

Java Server Pages

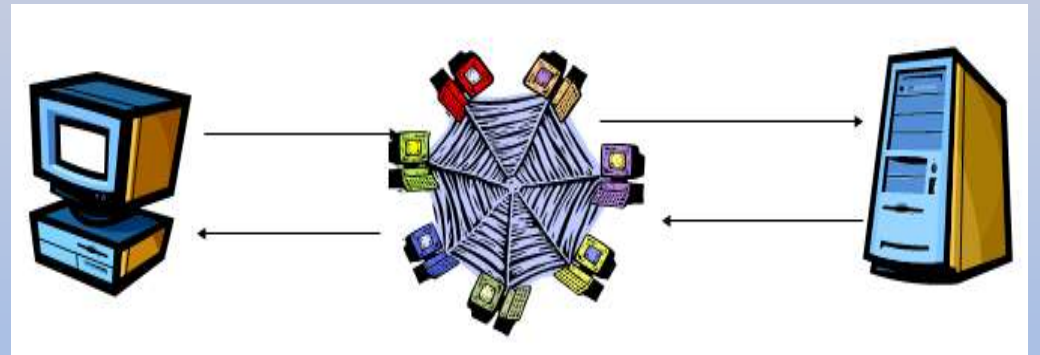
Unit - 4

Topics to be covered

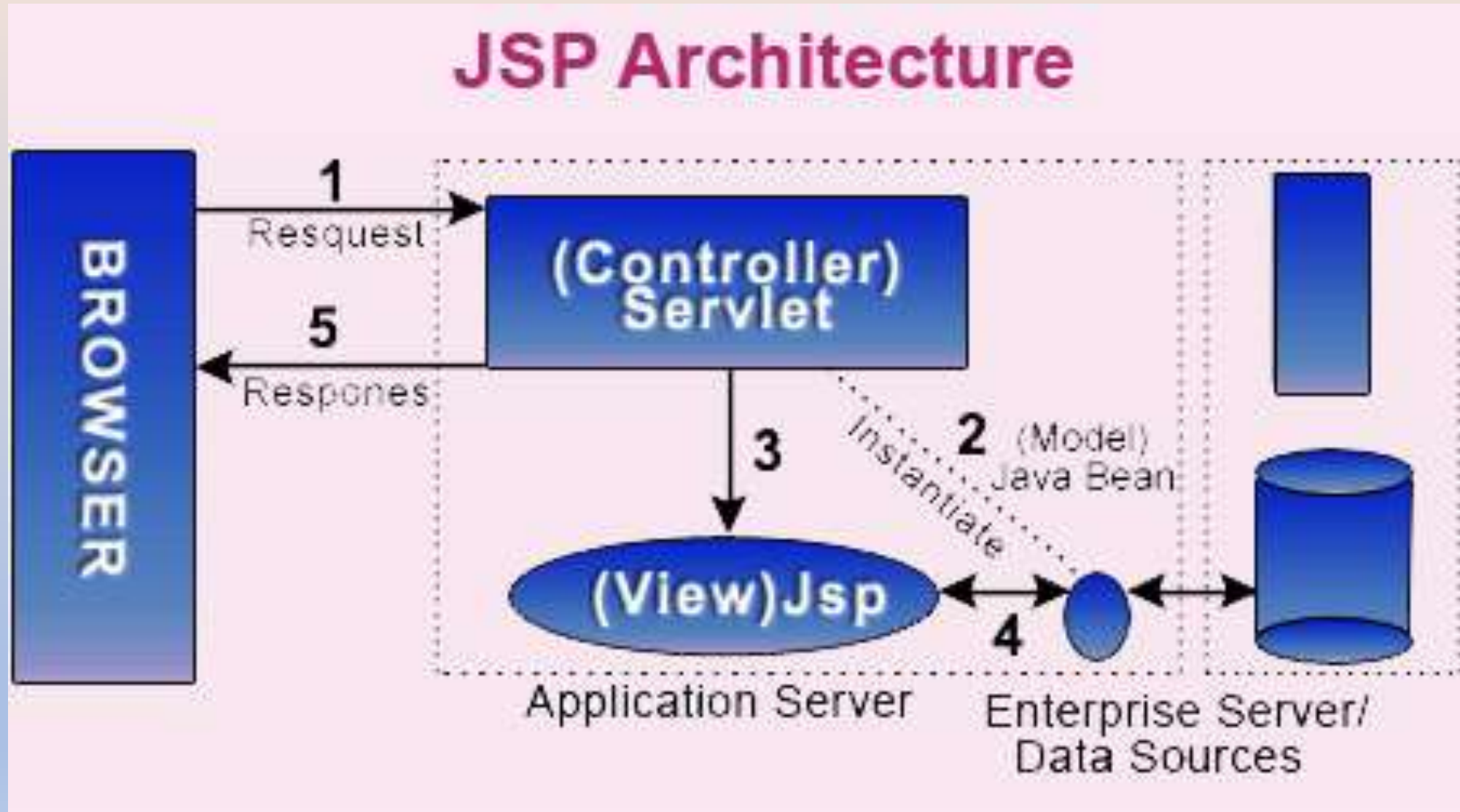
- ❏ Introduction to JSP
- ❏ Architecture of JSP
- ❏ JSP Life Cycle
- ❏ JSP Elements
 - ❏ Directives
 - ❏ Scripting
 - ❏ Action Tags
- ❏ JSP Page with Database Connectivity
- ❏ JSP v/s Servlet

