WEB TECHNOLOGY

UNIT 5

INTRODUCTION TO JSP

Multiple Choice Questions:

1. JSP stands for____.

a.Java Side Pages b.. **Java Server Pages**

c. Java Server Phase d. All of the above

2. Which technology do we mix our business logic with the presentation logic?

a. Servlet **b.** JSP

c. Both A and B d. None of the above

3. Which attribute specifies a JSP page that should process any exceptions thrown but not caught in the current page?

a.The ErrorPage Attribute **b.** The IsErrorPage Attribute

c.Both A & B d. None of the above

4. The ASP and JSP technologies are quite similar in the way they support the creation of Dynamic pages, using HTML templates, scripting code and components for business logic.

a.**True** b. False

c. none d.all

5. Which of the following is an advantage of the statement – Separation of business logic from JSP ?

a.Custom Tags in JSP b.JSP Standard Tag Library

c. All the above d.None of the above

6.Which is the Microsoft solution for providing dynamic Web content?

a.ASP **b.**JSP

c.Both A and B d. None of the above

7.Which tag is used to execute java source code in JSP?

a.Declaration Tag b. **Scriptlet tag**

c. Expression tag d. None of the above

8. A JSP page consists of which tags?

a.HTML tags b.JSP tags

c. Both A & B **d.**None of the above

9. Which of the scripting of JSP not putting content into service method of the convertedservlet?

a.**Declarations** b.Scriptlets

c.Expressions **d.** None of the above

10. The difference between Servlets and JSP is the …….

a.translation b.compilation

c. syntax **d.**Both A and B

11. Which of the following are the valid scopes in JSP?

a. **request, page, session, application** **b.** request, page, session, global

c.response, page, session, application d. request, page, context, application

12. JSP includes a mechanism for defining_____or custom tags.
a. static attributes b. local attributes
c.dynamic attributes d.global attributes

13. Why DB connections are not written directly in JSPs?
a.Response is slow b.Not a standard J2EE architecture
c.Load Balancing is not possible d.**Both B and C**

14. This object allows the JSP programmer access to the Servlet or JSP engine initialization parameters such as the paths or file locations etc . it is used to get initialization parameter from the web.xml file.
a.include b.page
c.export **d**.taglib

15. Which http method send by browser that asks the server to get the page only?
a.get b.option
c. put d.post

16. Which tag should be used to pass information from JSP to included JSP?
a.Using <%jsp:page> tag b.Using <%jsp:param> tag
c.Using <%jsp:import> tag d. Using <%jsp:useBean> tag

17. Attribute used to handle web flow requests.
a.servlet-mapping **b.** servlet-attr
c. servlet-flow d.servlet-requests

18. The method forward(request, response) will
a.return back to the same method from where the forward was invoked
b. not return back to the same method from where the forward was invoked and the web pages navigation continues
c.Both A and B are correct d. None of the above

19. "request" is instance of which one of the following classes?
a.HttpRequest b.Request
c.HttpServletRequest **d**.ServletRequest

20. Filters are defined in
a.deployment descriptor file web.xml. b**. jsp pages.**
c.Both of the above. d.None of these

21. Which of the following do not supports JSP directly?
a.Weblogic Server b. **Apache HTTP Server**
c. WebSphere Server d. Tomcat Server

22. Every JSP element is converted into corresponding _____.
a.HTML b. **JSP Code**
c. SQL d. ALL of the above

23. The out implicit object is an instance of a _____ object and is used to send content in a response.
a. javax.servlet.ServletContext b. HttpServletResponse

c. Both A and B d. **javax.servlet.jsp.JspWriter**

24. Which method of the Component class is used to set the position and size of a component in JSP?
a.setPosition() b. setSizePosition()
c. setBounds() **d.** etSize()

25.Every JSP element is converted into corresponding Java code.this phase is called _____.
a. **Translation phase** **b.**servelet phase
c. HTTP c. All of the above

Long Answers

1.What are the Advantages of using JSP?

Ans. Advantages of JSP:

- Jsp is useful for server side programming.
- Jsp can be used along with servlets. Hence business logic for any application canbe developed using Jsp.
- Dynamic contents can be handled using Jsp because jsp allows scripting and element based programming.
- Jsp allows creating and using our own custom tag libraries. Hence any application specific requirements can be satisfied using custom tag libraries.
- Jsp is a specification and not a product. Hence any variety of applications can bedeveloped.
- Jsp is essential component of J2ee.Hence using Jsp is possible to develop simpleas well as complex applications.

