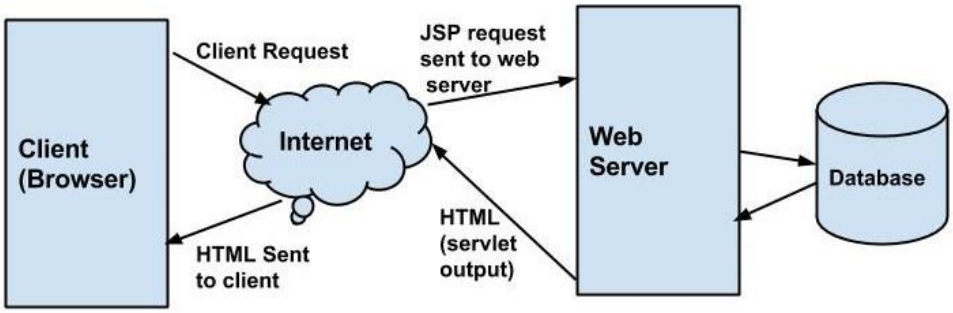
JSP stands for Java Server Pages. It is a server-side technology used for creating dynamic and platform independent web pages. JSP is the advanced version of Servlets.

JSP tags are used to insert JAVA code into HTML pages. This helps developers insert java code in HTML pages by making use of special JSP tags, most of which start with <% and end with %>.

JSP Architecture



Java Server Pages are part of a 3-tier architecture. A server(generally referred to as application or web server) supports the Java Server Pages. This server will act as a mediator between the client browser and a database.

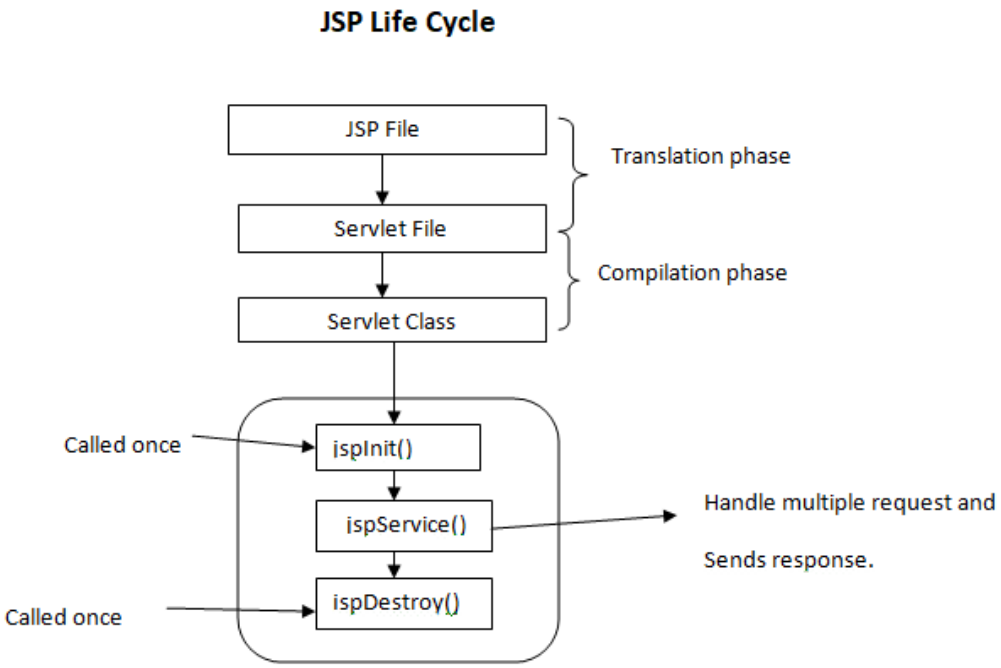
Architectural Flow:

1. The user goes to a JSP page and makes the request via internet in user's web browser.
2. The JSP request is sent to the Web Server.
3. Web server accepts the requested .jsp file and passes the JSP file to the JSP Servlet Engine.
4. If the JSP file has been called the first time then the JSP file is parsed otherwise servlet is instantiated. The next step is to generate a servlet from the JSP file. The generated servlet output is sent via the Internet form web server to users web browser.

5. Now in last step, HTML results are displayed on the users web browser.

JSP Life Cycle

A JSP life cycle is similar to a servlet life cycle with an added step wherein you need to compile a JSP into a servlet.



Phases involved in JSP Life cycle:

1. Translation of JSP page to Servlet
2. Compilation of JSP page : Compilation of JSP into test.java
3. Classloading : The classloader loads class file i.e.,convertion test.java to test.class takes place.
4. Instantiation : Object of the generated Servlet is created.
5. Initialization(jspInit() method is invoked by the container)
6. Request processing(_jspService()is invoked by the container)

7. JSP Cleanup (jspDestroy() method is invoked by the container)

Life Cycle Methods:

jspInit() : It is invoked only once during the life cycle of the JSP when JSP page is requested firstly. It is used to perform initialization. It is same as the init() method of Servlet interface.If you need to perform JSP-specific initialization, override the jspInit() method.

```
 public void jspInit()
{
   //initializing the code
}
```

_jspService() : It is invoked each time when request for the JSP page comes to the container. It is used to process the request. The underscore _ signifies that you cannot override this method.

```
void _jspservice(HttpServletRequest request HttpServletResponse response)
{
   //handling all request and responses
}
```

jspDestroy() : It is invoked only once during the life cycle of the JSP before the JSP page is destroyed. It can be used to perform some clean up operation.We can override jspdestroy() method when we perform any cleanup such as releasing database connections or closing open files.

```
public void _jspdestroy()
{
      //all clean up code
}
```

JSP Tags

Declaration Tag : We can declare variables and functions here

Syntax:-

```
<%!  Declare var/function  %>
```

Example :

```
<%!
    int age = 18;
    int value(int grade){
      return 10 +grade;
    }
 %>
```

Scriplet Tag : A scriptlet tag is used to execute java source code in JSP.

Syntax:-

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

Example :

```
<% out.print("welcome to jsp"); %>
```

Expression Tag

Syntax:-

```
<%= expression   %>
```

Example :

```
<% num1 = num1+num2 %>
```

Directive Tag : Tt commands JSP virtual engine to perform a specific task

Syntax:-

```
<%@ code   %>
```

Example :

```
<%@ page import="import java.sql.*" %>
<%@ include file="keogh/books.html" %>
<%@ taglib uri="myTags.tld" %>
```

Comments Tag : Used for writing comments in JSP.

Syntax:-

```
<%-- comments  --%>
```

Implicit Objects in JSP

1. **request** : This is the HttpServletRequest object associated with the request.
2. **response** : This is the HttpServletResponse object associated with the response to the client.
3. **session** : This is the HttpSession object associated with the request.
4. **out** : This is the PrintWriter object used to send output to the client.
5. **application** : This is the ServletContext object associated with the application context.
6. **config** : This is the ServletConfig object associated with the page.
7. **pageContext** : This encapsulates use of server-specific features like higher performance **JspWriters**.

8. **page** : This is simply a synonym for **this**, and is used to call the methods defined by the translated servlet class.
9. **Exception** : The Exception object allows the exception data to be accessed by designated JSP.