Web应用开发技术

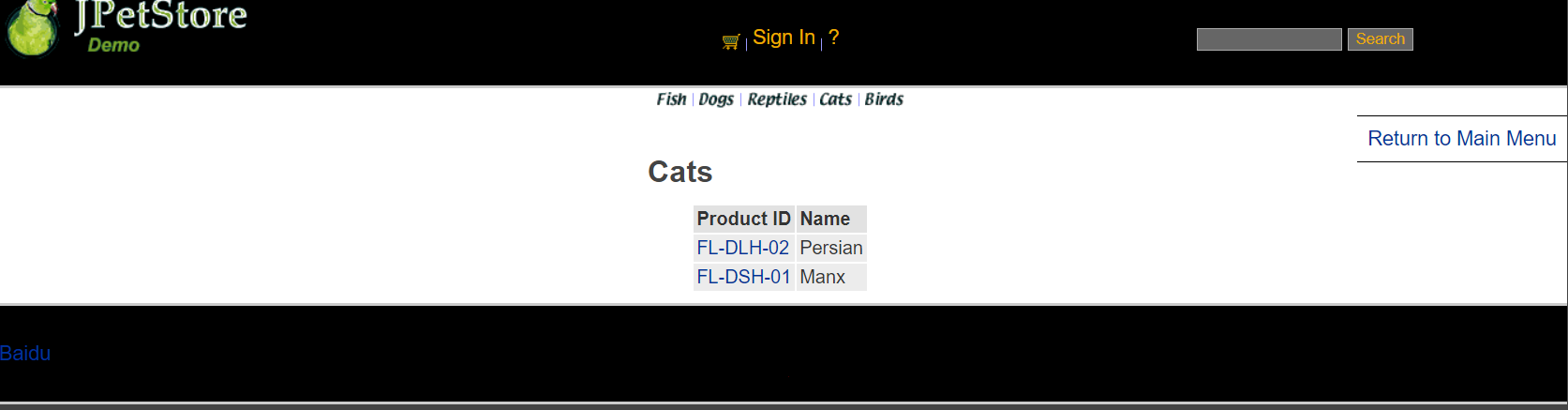
实验报告

**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。**

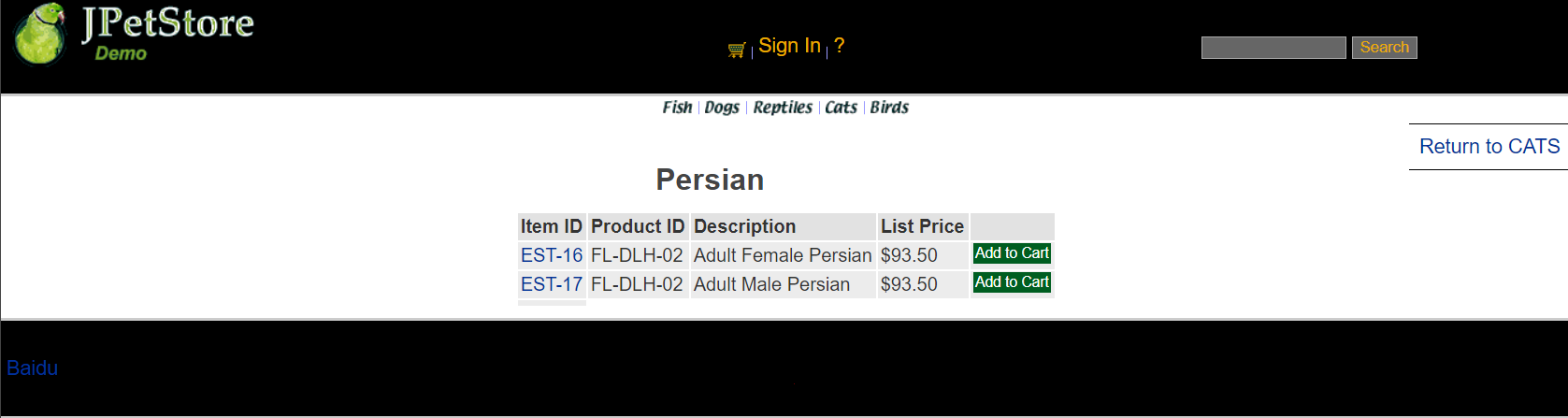
在实现大类Category时，先实现Category.jsp对大类进行具体的展示。jsp中的c:foreach语法可以实现大类的展示，通过ID选择不同的Product。在domain层中的Category.java中设计了大类相关的函数。在CategoryDAO中定义相关的数据库函数名称，在CategoryDAOImpl中对函数数据库的操作进行实现。最后通过ViewCategoryServlet实现页面之间信息的传递和jsp页面的跳转。



