Java Server Page(JSP)

- JSP stands for Java Server Pages.
- It is a server-side technology that is used for creating web applications.
- It is used to create dynamic web content.
- JSP consists of both HTML tags and JSP tags.
- JSP tags are used to insert JAVA code into HTML pages.
- It is an advanced version of Servlet Technology i.e. a web-based technology that helps us to create dynamic and platform-independent web pages.
- Java code can be inserted in HTML/ XML pages or both.
- JSP is first converted into a servlet by the JSP container before processing the client's request.
- JSP has various features like JSP Expressions, JSP tags, JSP Expression Language, etc.

Advantage of JSP over Servlet

- They are easy to maintain.
- No recompilation or redeployment is required.
- Less coding is required in JSP.
- JSP has access to the entire API of JAVA.
- JSP are extended version of Servlet.

Features of JSP

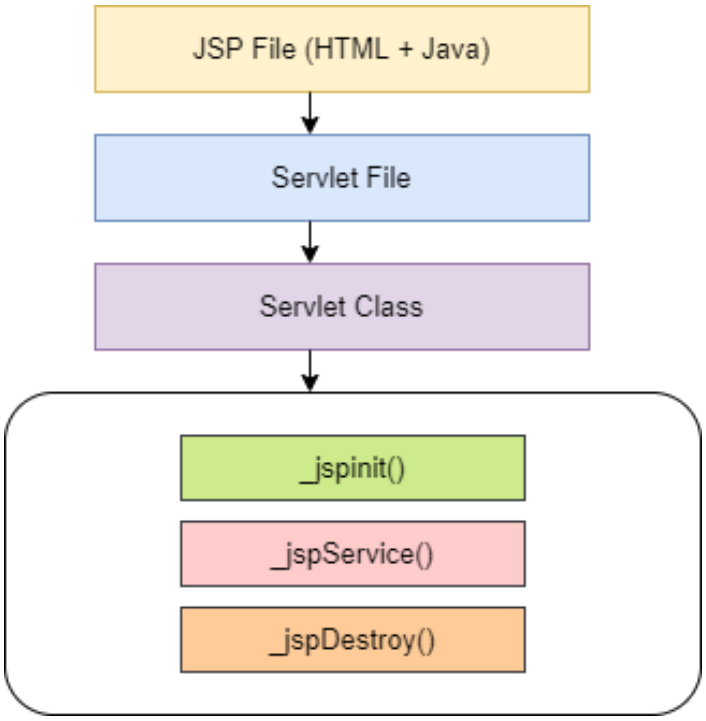
- Coding in JSP is easy: As it is just adding JAVA code to HTML/XML.
- Reduction in the length of Code: In JSP we use action tags, custom tags etc.
- Connection to Database is easier: It is easier to connect the website to a database and allows to read or write data easily to the database.
- Make Interactive Websites: In this, we can create dynamic web pages that helps user to interact in real time environment.
- Portable, Powerful, flexible and easy to maintain : as these are browser and server independent.
- No Redeployment and No Re-Compilation : It is dynamic, secure and platform independent so no need to re-compilation.
- It does not require advanced knowledge of JAVA
- It is capable of handling exceptions
- Easy to use and learn
- It contains tags which are easy to use and understand
- Implicit objects are there which reduces the length of code
- It is suitable for both JAVA and non JAVA programmer
- Difficult to debug for errors.
- First time access leads to wastage of time
- It's output is HTML which lacks features.

Lifecycle of JSP

Steps Involved in a JSP lifecycle:

1. Translation

2. Compilation
3. ClassLoading
4. Instantiation
5. Initialization
6. Request Processing
7. Destroy



Translation
This phase deals with the syntactic correctness of the page. JSP code is checked by the JSP container and is parsed by it to generate the servlet source code. The web container automatically translates the index.jsp file into index.java.

Compilation
Here, the JSP container compiles the JSP class source code. It converts the index.java file into an index.class file. The translation of the Java source to its implementation class can occur at any given time between the deployment of the JSP page into the container and processing of the JSP page.

ClassLoading
In this phase, the servlet class loaded from the JSP source is loaded into the container.

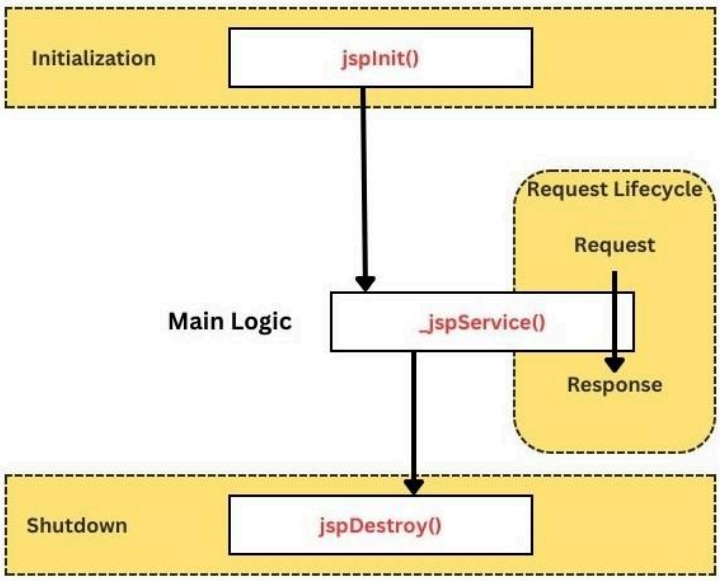
Instantiation
In this phase, an instance of the class is generated. The container manages one or more instances by responding to the requests.