Introduction to JSP

- Web browser send JSP request
- JSP request send via Internet to the web server
- The web server send the JSP file (template pages) to JSP servlet engine
- Parse JSP file
- Generate servlet source code
- Compile servlet to class
- Instantiate servlet
- HTML send back to the browser



JSP Architecture

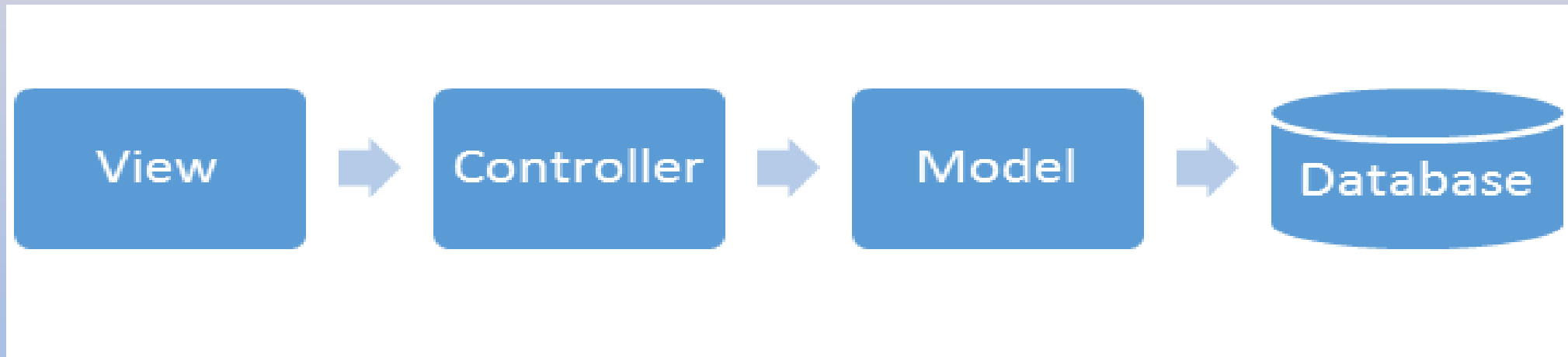


JSP Architecture

❏ M stands for Model

❏ V stands for View

❏ C stands for controller.








JSP Architecture

📁 Model Layer:

- 📄 This is the data layer which consists of the business logic of the system.
- 📄 It consists of all the data of the application
- 📄 It also represents the state of the application.
- 📄 It consists of classes which have the connection to the database.
- 📄 The controller connects with model and fetches the data and sends to the view layer.
- 📄 The model connects with the database as well and stores the data into a database which is connected to it.

JSP Architecture

View Layer:

-  This is a presentation layer.
-  It consists of HTML, JSP, etc. into it.
-  It normally presents the UI of the application.
-  It is used to display the data which is fetched from the controller which in turn fetching data from model layer classes.
-  This view layer shows the data on UI of the application.

JSP Architecture

🔧 Controller Layer:

- 🔧 It acts as an interface between View and Model.
- 🔧 It intercepts all the requests which are coming from the view layer.
- 🔧 It receives the requests from the view layer and processes the requests and does the necessary validation for the request.
- 🔧 This requests is further sent to model layer for data processing, and once the request is processed, it sends back to the controller with required information and displayed accordingly by the view.

