Advanced Java

- **JSP**
  - Introduction
  - Syntax
  - Implicit objects
  - Life cycle
  - Directives
  - Error pages
  - Java Beans
  - JSP Standard actions

JSP

- Servlet = Business logic* + Presentation logic
- JSP = Presentation logic* + Business logic
- **JSP is converted into the servlet while execution.**
- JSP is outdated.

JSP syntax

- Directive <%@ ... %>
  - Instructs JSP engine to process the jsp.
  - @page -- servlet creation/translation.
  - @include -- include a jsp/html into another jsp.
  - @taglib -- to use custom/third party tags in jsp.
- Declaration <%! ... %>
  - To declare fields and methods in generated servlet (other than service()).
- Scriptlet <% ... %>
  - For Java statements to be executed for each request (in jspService()).
- Expression <%= ... %>

    - For Java expressions whose output is to be embedded in produced response. Executes for each request (in jspService()).
- Comment <%-- ... --%>
    - Server side comment -- discarded while processing.

Example Servlet --> JSP

- Generated servlet

```
import java.util.Date;
class HelloServlet ... {
    private int count = 0;
    public void init(ServletConfig conf) ... {
        super.init(conf);
        System.out.println("init() called...");
    }
    public void destroy() {
        System.out.println("destroy() called...");
    }
    // HelloServlet.service()
    public void doGet(HttpServletRequest request, HttpServletResponse response) ... {
        processRequest(request, response);
    }
    public void doPost(HttpServletRequest request, HttpServletResponse response) ... {
        processRequest(request, response);
    }
    public void processRequest(HttpServletRequest request, HttpServletResponse response) ... {
        response.setContentType("text/html");
        PrintWriter out = response.getWriter();
        out.println("<html>");
        out.println("<head>");
        out.println("<title>Hello Servlet</title>");
        out.println("</head>");
        out.println("<body>");
        out.println("<h3>Congratulations, Sunbeam!</h3>");
        count++;
        if(count % 2 == 0) {
            out.println("Even Count: " + count);
        } else {
            out.println("Odd Count: " + count);
        }
        Date d = new Date();
        out.println("<br/><br/>Current Time: " + d.toString());
        out.println("</body>");
        out.println("</html>");
    }
}
```

- JSP

```
<%@ page language="java" %> <%@ page contentType="text/html"
import="java.util.Date" %> <%-- This is Hello JSP (Server side comment) --%>
<!-- This is Hello JSP (Client side/HTML comment) -->
<html>
  <head>
    <title>Hello JSP</title>
  </head>
  <body>
    <%! private int count = 0; %> <%! public void jspInit() {
    System.out.println("jspInit() called"); } public void jspDestroy() {
    System.out.println("jspDestroy() called."); } %>
    <h3>Congratulations, Sunbeam!</h3>
    <% count++; %> <% if(count % 2 == 0) { %> "Even Count: " <%= count %> <% }
    else { %> "Odd Count: " <%= count %> <% } %> <% Date d = new Date(); %>
    <br /><br />Current Time: <%= d.toString() %>
  </body>
</html>
```

JSP Syntax

- `<%@ ... %>` -- Directive
- `<%! ... %>` -- Declaration
- `<% ... %>` -- Scriptlet
- `<%= ... %>` -- Expression
- `<%-- ... --%>` -- Comment

JSP Implicit objects

- These objects are available for use in _jspService() i.e. scriptlets and expressions. We need not to declare them explicitly.
- Because these objects are local variables or arguments of generated _jspService() method.
- request: HttpServletRequest
- response: HttpServletResponse
- session: HttpSession
- out: JspWriter -- similar PrintWriter
- application: ServletContext
- config: ServletConfig
- pageContext: PageContext -- to store page attributes.
- page: Object -- represent current page/servlet instance (this).
- exception: Throwable -- available only in error pages.

JSP Life cycle

- JSP Engine
  - 1- Translation stage: Converts JSP into servlet java class. Check JSP syntax errors.
  - 2- Compilation stage: Compiles generated servlet java class into java byte code. Check java code errors (scriptlet, expression and declaration blocks).
- Servlet Engine
  - 3- Loading & Instantiation stage: Loads servlet class into JVM & create its object. Invokes jspInit().
  - 4- Request handling stage: Handles request & produce response. Invokes jspService(). For each request.
  - 5- Destruction stage: De-initialize the object. Invokes jspDestroy().
- For first request all stages 1 to 4 are executed.
- For subsequent requests only stage 4 is executed.

