JSP Life Cycle-

JSP Life Cycle is defined as translation of JSP Page into servlet as a JSP Page needs to be converted into servlet first in order to process the service requests. The Life Cycle starts with the creation of JSP and ends with the disintegration of that.

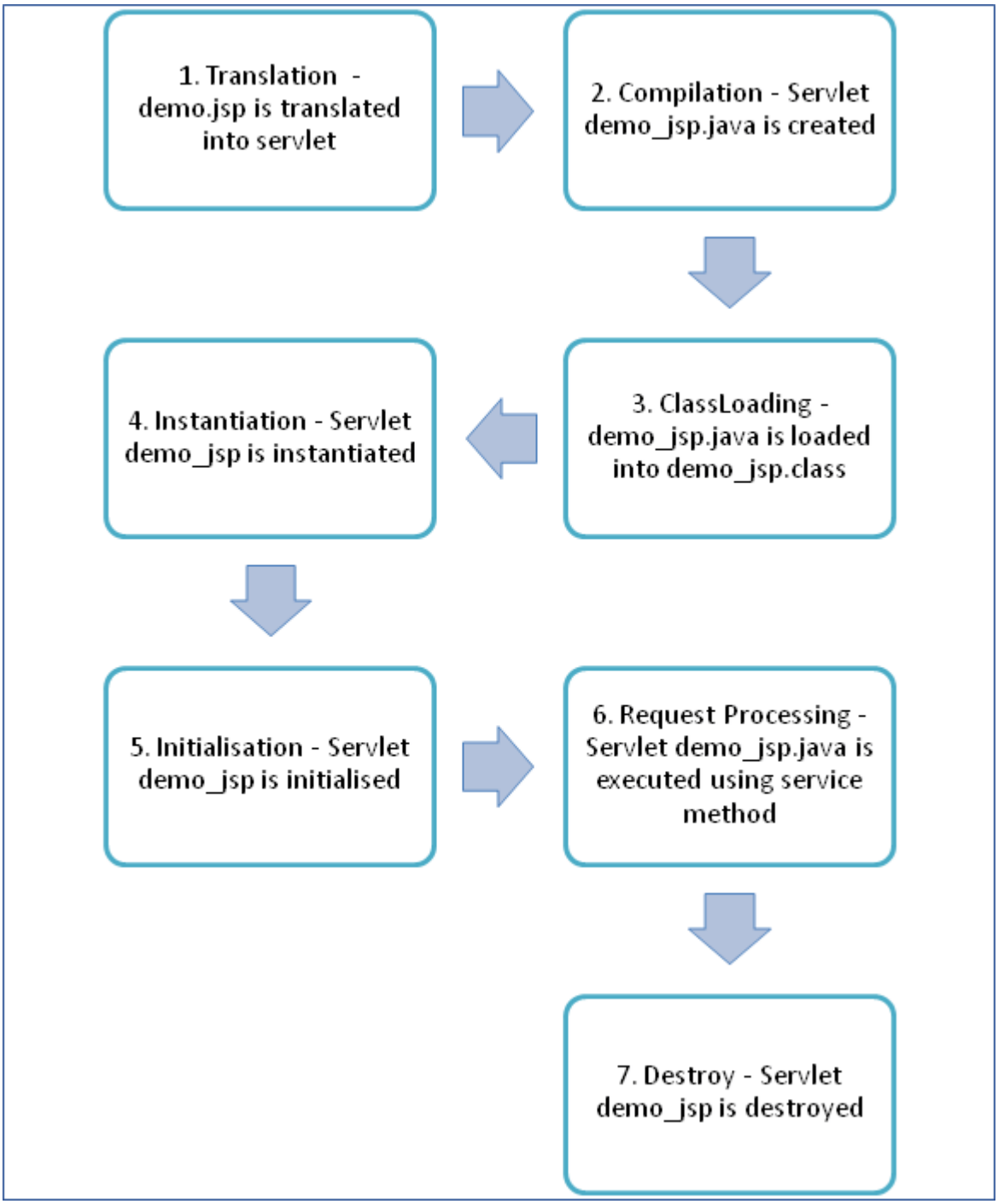
When the browser asks for a JSP, JSP engine first checks whether it needs to compile the page. If the JSP is last compiled or the recent modification is done in JSP, then the JSP engine compiles the page.

Compilation process of JSP page involves three steps:

1. Parsing of JSP
2. Turning JSP into servlet
3. Compiling the servlet

Following steps explain the JSP life cycle:

1. Translation of JSP page
2. Compilation of JSP page(Compilation of JSP page into _jsp.java)
3. Classloading (_jsp.java is converted to class file _jsp.class)
4. Instantiation(Object of generated servlet is created)
5. Initialisation(_jspinit() method is invoked by container)
6. Request Processing(_jspservice() method is invoked by the container)
7. Destroy (_jspDestroy() method invoked by the container)


1. Translation - demo.jsp is translated into servlet
2. Compilation - Servlet demo_jsp.java is created
3. ClassLoading - demo_jsp.java is loaded into demo_jsp.class
4. Instantiation - Servlet demo_jsp is instantiated
5. Initialisation - Servlet demo_jsp is initialised
6. Request Processing - Servlet demo_jsp.java is executed using service method
7. Destroy - Servlet demo_jsp is destroyed

1. Translation of the JSP Page:

A Java servlet file is generated from a JSP source file. This is the first step of JSP life cycle. In translation phase, container validates the syntactic correctness of JSP page and tag files.

