.responseText;  
 *//显示悬浮层* var inform = ***document***.getElementById("inform");  
 inform.innerText = resp;  
 inform.style.tableLayout  
 inform.style.display = "inline-block";  
 inform.style.backgroundColor = "rgb(147,112,219,0.1)";  
 inform.style.color = "rgb(0,205,205)";  
 }  
 }  
}  
  
*//隐藏悬浮层*function hiddenInform(event){  
 var informDiv = ***document***.getElementById('inform');  
 var x=event.clientX;  
 var y=event.clientY;  
 var divx1 = informDiv.offsetLeft;  
 var divy1 = informDiv.offsetTop;  
 var divx2 = informDiv.offsetLeft + informDiv.offsetWidth;  
 var divy2 = informDiv.offsetTop + informDiv.offsetHeight;  
 if( x < divx1 || x > divx2 || y < divy1 || y > divy2){  
 ***document***.getElementById('inform').style.display='none';  
 }  
}

Main.jsp中的相关内容：

<div id="MainImageContent" style="display: inline-block">  
 *<!--中间显示栏-->* <map name="estoremap">  
 <area alt="BIRDS" coords="72,2,280,250" href="viewCategory?categoryId=BIRDS" shape="rect"  
 onmouseover="showInform(alt);" onmouseout="hiddenInform(***event***)"/>  
 <area alt="FISH" coords="2,180,72,250" href="viewCategory?categoryId=FISH" shape="rect"  
 onmouseover="showInform(alt);" onmouseout="hiddenInform(***event***)"/>  
 <area alt="DOGS" coords="60,250,130,320" href="viewCategory?categoryId=DOGS" shape="rect"  
 onmouseover="showInform(alt);" onmouseout="hiddenInform(***event***)"/>  
 <area alt="REPTILES" coords="140,270,210,340" href="viewCategory?categoryId=REPTILES" shape="rect"  
 onmouseover="showInform(alt);" onmouseout="hiddenInform(***event***)"/>  
 <area alt="CATS" coords="225,240,295,310" href="viewCategory?categoryId=CATS" shape="rect"  
 onmouseover="showInform(alt);" onmouseout="hiddenInform(***event***)"/>  
 <area alt="BIRDS" coords="280,180,350,250" href="viewCategory?categoryId=BIRDS" shape="rect"  
 onmouseover="showInform(alt);" onmouseout="hiddenInform(***event***)"/>  
 </map>  
 <img height="355" src="images/splash.gif" align="middle" usemap="#estoremap" width="350" />  
</div>

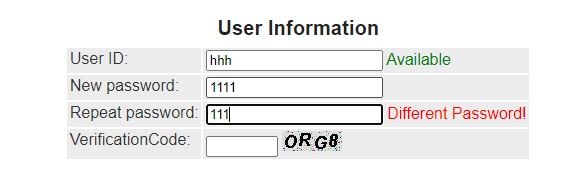
2、**订单模块。通过选项卡或模版内容实现订单多个信息部分的录入，不需要页面跳转，在一个页面完成订单所有内容的录入**

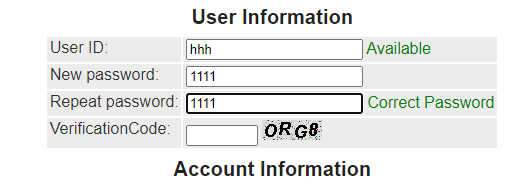
无

3、**其他效果：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**

1、在创建用户和修改信息界面加入了两次密码同异的判定

在这些界面的jsp的密码框设置一个onkeyup函数，每当抬起键盘上的按键时进行一次判断。插入一段java script脚本，获取密码和确认密码的值，对二者进行比较，如果相同则显示正确信息，如果不同则显示错误提示信息。



