JSP

Java Server Pages

- Servlet and JavaServer Pages (JSP) for JEE web application. A typical web application will contain both servlet and JSP complementing each other.
- JavaServer Pages (JSP) is a simple and fast way to create JEE web application.
- It enables addition of dynamically content with mixing of static HTML thus making programming and debugging very simple unlike servlets.
- JSP 2.2 version is what we will be taking about in these sessions

Advantages

- Allows the separation of developer and author roles.
  - While developers write dynamic content that interact with server-side, authors put static data and dynamic content together to create presentations . Hence a developer need not be a web page designer.
  - Allows reuse of components using Java Beans and tag libraries.
  - Allows easy embedding of dynamic components in a page.
  - Forms view part of MVC .

Tell me where

- Both Servlets and JSP can also be used to generate dynamic content. Where should JSP be used and where should Servlet be used?
- Servlets, JSP and Java Beans technologies like EJB form a very important part of Model View Controller Architecture.
- JSP is used for the View part.
- This is explained in the next slide when we attempt to understand MVC.

Model View Controller

- View is the component that interacts with the user. In web application the web page is used for user interaction. Web page could have static part as well as dynamic part. JSP, HTML become important part of View.
- Model represents data. It is used to interact with the data source like database to fetch data and initialize the data. Java Beans form the part of the model.
- Controller is acts in between Model and View. It is used to implement business logic. It fundamentally creates data objects and passes it on to the view. Servlets form the controller.

