Lab 13

Xây dựng module check out giỏ hàng

Mục tiêu

- Thực hành với expression language
- Xây dựng chức năng tiếp theo của module quản lý giỏ hàng ở lab 12
    - Check out giỏ hàng
    - Quản lý danh sách đơn hàng của khách hàng
    - Quản lý danh sách đơn hàng của admin (thêm đơn hàng mới, cập nhật trạng thái đơn hàng: preparation --> package --> delivery)

Phần I Bài tập step by step

Bài 1.1

Mục tiêu: Thực hiện module checkout giỏ hàng trong ứng dụng eMarket.

Trong bài trước, chúng ta cũng đã sử dụng expression language rồi, expression language thường đi kèm với JSTL. Nên từ bài 10 chúng ta đều thay thế scriptlet thành JSTL và kết hợp với Expression Language.

Module checkout cần hai trang view: checkout.jsp và confirmation.jsp.

***STEP 1*: Tạo các trang view cho module checkout**

Thêm xử lý checkout trong trang header.jspf, ngay sau đoạn xử lý giỏ hàng, sau thẻ <body> và :

```
<c:if test="${!empty cart && cart.numberOfItems != 0 &&

!fn:contains(pageContext.request.servletPath,'/checkout') &&
                        requestScope['javax.servlet.forward.servlet_path']
ne '/checkout' &&
```

```
!fn:contains(pageContext.request.servletPath,'/viewCart') &&
                        requestScope['javax.servlet.forward.servlet_path']
ne '/viewCart' &&

                        validationErrorFlag ne true &&
                        orderFailureFlag ne true}">

                <a href="<c:url value='checkout'/>">
                  <fmt:message key="proceedCheckout"/>
                </a>
            </c:if>
```

Chúng ta sử dụng biểu thức logic của expression language, proceed checkout chỉ hiển thị với người dùng khi: giỏ hàng không rỗng, không ở trong trang viewCart (trang view cart chúng ta proceed checkout ở menu khác) và không ở trong trang checkout.

Tạo trang checkout.jsp:

```
<c:set var='view' value='/checkout' scope='session' />
<script src="js/jquery.validate.js" type="text/javascript"></script>

<script type="text/javascript">

    $(document).ready(function () {
        $("#checkoutForm").validate({
            rules: {
                name: "required",
                email: {
```

```
                    required: true,
                    email: true
                },
                phone: {
                    required: true,
                    number: true,
                    minlength: 9
                },
                address: {
                    required: true
                },
                creditcard: {
                    required: true,
                    creditcard: true
                }
            }
        });
    });
</script>

<div id="container">
    <div class="one-half">
        <div class="heading_bg">
            <h2>Checkout</h2>
        </div>
```

```
        <p>
            <strong>In order to purchase the items in your shopping cart,
please provide us with the following information:</strong>
        </p>
        <c:if test="${!empty orderFailureFlag}">
            <p style="color: #c00; font-style: italic>We were unable to
process your order. Please try again!</p>
        </c:if>
        <form id="checkoutForm" action="<c:url value='purchase' />"
method="post">
            <fieldset>
                <label>Name<span class="required">*</span></label>
                <input type="text" name="name" id="name"
value="${param.name}" />
            </fieldset>
            <fieldset>
                <label>Email<span class="required">*</span></label>
                <input type="text" name="email" id="email"
value="${param.email}" />
            </fieldset>
            <fieldset>
                <label>Phone <span class="required">*</span></label>
                <input type="text" name="phone" id="phone"
value="${param.phone}" />
            </fieldset>
            <fieldset>
                <label>Address <span class="required">*</span></label>
```

```
                <input type="text" size="45" name="address" id="address"
value="${param.address}" />

            </fieldset>

            <fieldset>

                <label>City <span class="required">*</span></label>

                <input type="text" size="45" name="cityRegion"
id="cityRegion" value="${param.cityRegion}" />

            </fieldset>

            <fieldset>

                <label>Credit Card Number<span
class="required">*</span></label>

                <input type="text" size="45" name="creditcard"
id="creditcard" value="${param.creditcard}" />

            </fieldset>

            <fieldset>

                <input value="Submit purchase" class="button white"
type="submit">

            </fieldset>

        </form>

    </div>

    <div class="one-half last">

        <div class="heading_bg">

            <h2>Order Information</h2>

        </div>

        <p>

            <strong>Next-working day delivery is guaranteed</strong>

        </p>
```

```
        <p>
            <strong>
                A
                <fmt:formatNumber type="currency" currencySymbol="&euro; "
value="${initParam.deliveryFee}"/>
                delivery surcharge is applied to all purchase orders
            </strong>
        </p>
        <table>
            <th>Total</th>
            <th>Delivery Surcharge</th>
            <th>Credit Total</th>
            <tr>
                <td><fmt:formatNumber type="currency"
currencySymbol="&dollar; " value="${cart.subtotal}"/></td>
                <td><fmt:formatNumber type="currency"
currencySymbol="&dollar; " value="${initParam.deliveryFee}"/></td>
                <td><fmt:formatNumber type="currency"
currencySymbol="&dollar; " value="${cart.total}"/></td>
            </tr>
        </table>
    </div>
    <div style="clear:both; height: 40px"></div>
</div>
```

