JAVASERVER PAGES

Learning Goals

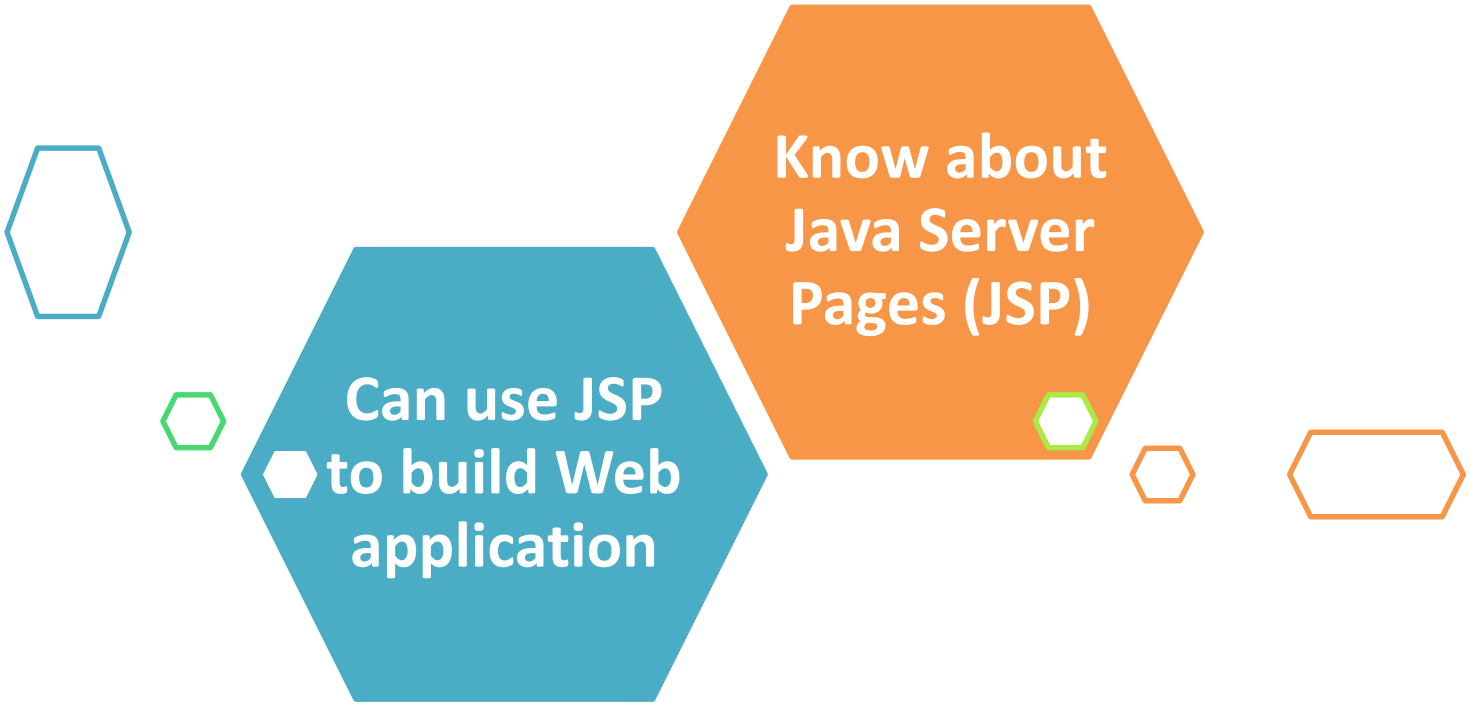


Table of contents

Section 1

JSP INTRODUCTION

JSP Introduction (1/2)

What is JSP?

- ✓ JavaServer Pages (JSP) is a technology for developing web pages that support **dynamic content** which helps developers **insert java code** in HTML.
- ✓ As an **extension** of Servlet technology
- ✓ JSPs **are** essential an **HTML page** with special **JSP tags embedded**.
- ✓ All JSP programs are stored as a **.jsp files**

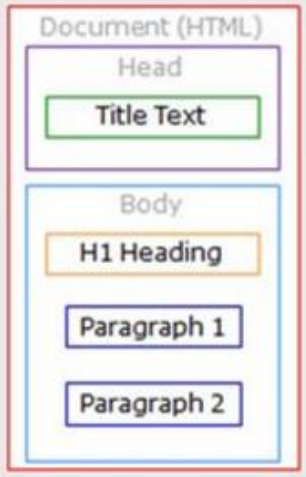
HTML page

```
<HTML>

 <HEAD>
  <TITLE>Title Text</TITLE>
 </HEAD>

 <BODY>
  <H1>H1 Heading</H1>
  <P>Paragraph 1</P>
  <P>Paragraph 2</P>
</BODY>

</HTML>
```



JSP page

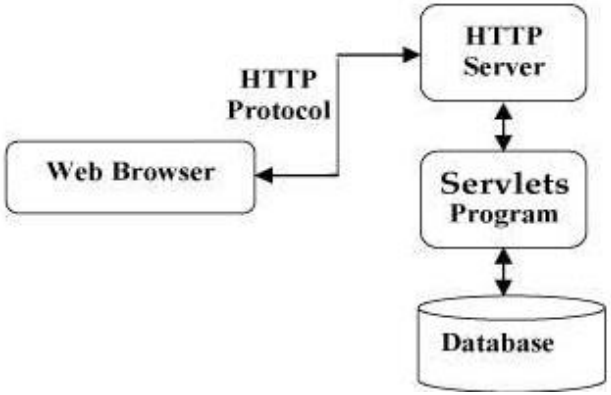
- Scriplet <% …java statements… %>
- Expression <%= …java expression… %>
- Declaration <%! …java variable declarations… %>
- Directive <%@ …special jsp directives… %>

```
<html>
  <head>
    <title>Hello World</head>
  </head>
<body>
<h1>Hello, World</h1>
It's <%= (new java.util.Date()).toString() %>
and all is well.
</body>
</html>
```

JSP Introduction (2/2)

Servlet:

- ✓ Java Servlets are programs that **run on a Web server** and act as a middle layer between a request coming from a Web browser and databases on the HTTP server.



