JSP Programming

Introduction

- A Java Server Page is a template for a web page that uses Java code to generate an HTML document dynamically. JSPs are run in a server side component known as a JSP container, which translate them into equivalent Java servlet. For this reason, servlets and JSP pages are intimately related. JSP pages are typically comprised of:
  - Static HTML / XML Component.
  - Special JSP tags.
  - Code written in Java called “scriptlets”.

- It is important to note that the JSP specification is a standard extension defined on top of the servlet API.
- Thus, it leverages all of your experience with servlets. There are significant differences between JSP and servlet technology. Unlike servlets, which is a programmatic technology requiring significant developer expertise, JSP appeals to a much wider audience. It can be used to not only by developers, but also by page designers, who can now play a more direct role in the development life cycle.

The Benefits of Using JSP

- Nobody can borrow the code
  - The JSP code written runs and remains on the web server. So issue of copy source code does not arise at all. All of JSP's functionality is handled before the page is sent to a browser.
- Faster loading pages
  - With JSP, decision can be made about what user wants to see at web server prior the pages being dispatched. So, only the content that the user is interested will be dispatched to the user. There is no extra code and no extra content.

- **No browser compatibility issue:**
  - **JSP pages can run same way in browser. The developer ends up sending standard HTML to a user browser. This largely eliminates scripting issues and cross browser compatibility.**
- **JSP support**
  - **JSP is supported by number of web servers like Apache, Microsoft IIS. Netscape's FastTrack and enterprise web servers and others. Built in support for JSP is available Java Web Server from Sun Microsystem.**
- **Compilation**

- This allows the server to handle JSP pages much faster, because in the older technologies such as CGI require the server to load an interpreter and the target script each time the page is requested.

- JSP elements in HTML/XML pages
  - A JSP page looks a lot like an HTML or XML page. It holds text marked with a collection of tags. While a regular JSP page is not a valid XML page, there is a variant JSP tag syntax that lets the developer use JSP tags within XML documents.

Disadvantages of JSP

- Attractive Java Code:
  - Putting java code within web page is really bad design, but JSP makes it tempting to do just that. avoid this as far as possible. It is done using template using.
- Java code required:
  - To relatively simple things in JSP can actually demand putting java code in a page.
- Simple task are hard to code:
  - Even including page headers and footers is a bit difficult with JSP.

- Difficult looping in JSP:
  - In regular JSP pages looping is difficult. In advance JSP we can use some custom tags for looping.
- Occupies a lot of space:
  - Java Server Pages consume extra hard drive and memory space.

Servlet V/S JSP

- It is true that both servlet and JSP pages have many features in common, and can be used for serving up dynamic web content.
- A servlet is a java class implementing the javax.servlet.Servlet interface that runs within a web or application server’s Servlet engine, servicing client requests forwarded to it through the server.
- A JSP is a slightly more complicated. JSP pages contain a mixture of HTML, java scripts, jsp elements, and jsp directives.
- The elements in a JSP will generally be compiled by the JSP engine into a servlet, but the JSP specification only requires that the JSP page execution entity follow the servlet protocol.