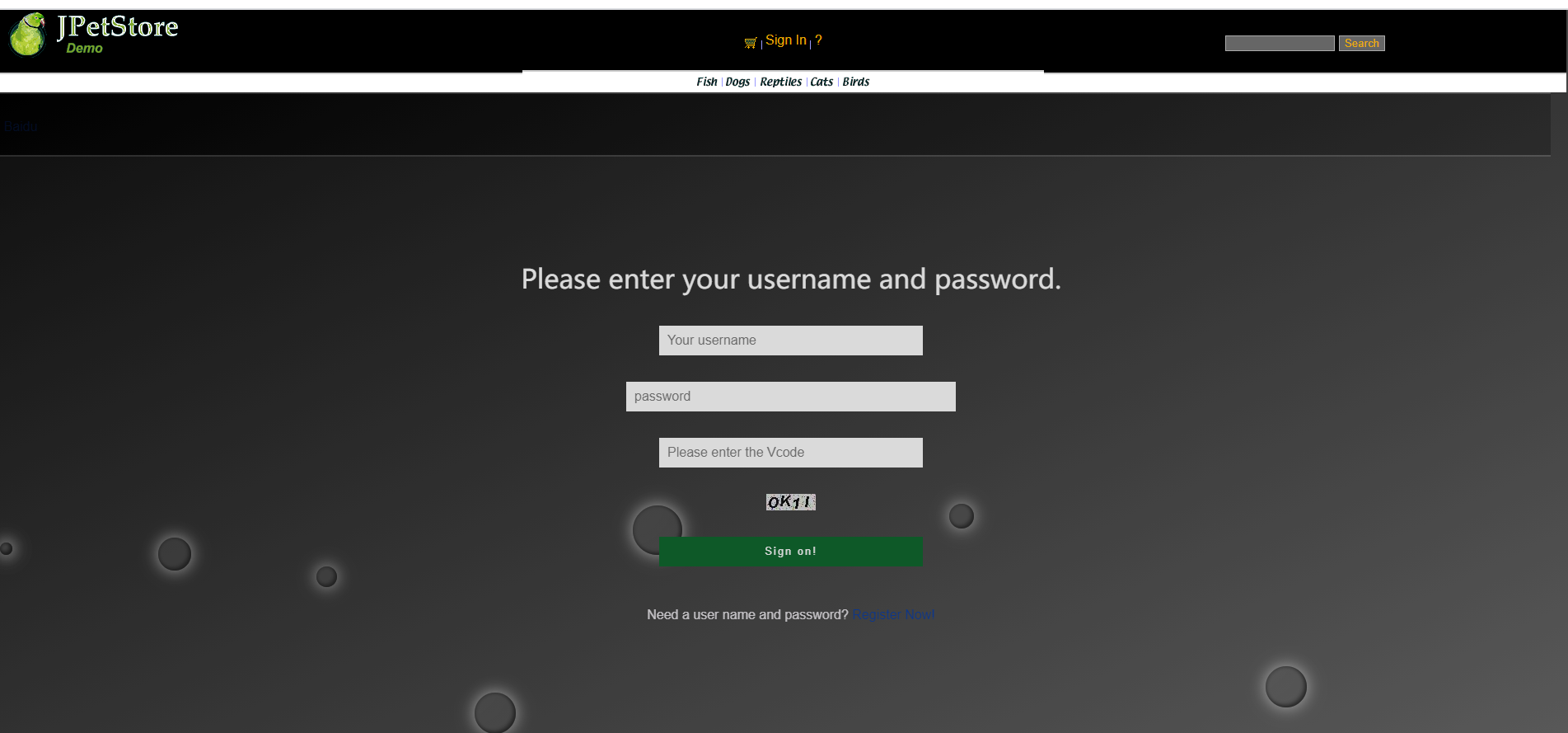


相关的java script代码：

<script>  
 function validate() {  
 var password = ***document***.getElementById("password").value;  
 var repeatedPassword = ***document***.getElementById("repeatedPassword").value;  
  
 var tips = ***document***.getElementById("passwordTips");  
 if(password==repeatedPassword){  
 tips.className='okmsg';  
 tips.innerText="Correct Password"  
 }  
 else if(password!=repeatedPassword){  
 tips.className='errormsg';  
 tips.innerText="Different Password!"  
 }  
  