JSP vs Servlet:

Servlets

HTML in Java

```
public void doGet(request, response)
{
PrintWriter out = response.getWriter();
String name =
    request.getParameter(uName);
out.println("<html><body>");
out.println("Username:" + name);
out.println("</body></html>");
}
```

JSP

Java in HTML

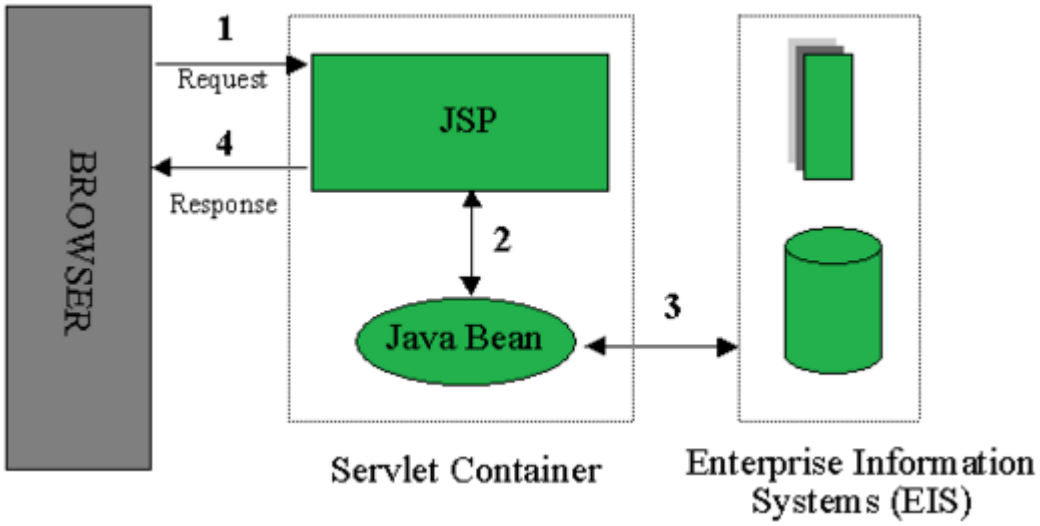
```
<html>
<body>
<% String name =
  request.getParameter(uName); %>

Username: <%= name %>

</body>
</html>
```

Architecture of a JSP Application

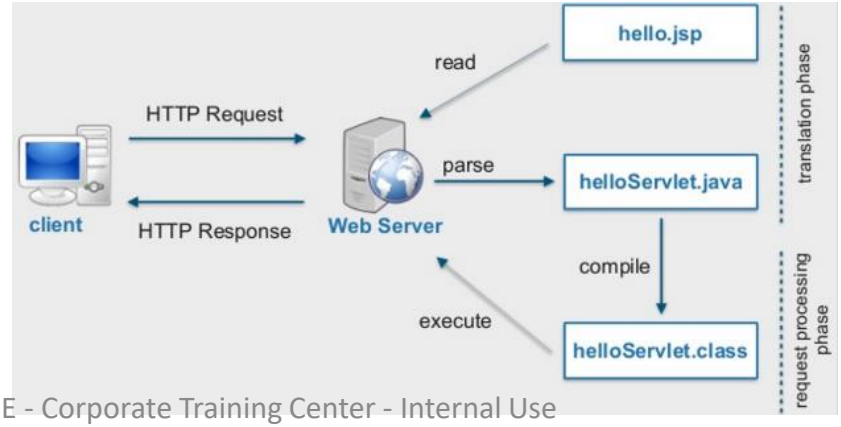
- JSP plays a key role and it is responsible for of processing the request made by client.
  - ✓ Client (Web browser) **makes a request**
  - ✓ JSP then creates a bean object which then fulfills the request and pass the **response** to JSP
  - ✓ JSP then **sends the response** back to client

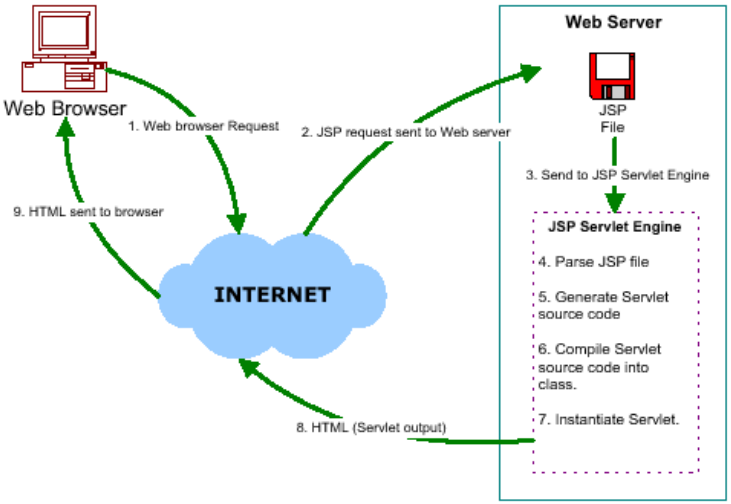


JSP Process

Steps required for a JSP request:

1. The **user goes to web side** made using JSP. The web browser makes the request via the Internet.
2. The JSP request gets **sent** to the Web server.
3. The Web server **recognizes** that the file required is special (.jsp), therefore passes the **JSP file** to the **JSP servlet engine**.
4. If the JSP file has been called the first time, the JSP file is parsed, otherwise goto step 7.
5. The next step is to **generate a special Servlet from the JSP file**. All the HTML required is converted to println statements.
6. The Servlet source code **is compiled** into a class.
7. The servlet **is instantiated**, calling the **init** and **service** methods.
8. **HTML** from the Servlet output is sent via the Internet.
9. HTML results are **displayed** on the user's web browser.