Thêm init parameter liên quan đến tiền vận chuyển cho context servlet trong trang web.xml:

```
    <context-param>
        <description>The delivery fee</description>
```

```
        <param-name>deliveryFee</param-name>
        <param-value>5.00</param-value>
    </context-param>
```

Tạo trang confirmation.jsp

```
<div id="container">

    <div class="one">

        <div class="heading_bg">

            <h2>Confirmation</h2>

        </div>

        <p id="confirmationText">

            <strong><fmt:message key="successMessage" /></strong>

            <br />

            <br />

            <fmt:message key="confirmationNumberMessage"/>

            <strong>${orderRecord.confirmationNumber}</strong>

            <br>

            <fmt:message key="contactMessage"/>

            <br><br>

            <fmt:message key="thankYouMessage"/>

        </p>

    </div>

    <div class="two-third">

        <div class="heading_bg">

            <h3><fmt:message key="orderSummary"/></h3>

        </div>

        <table>
```

```
            <th><fmt:message key="product"/></th>
            <th><fmt:message key="quantity"/></th>
            <th><fmt:message key="price"/></th>

            <c:forEach var="orderedProduct" items="${orderedProducts}"
varStatus="iter">
                <tr>
                    <td>
                        ${products[iter.index].name}
                    </td>
                    <td>
                        ${orderedProduct.quantity}
                    </td>
                    <td>
                        <fmt:formatNumber type="currency"
currencySymbol="&dollar; "
                                          value="${products[iter.index].price
* orderedProduct.quantity}"/>
                    </td>
                </tr>
            </c:forEach>

            <tr>
                <td colspan="2"><strong><fmt:message
key="surcharge"/>:</strong></td>
                <td>
                    <fmt:formatNumber type="currency"
```

```
                                    currencySymbol="&euro; "
                                    value="${initParam.deliveryFee}"/></td>
            </tr>

            <tr>
                <td colspan="2"><strong><fmt:message
key="total"/>:</strong></td>
                <td>
                    <fmt:formatNumber type="currency"
                                    currencySymbol="&euro; "
                                    value="${orderRecord.amount}"/></td>
            </tr>

           <tr>
                <td colspan="3"><strong><fmt:message
key="dateProcessed"/>:</strong>
                    ${orderRecord.dateCreated}
            </tr>
        </table>
    </div>
    <div class="sidebar_right">
        <div class="heading_bg">
            <h3><fmt:message key="deliveryAddress" /></h3>
        </div>
        <table>
            <tr>
```

```
                <td colspan="3">
                    ${customer.name}
                    <br>
                    ${customer.address}
                    <br>
                    <fmt:message key="city"/> ${customer.cityRegion}
                    <br>
                    <hr>
                    <strong><fmt:message key="email"/>:</strong>
${customer.email}
                    <br>
                    <strong><fmt:message key="phone"/>:</strong>
${customer.phone}
                </td>
            </tr>
        </table>
    </div>
    <div style="clear:both; height: 40px"></div>
</div>
```

STEP 2: Thực hiện xử lý trong ControllerServlet cho đơn hàng.