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。



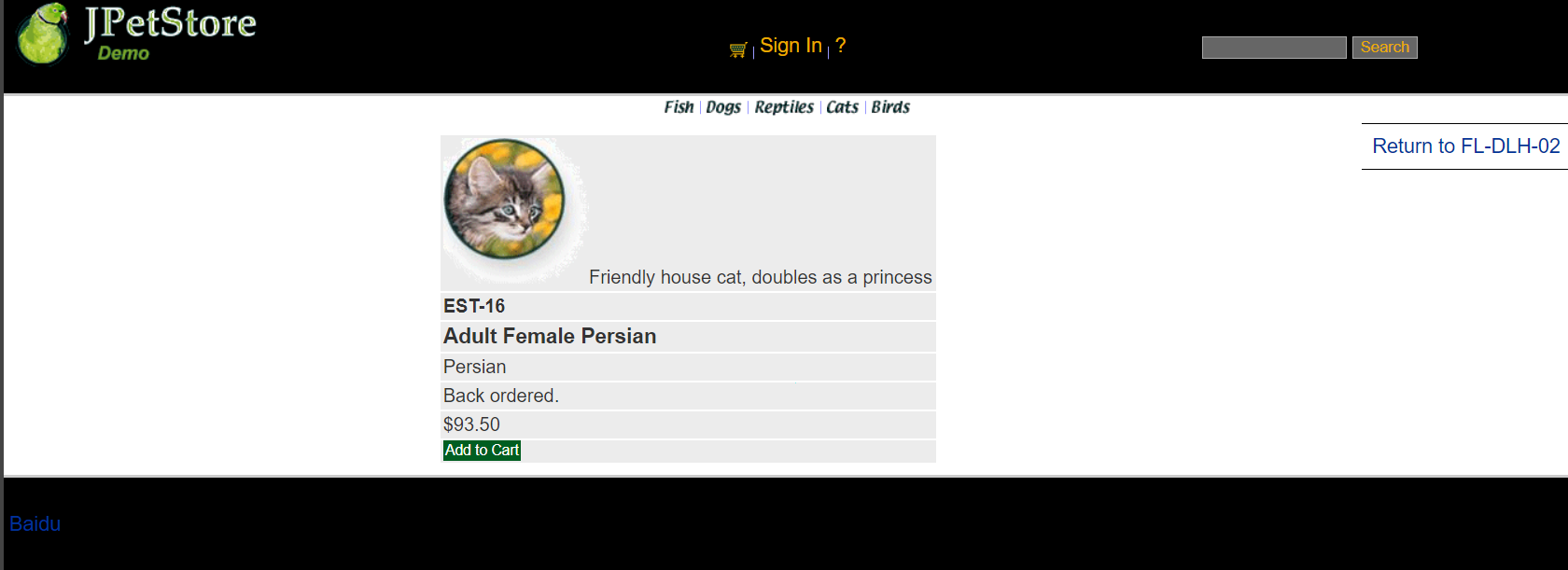
ViewProductServlet为：

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。

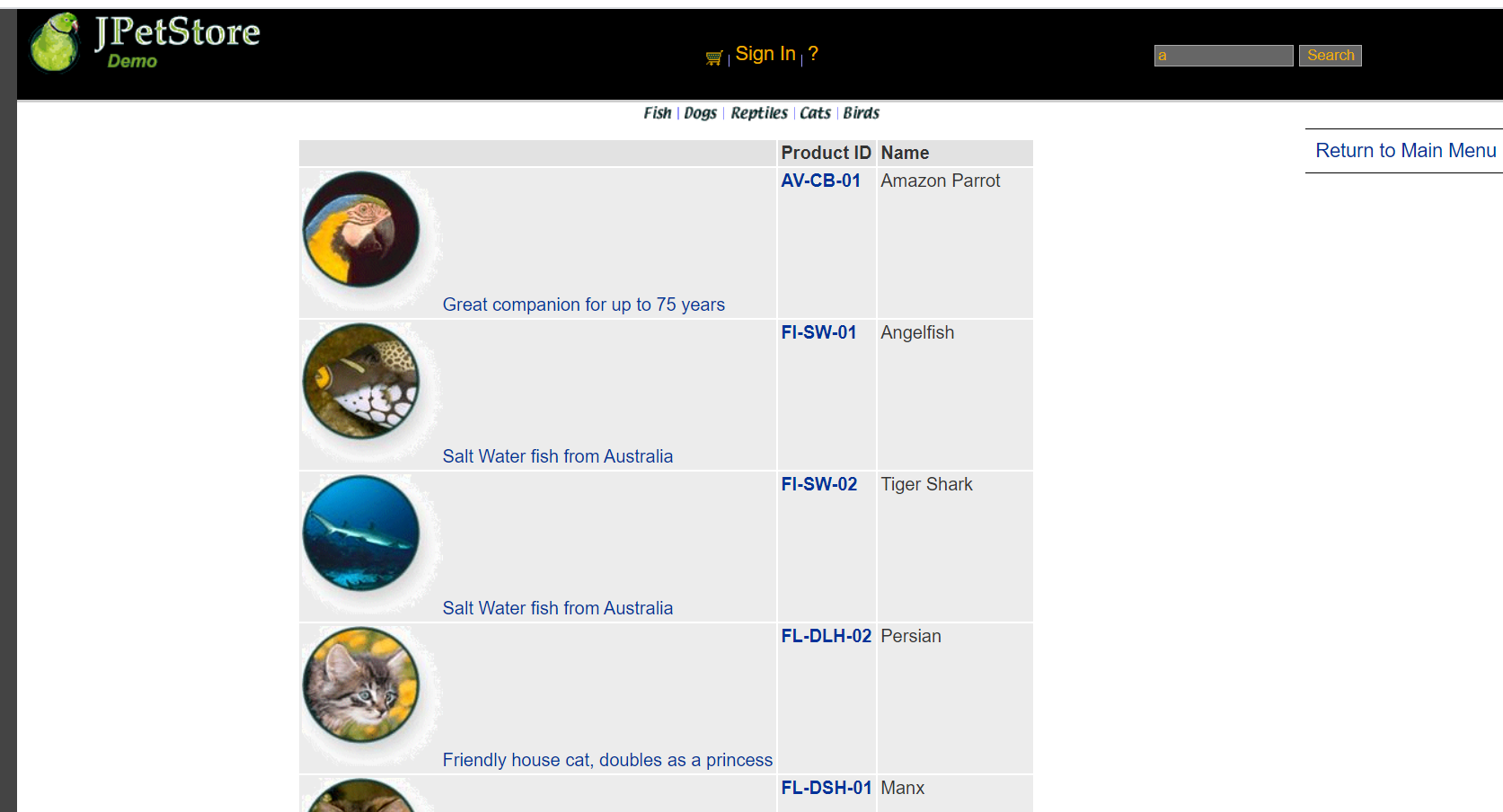


ViewItemServlet如下：

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);  
 }  
}

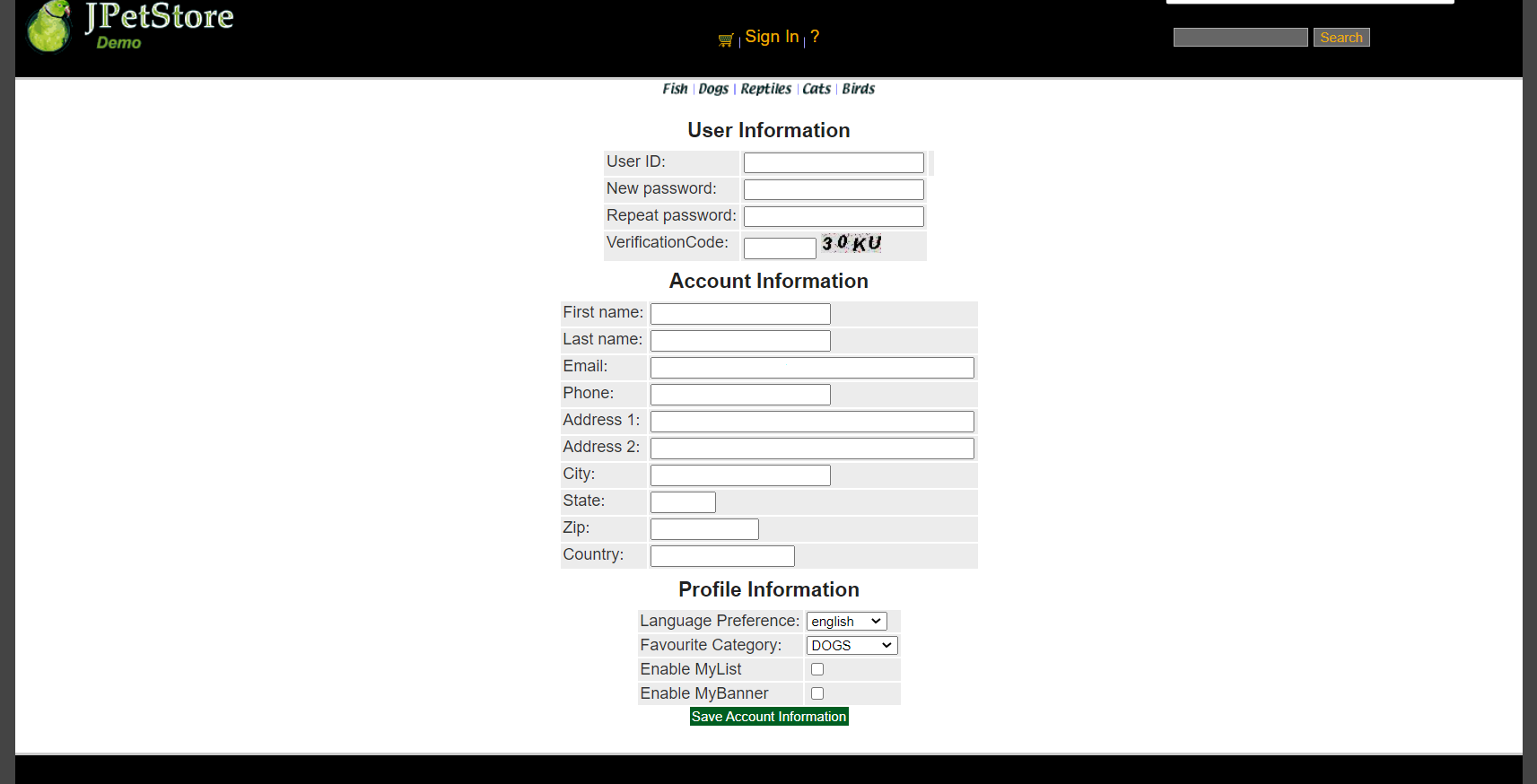
在实现搜索功能时，SearchProducts.jsp为搜索界面的展示界面。通过关键字进行查处，跳转页面，显示所有含有关键字的项目。