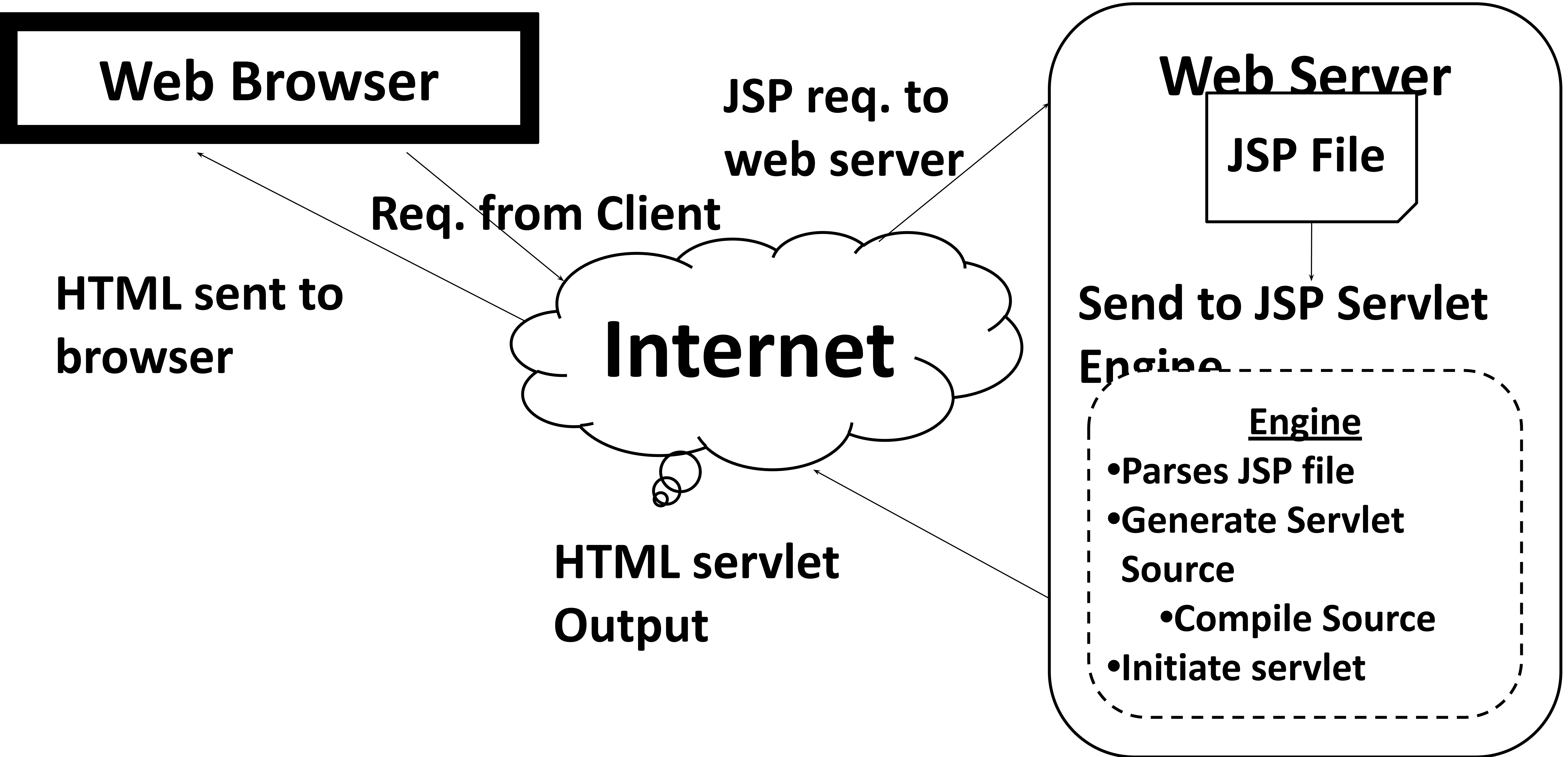
- One the advantage of the JSP over Servlets is that the JSP allows a logical division between what is displayed and the webserver side code specification that dedicates what content fills the page.
- It is easy to modify the look and feel of what is delivered by JSP without having to alter any web server side, java code specification.
- Other advantage of JSP is that they are document – centric. Servlets, on the other hand, look and act like programs.

- A JSP can contain Java program fragments that instantiate and execute java classes, but these occur inside an HTML template file are primarily used to generate dynamic content.
- Some of the JSP functionality can be achieved on the client, using JavaScript.
- While it’s true that anything done with a JSP can also be done with using a servlet, JSPs provides a nice clean separation of the application’s presentation layer from its data manipulating layer.
- JSP’s are simpler to craft than servlets.
- Servlets and JSPs work well together.

JSP Architecture

- The purpose of JSP is to provide a declarative, presentation – centric method of developing servlets.
- As noted before, the JSP specification itself is defined as a standard extension on top the servlet API.
- Typically, JSP pages are subject to a translation phase and a request processing phase. The translation phase is carried out only once, unless the JSP page changes, in which case it is repeated.