JSP v/s ASP

- ❏ JSP and ASP have some basic concepts in common.
 - ❏ They both make use of simple server-side scripting to provide access to Web server information and functionality.
 - ❏ They both have similar styles of delimiting this scripting from a page's content.
- ❏ JSP is a server side technology which helps to create a webpage dynamically using java as the programming language.
- ❏ JSP is a specification from Sun Microsystems. It is an extension to Servlet API.

JSP v/s ASP

Criteria	ASP Technology	JSP Technology
Web Server	Microsoft IIS or Personal Web Server	Any web server, including Apache, Netscape, IIS, Websphere, and Weblogic
Platforms	Microsoft Windows (Accessing other platforms requires third-party ASP porting products.)	Most popular platforms, including the Solaris™ Operating Environment, Microsoft Windows, Mac OS, Linux, and other UNIX® platform implementations
Reusable, Cross-Platform Components	No	JavaBeans, Enterprise JavaBeans, custom JSP tags
Security Against System Crashes	No	YES
Memory Leak Protection	No	YES
Scripting Language used	VBScript, JavaScript	JAVA

Servlet v/s JSP

Like JSP, Servlets are also used for generating dynamic web pages. Below is a comparison between them –

Servlets

- Servlets are Java programs which supports HTML tags .
- They are generally used for developing business layer of an enterprise application.
- Servlets are created and maintained by Java developers.

JSP

- JSP program is a HTML code which supports java statements.
- Which is used for developing presentation layer of an enterprise application
- Frequently used for designing websites by web designers.

Servlet v/s JSP

Servlet	JSP
Servlets are java classes .	JSP is webpage scripting language .
Servlets run faster compared to JSP.	JSP run slower compared to Servlet as it takes compilation time to convert into Java Servlets.
In MVC, servlet act as a controller .	In MVC, jsp act as a view .
servlets are best for use when there is more processing and manipulation involved.	JSP are generally preferred when there is not much processing of data required.
We cannot build custom tags using Servlet.	We can build custom tags using JSP.
To make servlet, java code knowledge must be there.	JSP are compiled to servlet effectively allowing you to produce a servlet by just writing the HTML page , without knowing Java.
In servlet implicit objects are not present.	In JSP implicit objects are present.
We cannot achieve functionality of servlets at client side .	We can achieve functionality of JSP at client side by running JavaScript at client side .
Consists of an html file for static content & java file for dynamic content.	Contains java code embedded directly to in an html page by using special tags.

