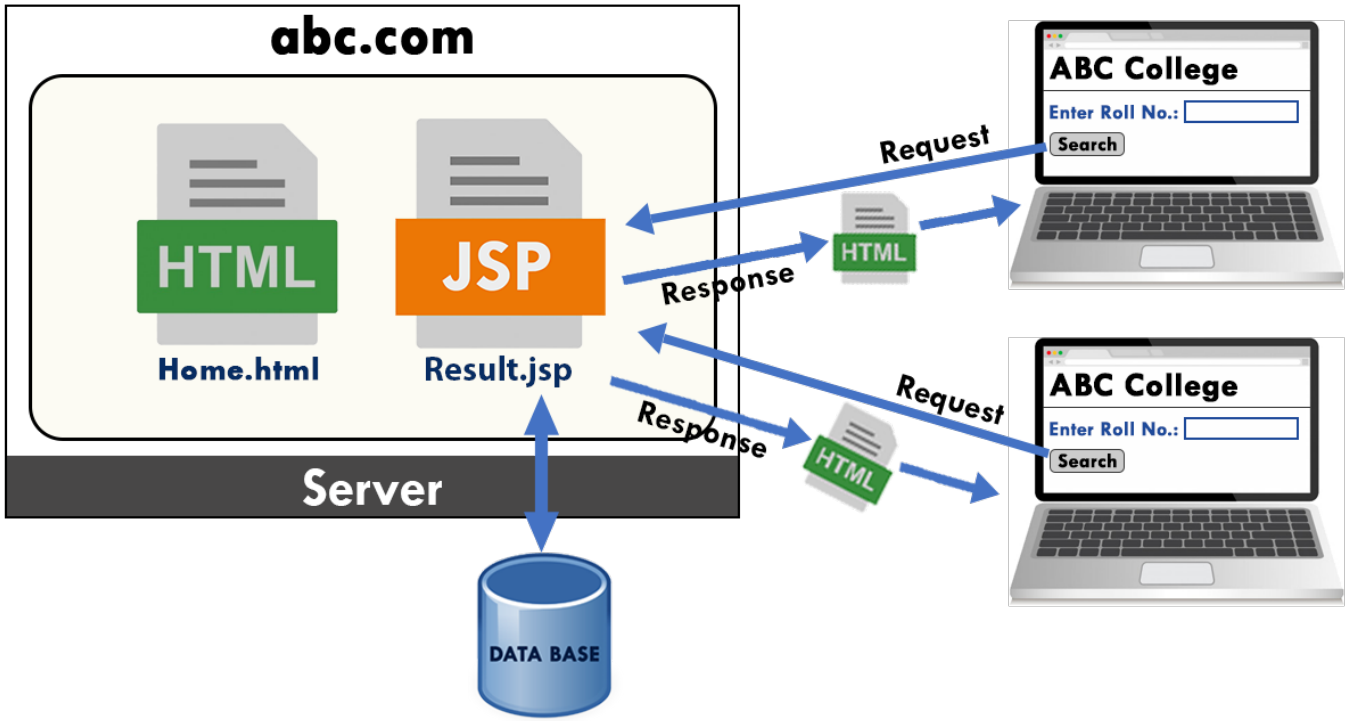