 }  
</script>

2、修改了登录界面，使得登录界面更加美观。

在登录界面中，通过css为其加入了一个动态背景，并通过css实现了鼠标点击输入框使输入框变大

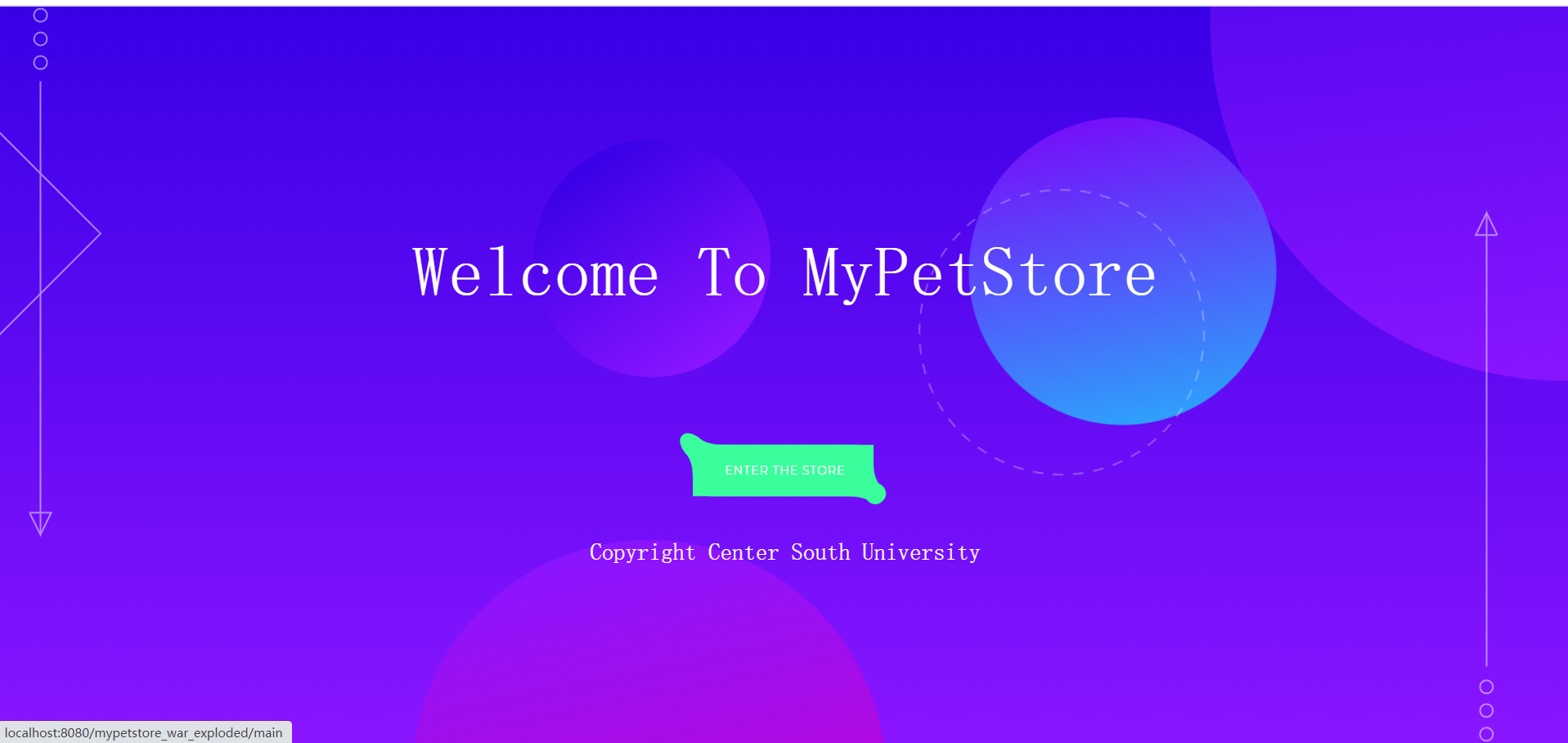


相关的操作为：为input标签加入transition属性，使得鼠标移动上去的时候大小发生改变，将时间设置为0.1ms，从而实现人眼看起来输入框为瞬间变大效果。之后通过nth-child(1)到nth-child(10)来分别空中从左到右的背景中的圆形，用animation-duration等相关数据来控制各个圆形的动画运动时间。

SignonForm.jsp:

<style type="text/css">  
 \* {  
 box-sizing: border-box;  
 }  
 body {  
 margin: 0;  
 padding: 0;  
 font: 16px/20px microsft yahei;  
 }  
 .wrap {  
 width: 100%;  
 height: 100%;  
 padding: 10% 0;  
 position: fixed;  
 opacity: 0.8;  
 background: linear-gradient(to bottom right,#000000, #656565);  
 background: -webkit-linear-gradient(to bottom right,#50a3a2,#53e3a6);  
 }  
 .container {  
 width: 60%;  
 margin: 0 auto;  
 }  
 .container h1 {  
 text-align: center;  
 color: #FFFFFF;  
 font-weight: 500;  
 }  
 .container input {  
 width: 320px;  
 display: block;  
 height: 36px;  
 border: 0;  
 outline: 0;  
 padding: 6px 10px;  
 line-height: 24px;  
 margin: 32px auto;  
 -webkit-transition: all 0s ease-in 0.1ms;  
 -moz-transition: all 0s ease-in 0.1ms;  
 transition: all 0s ease-in 0.1ms;  
 }  
 .container input[type="text"] , .container input[type="password"] {  
 background-color: #FFFFFF;  
 font-size: 16px;  
 color: #50a3a2;  
 }  
 .container input[type='submit'] {  
 font-size: 16px;  
 letter-spacing: 2px;  
 color: #666666;  
 background-color: #FFFFFF;  
 }  
 .container input:focus {  
 width: 400px;  
 }  
 .container input[type='submit']:hover {  
 cursor: pointer;  
 width: 400px;  
 }  
 .to\_login{  
 color: #a7c4c9;  
 }  
 .text{  
 color: #e2dfe4;  
 }  
 .wrap ul {  
 position: absolute;  
 top: 0;  
 left: 0;  
 width: 100%;  
 height: 100%;  
 z-index: -20;  
 }  
 .wrap ul li {  
 list-style-type: none;  
 display: block;  
 position: absolute;  
 bottom: -120px;  
 width: 15px;  
 height: 15px;  
 z-index: -8;  
 border-radius: 50%;  
 box-shadow: inset -30px -30px 75px rgba(0, 0, 0, .2),  
 inset 0px 0px 5px rgba(0, 0, 0, .5),  
 inset -3px -3px 5px rgba(0, 0, 0, .5),  
 0 0 20px rgba(255, 255, 255, .75);  
 background-color:rgba(255, 255, 255, 0.15);  
 animotion: square 25s infinite;  
 -webkit-animation: square 25s infinite;  
 }  
 .wrap ul li:nth-child(1) {  
 left: 0;  
 animation-duration: 10s;  
 -moz-animation-duration: 10s;  
 -o-animation-duration: 10s;  
 -webkit-animation-duration: 10s;  
 }  
 .wrap ul li:nth-child(2) {  
 width: 40px;  
 height: 40px;  
 left: 10%;  
 animation-duration: 15s;  
 -moz-animation-duration: 15s;  
 -o-animation-duration: 15s;  
 -webkit-animation-duration: 11s;  
 }  
 .wrap ul li:nth-child(3) {  
 left: 20%;  
 width: 25px;  
 height: 25px;  
 animation-duration: 12s;  
 -moz-animation-duration: 12s;  
 -o-animation-duration: 12s;  
 -webkit-animation-duration: 12s;  
 }  
 .wrap ul li:nth-child(4) {  
 width: 50px;  
 height: 50px;  
 left: 30%;  
 -webkit-animation-delay: 3s;  
 -moz-animation-delay: 3s;  
 -o-animation-delay: 3s;  
 animation-delay: 3s;  
 animation-duration: 12s;  
 -moz-animation-duration: 12s;  
 -o-animation-duration: 12s;  
 -webkit-animation-duration: 12s;  
 }  
 .wrap ul li:nth-child(5) {  
 width: 60px;  
 height: 60px;  
 left: 40%;  
 animation-duration: 10s;  
 -moz-animation-duration: 10s;  
 -o-animation-duration: 10s;  
 -webkit-animation-duration: 10s;  
 }  
 .wrap ul li:nth-child(6) {  
 width: 75px;  
 height: 75px;  
 left: 50%;  
 -webkit-animation-delay: 7s;  
 -moz-animation-delay: 7s;  
 -o-animation-delay: 7s;  
 animation-delay: 7s;  
 }  
 .wrap ul li:nth-child(7) {  
 left: 60%;  
 width: 30px;  
 height: 30px;  
 animation-duration: 8s;  
 -moz-animation-duration: 8s;  
 -o-animation-duration: 8s;  
 -webkit-animation-duration: 8s;  
 }  
 .wrap ul li:nth-child(8) {  
 width: 90px;  
 height: 90px;  
 left: 70%;  
 -webkit-animation-delay: 4s;  
 -moz-animation-delay: 4s;  
 -o-animation-delay: 4s;  
 animation-delay: 4s;  
 }  
 .wrap ul li:nth-child(9) {  
 width: 50px;  
 height: 50px;  
 left: 80%;  
 animation-duration: 20s;  
 -moz-animation-duration: 20s;  
 -o-animation-duration: 20s;  
 -webkit-animation-duration: 20s;  
 }  
 .wrap ul li:nth-child(10) {  
 width: 75px;  
 height: 75px;  
 left: 90%;  
 -webkit-animation-delay: 6s;  
 -moz-animation-delay: 6s;  
 -o-animation-delay: 6s;  
 animation-delay: 6s;  
 animation-duration: 30s;  
 -moz-animation-duration: 30s;  
 -o-animation-duration: 30s;  
 -webkit-animation-duration: 30s;  
 }  
 @keyframes square {  
 0% {  
 -webkit-transform: translateY(0);  
 transform: translateY(0)  
 }  
 100% {  
 bottom: 400px;  
 -webkit-transform: translateY(-500);  
 transform: translateY(-500)  
 }  
 }  
 @-webkit-keyframes square {  
 0% {  
 -webkit-transform: translateY(0);  
 transform: translateY(0)  
 }  
 100% {  
 bottom: 400px;  
 -webkit-transform: translateY(-500);  
 transform: translateY(-500)  
 }  
 }  
</style>  
<div id="Content">  
  