2. What are the problems that are associated with servlets?

Ans. Problem with Servlet:

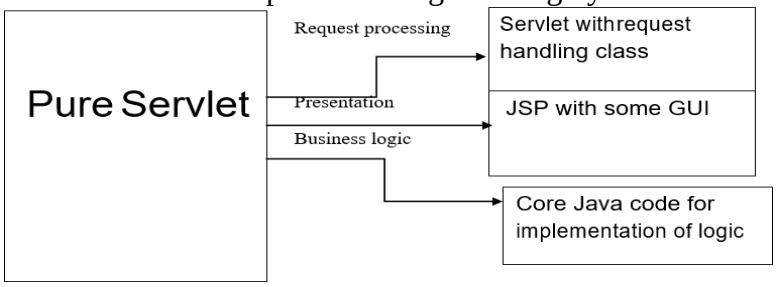
In only one class, the servlet alone has to do various tasks such as:

- Acceptance of request
- Processing of request
- Handling of business logic
- Generation of response

Hence there are some problems that are associated with servlets:

- For developing a servlet application, we need knowledge of Java as well as html code.
- While developing any web based application, look and feel of web based application needs to be changed then entire code needs to be changed and recompiled.
- There are some web page development tools available using which the developer can develop web based applications. But servlets don not support such tools. Even if such tools are used, we need to change embedded html code manually, which is time consuming and error prone.
- These problems associated with servlets are due to one and only one reason that is servlet has to handle all tasks of request processing. JSP is a technology that came up to overcome these problems.
- JSP is a technology in which request processing, business logic and presentations are separated out.
- JSP page is a simple web page which contains the **JSP elements** and **template text**.
- The template text can be scripting code such as Html, Xml or a simple plain text.
- Various JSP elements can be action tags, custom tags, JSTL library elements. These

JSP elements are responsible for generating dynamic contents.



3. What is JSP processing? Explain.

Ans. JSP Processing:

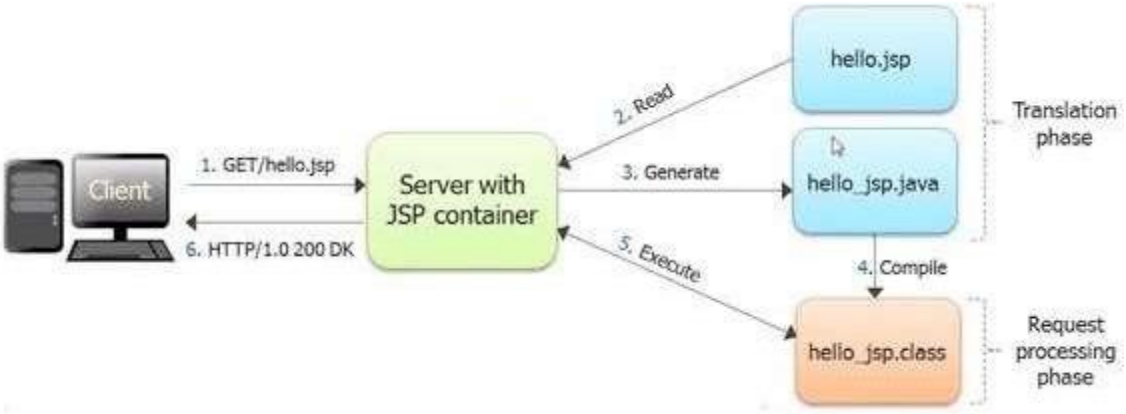
JSP pages can be processed using JSP Container only. Following are the steps that needto be followed while processing the request for JSP page:

- Client makes a request for required JSP page to server. The server must have JSP container so that JSP request can be processed. For instance: let the client makes request for xyz.jsp page.
- On receiving this request the JSP container searches and then reads the desired JSPpage. Then this JSP page is converted to corresponding servlet.
- Basically any JSP page is a combination of template text and JSP elements. Every template text is converted to corresponding println statement.

Every JSP element is converted into corresponding Java code.

This phase is called Translation phase, output of it is a servlet.for instance:xyz.jsp is converted to xyzservlet.java

- This servlet is compiled to generate servlet class file. Using this class response is generated. This phase is called request processing phase.
- The Jsp container thus executes servlet class file.
- A requested page is then returned to client as response.