Example

First Example

```
<%@page import="java.util.Date"%>
<%@ include file="index.html"%>
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
<title>The First JSP</title>
</head>
<body>
    <%!String stringHello = "Hello, Welcome to the Frist JSP!";%>
    <%=stringHello%>
    To day is:  <%=new Date()%>
    <jsp:setProperty property="HelloWorld" name="userName" />
    <jsp:forward page="Welcome.jsp"></jsp:forward>

</body>
</html>
```

1 Direction tag

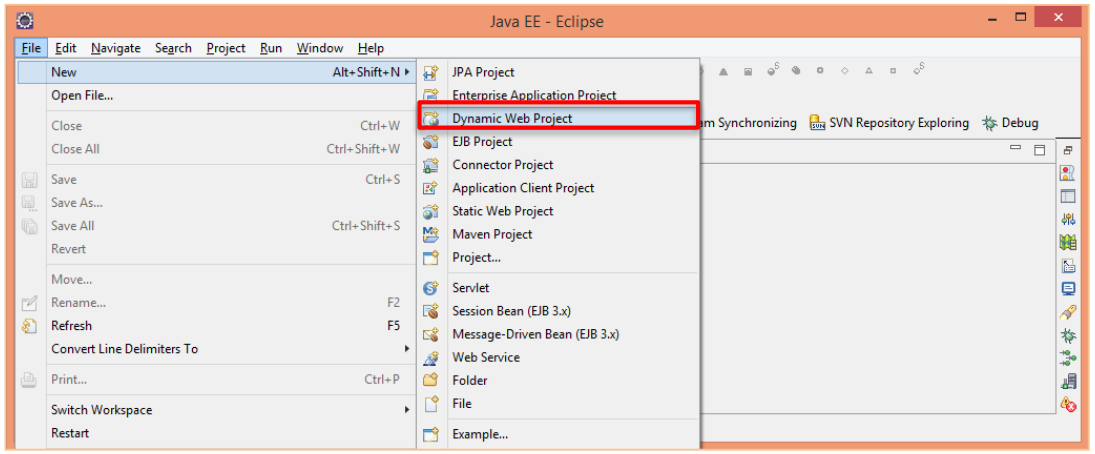
2 Decleration tag

3 Expression tag

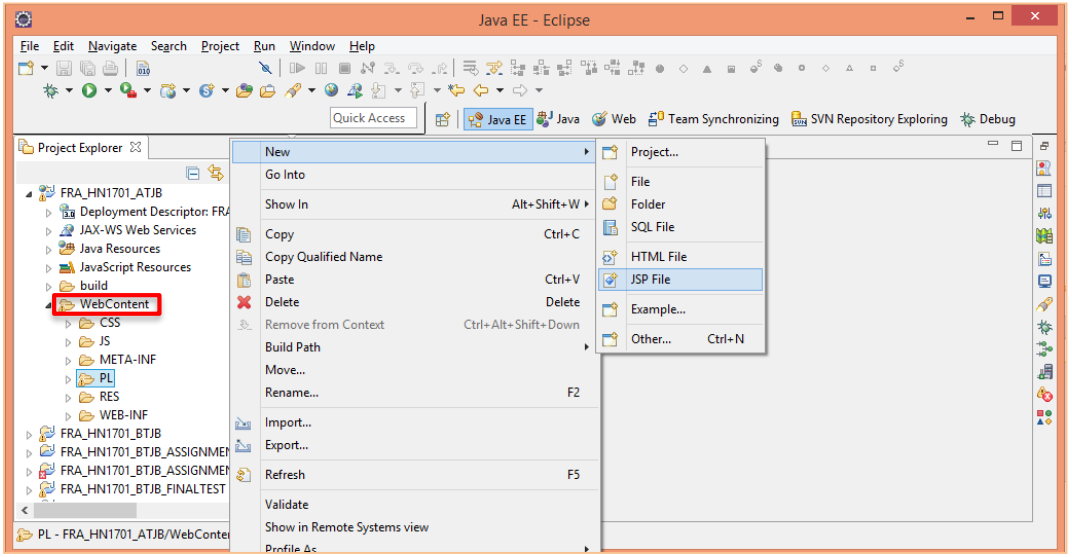
4 Action tag

Practical time

1. Create Dynamic Web Project:



2. Create jsp page in WebContent folder



Practical time

Create a simple jsp page:

FIFA World Cup 2014 News

The biggest scoreline in the history of the **FIFA World Cup** qualifiers - and indeed in the history of international football - was recorded on 11 April 2001, when Australia beat American Samoa 31-0.

This legendary match also brought global renown for *Archie Thompson*, whose 13-goal haul set a new world record, which stands to this day, for an individual player in a single international match.

And though the defeat earned American Samoa ignominy, so inspiring has their subsequent recovery been that it is now the subject of an acclaimed documentary, *'Next Goal Wins'*, showing across the world.

Destinations

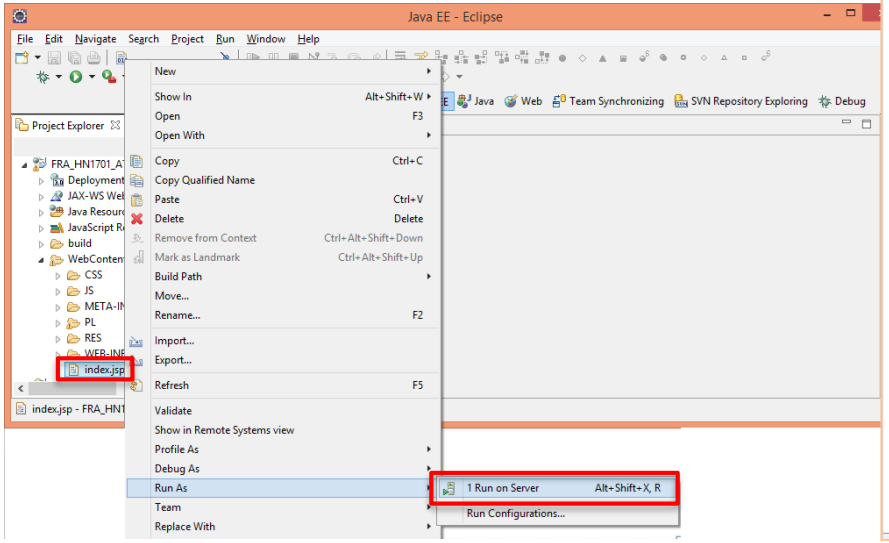
- BELO HORIZONTE, STADIUM : Estadio Mineirao
- BRASILIA, STADIUM : Estadio Nacional
- CUIABA, STADIUM : Arena Pantanal
- CURITIBA, STADIUM : Arena da Baixada
- FORTALEZA, STADIUM : Estadio Castelao
- MANAUS, STADIUM : Arena Amazonia
- NATAL, STADIUM : Estadio das Dunas
- PORTO ALEGRE, STADIUM : Estadio Beira-Rio
- RECIFE, STADIUM : Arena Pernambuco
- RIO DE JANEIRO, STADIUM : Maracanã - Estádio Jornalista Mário Filho
- SALVADOR, STADIUM : Arena Fonte Nova
- SAO PAULO, STADIUM : Arena de Sao Paulo