 <div class="wrap">  
 <div class="container">  
 <div id="Catalog">  
 <ul class="messages">  
 <li>${sessionScope.messageSignOn}</li>  
 </ul>  
 <form action="signOn" method="post">  
 <h1 style="color: white; margin: 0; text-align: center">Please enter your username and password.</h1>  
 <label><input type="text" placeholder="Your username" name="username"/></label>  
 <label><input type="password" placeholder="password" name="password"/></label>  
 <label><input type="text" placeholder="Please enter the Vcode" name="vCode"/></label>  
 <label><a href="signOn"><img border="0" src="verificationCode" name="checkcode"></a></label>  
 <input type="submit" value="Sign on!" name="signon"/>  
 </form>  
 <p class="change\_link" style="text-align: center">  
 <span class="text">Need a user name and password?</span>  
 <a href="newAccountForm" class="to\_login">Register Now!</a>  
 </p>  
 </div>  
 </div>  
 </div>

3、对于开始界面的修改



利用css，js和html修改了开始界面，使得开始界面有一个动态的背景效果。当鼠标放在按钮上时，可以触发一个按钮变色效果

Css和js的代码量较大，不在实验报告中展示。

相关的html：

<html lang="en">  
  
<head>  
 <meta charset="UTF-8">  
 <title>MyPetStore</title>  
 <link rel="stylesheet" href="css/reset.min.css">  
 <link rel="stylesheet" href="css/style.css">  
  
</head>  
  
<body>  
*<!-- partial:index.partial.html -->*<div class="arrow arrow--top">  
 <svg xmlns="http://www.w3.org/2000/svg" width="270.11" height="649.9" overflow="visible">  
 <style>  
 .geo-arrow {  
 fill: none;  
 stroke: #fff;  
 stroke-width: 2;  
 stroke-miterlimit: 10  
 }  
 </style>  
 <g class="item-to bounce-1">  
 <path class="geo-arrow draw-in" d="M135.06 142.564L267.995 275.5 135.06 408.434 2.125 275.499z" />  
 </g>  
 <circle class="geo-arrow item-to bounce-2" cx="194.65" cy="69.54" r="7.96" />  
 <circle class="geo-arrow draw-in" cx="194.65" cy="39.5" r="7.96" />  
 <circle class="geo-arrow item-to bounce-3" cx="194.65" cy="9.46" r="7.96" />  
 <g class="geo-arrow item-to bounce-2">  
 <path class="st0 draw-in" d="M181.21 619.5l13.27 27 13.27-27zM194.48 644.5v-552" />  
 </g>  
 </svg>  
</div>  
<div class="arrow arrow--bottom">  
 <svg xmlns="http://www.w3.org/2000/svg" width="31.35" height="649.9" overflow="visible">  
 <style>  
 .geo-arrow {  
 fill: none;  
 stroke: #fff;  
 stroke-width: 2;  
 stroke-miterlimit: 10  
 }  
 </style>  
 <g class="item-to bounce-1">  
 <circle class="geo-arrow item-to bounce-3" cx="15.5" cy="580.36" r="7.96" />  
 <circle class="geo-arrow draw-in" cx="15.5" cy="610.4" r="7.96" />  
 <circle class="geo-arrow item-to bounce-2" cx="15.5" cy="640.44" r="7.96" />  
 <g class="item-to bounce-2">  
 <path class="geo-arrow draw-in" d="M28.94 30.4l-13.26-27-13.27 27zM15.68 5.4v552" />  
 </g>  
 </g>  
 </svg>  
</div>  
<div class="main">  
 <div class="main\_\_text-wrapper">  
 <h1 class="main\_\_title">Welcome To MyPetStore</h1>  
  