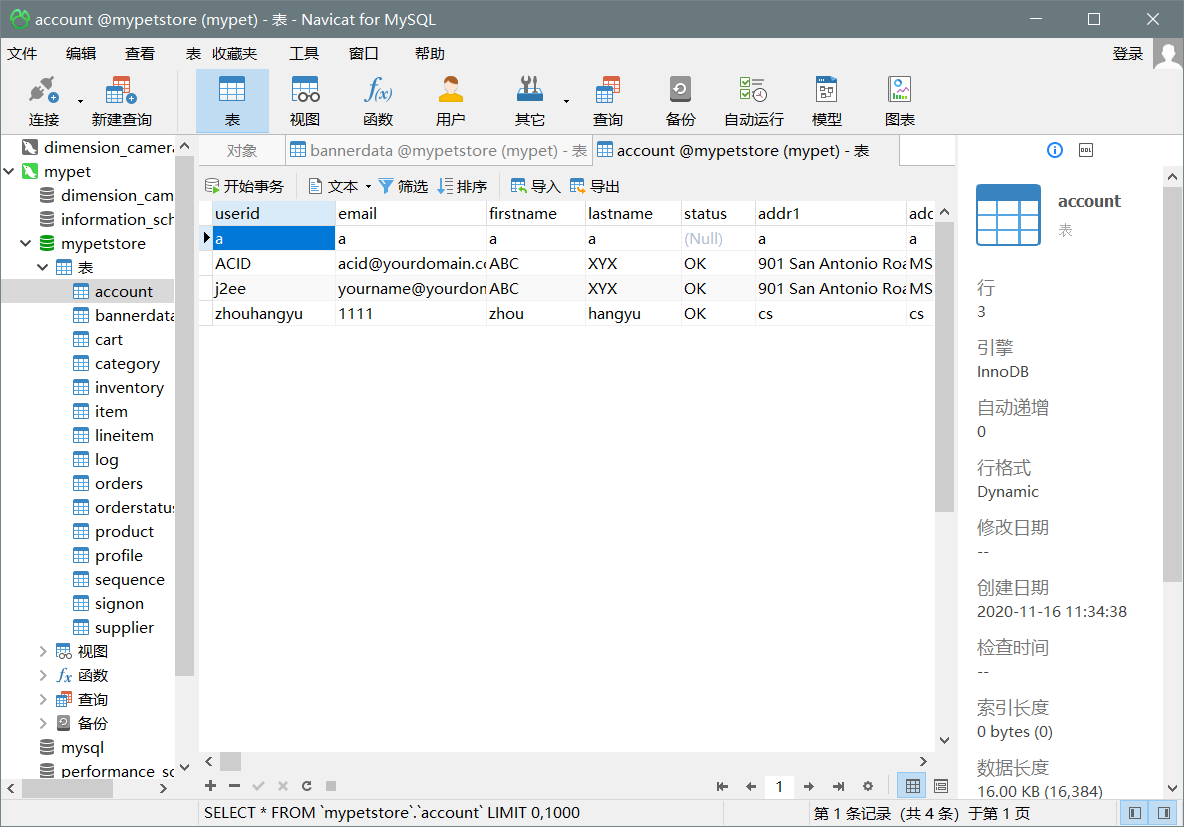
**2、用户管理业务模块，包括用户注册、登录、修改用户信息、查询用户相关订单等业务功能**