Chúng ta cần thêm thông tin vào CSDL cho mỗi đơn hàng, bao gồm:

- Thêm mới customer
- Thêm mới order
- Thêm mới order-product tương ứng với mỗi một mặt hàng trong 1 order (quan hệ giữa bảng order và product là quan hệ nhiều nhiều).

Chúng ta sẽ thực hiện tất cả giao dịch này trong transaction.

Trong package session_bean, tạo class OrderManager.java, stateless session bean, kiểu transaction là Container.

```
package session_bean;

import cart.ShoppingCart;
import cart.ShoppingCartItem;
import entity.Customer;
import entity.CustomerOrder;
import entity.OrderedProduct;
import entity.OrderedProductPK;
import entity.Product;
import java.math.BigDecimal;
import java.text.DateFormat;
import java.text.SimpleDateFormat;
import java.util.ArrayList;
import java.util.Date;
import java.util.HashMap;
import java.util.List;
import java.util.Map;
import java.util.Random;
import javax.annotation.Resource;
import javax.ejb.EJB;
import javax.ejb.SessionContext;
import javax.ejb.Stateless;
import javax.ejb.TransactionAttribute;
import javax.ejb.TransactionAttributeType;
import javax.ejb.TransactionManagement;
import javax.ejb.TransactionManagementType;
import javax.persistence.EntityManager;
import javax.persistence.PersistenceContext;

/**
 *
 * @author ThanDieu
 */
@Stateless
@TransactionManagement(TransactionManagementType.CONTAINER)
public class OrderManager {
    @EJB
    private CustomerOrderSessionBean customerOrderSB;
    @EJB
    private ProductSessionBean productSB;
    @EJB
    private OrderedProductSessionBean orderedProductSB;
    @EJB
    private CustomerSessionBean customerSB;

    @PersistenceContext(unitName = "eMarketPU")
    private EntityManager em;
    @Resource
    private SessionContext context;
```

```
    @TransactionAttribute(TransactionAttributeType.REQUIRED)
    public int placeOrder(String name, String email, String phone, String
address, String cityRegion, String ccNumber, ShoppingCart cart) {

        try {
            Customer customer = addCustomer(name, email, phone, address,
cityRegion, ccNumber);
            CustomerOrder order = addOrder(customer, cart);
            addOrderedItems(order, cart);
            return order.getOrderId();
        } catch (Exception e) {
            context.setRollbackOnly();
            e.printStackTrace();
            return 0;
        }
    }

    private Customer addCustomer(String name, String email, String phone,
String address, String cityRegion, String ccNumber) {

        Customer customer = new Customer();
        customer.setName(name);
        customer.setEmail(email);
        customer.setPhone(phone);
        customer.setAddress(address);
        customer.setCityRegion(cityRegion);
        customer.setCcNumber(ccNumber);

        customer = customerSB.create(customer);
        return customer;
    }

    private CustomerOrder addOrder(Customer customer, ShoppingCart cart) {

        // set up customer order
        CustomerOrder order = new CustomerOrder();
        order.setCustomerId(customer);
        order.setAmount(BigDecimal.valueOf(cart.getTotal()));

        // create confirmation number
        Random random = new Random();
        int i = random.nextInt(999999999);
        order.setConfirmationNumber(i);
        DateFormat df = new SimpleDateFormat("dd/MM/yy HH:mm:ss");
        order.setDateCreated(df.format(new Date()));

        order = customerOrderSB.create(order);
        return order;
    }

    private void addOrderedItems(CustomerOrder order, ShoppingCart cart) {

        List<ShoppingCartItem> items = cart.getItems();

        // iterate through shopping cart and create OrderedProducts
```

```
        for (ShoppingCartItem scItem : items) {

            int productId = scItem.getProduct().getProductId();

            // set up primary key object
            OrderedProductPK orderedProductPK = new OrderedProductPK();
            orderedProductPK.setOrderId(order.getOrderId());
            orderedProductPK.setProductId(productId);

            // create ordered item using PK object
            OrderedProduct orderedItem = new
OrderedProduct(orderedProductPK);

            // set quantity
            orderedItem.setQuantity(scItem.getQuantity());

            orderedProductSB.create(orderedItem);
        }
    }

    public Map getOrderDetails(int orderId) {

        Map orderMap = new HashMap();

        // get order
        CustomerOrder order = customerOrderSB.find(orderId);

        // get customer
        Customer customer = order.getCustomerId();

        // get all ordered products
        List<OrderedProduct> orderedProducts =
orderedProductSB.findByOrderId(orderId);

        // get product details for ordered items
        List<Product> products = new ArrayList<Product>();

        for (OrderedProduct op : orderedProducts) {

            Product p = (Product)
productSB.find(op.getOrderedProductPK().getProductId());
            products.add(p);
        }

        // add each item to orderMap
        orderMap.put("orderRecord", order);
        orderMap.put("customer", customer);
        orderMap.put("orderedProducts", orderedProducts);
        orderMap.put("products", products);

        return orderMap;
    }
}
```