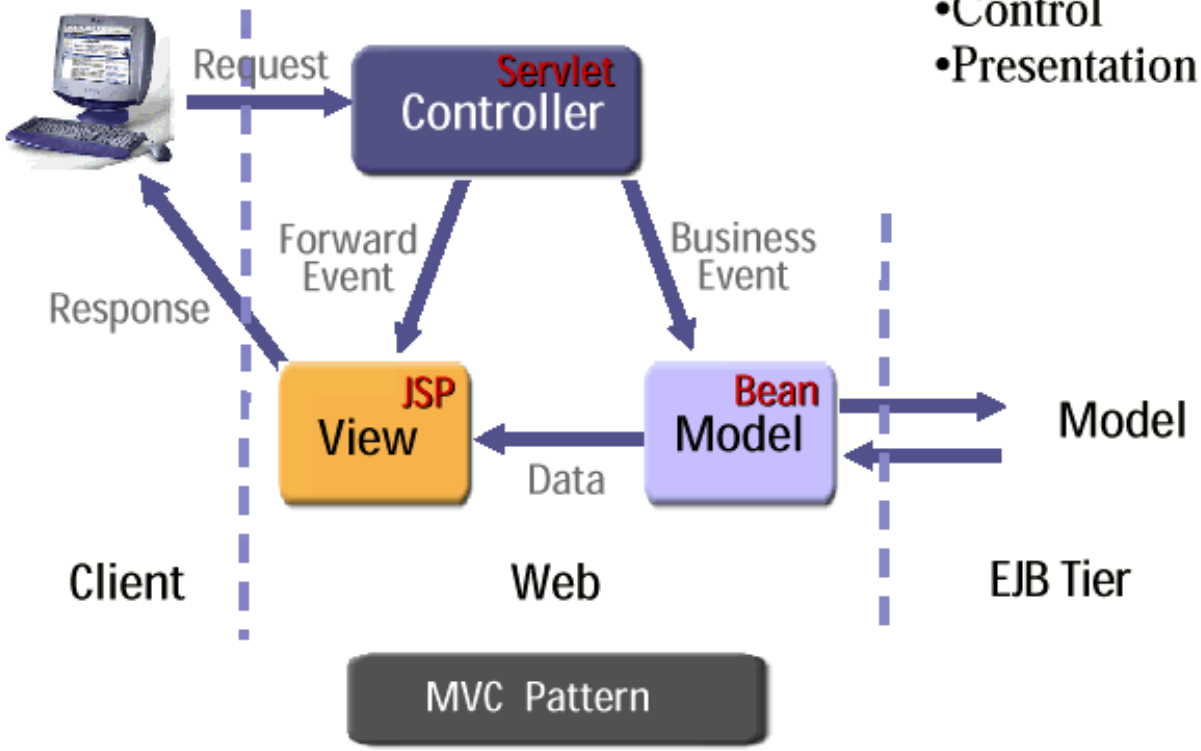
A typical MVC architecture

Model View Controller

Sun Tech Days

Sun microsystems

- Separate:
  - Model data
  - Control
  - Presentation of view



Syntactic Elements of a JSP Page

- A JSP page is composed of
  - template data
    - Template data is part of the JSP file that the JSP translator does not translate. In other words, it is anything that is not a scripting element → anything that JSP processor does not process.
    - A large percentage of the JSP page just consists of static HTML, known as template text
  - elements
    - An element is an instance of an element type known to the JSP container

JSP file in a web application

- JSP files are placed in the same place as html files i.e at the root of the web application.

JSP Comment

- A comment to that appears in the JSP page but not in the resultant document is a JSP comment
- `<%-- JSP Comment --%>`

Elements

- Scripting elements
  - Scripting elements provide glue around template text and actions.
  - EL expressions
- Directive elements
  - provide information for the translation phase.
- Action elements
  - Actions provide information for the request processing phase. An Actions can either be standard, that is, defined in this specification, or custom, that is provided via the portable tag extension mechanism.

Scripting elements

- Three language-based types of scripting elements
  - scriptlets `<% .... %>`

    `<jsp:scriptlet> jsp code </jsp:scriptlet>`
  - expressions `<%= ... %>`

    `<jsp:expression>expression</jsp:expression>`
  - declarations `<%! ... %>`

    `<jsp:declaration>declaration</jsp:declaration>`
- Since XML has become the de-facto standard for data transfer, HTML evolved into becoming XHTML. JSP also adopted XML syntax for its scripting elements.
- A JSP document is an XML document; this means that a JSP document is a well-formed, structured document and that this will be validated by the JSP container.

Scriptlets

- `<% .... %>`
- A set of one or more java statements intended to be used to process the request.
- A jsp file is saves with .jsp extension.
- It is not compiled like Servlets. It is compiled only when it is requested for the first time.

Example- Scriplet

```
<%--List of prime numbers between 2 to 100 --%>
<html>
 <head><title>Prime</title> </head>
 <body>
  <h1> List of prime numbers between 2 to 100</h1>
```

Scriplets

```
<% boolean flag=true;
 for(int i=2;i<100;i++){
     for(j=2;j=i/2;j++){
          if(i%j==0) flag=false;
          break;       }
    if(flag) out.println(i+ "<br>");
}
%>
```

Template data

Special object used to print the output to html page- more on this coming up …

```
</body>
</html>
```

Expressions

- Simpler way to print value of a java variable or expression into HTML page
- `<%= expression %>`

Note: no semicolon

Expressions- Example

```
<html>
 <head>
   <title> Date </title> </head>
 <body>
  <% java.util.Calendar now=Calendar.getInstance(); %>
  <h2> Server date and time is
  <%=now %>
</h2>
 </body>
</html>
```

Scriplet

Expression

JSP Declarations

- Allows the declaration of member variables and methods in JSP source code which finally become part of the generated servlet class.
- **`<%! expression %>`**

Example: Declaration

```
<html>
<body>Date and Time:<%=getDate()%>
    <%!
      int counter=0;
       java.util.Date getDate(){
            return new java.util.Date();
        }
    %>
  Thank you for visiting this page <BR>
  You are visitor number <%= ++counter%>
</body></html>
```

Declarations

Tell me what

- What is the difference between the two variable declarations given below?
  - `<% int x=10; %>`
  - `<% ! int x=10;%>`

- The 1st declaration is like a local variable declaration inside the **`service`** method of the servlet.
- The 2nd declaration is like a instance variable declaration of the servlet.