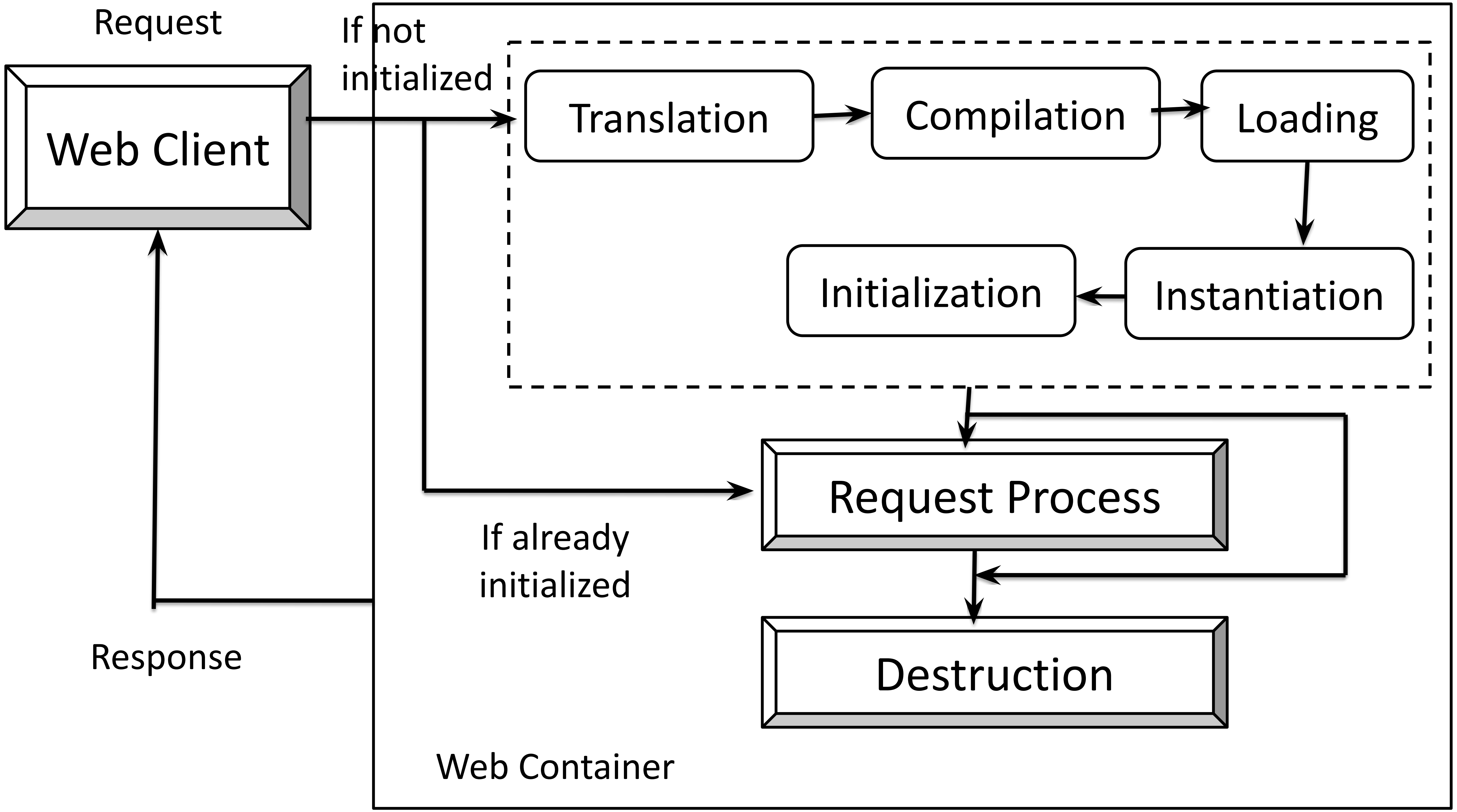
JSP Architecture



- JSP are built by SUN Microsystem servlet technology. JSP tag contains Java code and its file extension is .jsp.
- The JSP engine parses the .jsp and creates a Java servlet source file. Then it compile the source file into a class file, this is done first time only therefore JSP is probably slower when first time it is accessed.
- After this compiled servlet is executed and is therefore returned faster.