4. Explain the Anatomy of JSP page?

Ans. **ANATOMY OF JSP PAGE:**

- JSP page is a simple web page which contains the **JSP elements** and **template text**.
- The template text can be scripting code such as Html, Xml or a simple plain text.
- Various JSP elements can be action tags, custom tags, JSTL library elements. TheseJSP elements are responsible for generating dynamic contents.

```
<%@ page import="java.util.Date;" %>
<html>
<head>
<title>
first jsp program</title>
</head>
<body>
<%
out.println("Hello BTech CSE Students");
out.println("<br><br>");
out.println("Welcome to Jsp Programming");
out.println(new Date().toString());
%>
<center>Have a nice Day</center>
<br><br>
</body>
</html>
```

JSP Element

Template Text

JSP Element

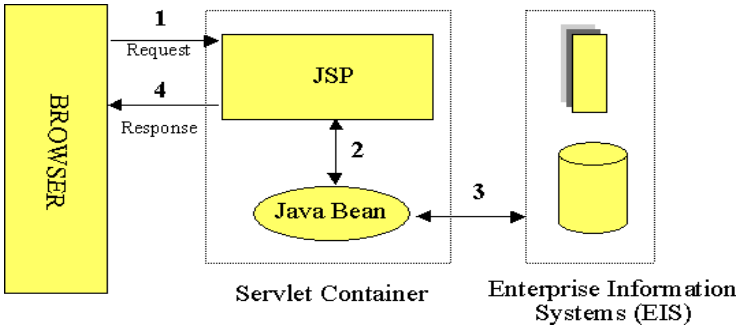
TempalteText

5. Explain the JSP Access model.

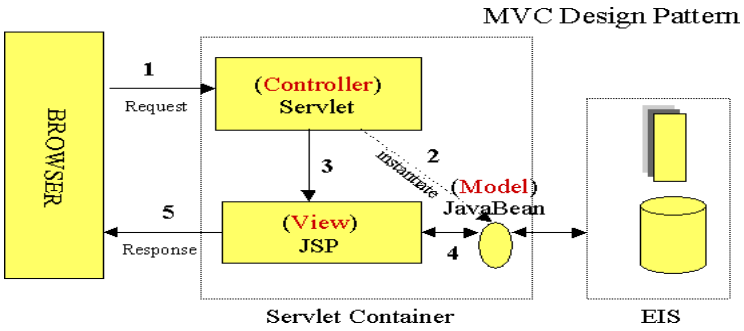
Ans. : JSP Access Models

The early JSP specifications advocated two philosophical approaches, popularly known as Model 1 and Model 2 architectures, for applying JSP technology. These approaches differ essentially in the location at which the bulk of the request processing was performed, and offera useful paradigm for building applications using JSP technology.

Consider the Model 1 architecture, shown below:



In the Model 1 architecture, the incoming request from a web browser is sent directly tothe JSP page, which is responsible for processing it and replying back to the client. There is still separation of presentation from content, because all data access is performed using beans. Although the Model 1 architecture is suitable for simple applications, it may not be desirable for complex implementations. Indiscriminate usage of this architecture usually leads to a significant amount of scriptlets or Java code embedded within the JSP page, especially if there is a significant amount of request processing to be performed. While this may not seem to be much of a problem for Java developers, it is certainly an issue if your JSP pages are created and maintained by designers--which is usually the norm on large projects. Another downside of this architecture is that each of the JSP pages must be individually responsible for managing application state and verifying authentication and security.



The Model 2 architecture, shown above, is a server-side implementation of the popular Model/View/Controller design pattern. Here, the processing is divided between presentation and front components. Presentation components are JSP pages that generate the HTML/XML response that determines the user interface when rendered by the browser. Front components (also known as controllers) do not handle any presentation issues, but rather, process all the HTTP requests.

Here, they are responsible for creating any beans or objects used by the presentation components, as well as deciding, depending on the user's actions, which presentation component to forward the request to. Front components can be implemented as either a servletor JSP page.

The advantage of this architecture is that there is no processing logic within the presentation component itself; it is simply responsible for retrieving any objects or beans that may have been previously created by the controller, and extracting the dynamic content within for insertion within its static templates.
Consequently, this clean separation of presentation from content leads to a clear delineation of the roles and responsibilities of the developers and page designers n the programming team. Another benefit of this approach is that the front components present a single point of entry into the application, thus making the management of application state, security, and presentation uniform and easier to maintain.