JSP Directives

- `<%@ page ... %>`
- `<%@ include ... %>`
- `<%@ taglib ... %>`

JSP Directive: @include

- Includes a file into the JSP page.
- Inclusion is static i.e. during translation stage.
- header.jsp

```
<h2>Sunbeam Online Bookshop</h2>
```

- footer.jsp

```
<h5>Copyright (c) 2023, Sunbeam Infotech</h5>
```

- index.jsp

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="UTF-8" />
    <title>Index</title>
  </head>
  <body>
    <table border="1" style="width:100%">
      <tr style="align:center">
        <td><%@include file="header.jsp"%></td>
```

```
          </tr>

          <tr></tr>
          <tr style="height: 800px">
            <td>
              <form method="post" action="login">
                Email: <input type="text" name="email" /> <br /><br />
                Password: <input type="password" name="passwd" /> <br /><br />
                <input type="submit" value="Sign In" />
                <a href="register.html">Sign Up</a>
              </form>
            </td>
          </tr>

          <tr></tr>
          <tr style="align:center">
            <td><%@include file="footer.jsp"%></td>
          </tr>

          <tr></tr>
        </table>
      </body>
    </html>
```

JSP Directive: @taglib

- JSP default tags (called as standard actions) starts with jsp prefix.
    - e.g. `<jsp:forward page="page2.jsp"/>`
- Used for third-party (e.g. JSTL) or custom tags in JSP page.

```
<%taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %> ...
<c:redirect url="https://sunbeaminfo.in" />
...
```

JSP Directive: @page

- `<%@page language="java"%>`
  - Server side processing lanaguage is java. Only java lanaguage is supported.
- `<%@page import="java.util.Date"%>`
  - Imports given package in generated servlet .java file.
- `<%@page contentType="text/html" %>`
  - response.setContentType("text/html");
- `<%@page session="true"%>`
  - Internally calls `session = request.getSession();`.
  - If session="false", then `session = null;`.
- `<%@page isErrorPage="false"%>`
  - This page is used only for displaying errors like 403, 404, 500 with custom error messages. isErrorPage="true". Will have access to "exception" object.
- `<%@ page errorPage="error.jsp" %>`
  - Errors produced in this page are to be displayed in error.jsp. Here error.jsp is a error page (user defined).
- `<%@page info="This is hello JSP"%>`
  - Keeps information/metadata about JSP page.
- `<%@page buffer = "8"%>`
  - JSP response is stored in a buffer. Default buffer size is 8 kb.
- `<%@page autoFlush = "false"%>`
  - false: Whenever buffer is full or response generated, then it is flushed to the client.
  - true: Whenever output is written into buffer, immediately send to the client.
- `<%@page extends = "javax.servlet.http.HttpServlet"%>`
  - Defines base class generated servlet class.
- `<%@page isELIgnored = "false"%>`
  - true: Do not process EL (expression language) syntax ${...} in JSP page.

Java Beans

- Java beans are simple java classes which contain constructor, fields, getters/setters and one/more business logic methods.
- Ideal JSPs do not contain scriptlets. So Java beans are used to encasulate 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 on 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.

Java Beans - Syntax Revision

jsp:useBean standard action

- Syntax

```
<jsp:useBean id="beanName" class="pkg.BeanClass" scope="page|request|session|application"/>
```

jsp:setProperty standard action

- To initialize bean fields.
- Syntax

```
<jsp:setProperty name="beanName" property="fieldName" param="reqParamName"/>
```

- Syntax

```
<jsp:setProperty name="beanName" property="*"/>
```

- Syntax

```
<jsp:setProperty name="beanName" property="fieldName" value="fixedValue"/>
```

```
<jsp:useBean id="lb" class="pkg.LoginBean"/>
<jsp:setProperty name="lb" property="email" value="nilesh@sunbeaminfo.com"/>
```

```
<jsp:useBean id="bb" class="pkg.BookBean"/>
<jsp:setProperty name="bb" property="name" value="The Alchemist"/>
<jsp:setProperty name="bb" property="subject" value="Novel"/>
<jsp:setProperty name="bb" property="price" value="482.32"/>
```

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. In request handling stage.
  - `<%@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 height="500" width="500" type="applet" code="MyApplet.class" name="my" codebase="."/>
    <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 ... />`