Advanced Java

Agenda

- Q & A
- QueryString
- Request
- ServletContext
- JSP concept
- JSP Syntax
- JSP Life Cycle
- JSP Implicit objects
- JSP Standard actions
- Java beans

Q & A

- What is difference between servlet init parameters and request parameters?
  - Servlet Init Parameters
    - Servlet config/settings are associated with servlet in web.xml or using @WebServlet.
    - Can be read using `config.getInitParameter("param-name")`.
    - e.g. Db url, Db username, Theme, ...
  - Request Parameters
    - Data received from the client along with the request (in URL or in body).
    - Can be read in request handling stage of servlet using `req.getParameter("param-name")` or `req.getParameterValues("param-name")`.
    - e.g. user credentials, product details, ...

State Management

QueryString

- Data can be added into URL after '?' in key=value pairs to send along with the request to that URL.
- If there are multiple key-value pairs, they should be separated by &.
- Examples

```
<a href='url?key1=value1&key2=value2'>Link</a>
```

```
out.printf("<a href='url?key1=%s&key2=%s'>Link</a>", value1, value2);
```

```
out.printf("<form action='url?key1=%s&key2=%s'>", value1, value2);
```

```
String url = String.format("url?key1=%s&key2=%s", value1, value2);
resp.sendRedirect(url);
```

- In next servlet (of given url) this data can be accessed using req.getParameter().

```
String value1 = req.getParameter("key1");
String value2 = req.getParameter("key2");
```

Request

- The request object can hold a set of key-value (String-Object) pairs called as request attributes.
- When same request is forwarded to the next web component (using RequestDispatcher forward() or include()), we can use the request attribute to transfer some data/information.
- To send request attribute

```
req.setAttribute("key", value);
```

- To retrieve request attribute in next web component

```
value = req.getAttribute("key");
```

- Once response is sent to the client, the request object (along with its attributes/parameters) and response object are destroyed.

Request parameter vs Request attribute

- Request parameter represents the data coming from the client along with http request (from HTML form controls or query string). It is accessed using req.getParameter() or req.getParameterValues(). Request param are always String.
- Request attributes are added by one web component and forwarded to the next web component. This server side state management is done using req.setAttribute() and req.getAttribute(). Request attribute can be of any type (Object).

ServletContext

- Web server creates a ServletContext object for each web application. It represents the whole "application".
- We can access current application's servlet context by several ways

```
ctx = req.getServletContext();
// OR
ctx = session.getServletContext();
// OR
ctx = config.getServletContext(); // ServletConfig
// OR
ctx = this.getServletContext(); // current servlet
```

- It keeps application metadata/information.

- It can also store state in form of ServletContext attributes (String-Object key-value).

```
ctx.setAttribute("key", value);
```

```
value = ctx.getAttribute("key");
```

- Servlet context can also be used to access context parameters of the application (from web.xml).

```
<context-param>
    <param-name>app.title</param-name>
    <param-value>Online Book Store</param-value>
</context-param>
<context-param>
    <param-name>color</param-name>
    <param-value>pink</param-value>
</context-param>
```

```
String paramValue = ctx.getInitParameter("app.title");
out.println("<h3>" + paramValue + "</h3>");
```

```
String color = ctx.getInitParameter("color");
out.printf("<body bgcolor='%s'>\r\n", color);
```

-

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

```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 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.
- When server stops or application undeplyed, stage 5 is executed.

JSP @Page Directive

- `<%@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 = req.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.
- `<%@ page errorPage="error.jsp" %>`
  - Errors produced in this page are to be displayed in error.jsp. Here error.jsp is a error page.
- `<%@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"%>`
  - Whenever buffer is full, it is flushed to the client.

- `<%@page extends = "javax.servlet.http.HttpServlet"%>`
  - Defines base class generated servlet class.
- `<%@page isELIgnored = "false"%>`
  - Do not process EL (expression language) syntax ${...} in JSP page.

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.

Assignments

1. Complete Bookshop application using Servlets. Add functionalities like user Sign up, Add Book, etc.