- The JSP container interprets the standard directives and actions, and the custom actions referencing tag libraries (they are all part of JSP page and will be discussed in the later section) used in this JSP page.
- In the above pictorial description, demo.jsp is translated to demo_jsp.java in the first step
- Let's take an example of "demo.jsp" as shown below:

Demo.jsp

```
1. <html>
2. <head>
3. <title>Demo JSP</title>
4. </head>
5. <%
6. int demvar=0;%>
7. <body>
8. Count is:
9. <% Out.println(demovar++); %>
10.<body>
11.</html>
```

Code Explanation for Demo.jsp

Code Line 1: html start tag

Code Line 2: Head tag

Code Line 3 - 4: Title Tag i.e. Demo JSP and closing head tag

Code Line 5,6: Scriptlet tag wherein initializing the variable demo

Code Line 7 - 8: In body tag, a text to be printed in the output (Count is:)

Code Line 9: Scriplet tag where trying to print the variable demovar with incremented value

Code Line 10-11: Body and HTML tags closed

Demo JSP Page is converted into demo_jsp servlet in the below code.

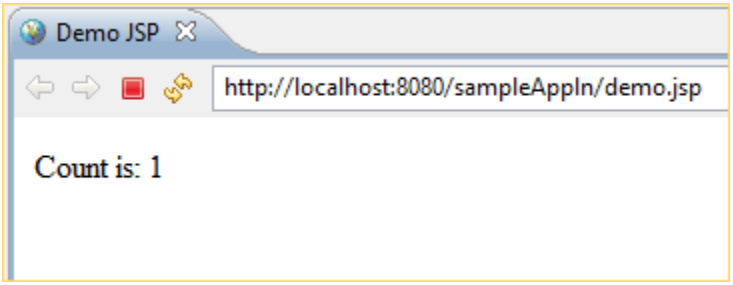
Code Line 8: Initializing demovar variable to 0

Code Line 9: Calling write() method of PrintWriter object to parse the text

Code Line 10: Calling print() method of PrintWriter object to increment the variable demovar from 0+1=1.Hence, the output will be 1

Code Line 11: Using write() method of PrintWriter object trying to parse html

Output:



- Here you can see that in the screenshot theOutput is 1 because demvar is initialized to 0 and then incremented to 0+1=1

In the above example,

- demo.jsp, is a JSP where one variable is initialized and incremented. This JSP is converted to the servlet (demo_jsp.class) wherein the JSP engine loads the JSP Page and converts to servlet content.
- When the conversion happens all template text is converted to println() statements and all JSP elements are converted to Java code.

This is how a simple JSP page is translated into a servlet class.

2. Compilation of the JSP Page

- The generated java servlet file is compiled into java servlet class
- The translation of java source page to its implementation class can happen at any time between the deployment of JSP page into the container and processing of the JSP page.
- In the above pictorial description demo_jsp.java is compiled to a class file demo_jsp.class

3. Classloading

- Servlet class that has been loaded from JSP source is now loaded into the container

4. Instantiation

- In this step the object i.e. the instance of the class is generated.
- The container manages one or more instances of this class in the response to requests and other events. Typically, a JSP container is built using a servlet container. A JSP container is an extension of servlet container as both the container support JSP and servlet.
- A JSPPage interface which is provided by container provides init() and destroy () methods.
- There is an interface HttpJSPPage which serves HTTP requests, and it also contains the service method.

5. Initialization

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

- _jspinit() method will initiate the servlet instance which was generated from JSP and will be invoked by the container in this phase.
- Once the instance gets created, init method will be invoked immediately after that
- It is only called once during a JSP life cycle, the method for initialization is declared as shown above

6. Request processing

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

- _jspservice() method is invoked by the container for all the requests raised by the JSP page during its life cycle
- For this phase, it has to go through all the above phases and then only service method can be invoked.
- It passes request and response objects
- This method cannot be overridden
- The method is shown above: It is responsible for generating of all HTTP methods i.eGET, POST, etc.

7.Destroy

```
public void _jspdestroy()
{
```

```
        //all clean up code
    }
```

- _jspdestroy() method is also invoked by the container
- This method is called when container decides it no longer needs the servlet instance to service requests.
- When the call to destroy method is made then, the servlet is ready for a garbage collection
- This is the end of the life cycle.
- We can override jspdestroy() method when we perform any cleanup such as releasing database connections or closing open files.

Code explanation for Demo_jsp.java

Code Line 1: Servlet class demo_jsp is extending parent class HttpServlet

Code Line 2,3: Overriding the service method of jsp i.e. _jspservice which has HttpServletRequest and HttpServletResponse objects as its parameters

Code Line 4: Opening method

Code Line 5: Calling the method getWriter() of response object to get PrintWriterobject (prints formatted representation of objects to text output stream)

Code Line 6: Calling setContentType method of response object to set the content type

Code Line 7: Using write () method of PrintWriter object trying to parse html