Advanced Java

Agenda

- Q & A
- JSP Standard actions
- Java beans
- JSP EL
- JSTL

JSP

JSP Standard Actions

- JSP Standard actions are predefined JSP tags for certain functionality. They can be used to reduce scriptlets in JSP code.
- `<jsp:forward page="subjects.jsp" />`

```
<%
    RequestDispatcher rd = request.getRequestDispatcher("subjects.jsp");
    rd.forward(request, response);
%>
```

- `<jsp:include page="page2.jsp" />`

```
<%
    RequestDispatcher rd = request.getRequestDispatcher("page2.jsp");
    rd.include(request, response);
%>
```

    - Dynamic/runtime inclusion i.e. page1.jsp <===> page2.jsp
    - <%@include file="page.jsp"%> is static inclusion i.e. contents of page2.jsp are included in page1.jsp during translation stage.
- <jsp:param name=... value=... />
    - Can be used as optional param as child tag of forward or include.

```
<%-- page1.jsp --%>
<jsp:forward page="page2.jsp">
    <jsp:param name="key" value="someValue"/>
</jsp:forward>
```

```
<%-- page2.jsp --%>
<%
    String value = request.getParameter("key");
%>
```

- <jsp:plugin type="applet" ... />
    - Applets are java classes that gets loaded into client browser and executed there in browser's JRE (plugin). Due to severe security concerns they are deprecated.
- <jsp:fallback .../>
    - fallback is child tag for plugin tag to show alternate message if plugin loading is failed.

```
<jsp:plugin type="applet" class="com.sunbeam.MyApplet" ...>
    <jsp:fallback>Applet Not Loaded.</jsp:fallback>
</jsp:plugin>
```

- <jsp:element name = "xmlElement">
- <jsp:attribute name = "xmlEleAttr">
- <jsp:body>...</jsp:body>

- `<jsp:text>...</jsp:text>`
  - Above four are XML generation tags.
- `<jsp:useBean ... />`
- `<jsp:setProperty ... />`
- `<jsp:getProperty ... />`

Java Beans

- Java beans are simple java classes which contain parameterless constructor, fields, getters/setters and one/more business logic methods.
- Ideal JSPs do not contain scriptlets. So Java beans are used to encapsulate all business logic required for the JSP processing.
- Java beans used in JSP pages using
  - `<jsp:useBean id="var" class="pkg.BeanClass" scope="..."/>`
  - `<jsp:setProperty name="var" property="... " value="..."/>`
  - `<jsp:setProperty name="var" property="... " param="..."/>`
  - `<jsp:setProperty name="var" property="*"/>`
  - `<jsp:getProperty name="var" property="... "/>`
- Java beans objects are created & accessed using reflection. So naming conventions must be strictly followed.

Java bean scopes

- page – PageContext attribute (default) -- lowest scope
  - Internally, bean object is stored in the current page context using pageContext.setAttribute("beanName", beanObject) and accessed using pageContext.getAttribute("beanName").
  - Bean is available for the current page current request only.
- request – Request attribute
  - Internally, bean object is stored in the current request using request.setAttribute("beanName", beanObject) and accessed using request.getAttribute("beanName").
  - If same request is forwarded or included (using RequestDispatcher), then the bean will be accessible in next page as well.
- session – HttpSession attribute
  - Internally, bean object is stored in the current user HttpSession using session.setAttribute("beanName", beanObject) and accessed using session.getAttribute("beanName").
  - The bean is accessible in all requests to all pages by the same client.

- application – ServletContext attribute -- highest scope
  - Internally, bean object is stored in the current application ServletContext using ctx.setAttribute("beanName", beanObject) and accessed using ctx.getAttribute("beanName").
  - The bean is accessible in all requests to all pages by all clients.

jsp:useBean

- Check if object with given name is present in given scope (using getAttribute()). If available, access it.
- If not available, create new bean object.
- Add the object into given scope (using setAttribute()).

```
// Internals of jsp:useBean
beanObj = scope.getAttribute("beanName");
if(beanObj == null) {
    beanObj = new BeanClass();
    scope.setAttribute("beanName", beanObj);
}
```

jsp:setProperty and jsp:getProperty

- These tags internally calls setter and getter methods on the bean object.
- jsp:setProperty, jsp:getProperty must be preceded by jsp:useBean.