JSP Life Cycle

- Java Server Pages technology enables you to write standard HTML pages containing tags that run powerful programs based on Java technology.
- The goal of JSP technology is to support separation of presentation and business logic.
- Web designers can design and update pages without learning the java programming language.
- Programmers for java platform can write codes without dealing with web page design's files are also converted into servlet at last.
- This process is done in the background by web container. **Javax.servlet.jsp.HttpJspPage** interface is used to create servlet from JSP file.


Request
If not initialized
Web Client
Translation
Compilation
Loading
Initialization
Instantiation
Request Process
If already initialized
Destruction
Response
Web Container

JSP Life Cycle

- it shows that if the JSP page is requested first time by web client to web container, then requested JSP code is converted into servlet code into the translation phase.
- After that in compilation phase converted servlet code is compiled any byte code is created in the form of servlet class. In next phases servlet class is loaded into servlet engine and servlet instance is created.
- After creating instance it goes to the Request Process phase. But if JSP page is already initialized then it goes to Request Process phase directly. When JSP servlet class completes its execution it goes into the destruction phase.
- After creating instance of JSP servlet class, following methods are called in turn accordingly which they are available under javax.servlet.jsp.HttpJspPage interface, which is sub – interface of javax.servlet.jsp.JspPage.

- jspInit()
  - When JSP servlet instance is created, jspInit() method will be called. You can use ServletContext object or ServletConfig object to get initial parameters. It is similar to init() of servlet.
  - Syntax: public void jspInit()
- _jspService():
  - This is similar to service() of Servlet. When JSP servlet instance is called, _jspService() is called where request and response object are sent.
  - Syntax: public void _jspService(HttpServletRequest,HttpServletResponse)throws ServletException, IOException
- jspDestroy():
  - This method is called when the JSP servlet instance is destroyed from the web container. It is also similar to destroy() of servlet.
  - Syntax: public void jspDestroy()

JSP Elements

- JSP elements are instruction to JSP container about what code to generate and how it should operate. JSP elements have a special identity to JSP compiler because it starts and ends with special kind of tags.
- Template data code is not compiled by the jsp compiler and also not recognized by the JSP container.
- It is also known as component of the JSP pages.
- There are basically three types of JSP elements which are as follows:
- Directive elements
- Scripting elements
- Action elements

- Directive Elements:
  - The role of the directives is to pass information to the JSP container. Following is the general syntax of directive element.
  - Syntax: <%@ directive {attribute name="value"}%>
  - There are 3 types of directive elements in JSP. They are as follow:
  - Page directive
  - Include directive
  - Taglib directive

Page directive

- Page directive is used to specify attributes for the whole JSP page.
- Following is the general syntax of page directive:
- <%@page [attribute = “value”]%>

Attribute name	Use (description)	Default value
Language	It is used to define the language which is used with the scriptlet elements. Most probably its valid value is java only.	Java
Extends	It defines the fully qualified name of the super class of the jsp page.	
Import	It defines the list of packages which are imported with the JSP page with its fully qualified name.	Following packages are automatically imported with JSP page: Javax.servlet.*; Javax.servlet.http.*; Java.servlet.jsp.*; Java.lang.*;
Session	It defines the boolean indicating value if the jsp page requires HTTP session then its value is true otherwise it become false.	True

Attribute name	Use (description)	Default value
Buffer	It specifies the size of output buffer	8 KB
autoFlush	It also defines the boolean indicating value. If its value is true it automatically flushed the buffer and if false then it throws an exception buffer overflow.	True
isThreadSafe	It also defines the boolean indicating value. If its value is true jsp page handles all the requests simultaneously from the multiple threads and if false generates servlet declares that it implements the SingleThreadModel interface	True
Info	It returns the string message information related to the jsp page using getServletInfo().	