Các classe: CustomerSessionBean, CustomerOrderSessionBean,

OrderedProductSessionbean được tạo ra tương ứng với các entity classes.

CustomerSessionBean.java

```
package session_bean;

import entity.Customer;
import javax.ejb.Stateless;
import javax.persistence.EntityManager;
import javax.persistence.PersistenceContext;

/**
 *
 * @author ThanDieu
 */
@Stateless
public class CustomerSessionBean extends AbstractSessionBean<Customer> {
    @PersistenceContext(unitName = "eMarketPU")
    private EntityManager em;

    protected EntityManager getEntityManager() {
        return em;
    }

    public CustomerSessionBean(){
        super(Customer.class);
    }
}
```

CustomerOrderSessionBean.java

```
package session_bean;

import entity.CustomerOrder;
import javax.ejb.Stateless;
import javax.persistence.EntityManager;
import javax.persistence.PersistenceContext;

/**
 *
 * @author ThanDieu
 */
@Stateless
public class CustomerOrderSessionBean extends
AbstractSessionBean<CustomerOrder> {
    @PersistenceContext(unitName = "eMarketPU")
    private EntityManager em;

    @Override
    protected EntityManager getEntityManager() {
```

```
        return em;
    }

    public CustomerOrderSessionBean() {
        super(CustomerOrder.class);
    }

    public CustomerOrder find(Object id) {
        CustomerOrder order = em.find(CustomerOrder.class, id);
        em.refresh(order);
        return order;
    }

    public CustomerOrder findByCustomer(Object customer) {
        return (CustomerOrder)
em.createNamedQuery("CustomerOrder.findByCustomer").setParameter("customer",
customer).getSingleResult();
    }
}
```

OrderedProductSessionBean.java

```
package session_bean;

import entity.OrderedProduct;
import java.util.List;
import javax.ejb.Stateless;
import javax.persistence.EntityManager;
import javax.persistence.PersistenceContext;

/**
 *
 * @author ThanDieu
 */
@Stateless
public class OrderedProductSessionBean extends
AbstractSessionBean<OrderedProduct> {
    @PersistenceContext(unitName = "eMarketPU")
    private EntityManager em;

    @Override
    protected EntityManager getEntityManager() {
        return em;
    }

    public OrderedProductSessionBean() {
        super(OrderedProduct.class);
    }

    public List<OrderedProduct> findByOrderId(Object id) {
        return
em.createNamedQuery("OrderedProduct.findByOrderId").setParameter("orderId",
id).getResultList();
```