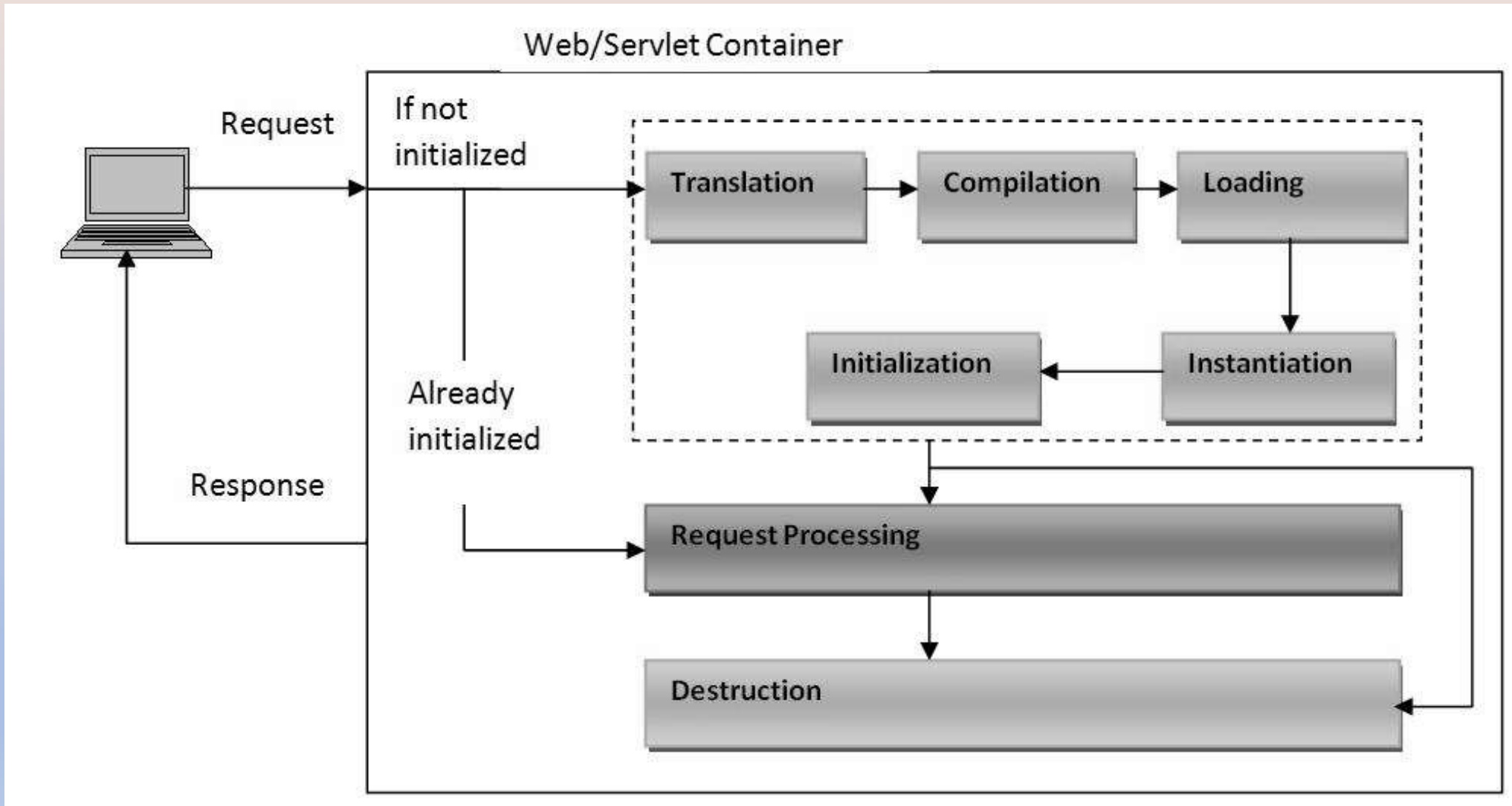
Advantages of JSP

- ❏ JSP has all the advantages that a servlet has, like
 - ❏ Better performance than CGI
 - ❏ Built in session features
 - ❏ it also inherits the features of java technology like – multithreading, exception handling, Database connectivity etc.
- ❏ Web Application Programmers can concentrate on how to process/build the information
- ❏ With the JSP, it is now easy for web designers to show case the information what is needed.
- ❏ JSP Enables the separation of content generation from content presentation. Which makes it more flexible.

Drawback of JSP

- ❏ JSP program is relatively difficult to debug.
- ❏ Database connection is not very easy.
- ❏ It is hard to choose proper servlet engine

JSP Life Cycle



JSP Life Cycle

☞ **Translation:** As stated above whenever container receives request from client, it does translation only when servlet class is older than JSP page otherwise it skips this phase.

☞ **_jspInit():** Then the container compiles the corresponding servlet program and loads the corresponding servlet class, instantiates it, calls the **jspInit() method** to initialize the servlet instance (JSP container will do this only when the instance of servlet file is not running or if it is older than the JSP file.)

```
public void jspInit()
```