Attribute name	Use (description)	Default value
isErrorPage	It also defines the boolean indicating value. If its value is true jsp page considered as an error page and if false it is a normal jsp page. In JSP exception is implicit variable so it is to create an error page and implements it with other jsp page to handle the exception.	False
errorPage	It defines the URL of the error page if we want to implement to implement the error page with the other jsp pages we can use this directive.	
contentType	It specifies the MIME type and character encoding which used with generated servlet.	<%@ page contentType="text /html" %>
PageEncoding	It defines the character encoding of jsp page itself	ISO – 8859 – 1 & UTF – 8

- Syntax for page directive:
- <%@ page [language = “java”] [extends = “package.class”] [import = “{package.class | package.*}, …..”] [session = “true | false”] [buffer=“none | 8kb | sizekb”] [autoFlush = “true | false”] [isThreadSafe = “true | false”] [info = “text”] [errorPage=“relativeURL”] [contentType = “MIME type [;charset = characterSet }” | “text/html; charset = ISO – 8859 – 1“] [isErrorPage = “true | false”]

  %>

Include directive

- Include directive is used to include the static HTML pages (template) and dynamic jsp pages with the other jsp pages.
- For example, if we want to set the same header for all the jsp pages we can create header.html and can include it with necessary jsp pages.
- In short the include directive can be used to insert a part of the code that is common to multiple pages.
- Syntax:
- <%@ include file=“relative path”%>
- The file attribute is used to specify the name of the file to be included.

Taglib directive

- Taglib directive declares that the JSP file uses custom tags, names the tag library that defines them, and specifies their tag prefix.
- Syntax:

```
<%@ taglib uri=“URIToTagLibrary” prefix=“tagPrefix” %>
```

Scripting elements

- Scripting elements are used to write java code with the jsp file.
- As you know in JSP java code is embedded within HTML code.
- You need to write some java language statements or use java features within the JSP page.
- Here the main questions arises is that how the HTML code and java code are differentiated within JSP file.
- The answer is scripting elements.
- There are three types of scripting elements which are as follows:
  - Scriptlets
  - Declaration
  - Expression

Scriptlets

- Scriptlets are block of java code for the jsp page.
- Scriptlets starts with <% tag and end with %> closing tag.
- As you know JSP is finally converted into servlet code ad then JSP engine adds all processing statements of JSP and processes them under _jspService().
- Syntax:

```
<%
    statement – 1;
    statement – n;
%>
```

Declarations

- Declaration tags are used to declare the variables, methods and instances of the classes within the jsp page.
- As we have discussed with the scriptlets we can declare these things all with the scriptlets also but scriptlet code becomes the part of the _jspService(), whereas declaration code is incorporated into generated source file outside the _jspService().
- Declarations starts with <%! tag and end with %> closing tag.

- Syntax:

```
<%! Statement – 1;
    statement – 2;
    statement – n;
%>
```

Expressions

- Expression element is used to print value of any variable or any valid expression when the jsp page is requested.
- All the expressions are printed automatically by converting values into string values.
- If the result cannot be converted into a string, an error will be raised at translation time.
- An expression starts with <%= and ends with %>
- Syntax:

```
<%= expressions %>
```

Action Elements

- Action elements are high level jsp elements which are used to create, modify and use other objects.
- Syntax of action element’s tags is just like XML syntax.
- Some standard action elements in JSP page are as follows:
- <jsp:param>
- <jsp:include>
- <jsp:forward>
- <jsp:plugin>

- <jsp:param>
  - This element is used to provide the tag/vlaue pairs of information, by including these as sub – attributes of the <jsp:include>, <jsp:forward> and the <jsp:plugin> actions.
  - Syntax:
  - <jsp:param name="pnm" value="pval"/> OR
  - <jsp:param name="pnm" value="pval"></jsp:param>
  - Where, pnm is the name of the parameter name which is being referenced and pval is the value of the parameter.

- <jsp:include>
  - This element is used to include static and dynamic resources of current jsp page. This object is just used to include resources on current JSP page. It can not be used to send response.
  - Syntax:
  - <jsp:include page="jsp page" flush="true/false">
  - <jsp:param name="pnm" value="pval"/>
  - </jsp:include>
  - Page: specifies the resource (JSP/HTML) file which will be included.
  - Flush: an optional parameter used to flush buffer. Specified as true or false.