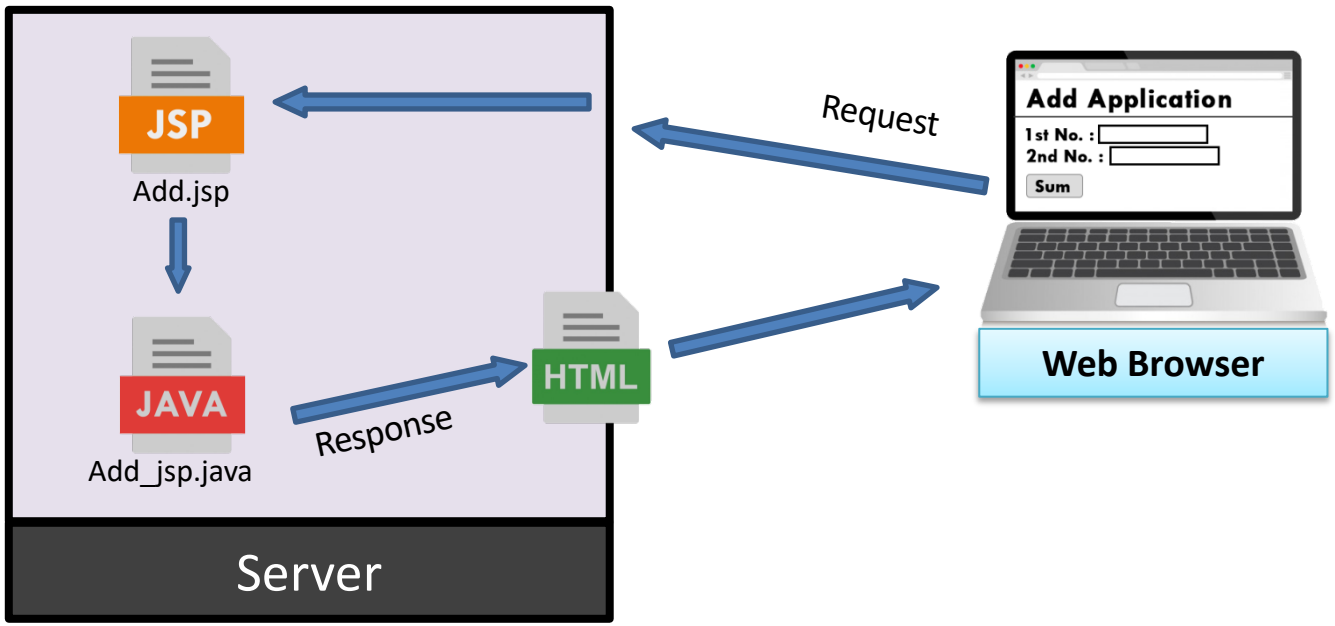
JSP(Jakarta Server Pages)