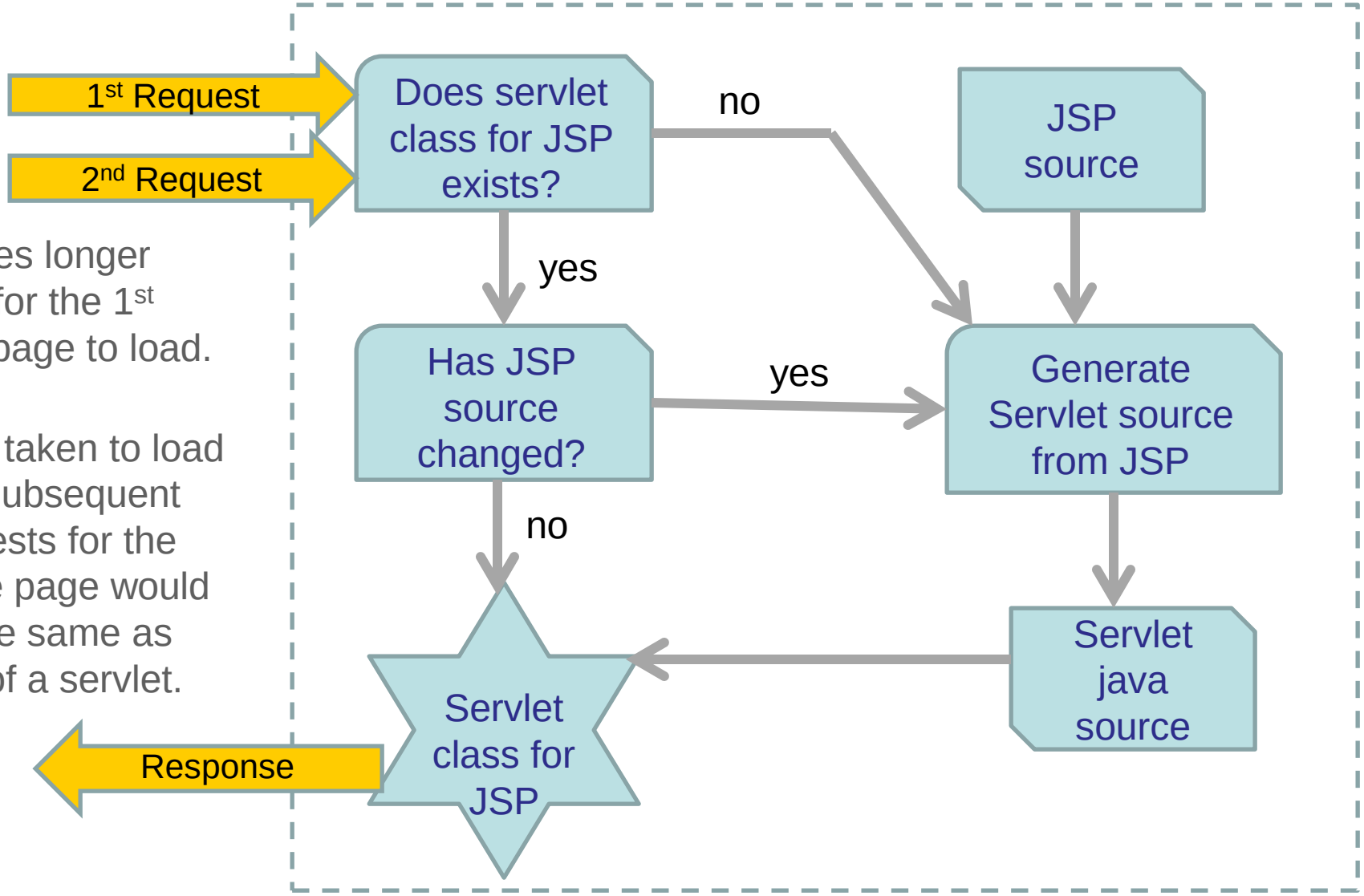
Let us look at the JSP life cycle to have a better insight.

