COMP S380F Web Applications: Design and Development
Lab 6: Session – Shopping cart

In this lab, we will implement a shopping cart. The following topics are covered:

- Java synchronized collection classes: CopyOnWriteArrayList, ConcurrentHashMap
- Configuring session properties in web.xml
- JSTL core tags: <c:url> and <c:param>

Task 1: Implementing shopping cart using the session object

1. In IntelliJ, create a **Gradle Web Application** project with the following properties:
   - Category: **Jakarta EE**
   - Name: **Lab06**
   - Template: **Web application**
   - Application Server: **Tomcat 10.1**
   - Build system: **Gradle**
   - Group: **hkmu.comps380f**
   - Jakarta EE Version: **Jakarta EE 10**

2. Update the Gradle dependencies as shown below. Note that the artifact `jakarta.servlet.jsp-api` allows IntelliJ to recognize Java code in JSP pages. Reload the project in the Gradle window (on the right-side menu).

```
dependencies {
    compileOnly('jakarta.servlet:jakarta.servlet-api:6.0.0')
    implementation 'jakarta.servlet.jsp:jakarta.servlet.jsp-api:3.1.1'
    implementation 'jakarta.servlet.jsp.jstl:jakarta.servlet.jsp.jstl-api:3.0.0'
    implementation 'org.glassfish.web:jakarta.servlet.jsp.jstl:3.0.1'
}
```

3. Create the JSP fragment /WEB-INF/jsp/**base.jspf** that declares the JSTL core tag library with XMLNS prefix "c":

```
<%@ taglib prefix="c" uri="jakarta.tags.core" %>
```

4. Add the following <jsp-config> and <session-config> tags to the DD (/WEB-INF/web.xml):

```
<jsp-config>
    <jsp-property-group>
        <url-pattern>*.jsp</url-pattern>
        <url-pattern>*.jspf</url-pattern>
        <page-encoding>UTF-8</page-encoding>
        <include-prelude>/WEB-INF/jsp/base.jspf</include-prelude>
        <trim-directive-whitespaces>true</trim-directive-whitespaces>
        <default-content-type>text/html</default-content-type>
    </jsp-property-group>
</jsp-config>

<session-config>
    <session-timeout>30</session-timeout>
    <cookie-config>
        <http-only>true</http-only>
    </cookie-config>
    <tracking-mode>COOKIE</tracking-mode>
</session-config>
```

The <session-config> settings cause sessions to last for 30 minutes, instruct the web container to only use cookies for session tracking, and make session cookies contain the HttpOnly attribute for security concerns.

5. Create the following servlet `StoreServlet` that has a Map variable `products` keeping a list of products:

```java
@WebServlet(name = "storeServlet", value = "/shop")
public class StoreServlet extends HttpServlet {
    private final Map<Integer, String> products = new ConcurrentHashMap<>();

    public StoreServlet() {
        this.products.put(1, "Sandpaper");
        this.products.put(2, "Nails");
        this.products.put(3, "Glue");
        this.products.put(4, "Paint");
        this.products.put(5, "Tape");
    }

    @Override
    protected void doGet(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {
        String action = request.getParameter("action");
        if (action == null)
            action = "browse";

        switch (action) {
            case "addToCart":
                this.addToCart(request, response);
                break;
            case "viewCart":
                this.viewCart(request, response);
                break;
            case "browse":
            default:
                this.browse(request, response);
                break;
        }
    }

    // Defining other methods ...
}
```

To ensure thread safety, we will use the synchronized `java.util.concurrent.ConcurrentHashMap`, instead of the unsynchronized `java.util.HashMap`. Note that in the previous lab, we used the synchronized list `java.util.concurrent.CopyOnWriteArrayList`.

