本章内容

- 购物车功能

01-CartItem类

- 模型分析
  - 

- CartItem类
  - 成员变量
    - 图书编号
    - 图书图片
    - 图书名称
    - 商品数量
    - 商品单价
    - 商品金额
      - 商品金额=商品数量*商品单价
  - 成员方法
    - 数量减1
    - 数量加1
- 代码实现

```
/**
 * 购物车项
 */
public class CartItem {

    private Integer bookId;//图书编号
    private String imgPath;//图书图片
    private String bookName;//图书名称
    private Integer count;//商品数量
```

```
    private Double price;//商品单价
    private Double amount;//商品金额


    /**
     * 商品数量减一
     */
    public void countDecrease() {
        this.count--;
    }

    /**
     * 商品数量加一
     */
    public void countIncrease() {
        this.count++;
    }

    /**
     * 商品金额 = 商品单价 * 商品数量
     * @return
     */
    public Double getAmount() {
        return price * count;
    }

    public void setAmount(Double amount) {
        this.amount = amount;
    }
}
```

02-Cart类

- Cart类
  - 成员变量
    - 所有购物车项
    - 商品总数量
    - 商品总金额
  - 成员方法
    - 获取所有购物车项
    - 指定购物车项数量减1
    - 指定购物车项数量加1
    - 删除指定购物车项
    - 修改指定购物车项的数量
    - 获取商品总数量
    - 获取商品总金额
    - 添加图书到购物车
      - 如果有图书，数量加1
      - 如果没有图书，数量为1

- 代码实现

```
public class Cart {

    private Integer totalCount;//商品总数量
    private Double totalAmount;//商品总金额
    private Map<Integer, CartItem> cartItemMap = new HashMap<>();//所有购物车项


    /**
     * 获取所有购物车项
     * @return
     */
    public Map<Integer, CartItem> getCartItemMap() {
        return cartItemMap;
    }

    /**
     * 指定购物车项数量减一
     * @param bookId : 指定购物车项
     */
    public void cartItemCountDecrease(Integer bookId) {
        cartItemMap.get(bookId).countDecrease();
    }

    /**
     * 指定购物车项数量级加一
     * @param bookId : 指定购物车项
     */
    public void cartItemCountIncrease(Integer bookId) {
        cartItemMap.get(bookId).countIncrease();
    }

    /**
     * 移除指定购物车项
     * @param bookId : 指定购物车项
     */
    public void removeCartItem(Integer bookId) {
        cartItemMap.remove(bookId);
    }


    /**
     * 修改指定购物车项的数量
     * @param bookId : 指定购物车项
     * @param newCount : 指定数量
     */
    public void updateCartItemCount(Integer bookId, Integer newCount) {
        cartItemMap.get(bookId).setCount(newCount);
    }


    /**
     * 获取商品总数量
     * @return
     */
    public Integer getTotalCount() {
```

```
        Collection<CartItem> cartItems = cartItemMap.values();
        Integer totalCount = 0;
        for (CartItem cartItem : cartItems) {
            totalCount += cartItem.getCount();
        }
        this.totalCount = totalCount;
        return this.totalCount;
    }

    /**
     * 获取商品总金额
     * @return
     */
    public Double getTotalAmount() {
        Collection<CartItem> cartItems = cartItemMap.values();
        Double totalAmount = 0.0;
        for (CartItem cartItem : cartItems) {
            totalAmount += cartItem.getAmount();
        }
        this.totalAmount = totalAmount;
        return this.totalAmount;
    }


    /**
     * 将指定图书添加到购物车
     * @param book : 指定图书
     */
    public void addBook2Cart(Book book) {
        Integer bookId = book.getBookId();
        CartItem cartItem = cartItemMap.get(bookId);
        if (cartItem == null) {
            //没有该图书，数量为1
            cartItem = new CartItem(
                    book.getBookId(),
                    book.getImgPath(),
                    book.getBookName(),
                    1,
                    book.getPrice(),
                    book.getPrice() * book.getCount()
            );
            cartItemMap.put(bookId, cartItem);
        } else {
            //有该图书，数量加1
            cartItemCountIncrease(bookId);
        }
    }
}
```

```
Book book1 = new Book(
        1,
        "西游记",
        "黄贯中",
        11.0,
```

```
        11,
        100,
        "static/img/xiyouji.jpg"
);
Book book2 = new Book(
        1,
        "西游记",
        "黄贯中",
        11.0,
        11,
        100,
        "static/img/xiyouji.jpg"
);
Book book3 = new Book(
        2,
        "红楼梦",
        "老邱",
        22.0,
        22,
        100,
        "static/img/hongloumeng.jpg"
);

Cart cart = new Cart();
cart.addBook2Cart(book1);
cart.addBook2Cart(book2);
cart.addBook2Cart(book3);

System.out.println("商品总数量 : " + cart.getTotalCount());
System.out.println("商品总金额 : " + cart.getTotalAmount());
```