JSP Execution Models

1. On-Demand Translation Model
2. Pretranslation Model

- Most of the application server support On-Demand Translation Model
- By default, On-Demand Translation Model is used for translation
- Some application server support Pretranslation Model.

On-Demand Translation



It takes longer time for the 1st JSP page to load.

Time taken to load any subsequent requests for the same page would be the same as that of a servlet.

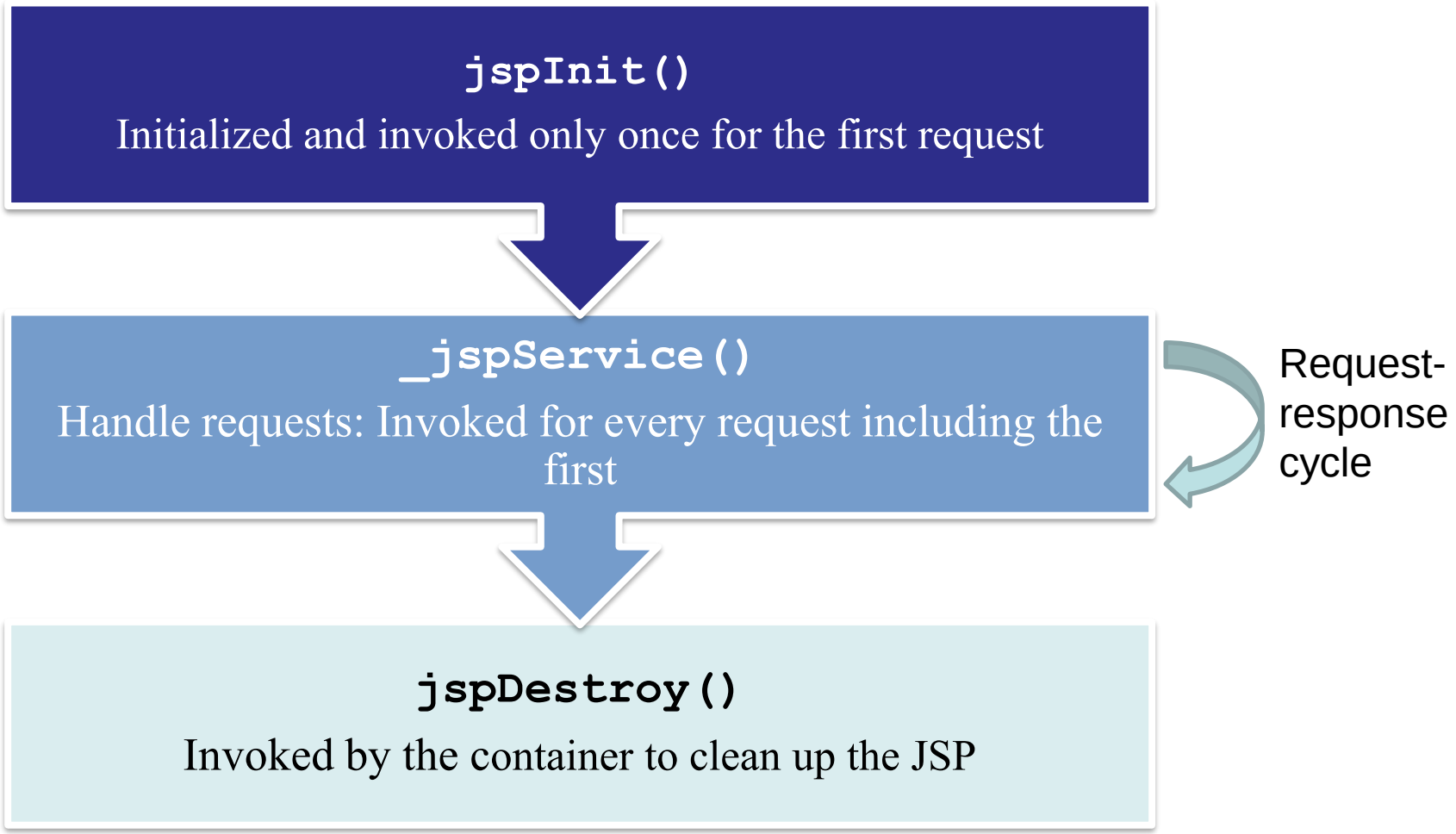
Pretranslation Model

- In some cases, it may be needed that pages are translated in advance and deploy them as working servlets.
- Advantages of these are:
  - Saving time when JSP page is requested first request time, because translation at execution time is not necessary.
  - It is useful when there are security concerns like proprietary code and so deployment has o happen in binary form
- Steps to accomplish Pretranslation Model is provided in http://docs.oracle.com/cd/A97336_01/buslog.102/a83726/instlcf2.htm#631353

javax.servlet.jsp.HttpJspPage

- JSP page is converted into servlet and this class implements **`HttpJspPage`** (inherits from **`JspPage`** which in turn inherits from Servlet)
- The life cycle methods of **`HttpJspPage`**
  - **`public void jspInit()`**
  - **`public void _jspService(HttpServletRequest req, HttpservlerResponse res) throws servletException, IOException`**
  - **`public void jspDestroy()`**
- Call back life cycle methods like **`jspInit()`** and **`jspDestroy()`** are written inside the JSP declaration.
- The scriptlet and expression are converted to code and are placed inside the **`_jspService`** method.

Life cycle methods



All the script elements goes into **`_jspService()`** method.

Implicit JSP objects

- Implicit JSP objects are automatically defined variables
- These are defined in the **`_jspService()`** method
- These are not accessible in declarations.

- **`out`**
- **`request`**
- **`response`**
- **`session`**
- **`application`**
- **`exception`**
- **`page`**