6. What are the different implicit objects used in JSP?

Ans. JSP - 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.
Java objects that the JSP Container makes available to the developers in each page and the developer can call them directly without being explicitly declared. JSP Implicit Objects are also called **pre-defined variables**.

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

1. out: This is the **PrintWriter** object where methods like print and println help for displaying the content to the client.
2. *request*: This is the object of **HttpServletRequest** class associated with the request.
3. *response*: This is the object of **HttpServletResponse** class associated with the response to the client.
4. *config*: This is the object of **ServletConfig** class associated with the page.
5. *application*: This is the object of **ServletContext** class associated with the application context.
6. *session*: This is the object of **HttpSession** class associated with the request.

7. *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.
8. *page object*: The manner we use the keyword **this** for current object, page object is used to refer to the current translated servlet class.
9. *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**.

7. Explain any three implicit objects in JSP.

Ans. JSP - 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.
Java objects that the JSP Container makes available to the developers in each page and the developer can call them directly without being explicitly declared. JSP Implicit Objects are also called **pre-defined variables**.\

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

1) JSP out implicit object
For writing any data to the buffer, JSP provides an implicit object named out. The out implicit object is an instance of a **javax.servlet.jsp.JspWriter** object and is used to send content in a response.

Following table lists out the important methods that we will use to write **boolean char, int, double, object, String**, etc.

S.No.	Method & Description
1	**out.print(dataType dt)** Print a data type value
2	**out.println(dataType dt)** Print a data type value then terminate the line with new line character.
3	**out.flush()** Flush the stream.

In case of servlet you need to write:

Syntax: PrintWriter out=response.getWriter();

Example of out implicit object
In this example we are simply displaying date and time.
index.jsp

```
<html>
<body>
<% out.print("Today is:"+java.util.Calendar.getInstance().getTime()); %>
</body>
</html>
```

2) JSP request implicit object

The **JSP request** is an implicit object of type HttpServletRequest i.e. created for each jsp request by the web container. It can be used to get request information such as parameter, header information, remote address, server name, server port, content type, character encoding etc.

It can also be used to set, get and remove attributes from the jsp request scope.

Following is the simple example of request implicit object where we are printing the name of the user withwelcome message.

Example of JSP request implicit object

index.html

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

welcome.jsp

```
<%
String name=request.getParameter("uname");
out.print("welcome "+name);
%>
```

3) JSP response implicit object

In JSP, response is an implicit object of type HttpServletResponse. The instance of HttpServletResponseis created by the web container for each jsp request.

It can be used to add or manipulate response such as redirect response to another resource, senderror etc.

Let's see the example of response implicit object where we are redirecting the response to the Google.

Example of response implicit object

index.html

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

welcome.jsp

```
<%
response.sendRedirect("http://www.google.com");
%>
```

4)JSP config implicit object

The config object is an instantiation of **javax.servlet.ServletConfig** and is a direct wrapper around the **ServletConfig** object for the generated servlet. This object allows the JSP programmer access to the Servlet or JSP engine initialization parameters such as the paths or file locations etc. Generally, it is used to get initialization parameter from the web.xml file.

The following **config** method is the only one you might ever use, and its usage is trivial −
config.getServletName();

This returns the servlet name, which is the string contained in the **<servlet-name>** element defined in the **WEB-INF\web.xml** file. This object can be used to get initialization parameter for a particular JSP page. The config object is created by the web container for each jsp page.

Example of config implicit object:
index.html

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

web.xml file

```
<web-app>
 <servlet>
<servlet-name>sonoojaiswal</servlet-name>
<jsp-file>/welcome.jsp</jsp-file>

<init-param>
<param-name>dname</param-name>
<param-value>sun.jdbc.odbc.JdbcOdbcDriver</param-value>
</init-param>

</servlet>

<servlet-mapping>
<servlet-name>sonoojaiswal</servlet-name>
<url-pattern>/welcome</url-pattern>
</servlet-mapping>
 </web-app>
```

welcome.jsp

```
<%
out.print("Welcome "+request.getParameter("uname"));

String driver=config.getInitParameter("dname");
out.print("driver name is="+driver);
%>
```

5) JSP application implicit object

The application object is direct wrapper around the **ServletContext** object for the generated Servlet and in reality an instance of a **javax.servlet.ServletContext** object.