03-添加商品进购物车

- 需求
  - 在index.html，点击加入购物车按钮，加入购物车成功后，跳转到index.html，显示商品总数量
    - 如果是第一次购物，那么需要创建购物车，再将图书添加到购物车
    - 如果不是第一次购物，那么直接将图书添加到购物车
- 代码实现

```
@WebServlet("/cart")
public class CartServlet extends ModelBaseServlet {


    /**
     * 添加图书到购物车
     *
     * @param request
     * @param response
     */
    public void addBook2Cart(HttpServletRequest request, HttpServletResponse
response) {
        String bookId = request.getParameter("bookId");
        BookService bookService = new BookServiceImpl();
```

```
        Book book = null;
        try {
            book = bookService.selectBookById(bookId);
            Cart cart = (Cart) request.getSession().getAttribute("cart");
            if (null == cart) {
                //第一次加入购物车 , 创建Cart对象
                cart = new Cart();
            }
            //将Book对象放入购物车中
            cart.addBook2Cart(book);
            request.getSession().setAttribute("cart", cart);
            //购物车添加成功后，跳转到index.html
            response.sendRedirect(request.getContextPath() + "/index.html");
        } catch (Exception e) {
            e.printStackTrace();
        }

    }


}
```

```
<!--未登录-->
<div class="topbar-right" th:if="${session.existUser==null}">
    <a href="user?method=toLoginPage" class="login">登录</a>
    <a href="user?method=toRegistPage" class="register">注册</a>
    <a href="protected/cart?method=toCartPage"
            class="cart iconfont icon-gouwuche">
        购物车
        <div class="cart-num"
th:if="${session.cart==null||session.cart.totalCount==0}">0</div>
        <div class="cart-num"
th:unless="${session.cart==null||session.cart.totalCount==0}"
th:text="${session.cart.totalCount}">0</div>
    </a>
    <a href="pages/manager/book_manager.html" class="admin">后台管理</a>
</div>
<!--已登录-->
<div class="topbar-right" th:unless="${session.existUser==null}">
    <span>欢迎你<b th:text="${session.existUser.userName}">张总</b></span>
    <a href="user?method=logout" class="register">注销</a>
    <a
            href="protected/cart?method=toCartPage"
            class="cart iconfont icon-gouwuche">
        购物车
        <div class="cart-num"
th:if="${session.cart==null||session.cart.totalCount==0}">0</div>
        <div class="cart-num"
th:unless="${session.cart==null||session.cart.totalCount==0}"
th:text="${session.cart.totalCount}">0</div>
    </a>
    <a href="pages/manager/book_manager.html" class="admin">后台管理</a>
</div>
```

```
<div class="list-content">
    <div class="list-item" th:each="book : ${bookList}">
        <img th:src="${book.imgPath}" alt="">
        <p th:text="${book.bookName}">书名:活着</p>
        <p th:text="${book.author}">作者:余华</p>
        <p th:text="${book.price}">价格:￥66.6</p>
        <p th:text="${book.sales}">销量:230</p>
        <p th:text="${book.stock}">库存:1000</p>
        <a th:href="@{/cart(method='addBook2Cart',bookId=${book.bookId})}">加入购物车
</a>
    </div>

</div>
```