JSP EL

- To reduce the scriptlets and expressions.
- Syntax: ${...}
- Used to
  - Access scoped objects
    - ${scopeName.objName}
    - pageScope, requestScope, sessionScope, applicationScope

        - e.g. ${sessionScope.lb.role}, ${pageScope.sb.bookList}
    - Auto search from lowest to highest scope
        - e.g. ${lb.role} --> lb will be searched first in pageScope, then requestScope, then sessionScope.
        - Using scopeName is useful if different beans with same name are present in multiple scopes.
- Access fields (via getters)
    - ${objName.fieldName} --> internally calls objName.getFieldName()
- Access methods
    - ${objName.method()}
- Perform arbitrary calculations
    - ${2 + 3 * 4 mod 5}
        - mod is "%" operator.
        - div is "/" operator.
- Work with EL implicit objects

EL implicit objects

- ${param.name} --> request.getParameter("name")
- ${paramValues.name} --> request.getParameterValues("name")
- ${header.name} --> request.getHeader("name")
- ${headerValues.name} --> request.getHeaderValues("name")
- ${initParam.name} --> ctx.getInitParameter("name") -- `<context-param>` in web.xml.
- ${cookie.name} --> returns value of the cookie with given "name".
- ${pageContext....} --> to access objects in the pageContext like request, ...

JSP Page Context

- PageContext object is created to process the JSP. It holds all required objects (for JSP processing) like request, response, out, session, application, config, etc.
- It is also useful for state management -- Page level.

```
pageContext.setAttribute("key", value);
value = pageContext.getAttribute("key");
```

JSTL

- JSTL = JSP Standard Tag Library
- Maven dependency

```
<!-- https://mvnrepository.com/artifact/jstl/jstl -->
<dependency>
    <groupId>jstl</groupId>
    <artifactId>jstl</artifactId>
    <version>1.2</version>
</dependency>
```

- Five components
  - Core -- Programming constructs (forEach, choose, if), redirect, Variables (set), ...
  - Formatting -- Formatting date and currency.
  - SQL -- execute SQL queries directly from JSP pages (not recommended).
  - XML -- XML processing/generation.
  - Functions -- Utility functions like String manipulation e.g. length, concat, ...
- `<@taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>`
  - <c:if .../>
  - <c:choose .../>
  - <c:forEach .../>
  - <c:set .../>
  - <c:url .../>
  - <c:redirect .../>
- `<%@ taglib prefix="fmt" uri="http://java.sun.com/jsp/jstl/fmt" %>`
  - `<fmt:formatDate pattern="dd-MM-yyyy" value="${cust.birth}" />`

Filters

- Not mentioned in DMC Syllabus
- Filters is way of implementing AOP in Java EE applications. Filters are used to perform pre-processing, post-processing or both for each request.
- Multiple filters can be executed in a chain/stack before/after handling request.
- javax.servlet.Filter interface is used to implement Filters.
    - void init(FilterConfig filterConfig);
    - void doFilter(ServletRequest req, ServletResponse resp, FilterChain chain);
    - void destroy();
- https://docs.oracle.com/javaee/7/api/javax/servlet/Filter.html
- Can be configured with @WebFilter or in web.xml (similar to servlets).

Listeners

- Not mentioned in DMC Syllabus
- Listeners are used to handle application level events.
- There are many listener interfaces. Refer docs.
    - ServletContextListener -- To handle application initialized and destroy events.
        - void contextInitialized(ServletContextEvent sce);
            - Called by web container when application is started/deployed i.e. servlet context is created.
            - Example: load and register JDBC driver, initialize a connection pool, ...
        - void contextDestroyed(ServletContextEvent sce);
            - Called by web container when application is stopped i.e. when web server shutdown.
            - Example: release a connection pool, ...
        - Implement this listener to perform one time initialization and destruction for the whole application.
    - HttpSessionListener -- To handle session initialized and destroy events.
        - sessionCreated() method is called when req.getSession() is called first time for any client. You may add any session attribute in it immediately after creating session.
        - sessionDestroyed() method is called when session is invalidated or time-out.
    - ServletRequestListener -- -- To handle request initialized and destroy events.
    - ServletContextAttributeListener
    - HttpSessionActivationListener
    - HttpSessionAttributeListener
    - ServletRequestAttributeListener

- Listener class must implement one or more listener interface.
- Can be configured with @WebListener OR in web.xml.

```
<listener>
    <listener-class>pkg.MyListener</listener-class>
</listener>
```

Assignments

1. Implement Bookshop using JSPs and Java beans. Make proper use of JSP EL and JSTL tags.