- JSP(formerly Java Server Pages) allows you to rapidly develop dynamic web pages.
- JSP has all the capabilities of Java Servlet technology.
- JSP may be viewed as a high-level abstraction of Java Servlets.
- JSP is translated into servlet at runtime.



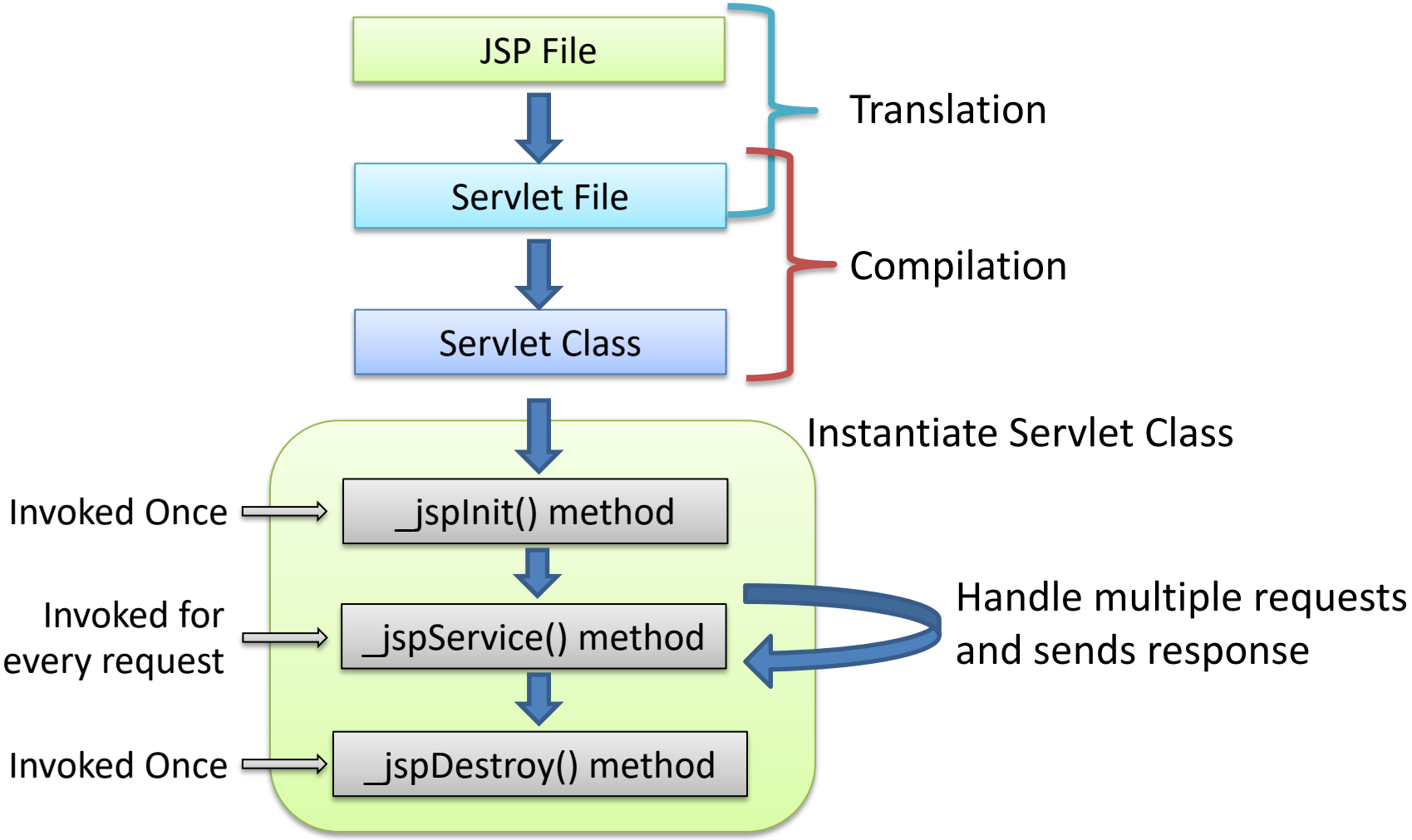
JSP Working

- When a JSP page is called, it will be compiled (by the JSP compiler) into a Java servlet. The servlet engine then loads the servlet class (using a class loader) and executes it to create dynamic HTML to be sent to the browser.
- The next time the page is requested, the JSP engine executes the already-loaded servlet.



JSP Life Cycle

incapp



JSP vs Servlet

incapp

JSP	Servlet
A scripting language to create dynamic web pages.	A server side java program to create dynamic web pages.
JSP page is html page that contains java.	Servlet is a java program that contains html.
Execution is slower than servlet.	Execution is faster than jsp.
Main focus on viewing the information.	Main focus on logic processing.
Coding is easy here.	Coding is not as easy here.

JSP Implicit Objects

Implicit Objects	Type
request	Subclass of *javax.servlet.ServletRequest*
response	Subclass of *javax.servlet.ServletResponse*
pageContext	*javax.servlet.jsp.PageContext*
session	*javax.servlet.http.HttpSession*
application	*javax.servlet.ServletContext*
out	*javax.servlet.jsp.JspWriter*
config	*javax.servlet.ServletConfig*
page	*java.lang.Object*
exception	*java.lang.Throwable*

JSP Tags

Scriptlet Tag	<% code %>
Expression Tag	<%= expression %>
Declaration Tag	<%! declaration %>
Comment Tag	<%-- comment --%>
Directive Tag	<%@ directive %>
Action Tag	<jsp: Action />

Create Add-Square-Counter App.