- **`pageContext`**
- **`config`**

`out` object

- This object handles text output to browser from the JSP page.
- An instance of **`javax.servlet.jsp.JspWriter`**
- Example:

```
<% out.println("<h1>Hello</h1>"); %>
```

`request` object

- This object can be used to access lots of information contained in the HTTP request header sent from a browser when requesting a JSP page.
- The request object is analogous to request object in servlet.
- Common Examples:

```
<%=request.getParameter("name") %>
```

response object

- This object is used in response header that are sent to the browser from the server along with the current page.
- The **response** object is analogous to request object in servlet.
- Examples:

```
<% response.setContentType("text/html"); %>
<%
response.sendRedirect("http://localhost:8080/a.html");
%>
```

application object

- This object is used to hold references to other objects that may be accessed by multiple users. One such object is database connectivity.
- Represents the **ServletContext** object in servlet.
- Methods: **getAttribute(), setAttribute().**
- Examples:

```
<%
  application.setAttribute("aName","CollPortal");
%>

<%= application.getAttribute("aName")%>
```

`session` object

- When a JSP page is opened in a browser a unique session identity is allocated for each user accessing that application.
- Represents **`HttpSession`** object in Servlet
- Methods:
  - **`getAttribute()`**
  - **`setAttribute()`**
  - **`invalidate()`**
  - **`getId()`**

`exception`, `page` and `config` object

- **`exception`**
  - The exception object is available only to pages which are defined as error pages by setting the value for the attribute “isErrorPage” in the page directive.
- **`page`**
  - This variable is simply a synonym for this and is not very useful in the Java programming language. It was created as a place holder for the time when the scripting language could be something other than Java.
- **`config`**
  - Similar to **`ServletConfig`** class

