Q1. List and Explain various stages of JSP life cycle. Briefly give the function of each phase.

Ans.
1. A JSP life cycle can be defined as the entire process from its creation till the destruction.
2. It is similar to a servlet life cycle with an additional step which is required to compile a JSP into servlet.
3. A JSP page is converted into Servlet in order to service requests.
4. The translation of a JSP page to a Servlet is called Lifecycle of JSP.

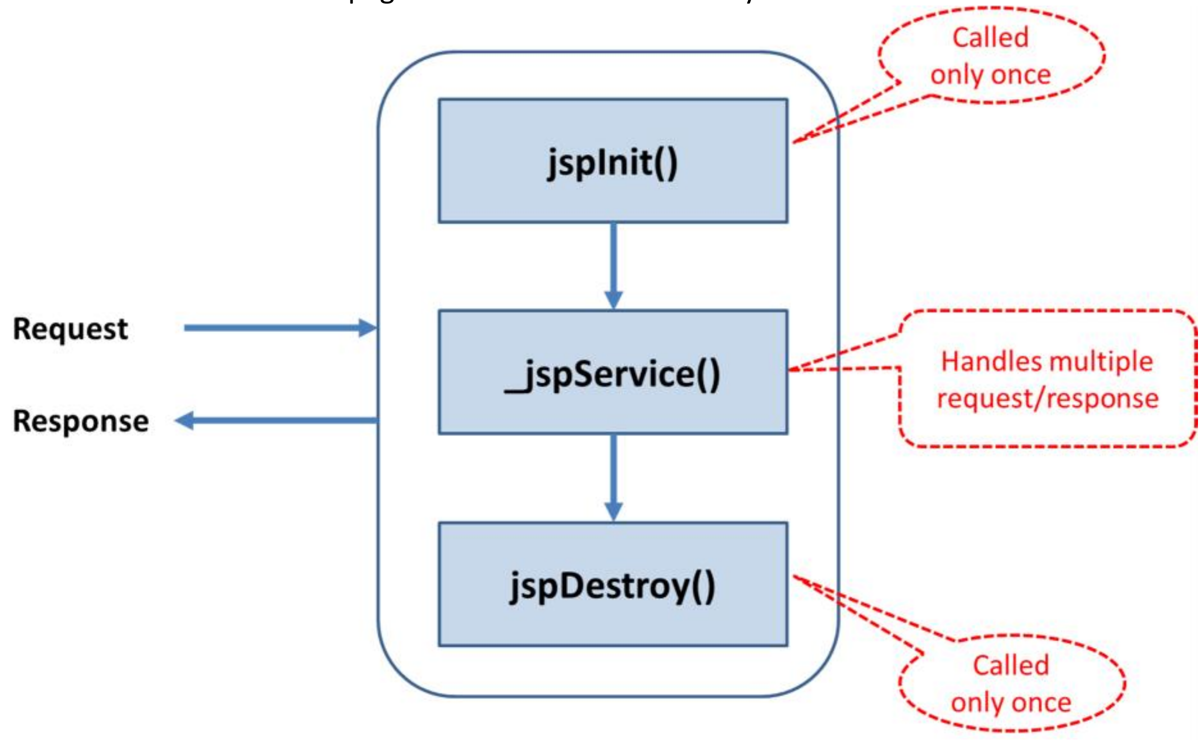


Figure: JSP Life Cycle

JSP Lifecycle Stages

1. **Translation of JSP to Servlet code**
    - Web Container translates JSP code into a servlet source(.java) file.
    - This is the first step in its tedious multiple phase life cycle.
    - In the translation phase, the container validates the syntactic correctness of the JSP pages and tag files.
    - The container interprets the standard directives and actions, and the custom actions referencing tag libraries used in the page.

2. **Compilation of Servlet to bytecode**
    - The JSP engine compiles the servlet into an executable class and forwards the original request to a servlet engine.
    - The translation of a JSP source page into its implementation class can happen at any time between initial deployment of the JSP page into the JSP container and the receipt and processing of a client request for the target JSP page.

3. **Loading Servlet class**
   The java servlet class that was compiled from the JSP source is loaded into the container.

4. **Creating servlet instance**
   In this execution phase the container manages one or more instances of this class in response to requests and other events.