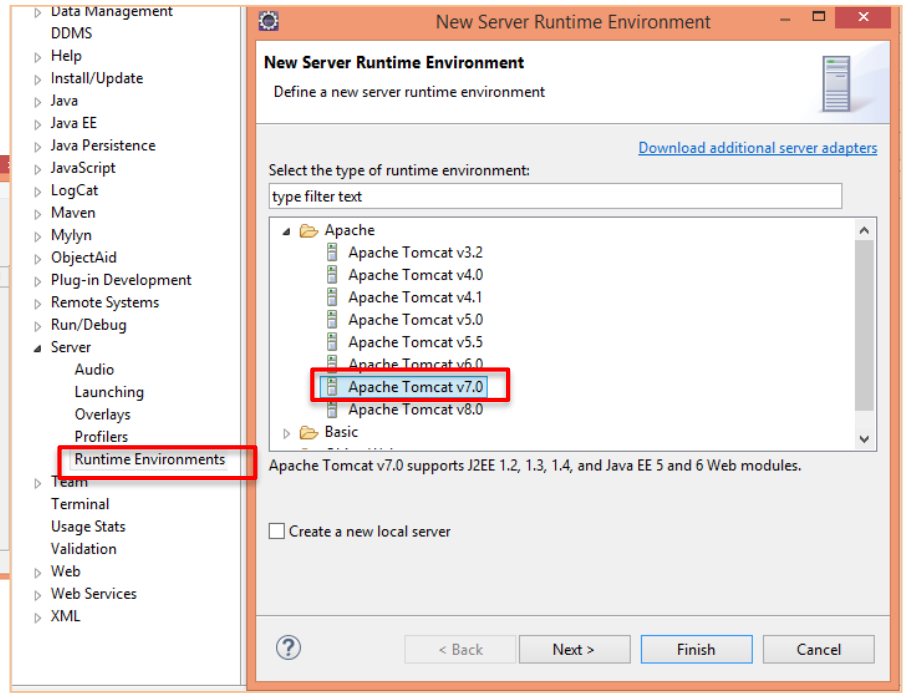
Practical time

3. Setup tomcat server:

- Download and unzip Apache Tomcat .
- Open **Window | Preferences | Server | Installed Runtimes** to create a Tomcat installed runtime.
- Click on **Add...** to open the **New Server Runtime** dialog, then select your runtime under **Apache**

4. Run your code





Section 2

JSP SCRIPTING ELEMENT

Scriptlet

Scriptlet (<% ... %>) - also called "Scripting Elements"

- ✓ Enable programmers to insert **Java code** in JSPs
- ✓ Performs request processing
- ✓ For example, to print a variable:

Code snippet

```
<%
    String username = "alliant";
    out.println(username);
%>
```

JSP Tags

Declaration tag (<%! %>)

- ✓ Allow the developer to declare variables or methods.

Code snippet

```
<%!
    private int counter = 0;
    private String getAccount(int accountNo);
%>
```

Expression tag (<%= %>)

- ✓ Allow the developer to embed any Java expression and is short for out.println()

Code snippet

```
<%!
    <%=new java.util.Date()%>
%>
```

Directive tags (1/3)

- **Directive[chỉ thị] (<%@ directive... %>**
  - ✓ Give special information about the page to the JSP container.
  - ✓ Enable programmers to specify:
    - Page settings (page directive):

```
Ex:<%@page import="java.sql.Statement"%>
```

    - Content to include from other resources (Include directive).

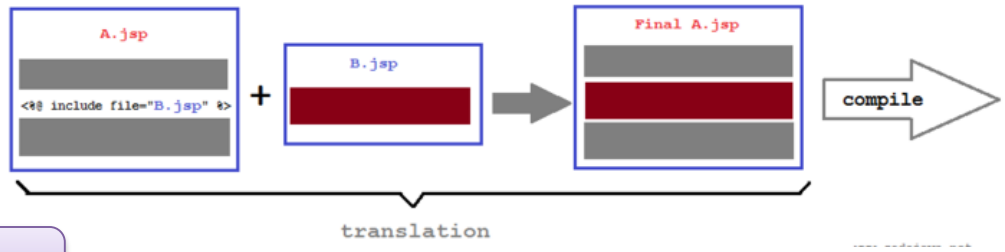
```
Ex:<%@include file="Connection.jsp"%>
```

    - Tag libraries to be used in the page (Tag library).

```
Ex:<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core"%>
```

Directive tags (2/3)

❖ Content to include from other resources (Include directive).

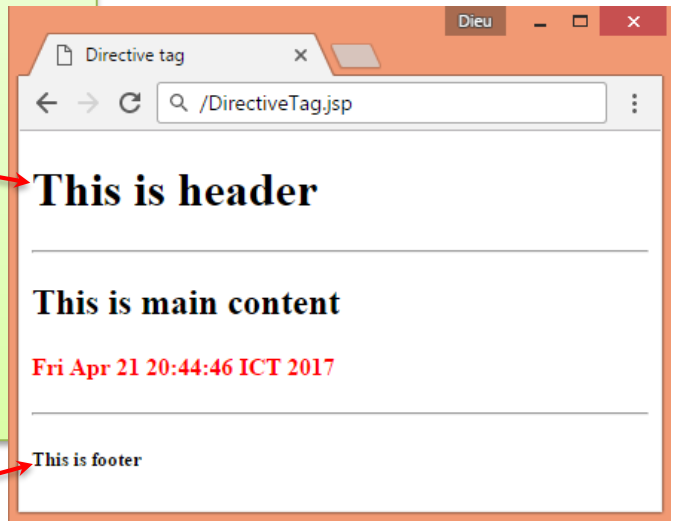


Code snippet

```
<%@page import="java.util.Date"%>
<%@ page     language="java" contentType="text/html; charset=ISO-8859-1"
             pageEncoding="ISO-8859-1"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
      <meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
      <title>Directive tag</title>
</head>
<body>
      <%@include file="Header.jsp"%>

      <hr />
      <h2>This is main content</h2>
      <h4 style="color: red"><%=new Date()%></h4>
      <hr />

      <%@include file="Footer.jsp"%>
</body>
</html>
```



Directive tags (3/3)

Tag libraries to be used in the page (Tag library).