6. The method `viewCart()` adds the products to a ***request attribute*** and shows the shopping cart:

```java
private void viewCart(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException {
    request.setAttribute("products", this.products);
    request.getRequestDispatcher("/WEB-INF/jsp/viewCart.jsp")
            .forward(request, response);
}
```

7. The method `browse()` also adds the products to a ***request attribute***, but it shows the list of available products for the user to browse through.

```java
private void browse(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException {
    request.setAttribute("products", this.products);
    request.getRequestDispatcher("/WEB-INF/jsp/browse.jsp")
            .forward(request, response);
}
```

8. The method `addToCart()` will add an item whose product ID equals the query parameter `productId` to the shopping cart. The shopping cart is a **`ConcurrentHashMap<Integer, Integer>`** object, stored as a ***session attribute***. The key is the **product ID** and the value is the **quantity** of that product in the cart.

```
private void addToCart(HttpServletRequest request, HttpServletResponse response)
        throws IOException {
    int productId;
    try {
        productId = Integer.parseInt(request.getParameter("productId"));
    } catch (Exception e) {
        response.sendRedirect("shop");
        return;
    }

    HttpSession session = request.getSession();
    if (session.getAttribute("cart") == null)
        session.setAttribute("cart", new ConcurrentHashMap<>());

    @SuppressWarnings("unchecked")
    Map<Integer, Integer> cart
            = (Map<Integer, Integer>) session.getAttribute("cart");
    if (!cart.containsKey(productId))
        cart.put(productId, 0);
    cart.put(productId, cart.get(productId) + 1);

    response.sendRedirect("shop?action=viewCart");
}
```

Question: Is it correct to change the URL in the last line to `"/shop?action=viewCart"`?

9. The JSP page **browse.jsp** is shown, as follows. Where should it be placed?

```
<%@ page import="java.util.Map" %>
<!DOCTYPE html>
<html>
<head>
    <title>Product List</title>
</head>
<body>
<h1>Product List</h1>
<a href="<c:url value="/shop?action=viewCart" />">View Cart</a><br/><br/>
<%
    @SuppressWarnings("unchecked")
    Map<Integer, String> products =
            (Map<Integer, String>) request.getAttribute("products");

    for (int id : products.keySet()) {
%><a href="<c:url value="/shop">
                <c:param name="action" value="addToCart" />
                <c:param name="productId" value="<%= Integer.toString(id) %>" />
            </c:url>"><%= products.get(id) %></a><br/>
<% } %>
</body>
</html>
```

The JSTL core tag **`<c:url>`** properly encodes URLs. If the URL is a relative URL, the tag prepends the URL with the ***base URL of your application*** so that the browser receives the correct URL. This tag can be used together with the tag **`<c:param>`** which encodes the query parameter string and appends it to the encoded URL. We will check the resulted HTML code later.

10. The JSP page **viewCart.jsp** is shown, as follows:

```
<%@ page import="java.util.Map" %>
<!DOCTYPE html>
<html>
<head>
    <title>View Cart</title>
</head>
<body>
<h1>View Cart</h1>
<a href="<c:url value="/shop" />">Product List</a><br/><br/>
<%
    @SuppressWarnings("unchecked")
    Map<Integer, String> products =
            (Map<Integer, String>) request.getAttribute("products");

    @SuppressWarnings("unchecked")
    Map<Integer, Integer> cart =
            (Map<Integer, Integer>) session.getAttribute("cart");

    if (cart == null || cart.size() == 0) { %>
Your cart is empty
<% } else { %>
<ul>
    <% for (int id : cart.keySet()) { %>
    <li><%=products.get(id)%> (qty: <%=cart.get(id) %>)</li>
    <% } %>
</ul>
<% } %>
</body>
</html>
```

11. Build and deploy the web app. Access storeServlet and observe the HTML output of browse.jsp.

Task 2: Adding the function of emptying the shopping cart

In our web application, we need to close and re-open the browser to empty the shopping cart. We will add the function of emptying the shopping cart directly.

1. In the servlet `storeServlet`, add a new case to the `doGet()` method:

```
case "emptyCart":
    this.emptyCart(request, response);
    break;
```

2. Add the method `emptyCart()`:

```
private void emptyCart(HttpServletRequest request, HttpServletResponse response)
        throws IOException {
    request.getSession().removeAttribute("cart");
    response.sendRedirect("shop?action=viewCart");
}
```

3. Add the following link to the beginning of the <body> of **viewCart.jsp:**

```
<a href="<c:url value="/shop?action=emptyCart" />">Empty Cart</a>
```

4. Test the web app again.