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。

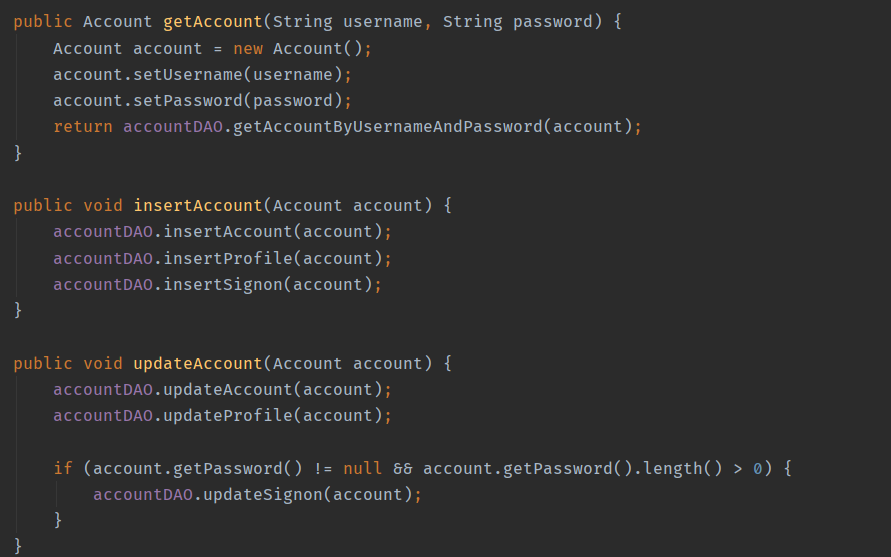
NewAccountServlet：

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);  
 }  
 }  
}

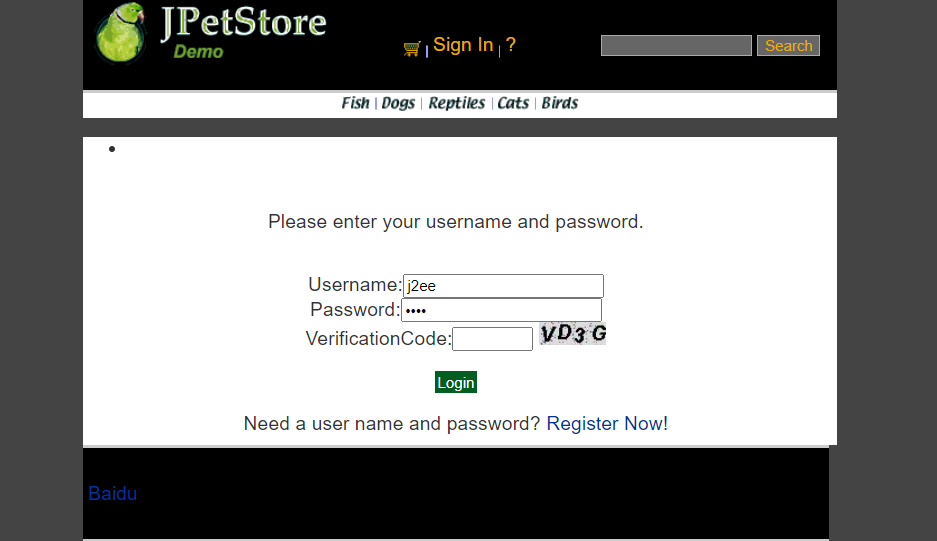
NewAccountFormServlet：

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。

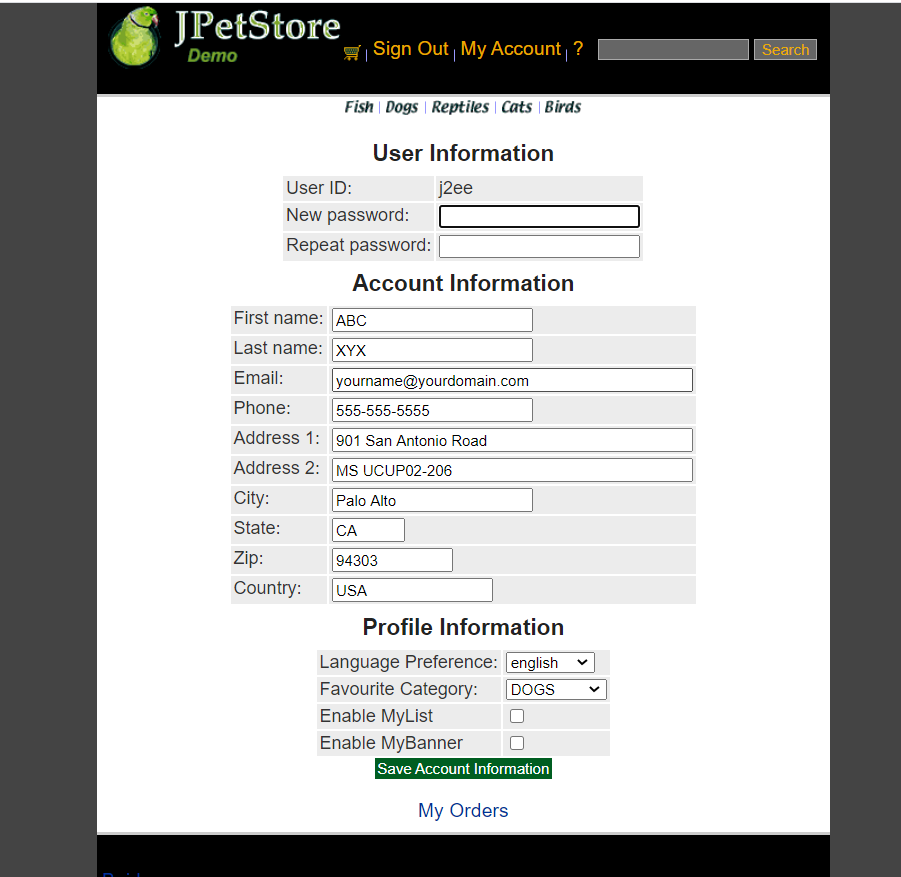
SignOnFormServlet：

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。

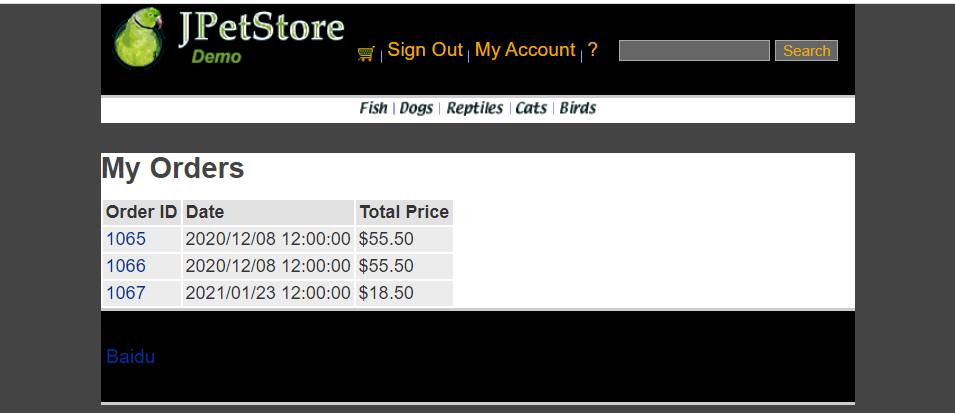
实现进入账户界面的Servlet为EditAccountServlet：

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：



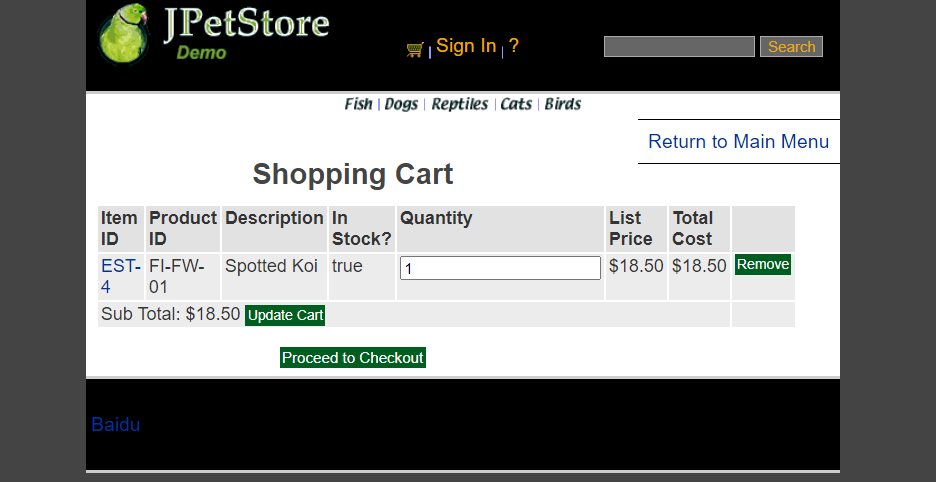
点击my order可以显示该账号的账单。首先设计ListOrders.jsp，接下来完成Order.java的基础模块。创建OrderDAO接口，设计相关的数据库操作。在OrderDAOImpl中完成数据库的具体操作。接下来实现OrderService，成功完成业务层调用。最后通过ViewListOrderServlet实现用户可以从数据库中查看到自己的订单信息。