- Add jstl lib

- Defining core tags:

```
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core"%>
```

Code snippet

```
<h3 align="left">TRAINEE FULL INFORMATION</h3>
<table>
      <tr>
            <th>Id</th>
            <th>Trainee Name</th>
            <th>Salary</th>
      </tr>
      <!-- traineeData: a list of trainee -->
      <c:forEach items="${traineeData}" var="trainee">
      <tr>
            <td>${trainee.getId()}</td>
            <td>${trainee.getName()}</td>
            <td>${trainee.getSalary()}</td>
      </tr>
      </c:forEach>
</table>
```

TRAINEE FULL INFORMATION

Id	Trainee Name	Salary
1	John	1000.0
2	Smith	960.0
3	Michel	1170.0

Action tags (1/2)

❖ Action (<%action... %>

- ✓ JSP Actions lets you **perform** some action.
- ✓ Provide access to common tasks performed in a JSP:
  - Including content from other resources.
  - Forwarding the user to another page.
  - Interacting with JavaBeans.
- ✓ Delimited by **<jsp:action>** and **</jsp:action>**.

Action tags (2/2)

Action	Description
<jsp:include>	Dynamically includes another resource in a JSP. As the JSP executes, the referenced resource is included and processed.
<jsp:forward>	Forwards request processing to another JSP, servlet or static page. This action terminates the current JSP's execution.
<jsp:plugin>	Allows a plug-in component to be added to a page in the form of a browser-specific **object** or **embed** HTML element. In the case of a Java applet, this action enables the downloading and installation of the *Java Plug-in*, if it is not already installed on the client computer.
<jsp:param>	Used with the **include**, **forward** and **plugin** actions to specify additional name/value pairs of information for use by these actions.
JavaBean Manipulation	
<jsp:useBean>	Specifies that the JSP uses a JavaBean instance. This action specifies the scope of the bean and assigns it an ID that scripting components can use to manipulate the bean.
<jsp:setProperty>	Sets a property in the specified JavaBean instance. A special feature of this action is automatic matching of request parameters to bean properties of the same name.
<jsp:getProperty>	Gets a property in the specified JavaBean instance and converts the result to a string for output in the response.

Include tag (1/3)

- Include action tag is used for **including another resource** to the **current JSP page.**
  - ✓ The included resource can be a **static page in HTML, JSP page,**
  - ✓ We can also **pass parameters** and **their values** to the resource which we are including.
- **Syntax:**

1) Include along with parameters.

```
<jsp:include page="Relative_URL_Of_Page">
<jsp:param ... />
<jsp:param ... />
...
</jsp:include>
```

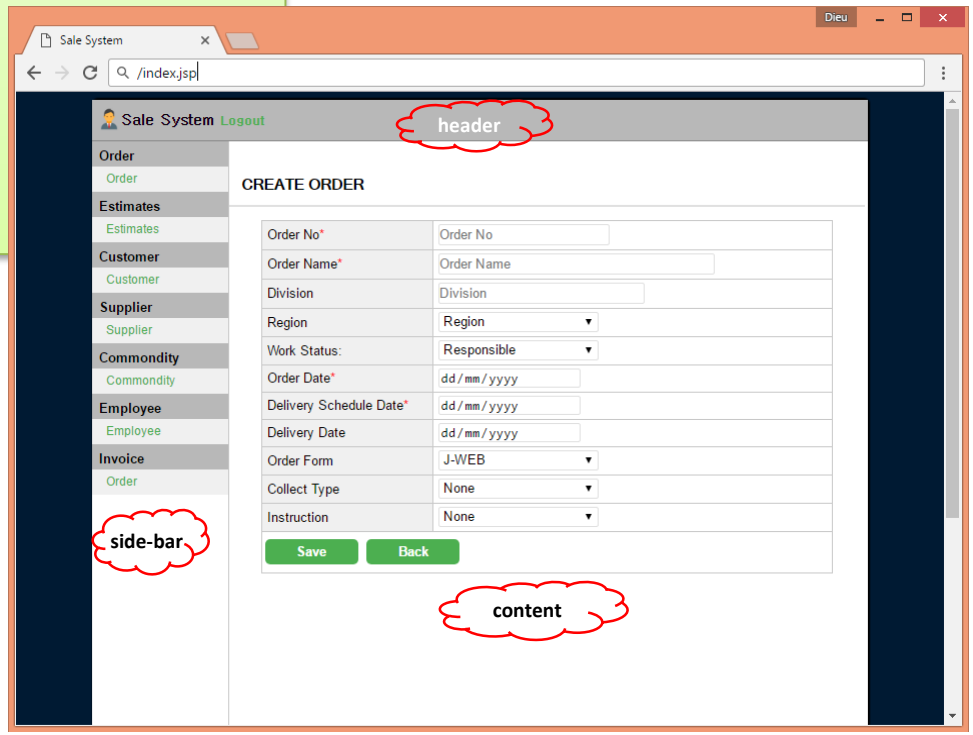
2) Include of another resource without sharing parameters.

```
<jsp:include page="Relative_URL_of_Page" />
```

Include tag (2/3)

Code snippet

```
<html>
<body>
<div class="main_wrap">
    <div class="header">
        <jsp:include page="Header.jsp"></jsp:include>
    </div>
    <div class="container">
        <div class="side-bar">
            <jsp:include page="SideBar.jsp"></jsp:include>
        </div>
        <div class="content">
            <jsp:include page="Order.jsp"></jsp:include>
        </div>
    </div>
</div>
</body>
</html>
```

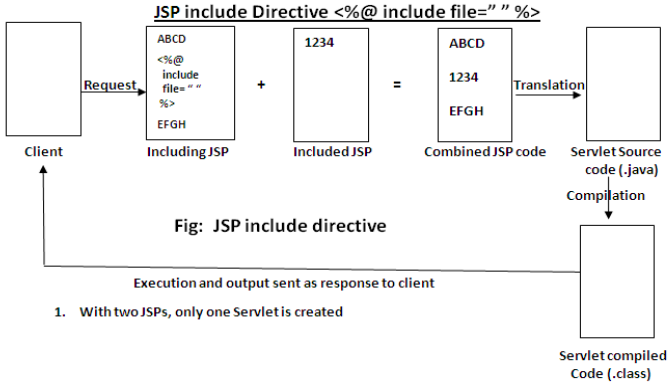


Include tag (3/3)

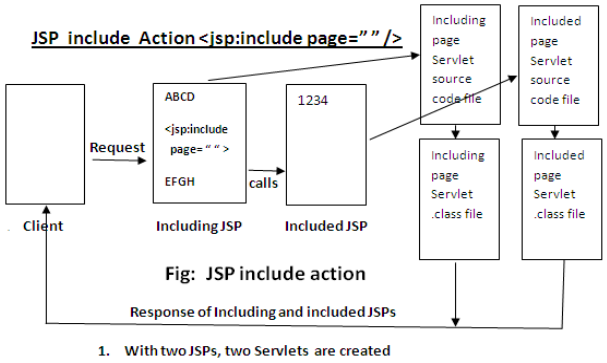
❖ Directives vs Actions:

1. Directives are used during translation phase (**at translate time**) while actions are used during request processing phase (**at request time**).