This object is a representation of the JSP page through its entire lifecycle. This object is created when the JSP page is initialized and will be removed when the JSP page is removed by the **jspDestroy()** method.

The instance of ServletContext is created only once by the web container when application or projectis deployed on the server.
This object can be used to get initialization parameter from configuaration file (web.xml). It can alsobe used to get, set or remove attribute from the application scope.This initialization parameter can be used by all jsp pages.

Example of application implicit object:

index.html

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

web.xml file

```
<web-app>
  <servlet>
<servlet-name>sonoojaiswal</servlet-name>
<jsp-file>/welcome.jsp</jsp-file>
</servlet>

<servlet-mapping>
<servlet-name>sonoojaiswal</servlet-name>
<url-pattern>/welcome</url-pattern>
</servlet-mapping>

<context-param>
<param-name>dname</param-name>
<param-value>sun.jdbc.odbc.JdbcOdbcDriver</param-value>
</context-param>
  </web-app>
```

welcome.jsp

```
<%
out.print("Welcome "+request.getParameter("uname"));
String driver=application.getInitParameter("dname");
out.print("driver name is="+driver);
  %>
```

6)session implicit object

In JSP, session is an implicit object of type HttpSession.The Java developer can use this object to set,get or remove attribute or to get session information. The session object is used to track client session between client requests.

Example of session implicit object

index.html

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

welcome.jsp

```
<html>
<body>
<%
  String name=request.getParameter("uname");
 out.print("Welcome "+name);
  session.setAttribute("user",name);
  <a href="second.jsp">second jsp page</a>
  %>
</body>
</html>
```

second.jsp

```
<html>
<body>
<%
  String name=(String)session.getAttribute("user");
out.print("Hello "+name);
%>
</body>
</html>
```

Output:

7)The pageContext Object

The pageContext object is an instance of a **javax.servlet.jsp.PageContext** object. The pageContext object is used to represent the entire JSP page. This object is intended as a means to access information about the page while avoiding most of the implementation details. This object stores references to the request and response objects for each request.

The **application, config, session**, and out objects are derived by accessing attributes of this object. The pageContext object also contains information about the directives issued to the JSP page, including the buffering information, the errorPageURL, and page scope.

The PageContext class defines several fields, including **PAGE_SCOPE, REQUEST_SCOPE, SESSION_SCOPE,** and **APPLICATION_SCOPE**, which identify the four scopes. It also supports more than 40 methods, about half of which are inherited from the **javax.servlet.jsp.JspContext class**.

One of the important methods is **removeAttribute**. This method accepts either one or two arguments. For example, **pageContext.removeAttribute ("attrName")** removes the attribute from all scopes,

8) The page Object

This object is an actual reference to the instance of the page. It can be thought of as an object that represents the entire JSP page. The page object is really a direct synonym for the **this** object.

9) The exception Object

The exception object is a wrapper containing the exception thrown from the previous page. It is typically used to generate an appropriate response to the error condition.

8. What are the different scripting elements used in JSP?

Ans. : Scripting elements in JSP must be written within the <% %> tags. The JSP engine will process any code you write within the pair of the <% and %> tags, and any other texts within the JSP page will be treated as HTML code or plain text while translating the JSP page.

The five different scripting elements:

1) Comment <%-- Set of comment statements --%>
2) Directive <%@ directive %>
3) Declaration <%! declarations %>
4) Scriptlet <% scriplets %>
5) Expression <%= expression %>

- **Comments** are marked as text or statements that are ignored by the JSP container. They are useful when you want to write some useful information or logic to remember in the future.

  Ex.

```
<%@ page import="java.util.Date"%>
<!DOCTYPE html>
<html>
  <head>
    <title>Comments in JSP Page</title>
  </head>
  <body>
    <%-- A JSP comment --%>
    <!-- An HTML comment -->
    <!—The current date is <%= new Date() %>
    -->
  </body>
</html>
```

- **Directive:** These tags are used to provide specific instructions to the web container when the page is translated. It has three subcategories:
- Page: <%@ page ... %>
- Include: <%@ include ... %>
- Taglib: <%@ taglib ... %>

- **Declaration:** As the name suggests, it is used to declare methods and variables you will use in your Java code within a JSP file. According to the rules of JSP, any variable must be declared before it can be used.