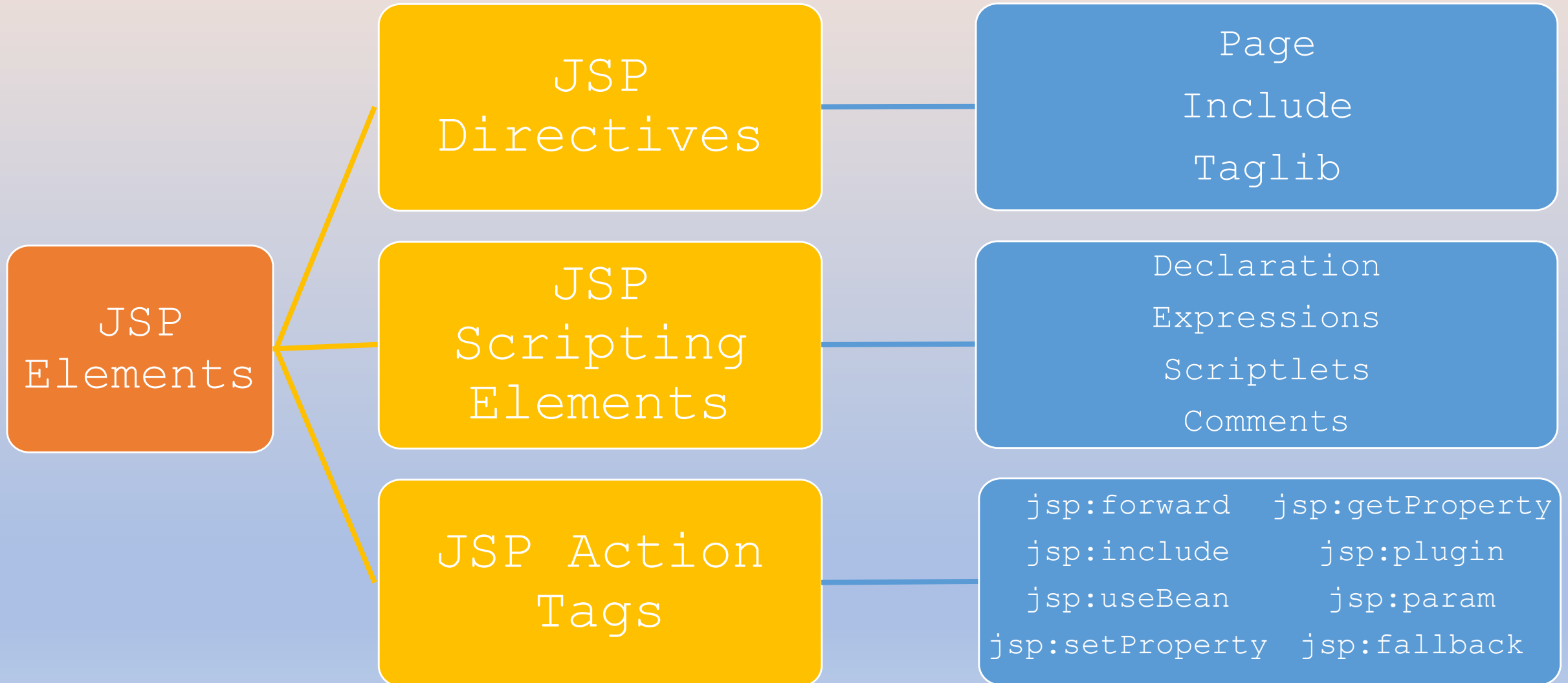
☞ **_jspService():** A new thread is then gets created, which invokes the **_jspService()** method, with a **request** (HttpServletRequest) and **response** (HttpServletResponse) objects as parameters like.

```
public void _jspService (HttpServletRequest request, HttpServletResponse response
```

☞ **jspDestroy():** JSP web container invokes the **jspDestroy()** method to destroy the instance of the servlet class. code will be like

```
public void jspDestory()
```


JSP Elements



JSP Scripting Elements

📄 JSP Scripting element are written inside `<% %>` tags. These code inside `<% %>` tags are processed by the JSP engine during translation of the JSP page. Any other text in the JSP page is considered as HTML code or plain text.

📄 Types of Scripting Elements

Expressions:

`<%= expression %>`

- Evaluated and inserted into the servlets' output. i.e., results in something like `out.println(expression)`.

Scriptlets:

`<% code %>`

- Inserted verbatim into the servlets' `_jspService` method (called by service).

Declarations:

`<%! code %>`

- Inserted verbatim into the body of the servlet class, outside of any existing methods.

Comments:

`<!-- Comment --%>`

- This will not be compiled into the servlet.

JSP Expressions

❏ Expression tag evaluates the expression placed in it, converts the result into String and sends the result back to the client through **response** object. Basically it writes the result to the client (browser).