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：



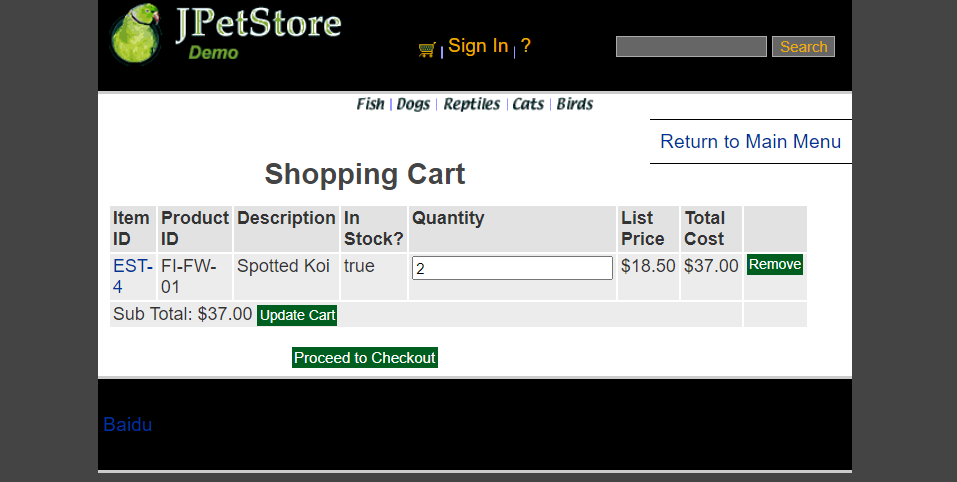
先设计Cart.jsp，用于网页显示购物车栏目信息，接下来具体实现Cart.java使得用户可以创建Cart类。通过AddItemToCartServlet来实现用户将商品添加到购物车。

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：



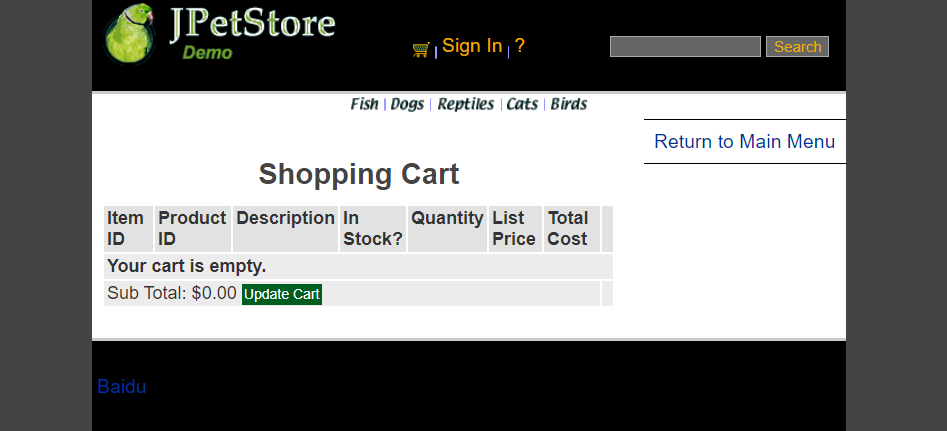
当修改Quantity的值时，点击Update，可以通过利用UpdateCartQuantitiesServlet实现信息更新以及页面更新，传递商品的数量信息获取后进行相关操作。

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。

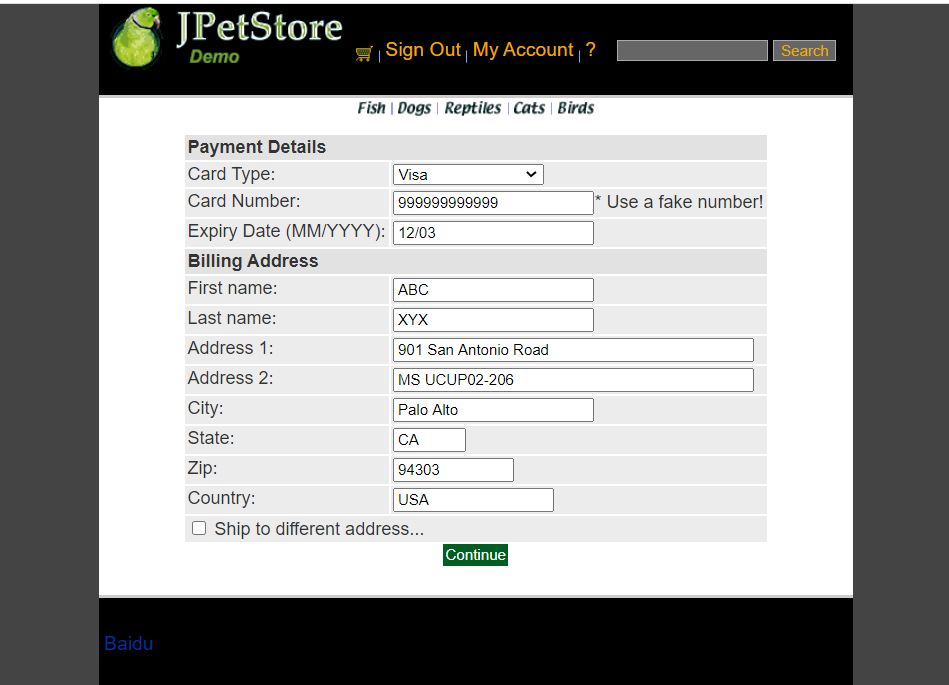
由于后续操作，RemoveItemFromCartServlet已经改为改变会持续化保持在数据库中，不再从session中获取