5. **Initialization by calling jspInit() method**
   - When a container loads a JSP it invokes the jspInit() method before servicing any requests. If you need to perform JSP-specific initialization, override the jspInit().
   - Typically, initialization is performed only once and as with the servlet init method, you generally initialize database connections, open files, and create lookup tables in the jspInit method.

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

6. **Request Processing by calling _jspService() method**
   - This phase of the JSP life cycle represents all interactions with requests until the JSP is destroyed.
   - Whenever a browser requests a JSP and the page has been loaded and initialized, the JSP engine invokes the _jspService() method in the JSP.
   - The _jspService() method takes an HttpServletRequest and an HttpServletResponse as its parameters .

```
void _jspService(HttpServletRequest request,
                 HttpServletResponse response) {
   // Service handling code...
}
```

   - The _jspService() method of a JSP is invoked on request basis. This is responsible for generating the response for that request.

7. **Destroying by calling jspDestroy() method**
   - The destruction phase of the JSP life cycle represents when a JSP is being removed from use by a container.
   - The jspDestroy() method is the JSP equivalent of the destroy method for servlets.
   - Override jspDestroy(), when you need to perform any cleanup, such as releasing database connections or closing open files.

```
public void jspDestroy() {
   // Your cleanup code goes here.
}
```

   - When the call to destroy method is made then, the servlet is ready for a garbage collection
   - This is the end of the JSP life cycle.

Q2. Compare JSP with Servlet. Also state advantages of JSP over Servlets.

Ans.

JSP	Servlet
JSP is a webpage scripting language that generates dynamic content.	Servlets are Java programs that are already compiled which also creates dynamic web content.
A JSP technically gets converted to a servlet We embed the java code into HTML. E.g. <html> <% java code %> </html>	A servlet is a java class. We can put HTML into print statements. E.g. out.println("<html code>");
JSPs are extension of servlets which minimizes the effort of developers to write User Interfaces using Java programming.	A servlet is a server-side program and written purely on Java.
JSP runs slower than servlet. As, it has the transition phase for converting from JSP to a Servlet. Once it is converted to a Servlet then it will start the compilation	Servlets run faster than JSP
In MVC architecture JSP acts as view.	In MVC architecture Servlet acts as controller.
We can build custom tags using JSP API	We cannot build any custom tags in servlet.

Advantages of JSP over Servlets

1. JSP needs no compilation. There is automatic deployment of a JSP, recompilation is done automatically when changes are made to JSP pages.
2. In a JSP page visual content and logic are separated, which is not possible in a servlet.
   i.e. JSP separates business logic from the presentation logic.
3. Servlets use *println* statements for printing an HTML document which is usually very difficult to use. JSP has no such tedious task to maintain.

Q3. Explain JSP Scripting Elements with appropriate example.

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

- scriptlet tag
- expression tag
- declaration tag

Scriptlet tag

- A scriptlet tag is used to execute java source code in JSP.
- A scriptlet can contain
  - Any number of JAVA language statements
  - Variable
  - Method declarations
  - Expressions that are valid in the page scripting language

Syntax

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

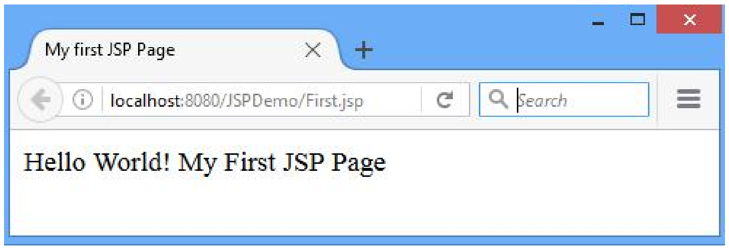
Example

```
<% out.print("welcome to jsp"); %>
<% int a=10;    %>
```

- Everything written inside the scriptlet tag is compiled as java code.
- JSP code is translated to Servlet code, in which **_jspService()** method is executed which has HttpServletRequest and HttpServletResponse as argument.
- JSP page can have any number of scriptlets, and each scriptlets are appended in _jspService ().

Program: First.jsp

```
<html>
<body>
<% out.println("Hello World! My First JSP Page"); %>
</body>
</html>
```



```
Hello World! My First JSP Page
```

Expression tag

- The code placed within **JSP expression tag** is *written to the output stream of the response*.
- So you need not write out.print() to write data.
- It is mainly used to print the values of variable or method.
- Do not end your statement with semicolon in case of expression tag.