Initialization
In this phase, the container invokes the init method of the JSP class. After that, servlet configuration with the init parameter is configured. Now, JSP can handle the client requests. Most often, the translation and initialization in JSP happen as soon as the first request for a JSP comes.

Request Processing
This is the most time-consuming phase of the JSP. Here, the JSP page processes the requests from a client. The request processing phase is multi-threaded. A new thread is used for every request, and a servlet request and response object are created. After this, the _jspService() method is called. This method cannot be overridden.

Destroy
The last phase of the JSP lifecycle is destroyed. The JSP class is unloaded or removed from the container in this phase. It is done when the application is undeployed or when the server is down. This is done using the jspDestroy() method and can be overridden.

JSP Lifecycle Methods



jspInit()
This is declared on the JSP page. This method is called only once in the complete lifecycle of JSP. This is used to initialize configuration params in a deployment descriptor. This method can be overridden using a JSP declaration scripting element.

_jspService()
It is invoked by the JSP container for each client request. This method passes the request and response objects. It cannot be overridden, and hence, it starts with an underscore. It is defined and declared in the HttpJspPage interface.

jspDestroy()

This is used for shutting down the application/container. This method is called only once in the complete JSP lifecycle when JSP is unloaded from the memory. This method should be overridden only to release resources created in the JSP init method.

Servlet vs JSP

Servlet	**JSP**
Servlet is a java code.	JSP is a HTML-based compilation code.
Writing code for servlet is harder than JSP as it is HTML in java.	JSP is easy to code as it is java in HTML.
Servlet plays a controller role in the ,MVC approach.	JSP is the view in the MVC approach for showing output
Servlet is faster than JSP.	JSP is slower than Servlet because the first step in the JSP lifecycle is the translation of JSP to java code and then compile.
Servlet can accept all protocol requests.	JSP only accepts HTTP requests.
In Servlet, we can override the service() method.	In JSP, we cannot override its service() method.
In Servlet by default session management is not enabled, the user have to enable it explicitly.	In JSP session management is automatically enabled.
It does not have inbuilt implicit objects.	In JSP there are inbuilt implicit objects.
Packages are to be imported on the top of the program.	Packages can be imported anywhere into the JSP program (i.e, bottom , middle client-side, or top)
It can handle extensive data processing.	It cannot handle extensive data processing very efficiently.

Sample Program: Create a web application with a form to enter name and faculty and a submit button. When the button is clicked send the data to the server and show the values entered in main.jsp page.

index.html

```
<html>
  <body>

    <form action = "main.jsp" method = "GET">
      First Name: <input type = "text" name = "first_name">
      <br />
      Last Name: <input type = "text" name = "last_name" />
      <input type = "submit" value = "Submit" />
    </form>

  </body>
</html>
```

main.jsp

```
<html>
  <head>
    <title>Using GET Method to Read Form Data</title>
  </head>

  <body>
    <h1>Using GET Method to Read Form Data</h1>
    <ul>
      <li><p><b>First Name:</b>
        <%= request.getParameter("first_name")%>
      </p></li>
      <li><p><b>Last  Name:</b>
        <%= request.getParameter("last_name")%>
      </p></li>
    </ul>

  </body>
</html>
```

Create a web application that allows users to enter their name. Pass the name to the server-side jsp page and store the name in the session. Using the value stored in the session object display the name in the second jsp page

index.html

```
<html>
<body>
<form action="welcome.jsp">
<input type="text" name="uname">
<input type="submit" value="go"><br/>
</form>
</body>
</html>
```

welcome.jsp [** *This jsp will get the value and store it in the session*]

```
<html>
<body>
<%

String name=request.getParameter("uname");   // GET THE VALUE ENTERED BY THE USER
out.print("Welcome "+name);  // PRINT THE NAME IN THE BROWSER

session.setAttribute("user",name);  // SAVE THE NAME IN A SESSION OBJECT

<a href="second.jsp">second jsp page</a>  //CREATE A LINK TO VISIT THE SECOND PAGE

%>
</body>
</html>
```

second.jsp

```
<html>
<body>
<%

String name=(String)session.getAttribute("user");  // GET THE NAME FROM THE SESSION
out.print("Hello "+name);   // DISPLAY THE NAME IN THE BROWSER

%>
</body>
</html>
```