JSP Directives

Directives are instructions for the JSP engine that are processed when the JSP page is translated into a servlet. There are three types of JSP directives:

include Directive	**<%@include** attributes **%>**
page Directive	**<%@page** attributes **%>**
taglib Directive	**<%@taglib** attributes **%>**

JSP include Directive

The include directive includes a file in a JSP file. Code Reusability is the advantage of this directive.

include Directive	**<%@include file=“ FileName ” %>**

JSP page Directive

The Page Directive defines many attributes that affect the whole page.

page Directive	<%@page attributes %>

JSP page Directive Attributes

incapp

name	purpose	syntax
import	used to import packages.	<%@page **import="**java.io.***"** %>
session	Used to specify JSP page will have session object or not.	<%@page **session="**true|false**"** %> The default value is true.
info	Used to set string for getServletInfo() method.	<%@page **info="**text**"** %>
language	Used to specify language used in the JSP file.	<%@page **language="**java**"** %> The default value is java.
buffer	Used to set buffersize of out object	<%@page **buffer="**20kb|none**"** %> The default value is 8kb.
isELIgnored	Used to specify JSP page will ignore EL or not.	<%@page **isELIgnored="**true|false**"** %> The default value is false.

JSP page Directive Attributes

name	purpose	syntax
errorPage	Used to specify the Error page	<%@page **errorPage="**page name**"** %>
isErrorPage	Used to specify whether the page can displays error or not	<%@page **isErrorPage="**true|false**"** %> The default value is false
extends	Used to specify the super class for generated servlet	<%@page **extends="**class name**"** %>
autoFlush	Used to set output buffer flushed automatically or not	<%@page **autoFlush="**true|false**"** %> The default value is true

JSP taglib Directive

This directive allows the page to use custom tags.

taglib Directive	<%@taglib **uri**="URlToTagLibrary" **prefix**="tagPrefix" %>

Exception Handling

Three ways to handle the Exception in JSP.

- Using **try-catch**.
- Using **errorPage and isErrorPage page directive**.
- Using **error-page element in the 'web.xml' file**.

Fetch and Display the Data in JSP

Login-Logout Project

JSP Actions

JSP provides a bunch of standard action tags that we can use for specific tasks such as working with java bean objects, including other resource, forward the request to other resource etc.

```
<jsp: ActionName  attributes />
```

OR

```
<jsp: ActionName  attributes  >
BODY
</jsp: ActionName >
```

Forward & Param Action

Forward action is used for redirecting the request to the another resource (JSP, Servlet or Html).

Forward Action without Param Action:

```
<jsp: forward  page="URL of the any Servlet, JSP or Static page" />
```

Forward Action with Param Action:

```
<jsp: forward  page="URL of the any Servlet, JSP or Static page" >
        <jsp:param name="ParameterName"
            value="ParameterValue | <%=expression%>" />
</jsp: forward>
```

Include & Param Action

Include action is used to include another resource (JSP, Servlet or Html) to the current JSP page.

Include Action without Param Action:

```
<jsp: include  page="URL of the any Servlet, JSP or Static page" />
```

Include Action with Param Action:

```
<jsp: include  page="URL of the any Servlet, JSP or Static page" >
        <jsp:param name="ParameterName"
            value="ParameterValue | <%=expression%>" />
</jsp: include >
```

useBean, setProperty & getProperty Action

UseBean action is used to use Java Beans in a JSP page.

```
<jsp: useBean id="unique_name_to_identify_bean"
class="class_name" />
```