RemoveItemFromCartServlet：

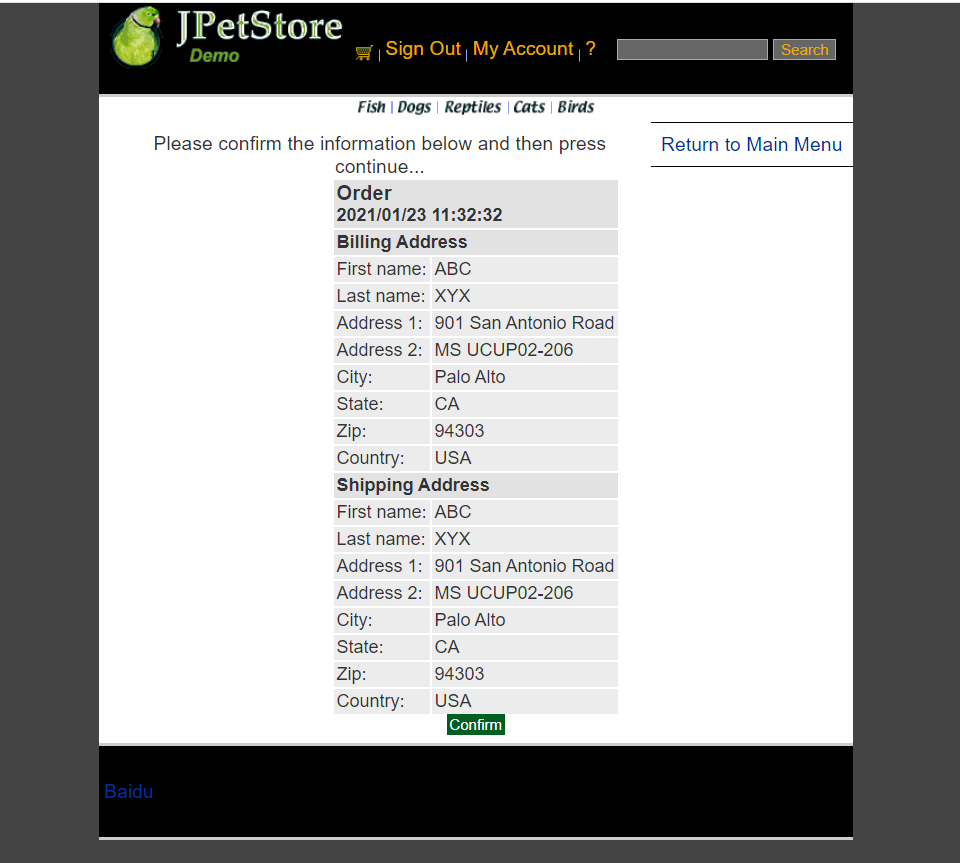
public class RemoveItemFromCartServlet extends HttpServlet {  
  
 private static final String *VIEW\_CART* = "/WEB-INF/jsp/cart/Cart.jsp";  
 private static final String *ERROR* = "/WEB-INF/jsp/common/Error.jsp";  
  
 private String workingItemId;  
 private Cart cart;  
 private CartItem cartItem = 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：