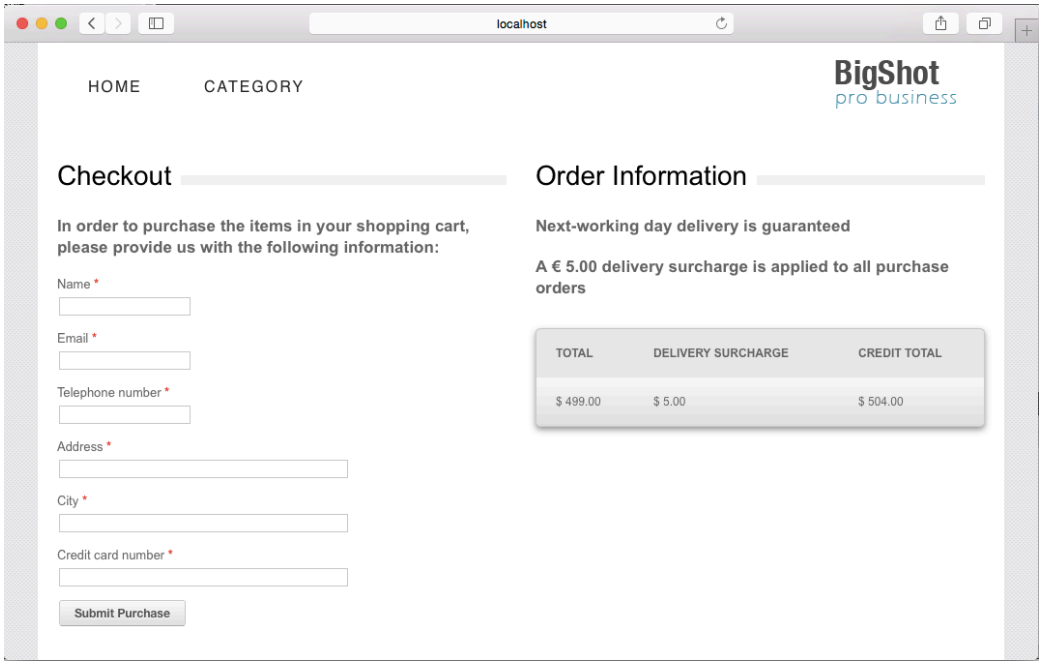
```
    }
}
```

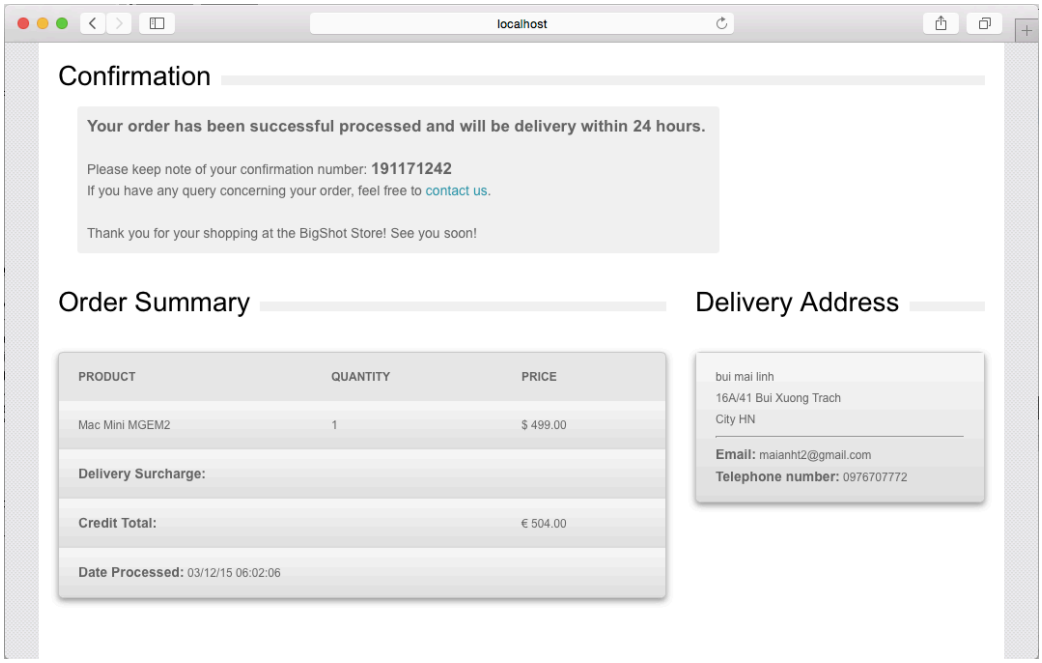
Thực hiện checkout trong hàm doPost của ControllerServlet, như sau:

```
protected void doPost(HttpServletRequest request, HttpServletResponse
response)
            throws ServletException, IOException {
        request.setCharacterEncoding("UTF-8");
        String userPath = request.getServletPath();
        HttpSession session = request.getSession();
        ShoppingCart cart = (ShoppingCart) session.getAttribute("cart");
        Validator validator = new Validator();

        if (userPath.equals("/updateCart")) {
            String productId = request.getParameter("productId");
            String quantity = request.getParameter("quantity");
            boolean invalidEntry = validator.validateQuantity(productId,
quantity);
            if (!invalidEntry) {
                Product product =
productSB.find(Integer.parseInt(productId));
                cart.update(product, quantity);
            }
            userPath = "/viewCart";
        }
        else if (userPath.equals("/purchase")) {
            if (cart != null) {
                String name = request.getParameter("name");
                String email = request.getParameter("email");
                String phone = request.getParameter("phone");
                String address = request.getParameter("address");
                String cityRegion = request.getParameter("cityRegion");
                String ccNumber = request.getParameter("creditcard");

                boolean validationErrorFlag = false;
                validationErrorFlag = validator.validateForm(name, email,
phone, address, cityRegion, ccNumber, request);

                if (validationErrorFlag) {
                    request.setAttribute("validationErrorFlag",
validationErrorFlag);
                    userPath = "/checkout";
                }
                else {
                    int orderId = orderManager.placeOrder(name, email, phone,
address, cityRegion, ccNumber, cart);
                    if (orderId != 0) {

                        // in case language was set using toggle, get
language choice before destroying session
                        Locale locale = (Locale)