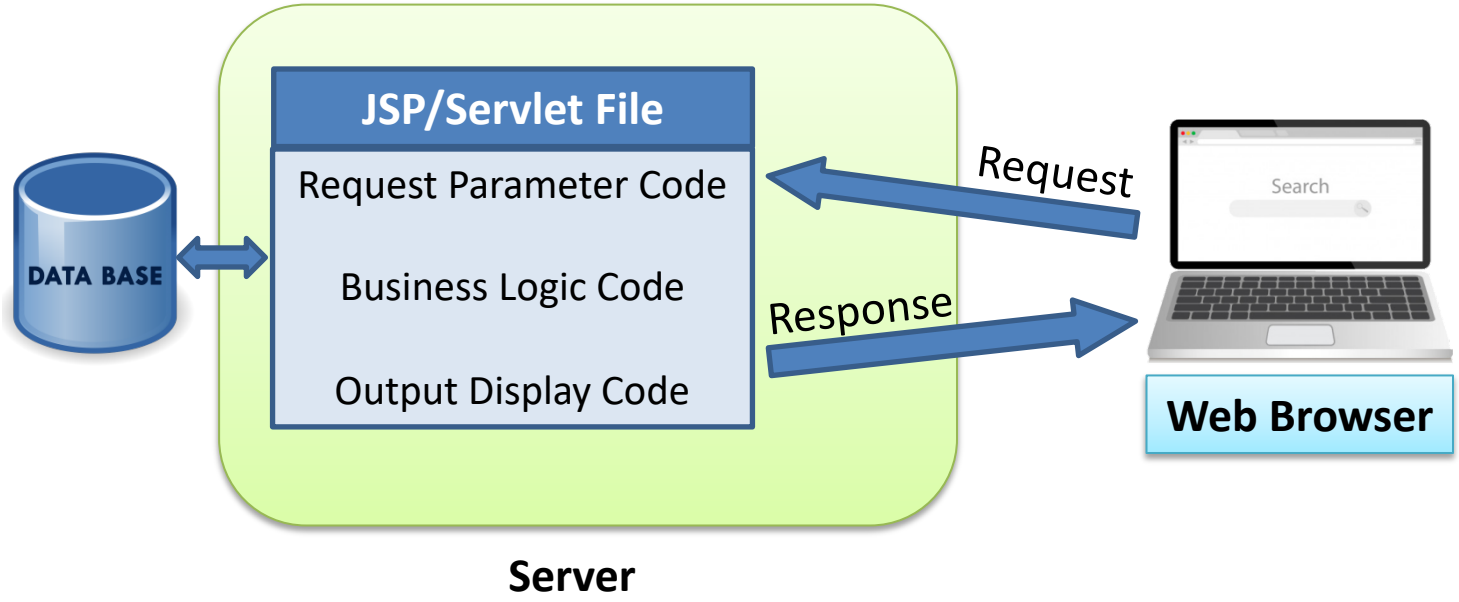
UseBean Action with SetProperty Action:

```
<jsp: useBean id="unique_name_to_identify_bean"
class="class_name" />
<jsp: setProperty name="unique_name_to_identify_bean"
property="property_name" value="property_value" />
```

UseBean Action with GetProperty Action:

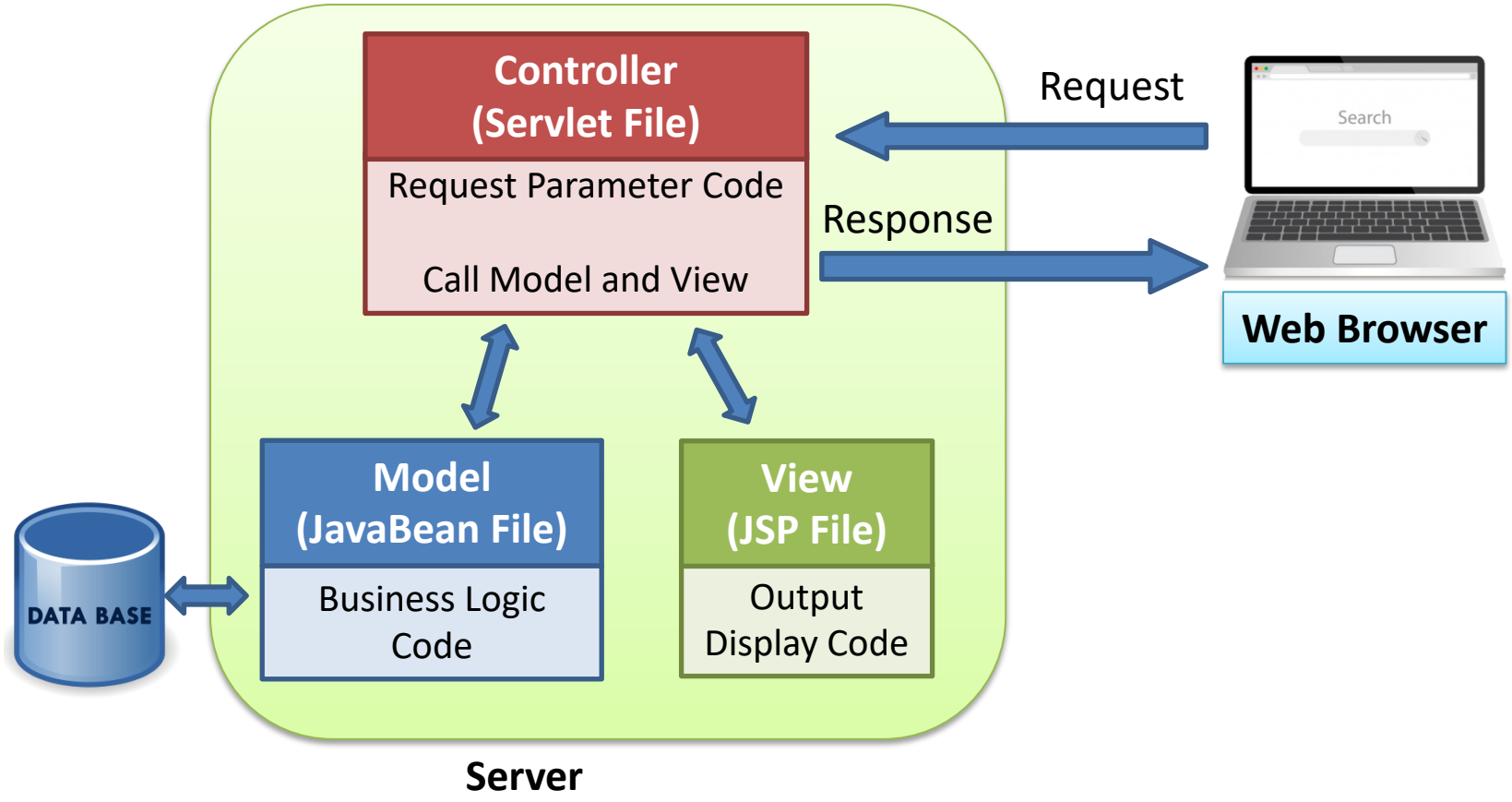
```
<jsp: useBean id="unique_name_to_identify_bean"
class="class_name" />
<jsp: getProperty name="unique_name_to_identify_bean"
property="property_name" />
```

Without MVC


JSP/Servlet File
Request Parameter Code
Business Logic Code
Output Display Code
DATA BASE
Request
Response
Search
Web Browser
Server

With MVC(Model-View-Controller)