❏ **Format**

<%= Java Expression %>

❏ **Result**

❏ Expression evaluated, converted to String, and placed into HTML page at the place it occurred in JSP page

❏ That is, expression placed in `_jspService` inside `out.print`

❏ **Examples**

Current time: <%= new java.util.Date() %>

Your hostname: <%= request.getRemoteHost() %>

JSP Expressions

```
<%= 2+4*5 %>
```

```
<% int n1=20;
```

```
    int n2=30;
```

```
    int n3=40;           %>
```

```
<%= n1+n2+n3 %>
```

```
<% application.setAttribute("MyName", "Nilesh"); %>
```

```
<a href="display.jsp">Click here for display</a>
```

```
<%= "This is a String" %><br>
```

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

JSP Scriptlets

- ❏ In a JSP program we may write java code may be of 100 lines, we can achieve this with scriptlets.
- ❏ A scriptlet is a fragment of code written in a scriptlet tag. You can write as many scriptlet tags you want. There is no restriction.
- ❏ The code fragment is enclosed between `<% and %>`.
- ❏ Therefore it is obvious that though the code fragments are written in different tags, they are contiguous. Scriptlets are java code enclosed within **`<% and %>`** tags.
- ❏ JSP container moves the statements enclosed in it to **`_jspService()`** method while generating servlet from JSP. The reason of copying this code to service method is: For each client's request the `_jspService()` method gets invoked, hence the code inside it executes for every request made by client.

JSP Scriptlets

Format:

 `<% Java Code %>`

Result

 Code is inserted verbatim into servlet's `_jspService`

Example

```
<%  
    String queryData = request.getQueryString();  
    out.println("Attached GET data: " + queryData);  
%>
```

`<% response.setContentType("text/plain"); %>`

 Lets take a simple example of how to exactly use it in our programs.

```
<H3> Sample JSP </H3>  
<% myMethod();%>
```

JSP Expression Language (EL)

- ❏ The Expression Language (EL) simplifies the accessibility of data stored in the Java Bean component, and other objects like request, session, application etc.
- ❏ There are many implicit objects, operators and reserve words in EL.
- ❏ It is the newly added feature in JSP technology version 2.0.
- ❏ Syntax
- ❏ `${ expression }`

JSP Implicit Objects

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



Object	Description
request	This is the <code>HttpServletRequest</code> object associated with the request.
response	This is the <code>HttpServletResponse</code> object associated with the response to the client.
out	This is the <code>PrintWriter</code> object used to send output to the client.
session	This is the <code>HttpSession</code> object associated with the request.
application	This is the <code>ServletContext</code> object associated with application context.
config	This is the <code>ServletConfig</code> object associated with the page.
pageContext	This encapsulates use of server-specific features like higher performance <code>JspWriters</code> .
page	This is simply a synonym for this, and is used to call the methods defined by the translated servlet class.

JSP Directives

❏ Directives are messages to the JSP container and do not produce output into the current output stream

❏ Syntax:

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

❏ There are three types of directives:

❏ page

❏ include

❏ taglib

❏ XML form:

```
<jsp:directive.directiveType attribute="value" />
```

JSP Directives: Page

 Defines attributes that apply to an entire JSP page.

`<%@ 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="mimeType [ ;charset=characterSet ]  
[ isErrorPage="true|false" ]
```

`%>`

JSP Directives: Page

❏ `import` attribute: A comma separated list of classes/packages to import

```
<%@ page import="java.util.*, java.io.*" %>
```

❏ `contentType` attribute: Sets the MIME-Type of the resulting document (default is `text/html`)

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

❏ *What is the difference between setting the `contentType` attribute, and writing.*

```
<%response.setContentType("...");%> ?
```

JSP Directives: Page

❏ `session="true|false"` - use a session?

❏ `buffer="sizekb|none"`

❏ Specifies the content-buffer size (**out**)

❏ `autoFlush="true|false"`

❏ Specifies whether the buffer should be flushed when it fills, or throw an exception otherwise

❏ `isELIgnored = "true|false"`

❏ Specifies whether *JSP expression language* is used

JSP Directives: Include

❏ Includes a static file in a JSP file, parsing the file's JSP elements.

❏ Syntax

```
<%@ include file="relativeURL" %>
```

❏ The `<%@ include %>` directive inserts a file of text or code in a JSP file at translation time, when the JSP file is compiled.

❏ `<%@ include %>` process is static. A static include means that the text of the included file is added to the JSP file.

❏ The included file can be:

❏ JSP file,

❏ HTML file,

❏ text file

JSP Directives: Taglib

📄 Defines a tag library and prefix for the custom tags used in the JSP page.

📄 Syntax

```
<%@ taglib uri="URIToTagLibrary" prefix="tagPrefix" %>
```

```
<%@ taglib uri="http://thathost/tags" prefix="public" %>
```

```
<public:loop>
```

```
</public:loop>
```

The `<%@ taglib %>` directive declares that the JSP file uses custom tags, names the tag library that defines them, and specifies their tag prefix.

JSP Action Tags

 `<jsp:include>`

 `<jsp:forward>`

 `<jsp:useBean>`

 `<jsp:setProperty>`

JSP Action Tags: <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 action tag includes the resource at request time so it is **better for dynamic pages** because there might be changes in future.