- <jsp:forward>
  - This element is used to transfer control from current JSP file to another source which may be any valid resource of application. It is obvious to understand that whenever this action element is called, execution of current JSP page is stopped and control is transferred to another specified URL into <jsp:forward>
  - Syntax:
  - <jsp:forward page="destination page"/>

- <jsp:plugin>
  - This element is used to embed an applet and java beans with the jsp page. The tag automatically detects the browser type and inserts the appropriate HTML tag either <embed> or <object> in the output.
  - Syntax:
  - <jsp:plugin type="plugintype" code="class file name" codebase="url">
  - </jsp:plugin>
  - Page: specifies the resource (JSP/HTML) file which will be included.
  - Flush: an optional parameter used to flush buffer. Specified as true or false.

Comments and Template Data

- We can use two types of comments with the JSP page one is for HTML and other for JSP.
- Standard HTML comments have this form: <!– this comment will appear in the client’s browser --> and JSP – specific comments that use this syntax: <% -- this comment will not appear in the client browser --%> JSP comments will not appear in the page output to the client.

- Template Data:
- In JSP page, everything that is not a directive, declaration, scriptlets, expression, action element, or JSP comment is termed template data.
- Usually all the HTML and text in the page in other words, template data is ignored by the JSP translator.
- This data is output to the client as if it had appeared within a static web page.

Scope of the JSP Objects

- Objects that are created as part of a JSP page have a certain lifetime and may or may not be accessible to other components or objects in the web application.
- The lifetime and accessibility of an object is known as scope.
- In some cases, such as with the implicit objects, the scope is set and cannot be changed.
- With other objects, you can set the scope of the object.
- There are four valid scopes:

- Page Scope
- Request Scope
- Session Scope
- Application Scope

Page Scope

- This scope is the most restrictive. With page scope, the object is accessible only within the current jsp page in which it is defined.
- JavaBeans created with page scope and objects created by scriptlets are therad – safe.
- JSP implicit objects out, exception, response, config, pageContext, and page have ‘page’ scope.

Request Scope

- JSP object created using the ‘request’ scope can be accessed from any pages that serve that request.
- This means that the object is available within the page in which it is created, and within pages to which the request is forwarded or included.
- Objects with request scope are thread – safe.
- Only the execution thread for a particular request can access these objects.
- The JSP object will be bound to the request object.
- Implicit object request has the ‘request’ scope.

Session Scope

- Objects with session scope are available to all application components that participate in the client’s session.
- These objects are not thread – safe.
- If multiple request could use the same session object at the same time, you may synchronize access to that object.
- The JSP object that is created using the session scope is bound to the session object.
- Implicit object session has the ‘session’ scope.

Application Scope

- This is the least restrictive scope.
- Objects that are created with application scope are available to the entire application for the life of the application.
- These objects are not thread – safe, and access to them must be synchronized if there is a chance that multiple requests will attempt to change the object at the same time.
- The JSP object is bound to the application object.
- Implicit object application has the ‘application’ scope.

Least restrictive & most visible	Application	Objects are accessible from pages that belongs to the same application
	Session	Objects are accessible from pages that belongs to the same session
	Request	Objects are accessible from pages processing the request where they were created
Most restrictive & least visible	Page	Objects are accessible only within the pages where they were created

Implicit Objects of JSP

- In servlet if we want to use some of the objects like request and response we need to explicitly declare it within the servlet program.
- Like these some other objects which are frequently used with the program are implicit within the JSP which are known as implicit objects of JSP.
- Implicit object means the object which is already created by JSP itself.
- Implicit objects are automatically created in JSP pages and can be used without declaring their objects.
- The reason behind implicit object is, JSP page is also converted into servlet at least. So some objects should be provided by jsp also for programming similar to servlet.

- Implicit objects are used within scriptlet and expression elements.
- Following are some implicit objects which are implicitly available in JSP:
  - Request
  - Response
  - Out
  - Session
  - Config
  - Exception
  - Application

Implicit Object	Super class	Description
request	HttpServletRequest	Provides HTTP request information.
response	HttpServletResponse	Send data back to the client
out	JspWriter	Write the data to the response stream
session	HttpSession	Track information about a user from one request to another
application	ServletContext	Data shared by all JSPs and servlets in the application
pageContext	PageContext	Contains data associated with the whole page
config	ServletConfig	Provides servlet configuration data.
page	Object	Similar with ‘this’ in java
Exception	Throwable	Exceptions not caught by application code.