JSP Include Directive



JSP Include Action



2. If the **included file is changed** but not the JSP which is including it then the changes will reflect only when we use **include action tag.** The changes will not reflect if you are using include.

3. **Syntax** difference.

4. When using include action tag we can also **pass the parameters** to the included page by using param action tag but in case of include directive it's not possible.

Forward tag (1/2)

- Forwards request processing **to another JSP**, servlet or static page. This action **terminates the current JSP's execution.**
  - ✓ Request can be forwarded **with** or **without parameter**
- **Syntax:**

1) Forwarding along with parameters.

```
<jsp:forward page="display.jsp">
<jsp:param ... />
<jsp:param ... />
...
</jsp:forward>
```

2) Forwarding without parameters.

```
<jsp:forward page="Relative_URL_of_Page" />
```

Forward tag (2/2)

Without passing parameters

```
<html>
<head>
      <title>Forward tag</title>
</head>
<body>
      <p align="center">My main JSP page</p>
      <jsp:forward page="OtherPage.jsp"></jsp:forward>
</body>
</html>
```

With passing parameters

```
<html>
<head>
      <title>Forward tag</title>
</head>
<body>
    <p align="center">My main JSP page</p>
    <jsp:forward page="OtherPage.jsp">
           <jsp:param name="name" value="Chaitanya" />
           <jsp:param name="tutorialname" value="Jsp forward action" />
    </jsp:forward>
</body>
</html>
```

Usebean tag (1/3)

- A **JavaBean** is a specially constructed Java class written in the Java. **JavaBeans** component design conventions govern the *properties* of the class and govern the *public methods* that give access to the properties.

Code snippet

```
package ctc.fr.atjb.bean;

public class Account implements Serializable {

    private String emailAddress;
    private String password;

    public String getEmailAddress() {
        return emailAddress;
    }

    public void setEmailAddress(String emailAddress) {
        this.emailAddress = emailAddress;
    }

    public String getPassword() {
        return password;
    }

    public void setPassword(String password) {
        this.password = password;
    }
}
```

Usebean tag (2/3)

Accessing JavaBeans:

- The **useBean** action declares a JavaBean for use in a JSP.
- The full syntax:

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

- **Example:**

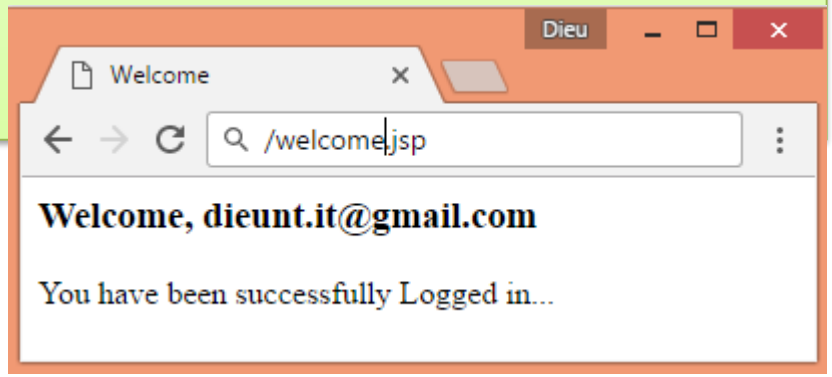
```
<html>
<head>
<title>useBean Example</title>
</head>
<body>
 <jsp:useBean id="date" class="java.util.Date" />
 <p>The date/time is <%= date %>
</body>
</html>
```

```
The date/time is Thu Sep 30 11:18:11 GST 2010
```

Usebean tag (3/3)

Code snippet

```
<jsp:useBean id="account" class="ctc.fr.atjb.bean.Account">
    <jsp:setProperty property="emailAddress" name="account" />
    <jsp:setProperty property="password" name="account" />
    <h3>
        Welcome,
        <jsp:getProperty property="emailAddress" name="account" /><br>
    </h3>
       You have been successfully Logged in...
</jsp:useBean>
```



Section 3

JSP IMPLICIT OBJECTS

Implicit Objects (1/3)

- These objects **are created by JSP Engine** during translation phase (while translating JSP to Servlet)
- All the implicit objects are divided by **four variable scopes:**
  - ✓ **Application:**
    - Objects owned by the container application
    - Any servlet or JSP can manipulate these objects
  - ✓ **Page:**
    - Objects that exist only in page in which they are defined
    - Each page has its own instance of these objects
  - ✓ **Request:**
    - Objects exist for duration of client request
    - Objects go out of scope when response sent to client
  - ✓ **Session:**
    - Objects exist for duration of client's browsing Session
    - Objects go out of scope when client terminates Session or when Session timeout occurs

Implicit Objects (2/3)

Implicit Object	Description
Application Scope	
`application`	This **`javax.servlet.ServletContext`** object represents the container in which the JSP executes.
Page Scope	
`config`	This **`javax.servlet.ServletConfig`** object represents the JSP configuration options. As with servlets, configuration options can be specified in a Web application descriptor.
`exception`	This **`java.lang.Throwable`** object represents the exception that is passed to the JSP error page. This object is available only in a JSP error page.
`out`	This **`javax.servlet.jsp.JspWriter`** object writes text as part of the response to a request. This object is used implicitly with JSP expressions and actions that insert string content in a response.
`page`	This **`java.lang.Object`** object represents the **`this`** reference for the current JSP instance.
`pageContext`	This **`javax.servlet.jsp.PageContext`** object hides the implementation details of the underlying servlet and JSP container and provides JSP programmers with access to the implicit objects discussed in this table.

JSP implicit objects (part 1 of 2).