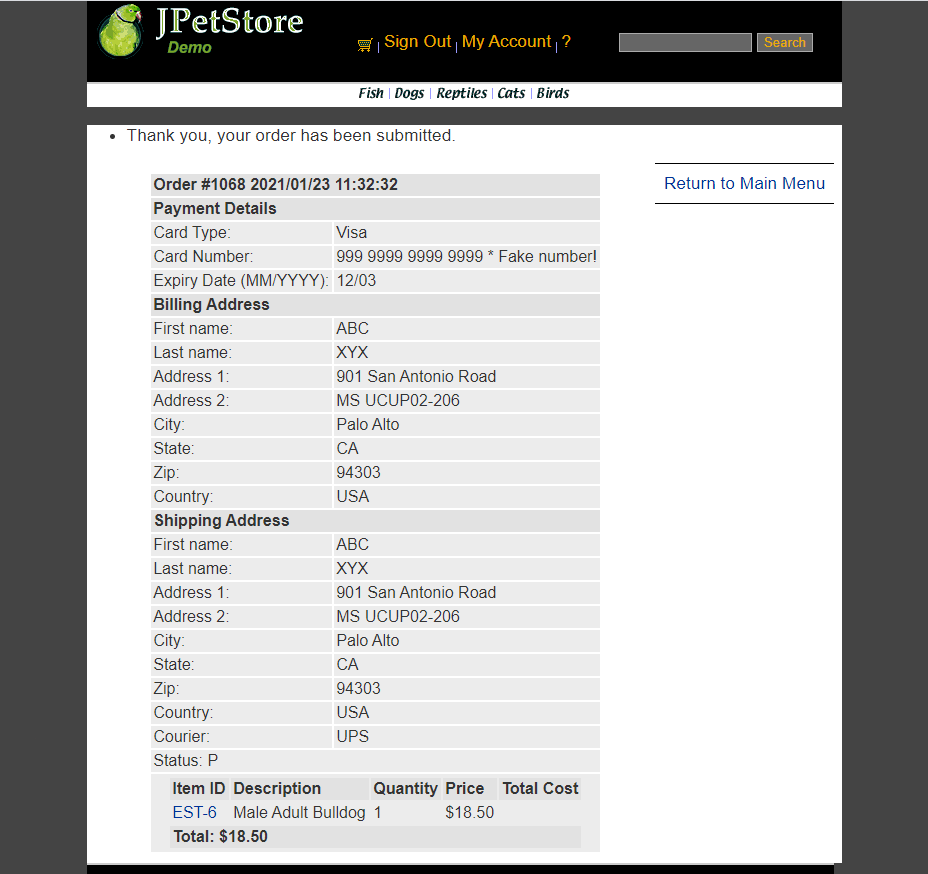
NewOrderForm.jsp



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);  
 }  
 }  
}

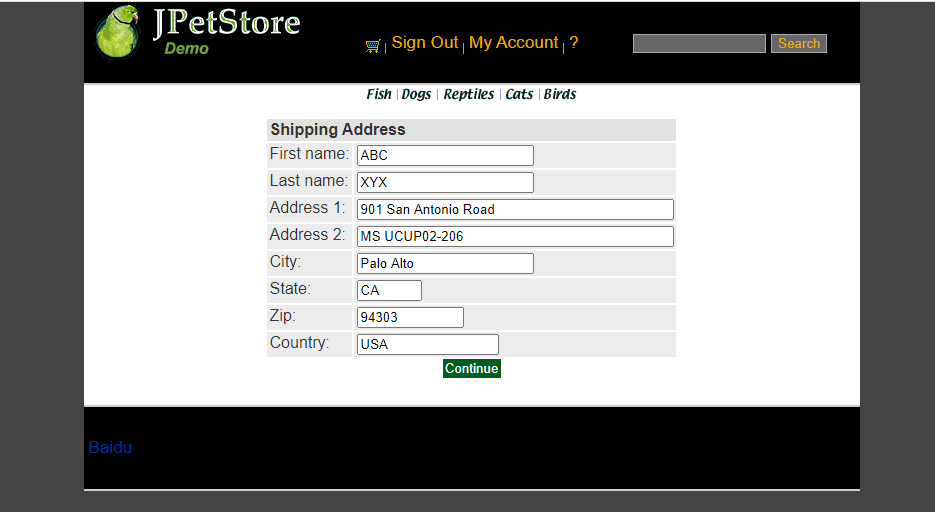
ConfirmOrderFormServlet：

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：

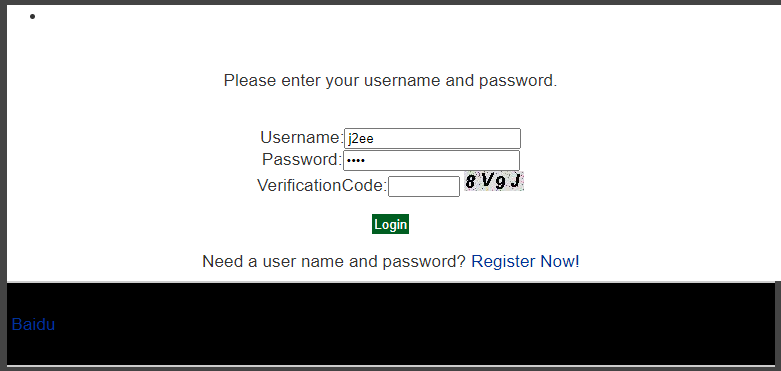
public class ShippingAddressServlet extends HttpServlet {  
 private static final String *CONFIRM\_ORDER\_FORM* = "/WEB-INF/jsp/order/ConfirmOrder.jsp";  
  
 private Order order;  
 private String shipToFirstName;  
 private String shipToLastName;  
 private String shipAddress1;  
 private String shipAddress2;  
 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);  
 }  
}