Syntax

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

Example

```
<%=(2*5) %>
```

Declaration

- The **JSP declaration tag** is used *to declare variables and methods*
- The declaration of jsp declaration tag is placed outside the _jspService() method.

Syntax

```
<%!  variable or method declaration %>
```

Example

```
<%! int a = 10; %>
<%! int a, b, c; %>
<%! Circle a = new Circle(2.0); %>
```

Comments

- The comments can be used for documentation.
- This JSP comment tag tells the JSP container to ignore the comment part from compilation.

Syntax

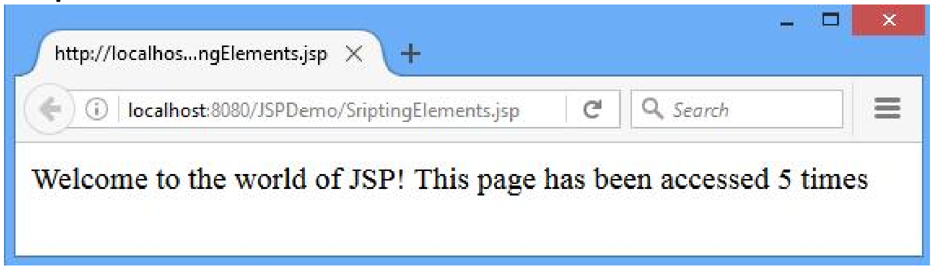
```
<%-- comments --%>
```

JSP comment	<%-- jsp comment --%>
Java comment	/* java comment */ or // for single line
Html comment	<!-- html comment -->

Write a JSP program to demonstrate use of all three scripting elements

```
<html>
<body>
<%-- comment:JSP Scipting elements --%>
<%! int i=0; %> <%--declaration--%>
<% i++; %> <%--Sciptlet--%>
Welcome to world of JSP!
<%= "This page has been accessed " + i + " times" %><%--expression--%>
</body>
</html>
```

Output


http://localhos...ngElements.jsp
localhost:8080/JSPDemo/SriptingElements.jsp
Search

Welcome to the world of JSP! This page has been accessed 5 times

Q4. Explain JSP Page Directives with appropriate example.

Ans.

- JSP directives provide directions and instructions to the container, telling it how to translate a JSP page into the corresponding servlet.
- A JSP directive affects the overall structure of the servlet class.
- JSP engine handles directives at Translation time.
- There are two types of directives:
    1. page directive
    2. include directive

Syntax

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

page directive

- The page directive defines attributes that apply to an entire JSP page.
- You may code page directives anywhere in your JSP page.
- By convention, page directives are coded at the top of the JSP page.

Syntax

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

Example

```
<%@page import="java.util.Date,java.util.List,java.io.*" %>
<%@page contentType="text/html; charset=US-ASCII" %>
```

Q5. Explain all the Attributes of Page Directive.

Ans.

Attributes of JSP page directive

import	Used to import class, interface or all the members of a package <%@ page **import**="java.util.Date" %> Today is: <%= **new** Date() %>
contentType	The contentType attribute defines the MIME type of the HTTP response. The default value is "text/html;charset=ISO-8859-1". <%@ page **contentType**=application/msword %>
extends	The extends attribute defines the parent class that will be inherited by the generated servlet <%@ page **extends**="javax.servlet.HttpServlet" %>
info	This attribute simply sets the information of the JSP page which is retrieved later by using getServletInfo() . <%@ page **info**="Authored by : AuthorName" %>
buffer	The buffer attribute sets the buffer size in kb to handle output generated by the JSP page. The default size of the buffer is 8Kb. <%@ page **buffer**="16kb" %>
language	The language attribute specifies the scripting language used in the JSP page. The default value is "java". <%@ page **language**="java" %>

isELIgnored	We can ignore the Expression Language (EL) in jsp by the isELIgnored attribute. By default its value is false i.e. EL is enabled by default. <%@ page **isELIgnored**="true" %>//Now EL will be ignored
autoFlush	The **autoFlush** attribute specifies whether buffered output should be flushed automatically when the buffer is filled.Bydefault it is true. <%@ page **autoFlush**="true" %>
isThreadSafe	This option marks a page as being thread-safe. By default, all JSPs are considered thread-safe(true). If you set the isThreadSafe = false, the JSP engine makes sure that only one thread at a time is executing your JSP. <%@ page **isThreadSafe** ="false" %>
session	The session attribute indicates whether or not the JSP page uses HTTP sessions. <%@ page **session**="true" %>//Bydefault it is true
pageEncoding	We can set response encoding type with this page directive attribute, its default value is "ISO-8859-1". <%@ page **pageEncoding** ="US-ASCII" %>
errorPage	It is used to define the error page, if exception occurs in the current page, it will be redirected to the error page. <%@ page **errorPage**="myerrorpage.jsp" %>
isErrorPage	The isErrorPage attribute is used to declare that the current page is the error page. <%@ page **isErrorPage**="true" %>

Q6. Explain JSP Include Directives with appropriate example.