Implicit Objects (3/3)

Implicit Object	Description
`response`	This object represents the response to the client. The object normally is an instance of a class that implements **`HttpServletResponse`** (package **`javax.servlet.http`**). If a protocol other than HTTP is used, this object is an instance of a class that implements **`javax.servlet.ServletResponse`**.
Request Scope	
`request`	This object represents the client request. The object normally is an instance of a class that implements **`HttpServletRequest`** (package **`javax.servlet.http`**). If a protocol other than HTTP is used, this object is an instance of a subclass of **`javax.servlet.ServletRequest`**.
Session Scope	
`session`	This **`javax.servlet.http.HttpSession`** object represents the client session information if such a session has been created. This object is available only in pages that participate in a session.

JSP implicit objects (part 2 of 2).

Implicit Objects

Request

- 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.
- **Methods:**
  - ✓ **getParameter(String name):** get the value of a request's parameter;
  - ✓ **getParameterNames():** It returns enumeration of all the parameter names associated to the

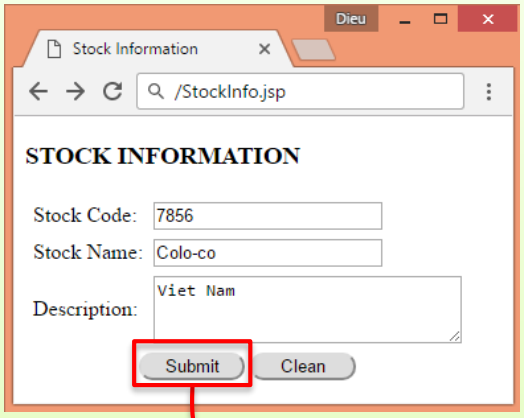
    request: Enumeration enum = request.getParameterNames();
  - ✓ **getParameterValues(String name) :** It returns the array of parameter values.

    String[] allpasswords = request.getParameterValues("password");
  - ✓ **getAttribute(String name) –** Used to get the attribute value. request.getAttribute("admin") would give you the value of attribute admin.
  - ✓ **setAttribute(String,Object) –** It assigns an object's value to the attribute.
  - ✓ **getCookies() –** It returns an array of cookie objects received from the client.

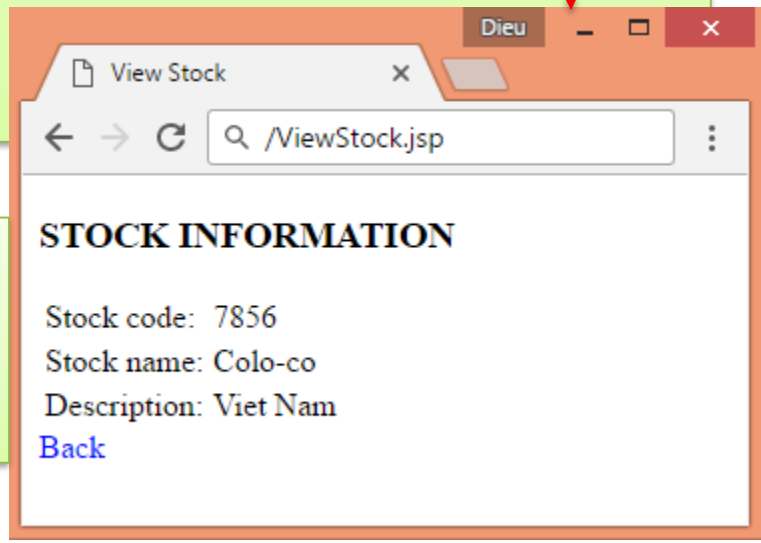
Implicit Objects

Request

```
<h3>STOCK INFORMATION</h3>
<form action="ViewStock.jsp" method="post">
	<table cellspacing="5px">
		<tr>
			<td>Stock Code:</td>
			<td><input type="text" size="20px" name="stockCode"></td>
		</tr>
		<tr>
			<td>Stock Name:</td>
			<td><input type="text" size="20px" name="stockName"></td>
		</tr>
		<tr>
			<td>Description:</td>
			<td><textarea rows="3px" cols="30" name="des"></textarea></td>
		</tr>
		<tr>
			<td colspan="2" align="center">
				<input type="submit" value="Submit" style="border-radius: 10px; width: 80px">
				<input type="reset" value="Clean" style="border-radius: 10px; width: 80px">
			</td>
		</tr>
	</table>
</form>
```




```
<!--ViewStock.jsp  -->
<%
    String stockCode = request.getParameter("stockCode");
    String stockName = request.getParameter("stockName");
    String description = request.getParameter("des");
%>
```



Implicit Objects

Response

- In JSP, **response** is an implicit object of type **HttpServletResponse**.
  - ✓ The instance of HttpServletResponse **is 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, send error etc.
- **Methods:**
  - ✓ void **sendRedirect**(String address) – It redirects the control **to a new JSP page**. For e.g.
    - Example: `response.sendRedirect("http://beginnersbook.com");`
  - ✓ void **addCookie**(Cookie cookie) – This method adds a cookie to the response. The below statements would add 2 Cookies Author and Siteinfo to the response.
    - Example: response.addCookie(Cookie Author); response.addCookie(Cookie Siteinfo);
  - ✓ void **sendError**(int status_code, String message) – It is used to send error response with a code and an error message.
    - For example: response.sendError(404, "Page not found error");

Implicit Objects

Session

- This **javax.servlet.http.HttpSession** object represents the **client session** information if such a session has been created.
- Methods:
  - ✓ **setAttribute**(String, object): This method is used to save an object in session by assigning a unique string to the object.
  - ✓ **getAttribute**(String name): The object stored by setAttribute method is fetched from session using getAttribute method.
  - ✓ **removeAttribute**(String name): The objects which are stored in session can be removed from session using this method. Pass the unique string identifier as removeAttribute's method.
  - ✓ **getAttributeNames**: It returns all the objects stored in session. Basically, it results in an enumeration of objects.