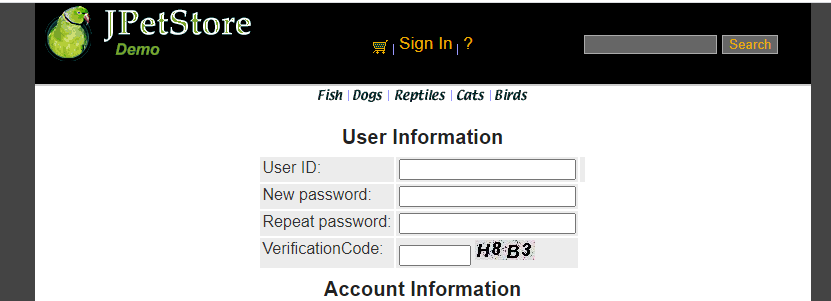
ShippingFormServlet：

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**





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。

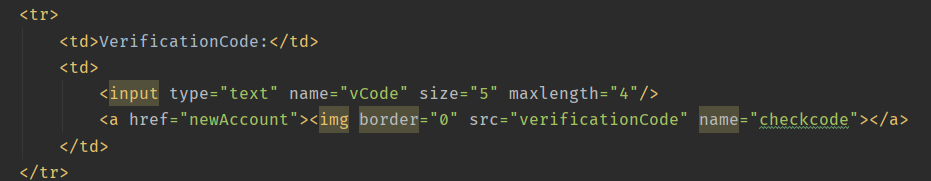
VerificationCode.java：

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：

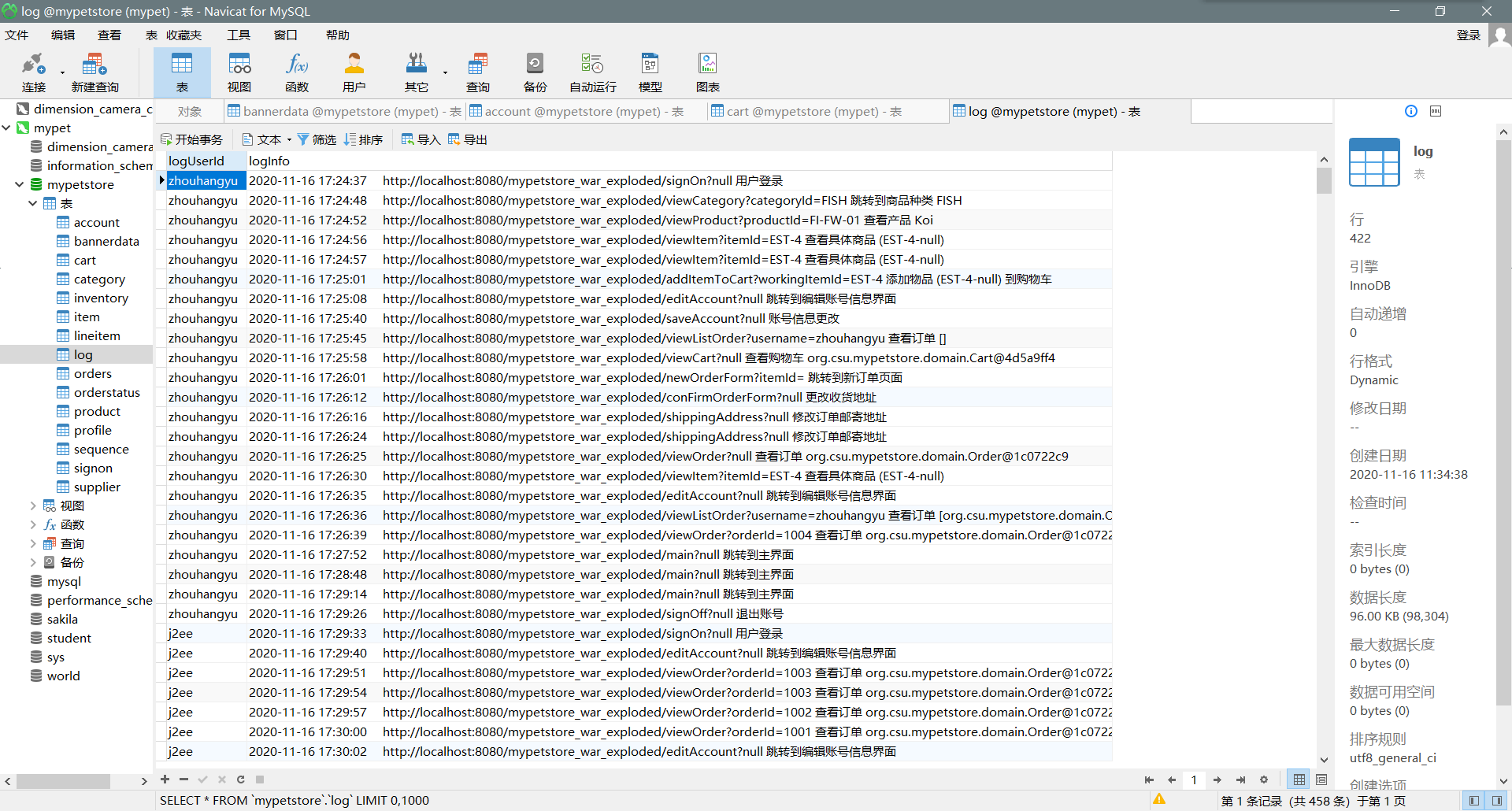


在SignOnServlet和NewAccountServlet中设置相应的获取验证码的操作：



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



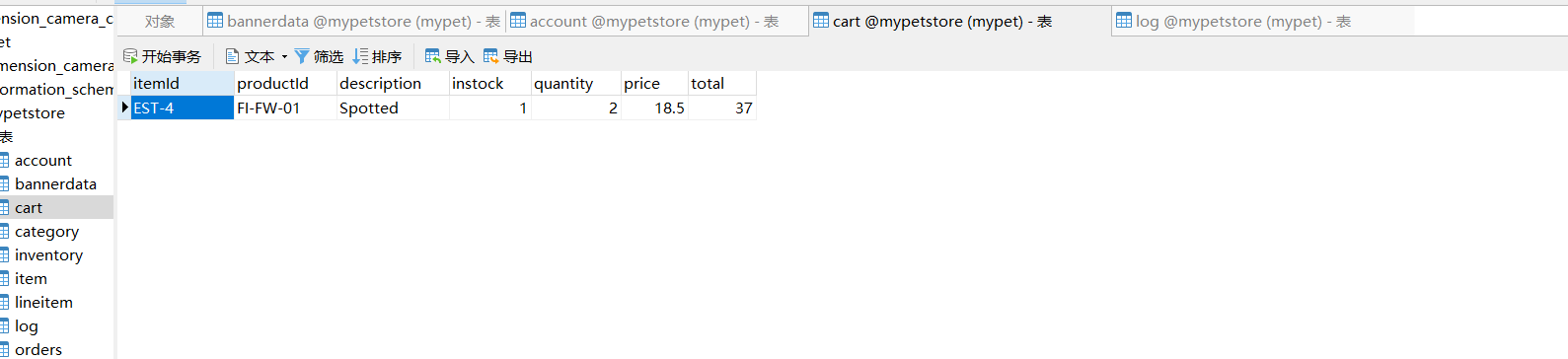
利用Log.java和HTTPServletRequest的各种方法获取当前页面跳转时的用户账号、跳转时间、当前操作页面的网址、具体操作页面的名字、操作的商品信息，再加上自定义的具体操作说明，传输给LogService的相关方法即可将其插入到数据库中记录下来。

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。

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：

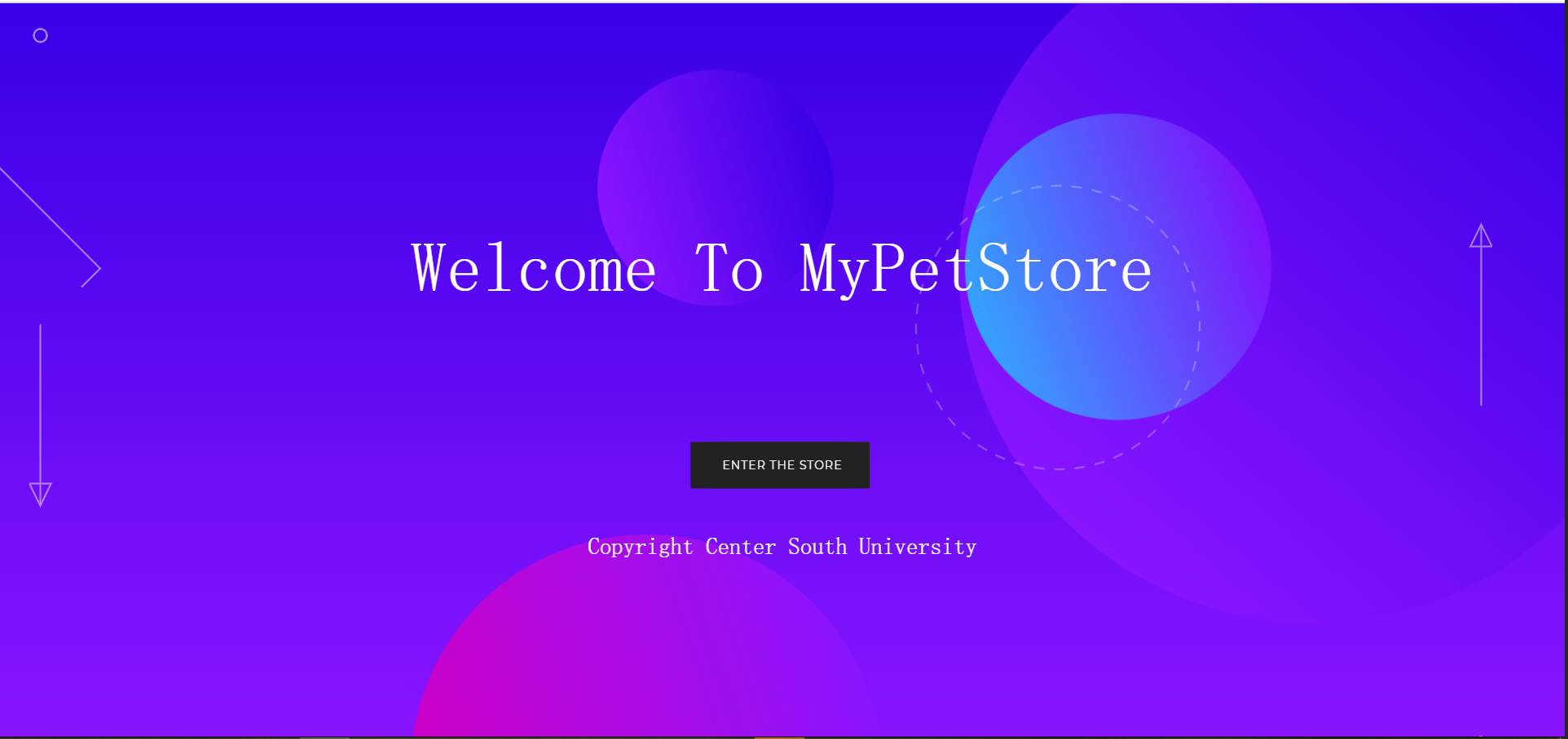
CartItemDAOImpl cartItemDAO = new CartItemDAOImpl();  
if(cart.containsItemId(workingItemId)){  
 *//已有该物品，数量加一* cart.incrementQuantityByItemId(workingItemId);  
  
 if(account != null){  
 HttpServletRequest httpRequest= (HttpServletRequest) request;  
 String strBackUrl = "http://" + request.getServerName() + ":" + 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、额外任务**

**在额外方面，我们小组修改了进入时的界面，借助了css，js和html5的技术实现了一个美观的动态网页。**



**Index.html**

<!DOCTYPE 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>

**四、小组分工**

**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**