04-显示购物车列表

- 需求
  - 

  - 在index.html，点击购物车按钮，跳转到cart.html，并展示购物车列表。
- 代码实现

```
@WebServlet("/cart")
public class CartServlet extends ModelBaseServlet {


    /**
     * 跳转到购物车页面
     *
     * @param request
     * @param response
     */
    public void toCartPage(HttpServletRequest request, HttpServletResponse response)
{
        try {
            processTemplate("cart/cart", request, response);
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
```

```
}
```

```
//cart.html

<div class="list">
    <div class="w">
        <table>
            <thead>
            <tr>
                <th>图片</th>
                <th>商品名称</th>

                <th>数量</th>
                <th>单价</th>
                <th>金额</th>
                <th>操作</th>
            </tr>
            </thead>
            <!-- 购物车为空 -->
            <tbody th:if="${session.cart==null||session.cart.totalCount==0}">
            <tr>
                <td colspan="6">
                    <a href="index.html">购物车空空如也，请抓紧购物吧!!!!</a>
                </td>
            </tr>
            </tbody>
            <!-- 购物车不为空 -->
            <tbody th:unless="${session.cart==null||session.cart.totalCount==0}">
            <tr th:each="cartItemEntry : ${session.cart.cartItemMap}">
                <td>
                    <img th:src="${cartItemEntry.value.imgPath}" alt=""/>
                </td>
                <td th:text="${cartItemEntry.value.bookName}">活着</td>
                <td>
                    <span class="count">-</span>
                    <input class="count-num" type="text"
th:value="${cartItemEntry.value.count}"/>
                    <span class="count">+</span>
                </td>
                <td th:text="${cartItemEntry.value.price}">36.8</td>
                <td th:text="${cartItemEntry.value.amount}">36.8</td>
                <td><a href="">删除</a></td>
            </tr>

            </tbody>
        </table>
        <div class="footer">
            <div class="footer-left">
                <a href="" class="clear-cart">清空购物车</a>
                <a href="#">继续购物</a>
            </div>
            <div class="footer-right">

                <!--购物车为空-->
```

```
                <div th:if="${session.cart==null||session.cart.totalCount==0}">
                    共<span >0</span>件商品
                </div>
                <div class="total-price"
th:if="${session.cart==null||session.cart.totalCount==0}">
                    总金额<span >0.0</span>元
                </div>

                <!--购物车不为空-->
                <div th:unless="${session.cart==null||session.cart.totalCount==0}">
                    共<span  th:text="${session.cart.totalCount}">3</span>件商品
                </div>
                <div class="total-price"
th:unless="${session.cart==null||session.cart.totalCount==0}">
                    总金额<span th:text="${session.cart.totalAmount}">99.9</span>元
                </div>

                <a class="pay" href="checkout.html">去结账</a>
            </div>
        </div>
    </div>
</div>
```

05-清空购物车

- 需求
  - 在cart.html中，点击清空购物车按钮，将Session域中的Cart对象移除，从而实现清空购物车功能。
- 代码实现

```
<a th:href="@{/cart(method='clearCart')}" class="clear-cart">清空购物车</a>
```

```
/**
 * 清空购物车
 *
 * @param request
 * @param response
 */
public void clearCart(HttpServletRequest request, HttpServletResponse response) {
    request.getSession().removeAttribute("cart");

    try {
        response.sendRedirect(request.getContextPath() + "/cart?method=toCartPage");
    } catch (IOException e) {
        e.printStackTrace();
    }
}
```

06-购物项减一

- 需求
  - 在cart.html中，点击减号按钮,如果当前购物车项数量为1时删除；如果当前购物车项数量大于1时减一。
- 代码实现

```
<span class="count">
    <a
th:href="@{/protected/cart(method='cartItemCountDecrease',bookId=${cartItemEntry.valu
e.bookId})}">
        -
    </a>
</span>
```

```
/**
 * 购物车项数量减一
 * @param request
 * @param response
 */
public void cartItemCountDecrease(HttpServletRequest request, HttpServletResponse
response){
    Integer bookId = Integer.parseInt(request.getParameter("bookId"));
    Cart cart = (Cart) request.getSession().getAttribute("cart");
    Integer count = cart.getCartItemMap().get(bookId).getCount();
    if (count <= 1) {
        //如果当前购物车项数量为1时删除
        cart.removeCartItem(bookId);
    } else {
        //如果当前购物车项数量大于1时减一
        cart.cartItemCountDecrease(bookId);
    }

    try {
        response.sendRedirect(request.getContextPath()+"/protected/cart?
method=toCartPage");
    } catch (IOException e) {
        e.printStackTrace();
    }
}
```