incapp

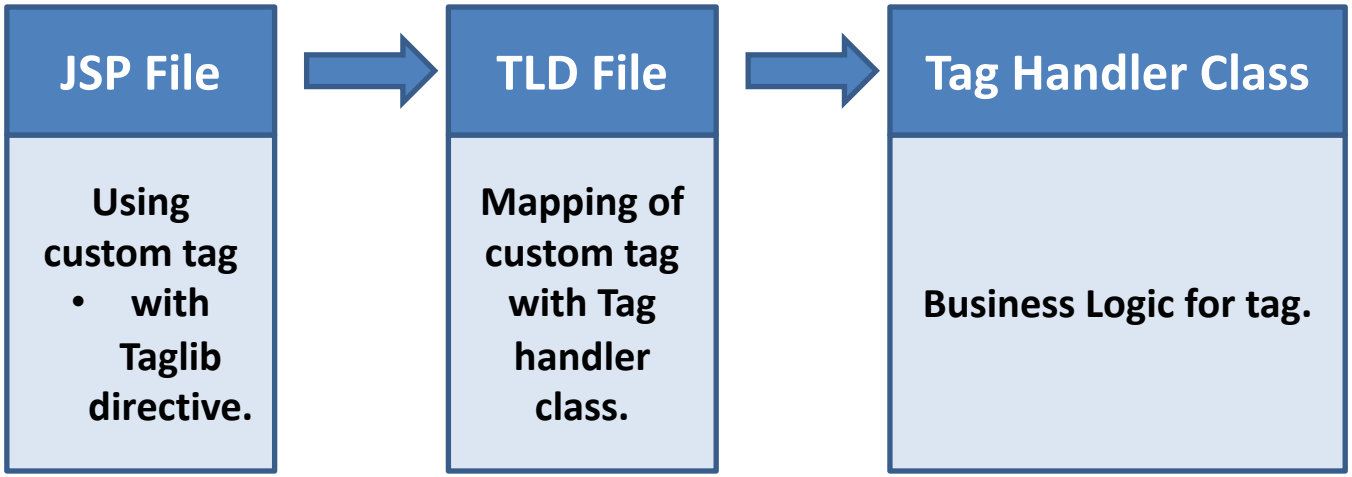


Custom Tag or Tag Extension

JSP custom tags provide a standardized mechanism for separating the presentation and business logic in a dynamic web page, allowing page designers to focus on the presentation while application developers code the backend.

```
<prefix:TagName attributeName= "attributeValue" />
```

3 files required to work with Custom Tag:



Custom Tag to print Hello


```
<my:Demo />
```

Reference to TLD file

- Reference to the TLD file via simple Taglib tag:

<%@ taglib uri=“/WEB-INF/MyTag.tld” prefix=“my” %>

Custom Tag with attribute

```
<my:AddDemo num1="10" num2="20" />
```

Custom Tag with body

```
<my:AddDemo num1="10" num2="20" >
        BODY
</my:AddDemo>
```

JSTL

The Java Standard Tag Library (JSTL) is a collection of useful JSP tags which encapsulates core functionality common to many JSP applications.
The JSTL tags library groups are.

- Core Tags
- XML tags
- Formatting / Internationalization tags
- Database / SQL tags
- Functions

JSTL URIs and Prefix

Library	URI	Prefix
Core	http://java.sun.com/jsp/jstl/core	c
XML Processing	http://java.sun.com/jsp/jstl/xml	x
Formatting	http://java.sun.com/jsp/jstl/fmt	fmt
Database Access	http://java.sun.com/jsp/jstl/sql	sql
Functions	http://java.sun.com/jsp/jstl/functions	fn

Core JSTL Tags

incapp

Tag	Description
<c:out >	Like <%= ... >, but for expressions.
<c:set >	Sets the result of an expression evaluation in a 'scope'
<c:remove >	Removes a scoped variable (from a particular scope, if specified).
<c:catch>	Catches any Throwable that occurs in its body and optionally exposes it.
<c:if>	Simple conditional tag which evalutes its body if the supplied condition is true.
<c:choose>	Simple conditional tag that establishes a context for mutually exclusive conditional operations, marked by <when> and <otherwise>
<c:when>	Subtag of <choose> that includes its body if its condition evalutes to 'true'.
<c:otherwise >	Subtag of <choose> that follows <when> tags and runs only if all of the prior conditions evaluated to 'false'.
<c:import>	Retrieves an absolute or relative URL and exposes its contents to either the page, a String in 'var', or a Reader in 'varReader'.
<c:forEach >	The basic iteration tag, accepting many different collection types and supporting subsetting and other functionality .
<c:forTokens>	Iterates over tokens, separated by the supplied delimeters.
<c:param>	Adds a parameter to a containing 'import' tag's URL.
<c:redirect >	Redirects to a new URL.
<c:url>	Creates a URL with optional query parameters

Xml JSTL Tags

incapp

Tag	Description
x:out	Similar to <%= ... > tag, but for XPath expressions.
x:parse	It is used for parse the XML data specified either in the tag body or an attribute.
x:set	It is used to sets a variable to the value of an XPath expression.
x:choose	It is a conditional tag that establish a context for mutually exclusive conditional operations.
x:when	It is a subtag of that will include its body if the condition evaluated be 'true'.
x:otherwise	It is subtag of that follows tags and runs only if all the prior conditions evaluated be 'false'.