Ans.

- JSP include directive is used to include the contents of another file to the current JSP page during translation time.
- Include directive is used for merging external files to the current JSP page during translation phase
- The included file can be HTML, JSP, text files etc.

Advantage of Include directive

Code Reusability

Syntax

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

Example

```
<%@ include file="1.jsp" %>
```

Q7. Explain JSP implicit objects with appropriate example.

Ans.
- There are **9 jsp implicit objects**.
- These objects are *created by the web container* that are available to all the jsp pages.

Sr.No.	Implicit Object	Example

1.	out	• For writing any data to the buffer, JSP provides an implicit object named *out*.
		• It is an object of JspWriter
		`<html>`
		`<body>`
		`<% out.print(“DIET”); %>`
		`</body>`
		`</html>`

2.	request	• Instance of *javax.servlet.http.HttpServletRequest* object associated with the request. • Each time a client requests a page the JSP engine creates a new object to represent that request. • The request object provides methods to get HTTP header information including from data, cookies, HTTP methods etc. `<% out.println(request.getParameter("login"));%>`
3.	response	• The response object is an instance of a javax.servlet.http.HttpServletResponse object. • Through this object the JSP programmer can add new cookies or date stamps, HTTP status codes, redirect response to another resource, send error etc. `<%response.sendRedirect("www.darshan.ac.in"); %>`

4.	config	• Config is an implicit object of type ***javax.servlet.ServletConfig.***
		• This object can be used to get initialization parameter for a particular JSP page.
		
		`<% out.print("Welcome "+ request.getParameter("login"));`
		
		`String c_name= config.getInitParameter("College");`
		`out.print("<p>College name is="+c_name+"</p>");%>`

5.	session	• In JSP, session is an implicit object of type ***javax.servlet.http.HttpSession***. • The Java developer can use this object to set, get or remove attribute or to get session information.

6.	pageContext	• Instance of ***javax.servlet.jsp.PageContext***
		• The pageContext object can be used to set, get or remove attribute.
		• The PageContext class defines several fields, including PAGE_SCOPE, REQUEST_SCOPE, SESSION_SCOPE, and APPLICATION_SCOPE, which identify the four scopes.
		
		`<% String name=`
		`(String)pageContext.getAttribute`
		`   ("user",PageContext.APPLICATION_SCOPE);`
		`out.print("Hello "+name);  %>`

7.	page	• This object is an actual reference to the instance of the page. • It is an instance of ***java.lang.Object*** • Direct synonym for the **this** object. ***Example: returns the name of generated servlet file*** `<%= page.getClass().getName() %>`

8.	application	• Instance of ***javax.servlet.ServletContext***
		• The instance of ServletContext is created only once by the web container when application or project is deployed on the server.
		• This object can be used to get initialization parameter from configuration file (web.xml).
		• This initialization parameter can be used by all jsp pages.
		
		`<%//refers to context parameter of web.xml`
		`String driver=application.getInitParameter("`
		`name");`
		`out.print("name is="+name); %>`

9.	exception	• Exception is an implicit object of type ***java.lang.Throwable*** class. This object can be used to print the exception.
		• But it can only be used in error pages.
		
		`<%@ page isErrorPage="true" %>`
		
		`<html> <body>`

		exception occured: <%=exception %> </body> </html>

Q8. Explain JSP Action elements with appropriate example.

Ans.

- JSP actions use constructs in XML syntax to control the behavior of the servlet engine.
- We can dynamically insert a file, reuse JavaBeans components, forward the user to another page, or generate HTML for the Java plugin.

Syntax

```
<jsp:action_name attribute="value" />
```

There are four types of JSP Action elements

1. <jsp:param>

- This action is useful for passing the parameters to other JSP action tags such as JSP include & JSP forward tag.
- This way new JSP pages can have access to those parameters using request object itself.