  Syntax:

  ```
  <%! declaration; [ declaration; ]+ ... %>
  ```

  Example:

  ```
  <!DOCTYPE html>
  <html>
     <body>
        <%! int variable_value=62; %>
        <%= " Your integer data is :"+ variable_value %>
     </body>
  </html>
  ```

- **Scriptlet:** The scriptlet tag allows writing Java code statements within the JSP page. This tag is responsible for implementing the functionality of _jspService() by scripting the java code.

  Syntax:

  <% JAVA CODE %>

  Example:

  ```
  <!DOCTYPE html>
  <html>
     <head>
        <title>Scriptlet Tag</title>
     </head>
     <% int count = 2; %>
     <body>
        Calculated page count <% out.println(cnt); %>
     </body>
  </html>
  ```

- **Expression:** Expressions elements are responsible for containing scripting language expression, which gets evaluated and converted to Strings by the JSP engine and is meant to the output stream of the response. Hence, you are not required to write out.print() for writing your data. This is mostly used for printing the values of variables or methods in the form of output.

  Example:

  ```
  <!DOCTYPE html>
  <html>
     <body>
        <%= "A JSP based string" %>
     </body>
  </html>
  ```

9. Explain briefly the MVC architecture in JSP.

Ans.: **Model-View-Controller Architecture (MVC)**

Model: is used for Buisness Logic

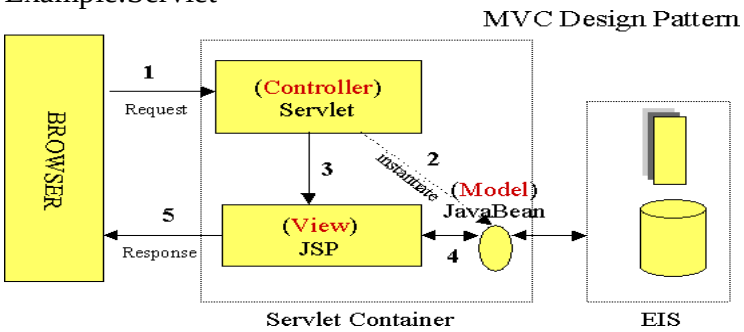
Example:Javabean,EJB

View: is used for Presentation Logic

Example:HTML,JSP

Controller: is used for RequestProcessing

Example:Servlet



Model/View/Controller design pattern. Here, the processing is divided between presentation and front components. Presentation components are JSP pages that generate the HTML/XML response that determines the user interface when rendered by the browser. Front components (also known as controllers) do not handle any presentation issues, but rather, process all the HTTP requests.

Here, they are responsible for creating any beans or objects used by the presentation components, as well as deciding, depending on the user's actions, which presentation component to forward the request to. Front components can be implemented as either a servletor JSP page.

The advantage of this architecture is that there is no processing logic within the presentation component itself; it is simply responsible for retrieving any objects or beans that may have been previously created by the controller, and extracting the dynamic content within for insertion within its static templates.

Consequently, this clean separation of presentation from content leads to a clear delineation of the roles and responsibilities of the developers and page designers n the programming team. Another benefit of this approach is that the front components present a single point of entry into the application, thus making the management of application state, security, and presentation uniform and easier to maintain.

10.Explain exception handling in JSP.

Ans. Exception Handling

JSP provides a rather elegant mechanism for handling runtime exceptions. Although you can provide your own exception handling within JSP pages, it may not be possible to anticipate all situations. By making use of the page directive's errorPage attribute, it is possible to forward an uncaught exception to an errorhandling JSP page for processing.

JSP declares 9 implicit objects, the exception object being one of them. It is an object of java.lang.Throwable class, and is used to print exceptions. However, it can only be used in error pages.

There are two ways of handling exceptions in JSP. They are:

- By errorPage and isErrorPage attributes of page directive
- By <error-page> element in web.xml file

Handling Exception using page directive attributes
The page directive in JSP provides two attributes to be used in exception handling. They're:

- **errorPage**: Used to site which page to be displayed when exception occurred.
  Syntax :

```
<%@page errorPage="url of the error page"%>
```

- **isErrorPage**: Used to mark a page as an error page where exceptions are displayed.
  Syntax :

```
<%@page isErrorPage="true"%>
```