Database Connectivity

- **Review:**
  - ✓ Provides a programming interface that is used to request a connection be*application* and *database*
  - ✓ JDBC API executes SQL statements (in the Java code to retrieve data) and results through a single API
- **Five steps to work with database:**
  - ✓ **Load driver:**

```
Class.forName
("com.microsoft.sqlserver.jdbc.SQLServerDriver");
```

  - ✓ **Create Connection object**

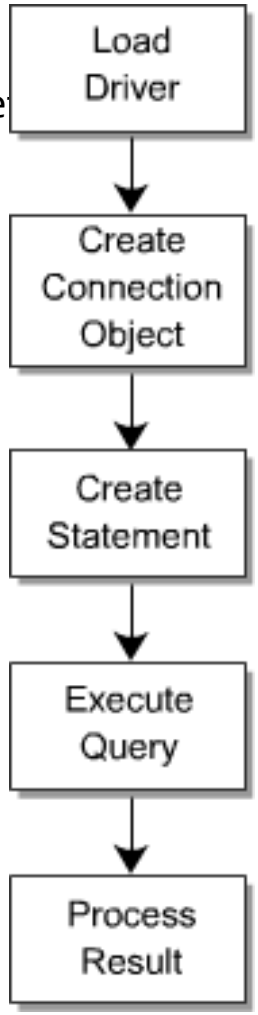
```
String connectionUrl ="jdbc:sqlserver:
//1FWADIEUNT1-LT:1433; databaseName=MY_DB";
```

Connection **con**= DriverManager.getConnection(
connectionUrl, "**userid**", "**pwd**");

  - ✓ **Create Statement:** Statement stat = con.createStatement() ;
  - ✓ **Execute Query:**

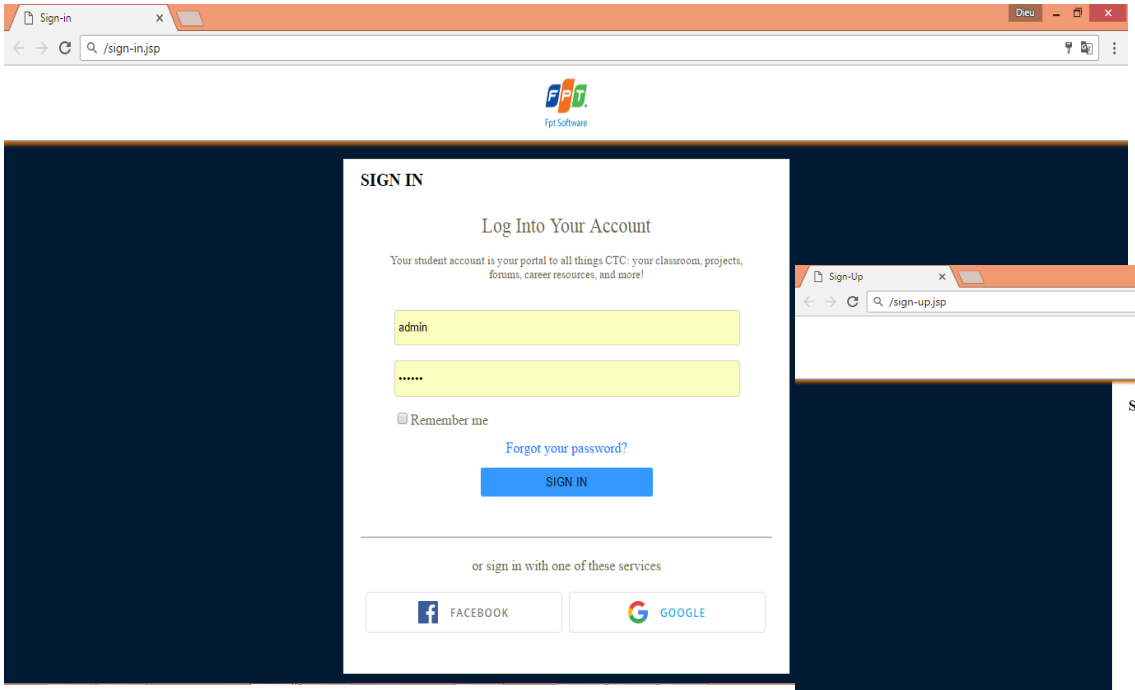
ResultSet resultset = stat.executeQuery("Select * from table_name");

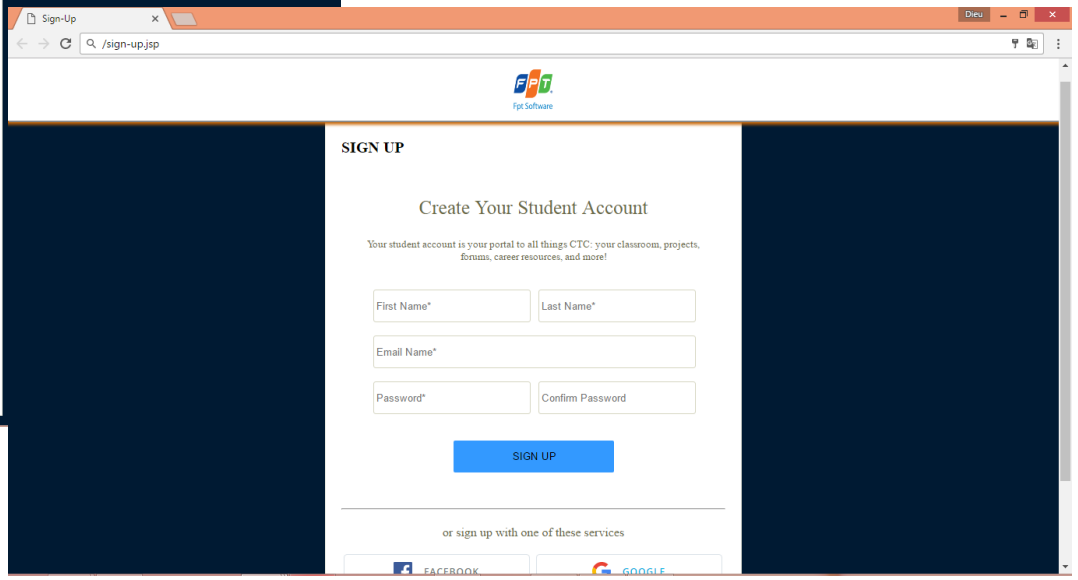
  - ✓ **Process Results:** `next()` method of the ResultSet object is used to process the results from the database.



Practical time

- Sử dụng JSP Scripting Element, JavaBean, Implicit Objects xử lý màn hình Login và Màn hình đăng ký đã thiết kế trong bài trước:





Summary

- JSP Introduction
- JSP Scripting Element
- Implicit Objects

Thank you