JSP

JSP

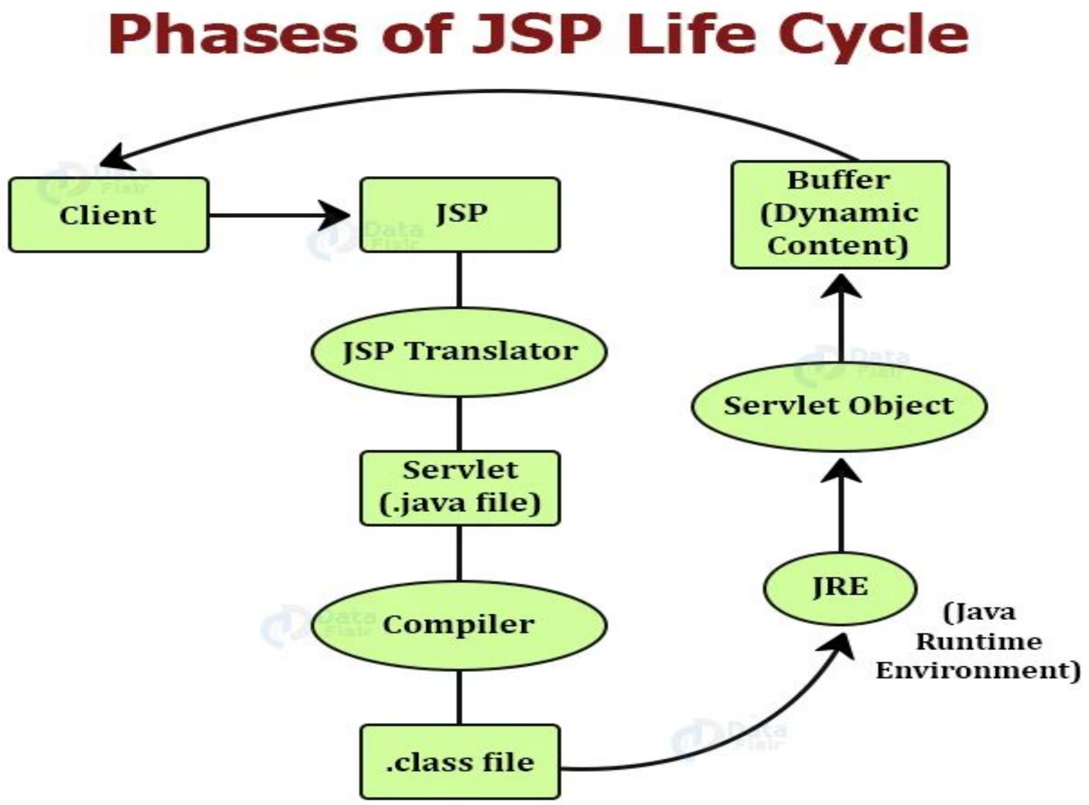
- **JSP** stands for **Java Server Pages**.
- It is a server-side technology which is used for creating web applications.
- JSP provides excellent server scripting support for creating database driven web applications.
- JSP enables the developer to directly insert Java code into jsp file, this makes the development process very simple and its maintenance also becomes very easy.
- Java is known for its characteristic of ‘ write once run anywhere’.
- i.e. JSP’s are platform independent.

Advantages of JSP

- JSP is a server-side technology, whereas Servlet is a small Java class program that runs on the server.
- JSP is easy to manage because in JSP separating business logic from presentation logic is easy, while in Servlet, business logic and presentation logic are together.
- If you want to modify the JSP page, you do not need to recompile and redeploy the project. If you are going to change the UI of the application in Servlet, you need to update the code.
- Coding of JSP is easier than Servlet because JSP is a tag-based approach, whereas Servlet is a Java code.
- Servlet can accept any type of request, while JSP accepts only HTTP protocol requests.
- JSP allows custom tags, but in Servlet, you can not build any custom tags.

- In JSP, you can import packages anywhere in the file, whereas in Servlet, you need to import every package on top of the Servlet.
- Session management is enabled by default in JSP, but in Servlet, users need to enable it separately.
- In (MVC) Model View Controller architecture, JSP acts as a viewer while Servlet acts as a controller.
- JSP is the preferred choice when there's less processing requirement. Servlet is the first choice when the processing and manipulation requirement is high.

The Life Cycle Of JSP



Phases of JSP Life Cycle:

1. **Translation of JSP to Servlet:** Here, in the first step, the .jsp file translates to _jsp.java.
2. **Compilation of JSP page:** The compilation of the translated Java Servlet file (_jsp.java) occurs in a Servlet class file (_jsp.class).
3. **Classloading:** Now, the compiled Servlet class loads into the container using classloader.
4. **Instantiation:** In this step, the web container generates an instance of that Servlet class.
5. **Initialisation:** The container invokes the _jspinit() method. _jspinit() method is invoked only once in a life cycle after Servlet instance generation.
6. **Request Processing:** Now, the container invokes _jspservice() method to process the request. You cannot override this method.
7. **Destroy:** _jspdestroy() method is used to destroy the Servlet instance from use. The container invokes the _jspDestroy() method to perform any required clean up. _jspdestroy() method is invoked only once. You can override this method.