Syntax

```
<jsp:param name ="name" value="value"  />
```

Example

```
<jsp:param name ="date" value="10-03-2017" />
<jsp:param name ="time" value="10:15AM" />
<jsp:param name ="data" value="ABC" />
```

2. <jsp:include>

- The **jsp:include action tag** is used to include the content of another resource it may be jsp, html or servlet.
- The jsp:include tag can be used to include static as well as dynamic pages

Attribute	Description
page	The relative URL of the page to be included.
flush	The boolean attribute determines whether the included resource has its buffer flushed before it is included. By default value is *false*.

Syntax

```
<jsp:include page="relative URL"  flush="true" />
```

Example

```
<jsp:include page="2.jsp" />
```

3. <jsp:forward>

Forwards the request and response to another resource.

Syntax

```
     <jsp:forward page="Relative URL" />
```

Example

```
     <jsp:forward page="2.jsp" />
```

4. <jsp:plugin>

- This tag is used when there is a need of a plugin to run a Bean class or an Applet.
- The <jsp:plugin> action tag is used to embed applet in the jsp file.
- The <jsp:plugin> action tag downloads plugin at client side to execute an applet or bean.

Syntax

```
<jsp:plugin type="applet|bean"
            code="nameOfClassFile"
            codebase= "URL"
/>
```

Example

MyApplet.java

```
import java.applet.*;
import java.awt.*;
public class MyApplet extends Applet {
   public void paint(Graphics g) {
      g.drawString("Welcome in Java Applet.",40,20);
}}
```

MyPlugin.jsp

```
<html> <body>
        <jsp:plugin
           type="applet"
           code="MyApplet.class"
           codebase="/JSPClass/MyApplet"/>
</body></html>
```

Q9. What is EL Scripting? Explain EL implicit object and EL operator with appropriate example.

Ans. What is EL Scripting?

- Expression Language(EL) Scripting.
- The Java **Expression Language** is a special purpose programming language mostly used in Java web applications for embedding expressions into web pages.
- It is the newly added feature in JSP technology version 2.0.
- **The purpose of EL is to produce script less JSP pages.**

Syntax

```
${expr}
```

Example

EL	Output
${a=10}	10
${10+20}	30
${20*2}	40
${10==20}	false
${'a'<'b'}	true

EL Implicit Object

pageScope	It is used to access the value of any variable which is set in the Page scope
requestScope	It is used to access the value of any variable which is set in the Request scope.
sessionScope	It is used to access the value of any variable which is set in the Session scope
applicationScope	It is used to access the value of any variable which is set in the Application scope
pageContext	It represents the PageContext object.
param	Map a request parameter name to a single value
paramValues	Map a request parameter name to corresponding array of string values.
header	Map containing header names and single string values.
headerValues	Map containing header names to corresponding array of string values.
cookie	Map containing cookie names and single string values.

- An expression can be mixed with static text/values and can also be combined with other expressions

Example1

```
${param.name}
${sessionScope.user}
```

Example2

EL1.jsp

```
<form action="EL1.jsp">
 Enter Name:<input type="text" name="name" >
 <input type="submit" value="go">
</form>
```

EL2.jsp

```
Welcome, ${ param.name }
```

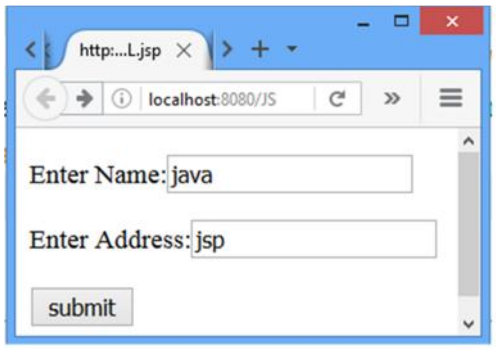
Example3:

Cookie_Session1.jsp

```
<form action="EL2.jsp">
<% Cookie ck=new Cookie("c1","abc");
   response.addCookie(ck);
   session.setAttribute("sid","054");  //for session
%>
Enter Name:<input type="text" name="name" >
Enter Address:<input type="text" name="address" >
<input type="submit" value="submit">
</form>
```