session.getAttribute("javax.servlet.jsp.jstl.fmt.locale.session");
                        String language = "";
```

```
                        if (locale != null) {

                            language = (String) locale.getLanguage();
                        }

                        // dissociate shopping cart from session
                        cart = null;

                        // end session
                        session.invalidate();

                        if (!language.isEmpty()) {                        //
if user changed language using the toggle,
                                                                          //
reset the language attribute - otherwise
                            request.setAttribute("language", language);  //
language will be switched on confirmation page!
                        }

                        // get order details
                        Map orderMap = orderManager.getOrderDetails(orderId);

                        // place order details in request scope
                        request.setAttribute("customer",
orderMap.get("customer"));
                        request.setAttribute("products",
orderMap.get("products"));
                        request.setAttribute("orderRecord",
orderMap.get("orderRecord"));
                        request.setAttribute("orderedProducts",
orderMap.get("orderedProducts"));

                        userPath = "/confirmation";

                    // otherwise, send back to checkout page and display
error
                    } else {
                        userPath = "/checkout";
                        request.setAttribute("orderFailureFlag", true);
                    }
                }
            }
        }
        String url = userPath + ".jsp";
        try {
            request.getRequestDispatcher(url).forward(request, response);
        }
        catch (Exception ex) {
            ex.printStackTrace();
       }

}
```

Chạy chương trình và review kết quả:





PHẦN 2: BÀI TẬP LÀM THÊM

Tạo chức năng xem tất cả các order của khách hàng. Sau khi check xử lý order thì xoá các order và khách hàng tương ứng trong CSDL. Chức năng này được đặt trong admin, giống addProduct đã làm ở lab 09.

HẾT