Implicit objects

- Implicit objects are a set of Java objects that the JSP Container makes available to developers on each page.
- These objects may be accessed as built-in variables via scripting elements and can also be accessed programmatically by JavaBeans and Servlets.
- Following are implicit objects:

1. **request**: This is the object of HttpServletRequest class associated with the request.
2. **response**: This is the object of HttpServletResponse class associated with the response to the client.
3. **config**: This is the object of ServletConfig class associated with the page.
4. **application**: This is the object of ServletContext class associated with the application context.
5. **session**: This is the object of HttpSession class associated with the request.

5. **page context:** This is the object of PageContext class that encapsulates the use of server-specific features. This object can be used to find, get or remove an attribute.

6. **page object:** The manner we use the keyword this for current object, page object is used to refer to the current translated servlet class.

7. **exception:** The exception object represents all errors and exceptions which is accessed by the respective jsp. The exception implicit object is of type java.lang.Throwable.

8. **out:** This is the PrintWriter object where methods like print and println help for displaying the content to the client.

JSP Scripting elements

The scripting elements provides the ability to insert java code inside the jsp. There are three types of scripting elements:

1. scriptlet tag
2. expression tag
3. declaration tag

Scriptlet tag

- This tag allow user to insert java code in JSP.
- The statement which is written will be moved to jspservice() using JSP container while generating servlet from JSP.
- When client make a request, JSP service method is invoked and after that the content which is written inside the scriptlet tag executes.
- Syntax:

```
<%  java source code %>
```

Example:

```
<html>

<body>

<%

 out.print("Hello World");

%>

</body>

</html>
```

Expression tag

- The code placed within JSP expression tag is written to the output stream of the response
- There’s no need to write out.print() to write data.
- It is mainly used to print the values of variable or method.
- The JSP Expression tag transforms the code into an expression statement that converts into a value in the form of a string object and inserts into the implicit output object.
- Syntax:

```
<%=  statement %>
```

Example:

```
<html>

<body>

<%= "Hello World" %>

</body>

</html>
```

Declaration Tag

- Declaration tag is one of the scripting elements in JSP.
- This Tag is used for declare the variables.
- Declaration Tag can also declare method and classes.
- Jsp initializer scans the code and find the declaration tag and initializes all the variables, methods, and classes. JSP container keeps this code outside of the service method (_JSPService()) to make them class level variables and methods.
- Syntax:

  <%! Variable or method declaration %>

Example:

```
<html>

<body>

<%! int a=10, b=20; %>

<%= "Addition : "+(a+b)

 %>

</body>

</html>
```

JSP comments

JSP comments <%-- comments --%> are ignored by the JSP engine. For example,

<%-- anything but a closing tag here will be ignored -->

Note that HTML comment is <!-- comments -->.

JSP expression within the HTML comment will be evaluated.

For example,

<!-- HTML comments here <%= Math.random() %> more comments -->

JSP Directives

- JSP directives are the elements of a JSP source code that guide the web container on how to translate the JSP page into its respective servlet.
- **Syntax :**

  ```
  <%@ directive attribute = "value"%>
  ```

- Directives can have a number of attributes that you can list down as **key-value pairs** and separate by commas.
- The blanks between the @ symbol and the directive name, and between the last attribute and the closing %>, are optional.

There are three types of directive tag −

1. **<%@ page ... %>**
   Defines page-dependent attributes, such as scripting language, error page, and buffering requirements.

1. **<%@ include ... %>**
   Includes a file during the translation phase.

1. **<%@ taglib ... %>**
   Declares a tag library, containing custom actions, used in the page

Page Directives

JSP page directive is used to define the properties applying the JSP page, such as the size of the allocated buffer, imported packages, and classes/interfaces, defining what type of page it is, etc. The syntax of JSP page directive is as follows:

```
<%@page attribute = "value"%>
```

Different properties/attributes :

The following are the different properties that can be defined using page directive :

1. **import**: This tells the container what packages/classes are needed to be imported into the program.

   Syntax:

```
<%@page import = "value"%>
```

2. contentType: This defines the format of data that is being exchanged between the client and the server. It does the same thing as the setContentType method in servlet used to.

Syntax:

```
<%@page contentType="value"%>
```

3. info: Defines the string which can be printed using the ‘getServletInfo()’ method.

Syntax:

```
<%@page info="value"%>
```

4. buffer: Defines the size of the buffer that is allocated for handling the JSP page. The size is defined in Kilo Bytes.

Syntax:

```
<%@page buffer = "size in kb"%>
```

5. language: Defines the scripting language used in the page. By default, this attribute contains the value 'java'.

6. isELIgnored: This attribute tells if the page supports expression language. By default, it is set to false. If set to true, it will disable expression language.

Syntax:

```
<%@page isElIgnored = "true/false"%>
```

7. errorPage: Defines which page to redirect to, in case the current page encounters an exception.

Syntax:

```
<%@page errorPage = "true/false"%>
```

Example of Page directive:

```
<%@page import = "java.util.Date"%>
<%Date d = new Date();%>
<%=d%>
```

Include directive

- JSP include directive is used to include other files into the current jsp page. These files can be html files, other sp files etc. The advantage of using an include directive is that it allows code re-usability.
- **Syntax:**

```
<%@include file = "file location"%>
```

Example:

<%@include file = "abc.html"%>

abc.html

<**h1**>Welcome to Pune</**h1**>