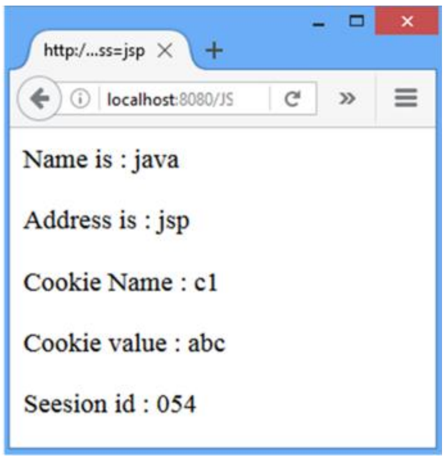
Cookie_Session2.jsp

```
<p>Name is :        ${param.name}</p>
<p>Address is :     ${param.address}</p>
<p>Cookie Name :    ${cookie.c1.name}</p>
<p>Cookie value :   ${cookie.c1.value}</p>
<p>Session id :     ${sessionScope.sid}</p>
```



Enter Name: java

Enter Address: jsp



Name is : java

Address is : jsp

Cookie Name : c1

Cookie value : abc

Seesion id : 054

JSP EL Operator

JSP EL Arithmetic Operators

Arithmetic operators are provided for simple calculations in EL expressions.
They are +, -, *, / or div, % or mod.

JSP EL Logical Operators

They are && (and), || (or) and ! (not).

JSP EL Relational Operators

They are == (eq), != (ne), < (lt), > (gt), <= (le) and >= (ge).

JSP EL Important Points

- EL expressions are always within curly braces prefixed with $ sign, for example ${expr}
- We can disable EL expression in JSP by setting JSP page directive isELIgnored attribute value to TRUE.
  ```
  <%@ page isELIgnored="true" %>
  ```
- JSP EL can be used to get attributes, header, cookies, init params etc, but we can't set the values.
- JSP EL implicit objects are different from JSP implicit objects except pageContext
- JSP EL is NULL friendly, if given attribute is not found or expression returns null, it doesn't throw any exception.

Q10. Explain Exception Handling in JSP.

Ans. **JSP provide 3 different ways to perform exception handling:**

1. Using simple **try...catch** block.
2. Using **isErrorPage** and **errorPage** attribute of **page directive**.
3. Using <**error-page**> tag in **Deployment Descriptor.**

1. **Using try...catch block** is just like how it is used in Core Java.

   Example
   ```
   <html>
   <body>
     <%
      try{
         int i = 100;
         i = i / 0;
         out.println("The answer is " + i);
      }
      catch (Exception e){
         out.println("An exception occurred: " + e.getMessage());
      }
     %>
   </body>
   </html>
   ```

2. **Using isErrorPage and errorPage attribute of page directive**

Example

1.jsp

```
<%@page  errorPage= "2.jsp" %>
<%    int i=10;
      i=i/0; %>
```

2.jsp

```
<%@page isErrorPage="true" %>
<html> <body>
     An Exception had occured
              <%  out.println(exception.toString());%>
</body> </html>
```

3. **Using <error-page> tag in Deployment Descriptor**
   - Declaring error page in Deployment Descriptor for entire web application.
   - Specify Exception inside
   - **<error-page>** tag in the Deployment Descriptor.
   - We can even configure different error pages for different exception types, or HTTP error code type(503, 500 etc).

Example1:web.xml

```
<error-page>
    <exception-type>
         java.lang.Throwable
    </exception-type>
    <location>/error.jsp</location>
</error-page>
```

Example2:web.xml

```
<error-page>
    <exception-type>
         java.lang.ArithmeticException
    </exception-type>
<location>/error.jsp</location>
</error-page>
```

Example3:web.xml

```
<error-page>
    <error-code>404</error-code>
    <location>/error.jsp</location>
</error-page>
<error-page>
    <error-code>500</error-code>
    <location>/error.jsp</location>
</error-page>
```

Q11. Write a JSP program to retrieve record from database.

Ans.

```
<%@page import="java.sql.*" %>
<%
Class.forName("com.mysql.jdbc.Driver");
Connection con=DriverManager.getConnection(
           "jdbc:mysql://localhost:3306/GTU","root","pwd");
Statement stmt=con.createStatement();
ResultSet rs=stmt.executeQuery("select * from diet");
while(rs.next())  {
          out.println("<p>"+rs.getString(1));
          out.println(rs.getString(2));
          out.println(rs.getString(3)+"</p>");
     }
     con.close();
     %>
```

---Mid Sem Syllabus Unit-4---