`pageContext` object

- Used to organize references to all of the objects involved in creating a jsp page.
- JSP engine writes the code that obtains the implicit objects from this object.
- `pageContext` is of type **`PageContext`** class and this class **`inherits from JspContext.`**
  - **`getException()`**
  - **`getPage()`**
  - **`getOut()`**
  - **`getRequest()`**
  - **`getResponse()`**
  - **`getSession()`**
  - **`getServletConfig()`**
  - **`getServletContext()`**

We will cover more methods and ***`JspContext in Tag handler session`***

JSP Directives

- JSP directive are messages that are sent to JSP processor from JSP.
- These are used to set global values for the jsp page and do not produce any output.
- **<%@ *directivename attribute="value"* %>**
  - The page directive
  - The include directive
  - taglib directives → *we will see this later*

Page directive

- A page can contain any no. of page directives, anywhere in the jsp.
- The page directive defines a number of page dependent properties and communicates these to the JSP container.
- However, there can be only one occurrence of any attribute/value pair defined by the page directive in a given jsp. Only exception is the import attributes.
- **<%@ page *attribute="value"* %>**
- XML equivalent:
  - **<jsp:directive.page page_directive_attr_list />**

Attribute List

Attributes	Default
language="*scriptingLanguage*"	**java**
extends="*className*"	
import="*importList*"	
session="true|false"	true
buffer="none|*size*kb"	8kb
autoFlush="true|*false*"	true
isThreadSafe="true|false"	true
info="*info_text*"	
errorPage="*error_url*"	
contentType="ctinfo"	text/html ; charset=ISO-8859-1
isErrorPage="true|false"	false
pageEncoding="peinfo"	

import

- The following packages are implicitly imported in a JSP file:
  - `java.lang.*`
  - `javax.servlet.*`
  - `javax.servlet.jsp.*`
  - `javax.servlet.http.*`

Example: using ‘import’

```
<%@ page language="Java"
   import="java.util.*,java.io.*">
<html> <head> <title> Example 1 </title> </head>
 <body>
<% Date now=new Date();%>
<h2> Server date and time is <%=now %></h2>
</body>
</html>
```

session

- The session attribute determines the availability of session implicit object.
- If the client must join an HTTP session the value must be true. This will return the current or new session.
- If the value is false, the session object or a <jsp:useBean> element with scope=session cannot be used in the JSP file. Either of these usages would cause a translation-time error.
- The default value is true.

Example: Runtime Error handling

JSP page displays 2 text boxes where user is expected to enter numbers. On clicking submit button the result of division of 2 numbers is displayed. In case an error occurs on account of a non-number being entered or if an attempt to divide by 0 occurs the page is redirected to error page.

```
<%@ page contentType="text/html" %>

<%@ page errorPage="error.jsp" %>

<html><head><title>Error Handling</title></head>

<body>

<form method="post" action="divide.jsp">

Number1: <input type="text" name="num1"> divided by

Number2: <input type="text" name="num2">

<input type="submit" value="=">
```

```
Result: <input type="text" name="result" value="
<% String num1=request.getParameter("num1");
   String num2=request.getParameter("num2");
  if (num1!=null && num2!=null){
  int n1=Integer.parseInt(num1);
  int n2=Integer.parseInt(num2);
  int r= n1/n2;
        out.print( r);    } } %>
">
</form></body></html>
```

exception object is automatically to an errorm page

error.jsp

```
<%@ page contentType="text/html" %>
<%@ page isErrorPage="true" %>
<html>
<head><title>Error Handling</title></head>
<body>
Error: <%= exception.getMessage() %>
</body>
</html>
```

Include directive

- This tag is useful to merge the content of an external static file with the contents of a JSP page. Including data is a significant part of the tasks in a JSP page.
- **<%@ include file="filename" %> or**

  <jsp:directive.include file="filename" >
- Example:
- **<%@ include file="agreement.txt" %>**
- **<%@ include file="date.jsp" %>**
- The included file may contain any valid JSP tags. These tags will be translated to Java source and included in the JSP page compilation class.
- Mostly used to including a common page header or footer.