07-购物项加一

- 需求
  - 在cart.html中，点击加号按钮，在原有商品数量基础上加1
- 代码实现

```
<span class="count">
    <a
th:href="@{/protected/cart(method='cartItemCountIncrease',bookId=${cartItemEntry.valu
e.bookId})}">
        +
    </a>
</span>
```

```
 * 购物车项加一
 * @param request
 * @param response
 */
public void cartItemCountIncrease(HttpServletRequest request, HttpServletResponse
response){
    Integer bookId = Integer.parseInt(request.getParameter("bookId"));
    Cart cart = (Cart) request.getSession().getAttribute("cart");
    cart.cartItemCountIncrease(bookId);
    try {
        response.sendRedirect(request.getContextPath()+"/protected/cart?
method=toCartPage");
    } catch (IOException e) {
        e.printStackTrace();
    }
}
```

08-删除购物车项

- 需求
  - 在cart.html中，点击删除按钮，删除指定购物车项
- 代码实现

```
<a
th:href="@{/protected/cart(method='removeCartItem',bookId=${cartItemEntry.value.bookI
d})}">删除</a>
```

```
/**
 * 删除购物车项
 * @param request
 * @param response
 */
public void removeCartItem(HttpServletRequest request, HttpServletResponse response){
    Integer bookId = Integer.parseInt(request.getParameter("bookId"));
    Cart cart = (Cart) request.getSession().getAttribute("cart");
    cart.removeCartItem(bookId);
    try {
        response.sendRedirect(request.getContextPath()+"/protected/cart?
method=toCartPage");
    } catch (IOException e) {
        e.printStackTrace();
    }
}
```

09-修改购物车项数量

- 需求

- 在cart.html中，用户在文本框输入新数据后，根据用户输入修改购物车项数量

- 代码实现

```
<input class="count-num" type="text" th:value="${cartItemEntry.value.count}"
       @change="updateCartItem()"/>
<input type="hidden" th:value="${cartItemEntry.value.bookId}">


var vue = new Vue({
    el: "#app",
    methods: {
        updateItemCount() {
            //获取新的数量
            var count = event.target.value;
            //获取bookId
            var bookId = event.target.nextElementSibling.value;
            if (count >= 1) {
                //修改指定购物项数量
                location.href = "cart?method=updateItemCount&id=" + bookId +
"&count=" + count;
            } else {
                //删除指定购物项
                location.href = "cart?method=removeCartItem&id=" + bookId;
            }

        }
    }
});
```

```
/**
 * 修改指定购物车项数量
 * @param request
 * @param response
 */
public void updateItemCount(HttpServletRequest request, HttpServletResponse response)
{
    Integer bookId = Integer.parseInt(request.getParameter("id"));
    Integer count = Integer.parseInt(request.getParameter("count"));
    Cart cart = (Cart) request.getSession().getAttribute("cart");
    cart.updateItemCount(bookId, count);

    try {
        response.sendRedirect(request.getContextPath() + "/cart?method=toCartPage");
    } catch (IOException e) {
        e.printStackTrace();
    }
}
```

10-BigDecimal精度调整

- 问题现象

- 概述
  - 使用BigDecimal类型来进行Double类型数据运算，创建BigDecimal类型对象时将Double类型的数据转换为字符串
- 开发步骤
  - CartItem类使用BigDecimal
  - Cart类使用BigDecimal
- 代码实现

```
public class CartItem {
    public Double getAmount() {
        BigDecimal price = new BigDecimal(this.price + "");
        BigDecimal count = new BigDecimal(this.count + "");
        return price.multiply(count).doubleValue();
    }
}
```

```
public class Cart {

    public Double getTotalAmount() {
        BigDecimal totalAmount = new BigDecimal(0.0 + "");
        Collection<CartItem> cartItems = cartItemMap.values();
        for (CartItem cartItem : cartItems) {
            totalAmount = totalAmount.add(new BigDecimal(cartItem.getAmount() + ""));
        }
        this.totalAmount = totalAmount.doubleValue();
        return totalAmount.doubleValue();
    }

}
```