Module 5: JSP Basics

Thanisa Numnonda
Faculty of Information Technology
King Mongkut's Institute of Technology Ladkrabang

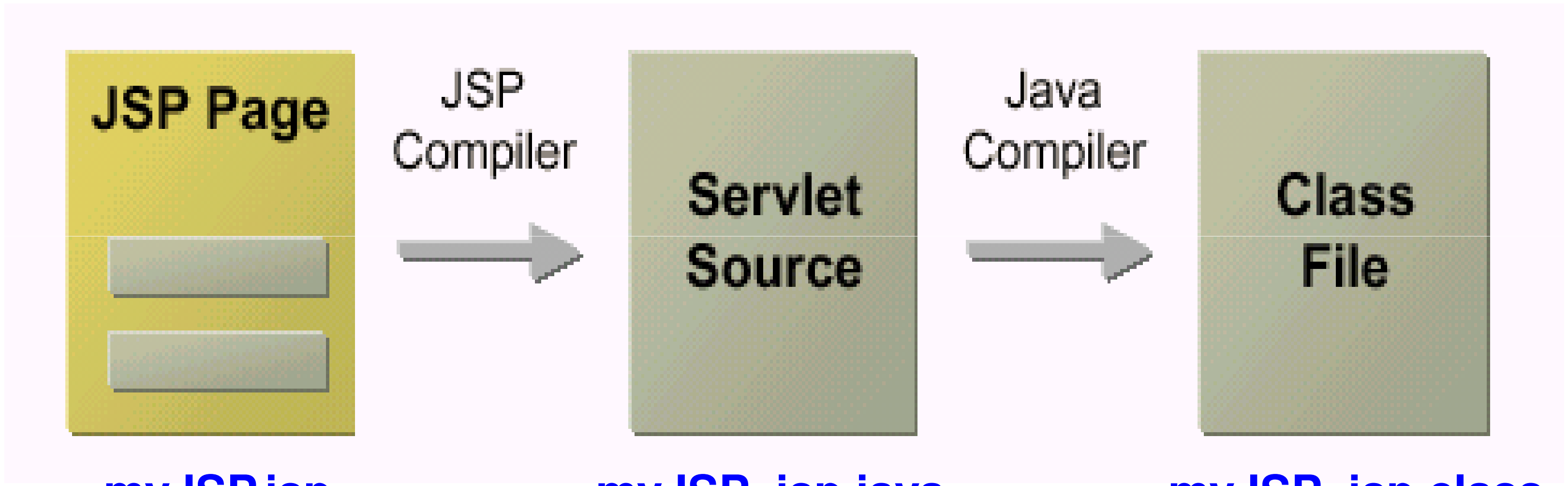
Objectives

- What is JSP?
- What are the Advantages of JSP?
- Elements of a JSP files and tags
- JSP and Java Beans
- Link Servlet and JSP

What is JSP?

- A way to create dynamic web pages
- A text-based documents capable of returning dynamic content to a client browser
- Server side processing
- Based on Java Technology
  - Large library base
  - Platform independence
- Contains HTML, XML, programming code, and JSP tags (HTML and XML tags) allowing access to components such as JavaBeans
- Separates the graphical design from the dynamic content