Format JSTL Tags

incapp

Tag	Description
<fmt:formatNumber>	To render numerical value with specific precision or format.
<fmt:parseNumber>	Parses the string representation of a number, currency, or percentage.
<fmt:formatDate>	Formats a date and/or time using the supplied styles and pattern
<fmt:parseDate>	Parses the string representation of a date and/or time
<fmt:bundle>	Loads a resource bundle to be used by its tag body.
<fmt:setLocale>	Stores the given locale in the locale configuration variable.
<fmt:setBundle>	Loads a resource bundle and stores it in the named scoped variable or the bundle configuration variable.
<fmt:timeZone>	Specifies the time zone for any time formatting or parsing actions nested in its body.
<fmt:setTimeZone>	Stores the given time zone in the time zone configuration variable
<fmt:message>	To display an internationalized message.
<fmt:requestEncoding>	Sets the request character encoding

SQL JSTL Tags

incapp

Tag	Description
<sql:setDataSource>	Creates a simple DataSource suitable only for prototyping
<sql:query>	Executes the SQL query defined in its body or through the sql attribute.
<sql:update>	Executes the SQL update defined in its body or through the sql attribute.
<sql:param>	Sets a parameter in an SQL statement to the specified value.
<sql:dateParam>	Sets a parameter in an SQL statement to the specified java.util.Date value.
<sql:transaction >	Provides nested database action elements with a shared Connection, set up to execute all statements as one transaction.

Function JSTL Tags

incapp

Function	Description
fn:contains()	Tests if an input string contains the specified substring.
fn:containsIgnoreCase()	Tests if an input string contains the specified substring in a case insensitive way.
fn:endsWith()	Tests if an input string ends with the specified suffix.
fn:escapeXml()	Escapes characters that could be interpreted as XML markup.
fn:indexOf()	Returns the index withing a string of the first occurrence of a specified substring.
fn:join()	Joins all elements of an array into a string.
fn:length()	Returns the number of items in a collection, or the number of characters in a string.
fn:replace()	Returns a string resulting from replacing in an input string all occurrences with a given string.
fn:startsWith()	Tests if an input string starts with the specified prefix.
fn:substring()	Returns a subset of a string.
fn:substringAfter()	Returns a subset of a string following a specific substring.
fn:substringBefore()	Returns a subset of a string before a specific substring.
fn:toLowerCase()	Converts all of the characters of a string to lower case.
fn:toUpperCase()	Converts all of the characters of a string to upper case.
fn:trim()	Removes white spaces from both ends of a string.

EL(Expression Language)

The EL allows page developers to use simple expressions to dynamically access data from JavaBeans components and other objects like request, session, application etc. It is the newly added feature in JSP technology version 2.0.

Syntax: `${expression}`

EL Predefined Variables

Similar to JSP implicit objects, we have predefined variables in EL.
pageScope: It helps in getting the attribute stored in Page scope.
pageContext: Same as JSP PageContext object.
sessionScope: Fetches attributes from session scope.
requestScope: It used for getting the attributes from request scope.
param: Similar to ServletRequest.getParameter.
applicationScope: Used for getting Application level attributes.
header: It helps in getting HTTP request headers as Strings.
headerValues: Used for fetching all the HTTP request headers.
initParam: It links to context initialization parameters.
paramValues: Same as ServletRequest.getParmeterValues.
cookie: It maps to Cookie object.

Getting values from request parameter

incapp

Syntax: `${ param.ParamaterName }`

or

Syntax: `${ param["ParamaterName"] }`

Getting values from application object

Syntax: **${applicationScope**.AttributeName **}**

or

Syntax: **${applicationScope[**“AttributeName”**]** **}**

Project Work