```
<jsp:include page="relativeURL | <%= expression %>">
```

❏ Example:

```
<h2>this is index page</h2>
```

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

```
<h2>end section of index page</h2>
```


JSP Action Tags: <jsp:forward>

❏ The `jsp:forward` action tag is used to forward the request to another resource it may be jsp, html or another resource.

```
<jsp:forward page="relativeURL | <%= expression %>" />
```

❏ Example:

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

JSP Action Tags: <jsp:useBean>

- ❏ The `jsp:useBean` action tag is used to locate or instantiate a bean class.
- ❏ If bean object of the Bean class is already created, it doesn't create the bean depending on the scope.
- ❏ But if object of bean is not created, it instantiates the bean.
- ❏ Syntax:

```
<jsp:useBean id= "instanceName" scope= "page | request | session | application"  
class= "packageName.className" type= "packageName.className"  
beanName="packageName.className | <%= expression >" >  
</jsp:useBean>
```

JSP Action Tags: <jsp:useBean>

❏ **id:** is used to identify the bean in the specified scope.

❏ **scope:** represents the scope of the bean. It may be page, request, session or application. The default scope is page.

❏ **page:** specifies that you can use this bean within the JSP page. The default scope is page.

❏ **request:** specifies that you can use this bean from any JSP page that processes the same request. It has wider scope than page.

❏ **session:** specifies that you can use this bean from any JSP page in the same session whether processes the same request or not. It has wider scope than request.

❏ **application:** specifies that you can use this bean from any JSP page in the same application. It has wider scope than

JSP Action Tags: <jsp:useBean>

- ❏ **class:** instantiates the specified bean class (i.e. creates an object of the bean class) but it must have no-arg or no constructor and must not be abstract.
- ❏ **type:** provides the bean a data type if the bean already exists in the scope. It is mainly used with class or beanName attribute. If you use it without class or beanName, no bean is instantiated.
- ❏ **beanName:** instantiates the bean using the `java.beans.Beans.instantiate()` method.

JSP Action Tags: <jsp:setProperty>

- ❏ The setProperty and getProperty action tags are used for developing web application with Java Bean.
- ❏ In web development, bean class is mostly used because it is a reusable software component that represents data.
- ❏ The jsp:setProperty action tag sets a property value or values in a bean using the setter method.

❏ Syntax:

```
<jsp:setProperty name="instanceOfBean" property= "*" |  
property="propertyName" param="parameterName" |  
property="propertyName" value="{ string | <%= expression %>}" />
```

❏ Example:

```
<jsp:setProperty name="bean" property="username" value="Abc" />
```

JSP Database Access using its standard taglib

```
<%@ page import = "java.io.*,java.util.*,java.sql.*"%>
<%@ page import = "javax.servlet.http.*,javax.servlet.*" %>
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix = "c"%>
<%@ taglib uri="http://java.sun.com/jsp/jstl/sql" prefix = "sql"%>
<html>
    <head><title>SELECT Operation</title></head>
    <body>
        <sql:setDataSource var = "snapshot" driver = "com.mysql.jdbc.Driver"
url = "jdbc:mysql://localhost/TEST" user = "root" password = "pass123"/>
        <sql:query dataSource = "${snapshot}" var = "result">
            SELECT * from Employees;
        </sql:query>
```

JSP Database Access using its standard taglib

```
<table border = "1" width = "100%">
    <tr>
        <th>Emp ID</th>
        <th>First Name</th>
        <th>Last Name</th>
        <th>Age</th>
    </tr>
    <c:forEach var = "row" items = "${result.rows}">
        <tr>
            <td><c:out value = "${row.id}"/></td>
            <td><c:out value = "${row.first}"/></td>
            <td><c:out value = "${row.last}"/></td>
            <td><c:out value = "${row.age}"/></td>
        </tr>
    </c:forEach>
</table>
</body>
</html>
```