Compiled into a Servlet



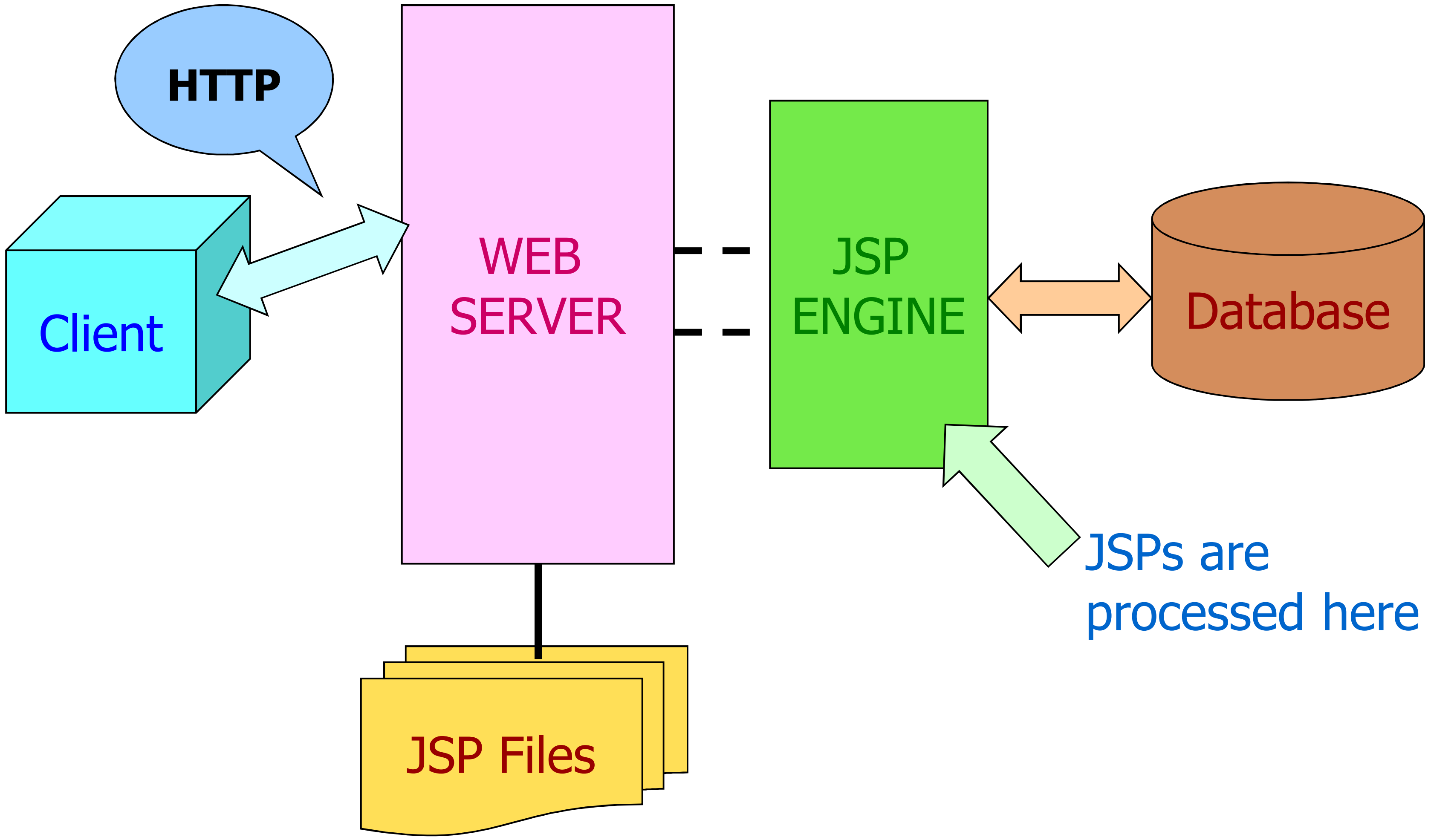
myJSP.jsp myJSP_jsp.java myJSP_jsp.class

```
jspInit()
_jspService()
jspDestroy()
```