 <div id='mainButton'>  
 <svg xmlns="http://www.w3.org/2000/svg" version="1.1" class="goo">  
 <defs>  
 <filter id="goo">  
 <feGaussianBlur in="SourceGraphic" stdDeviation="10" result="blur" />  
 <feColorMatrix in="blur" mode="matrix" values="1 0 0 0 0 0 1 0 0 0 0 0 1 0 0 0 0 0 19 -9"  
 result="goo" />  
 <feComposite in="SourceGraphic" in2="goo" />  
 </filter>  
 </defs>  
 </svg>  
  
 <span class="button--bubble\_\_container">  
 <a href="main" class="button button--bubble">  
 Enter the Store  
 </a>  
 <span class="button--bubble\_\_effect-container">  
 <span class="circle top-left"></span>  
 <span class="circle top-left"></span>  
 <span class="circle top-left"></span>  
  
 <span class="button effect-button"></span>  
  
 <span class="circle bottom-right"></span>  
 <span class="circle bottom-right"></span>  
 <span class="circle bottom-right"></span>  
 </span>  
 </span>  
 </div>  
  
 <svg xmlns="http://www.w3.org/2000/svg" class="dotted-circle" width="352" height="352" overflow="visible">  
 <circle cx="176" cy="176" r="174" fill="none" stroke="#fff" stroke-width="2" stroke-miterlimit="10"  
 stroke-dasharray="12.921,11.9271" />  
 </svg>  
 </div>  
 <footer>Copyright Center South University</footer>  
</div>  
*<!-- partial -->  
<!-- partial -->*<script src='js/cssPlugin.js'></script>  
<script src='js/easePack.js'></script>  
<script src='js/tweenLite.js'></script>  
<script src='js/jquery.js'></script>  
<script src='js/timeLineLite.js'></script>  
<script src="js/effect.js"></script>  
  
</body>  
  
</html>

**三、小组分工和实验总结**

**周航宇为组长，王鹤淞为组员**

**王鹤淞：**

1、商品信息的悬浮窗显示

2、购物车模块的局部刷新，实现异步更改

**周航宇：**

1、用户名重复性的判断

2、两次密码一致性的判断

3、查询商品模块实现自动补全

4、登录界面和初始界面的美化

5、实验报告的编写

**实验总结：**

实验二相比实验一的难度大了很多。一方面是因为实验二相关的ajax、jQuery等知识的难度更大，另一方面也是实验二的选题更加开放，老师讲的东西不多，更多的是需要我们从网上和书本中自己去学习和摸索。

在这次实验中，我认为的第一个难点在于从传统的java script向jQuery的过渡。相比而言，传统的js语法和我们熟知的java更接近，但是代码量更大。使用jQuery可以很好的降低我们的代码量，使代码变得简洁。在刚开始将js转化为jQuery的形式时，我并不是很理解jQuery中键值对这一表达形式。在传递数据时，我仍还是想当然的以为是传递一个text类型，后来在看了老师的视频后，我知道了传递的值应该是一个JSON对象，将这个JSON对象转化为相应的字符串并找到我们需要的数据。同时jQuery的语法中$也与一般的语法相差较大，我仍然需要一段时间来学习。

其次的难点是修改别人的css和js。因为网上好看的网页模板有很多，所以我本以为模仿别人的模板是一个比较简单的过程，但是后来发现其实并不简单。套模板很简单，但是要看懂别人的操作和学习别人的语法并为自己所用并不是一个容易的事。尤其是css和js这种有很多小细节的脚本语言。所以我尽量花了比较多的时间对css中的大部分参数都进行修改，以便来判断它影响的是哪一个控件，这一部分虽然比较枯燥，但是确让我对代码有着比较深的理解。

这次的实验总体来说是一个开放性很强的实验，通过这次实验也让我了解到光凭借老师上课的内容是完全不够的，前端的知识是很宽广的，需要不断的从书本和网络中学习其他的知识来丰富自己。