Handling errors and exceptions with JSP page

- The Exception refers to the error which occurred at runtime.
- As we have seen before that in java we can handle exception very easily through the Exception object.
- In JSP exception is an implicit object with page scope.
- It is an instance of java.lang.Throwable, as you know that throwable class is the super class of all the exception and error classes in the java.
- Here with the JSP we can handle exception with scriptlets, by creating an error page and with deployment descriptor.

Exception handling using scriptlets

- Exception can be handled in a scriptlet in the same way as in java, by using the try – catch block.

Exception handling using page directive

- This is the second way to handle exception by creating an error page.
- To create an error page with JSP we can use isErrorPage=true attribute of page directive.
- After creating an error page we can handle the exception with any JSP page by invoking this error page with it using errorPage="URL of the error page" attribute of the page directive.
- With the use of page directive we can only handle the exception at the page level if we want to handle exception at the application level we can handle it using deployment descriptor.

Exception handling using deployment descriptor

- This is the third way by which we can handle an exception with JSP page.
- Deployment Descriptor is a web.xml file which defines the classes, resources and configuration of the application and how web server uses them to serve web requests.
- It resides in the application under the WEB-INF/ directory.
- If an error page is defined for handling an exception, the request is directed to the error page's URL.
- The web application deployment descriptor uses the <error-page> tag to define web components that handle errors.

Exception handling using deployment descriptor

- We can set deployment descriptor for error handling in two ways, either using exception type or using error code. Deployment descriptor code:
- <error-page>
  - <exception-type>java.lang.Throwable</exception-type>
  - <location>/errorPg.jsp</location>
- </error-page>
- <error-page>
  - <error-code>500</error-code>
  - <location>/errPage.jsp</location>
- </error-page>

JSP Expression Language

- Expression language was first introduced in JSTL 1.0 (JSP Standard Tag Library).
- Before the introduction of JSTL, scriptlets were used to manipulate application data.
- JSTL introduced the concept of an expression language (EL) which simplified the page development by providing standard tag libraries.
- These tag libraries provide support for common, structural tasks, such as: iteration and conditionals, processing XML documents, internationalization and database access using the Structured Query Language (SQL).

- The Expression Language introduced in JSTL 1.0 is now incorporated in JSP specification JSP 2.0.
- The JSTL is a collection of custom tag libraries that encapsulates as simple tags, core functionality common to many JSP applications.
- It eliminates the need to use JSP scriptlets and expressions and uses higher – level syntax for expressions.
- It also implements general – purpose functionality such as iteration and conditionalization, data management formatting manipulation of XML, database access, formatting tags, and SQL tags.
- JSTL 1.0 introduced the concept of the EL but it was constrained to only the JSTL tags.
- With JSP 2.0 you can use the EL with template text and even get programmatic access via javax.servlet.jsp.el.

- EL expressions can be used in two situations:
  - As attribute values in standard and custom actions.
  - In template text, such as HTML or non – JSP elements, in the JSP file – in this situation, the value of the expression in template text is evaluated and inserted into the current output. However, it must be noted that an expression will not be evaluated if the body of the tag is declared to be tag dependent.
- EL provides the ability to use run – time expressions outside JSP scripting elements.
- Scripting elements are the elements in a page that can be used to embed Java code in the JSP file.
- They are commonly used for object manipulation and performing computation that affects the generated content.
- JSP 2.0 adds EL expressions as a scripting element.

- Using EL the above code can be written using a simpler syntax yet achieving the same results as the JSP elements above.
- Another advantage of EL expression is its use in script less JSP pages that do not permit the usage of any of the scripting element forms.
- If we want a clear separation between your presentation and business logic, then we also have the choice to force the page to go script less.
- By enforcing script less pages, the dynamic behavior of JSP pages must be provided through other elements such as JavaBeans, EL, expressions, Custom actions and standard tag libraries.

Syntax of Expression Language

- The JSP expression language allows a page author to access a bean using a simple syntax such as:
- ${expr}
- Logical operators: (<,<=,>,>=,==,lt,le,eq,ge,gt)
- Arithmetic operators: (+,- (binary & unary),*,/,div,%,mod)
- Indexing operators: (.) and ([ ])
- Unary negation operators (!, not)
- Conditional operator: ? :