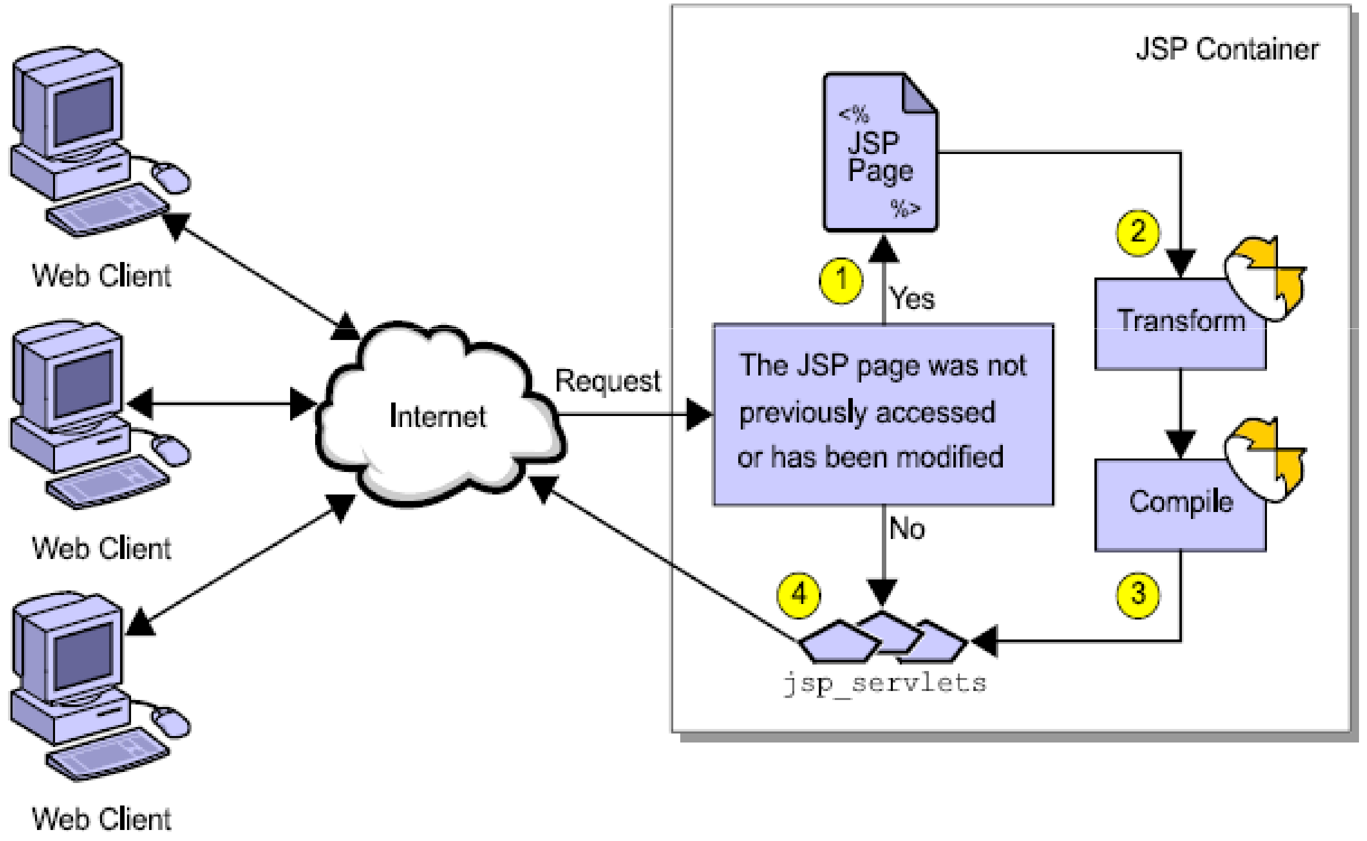
The JSP Life Cycle

- The life cycle of a JSP can be split into approximately four phases:
  - **Translation**
    - The JSP engine will **translate** the JSP **into** its page implementation **servlet** and compile it into a class file
  - **Initialization**
    - The JSP engine will need to load the generated class file and **create** an **instance** of the **servlet**
  - **Servicing**
    - The methods **`jspInit()`** and **`_jspService()`** will be called
  - **Destruction**
    - The method **`jspDestroy()`** will be called

How Do JavaServer Pages Work?



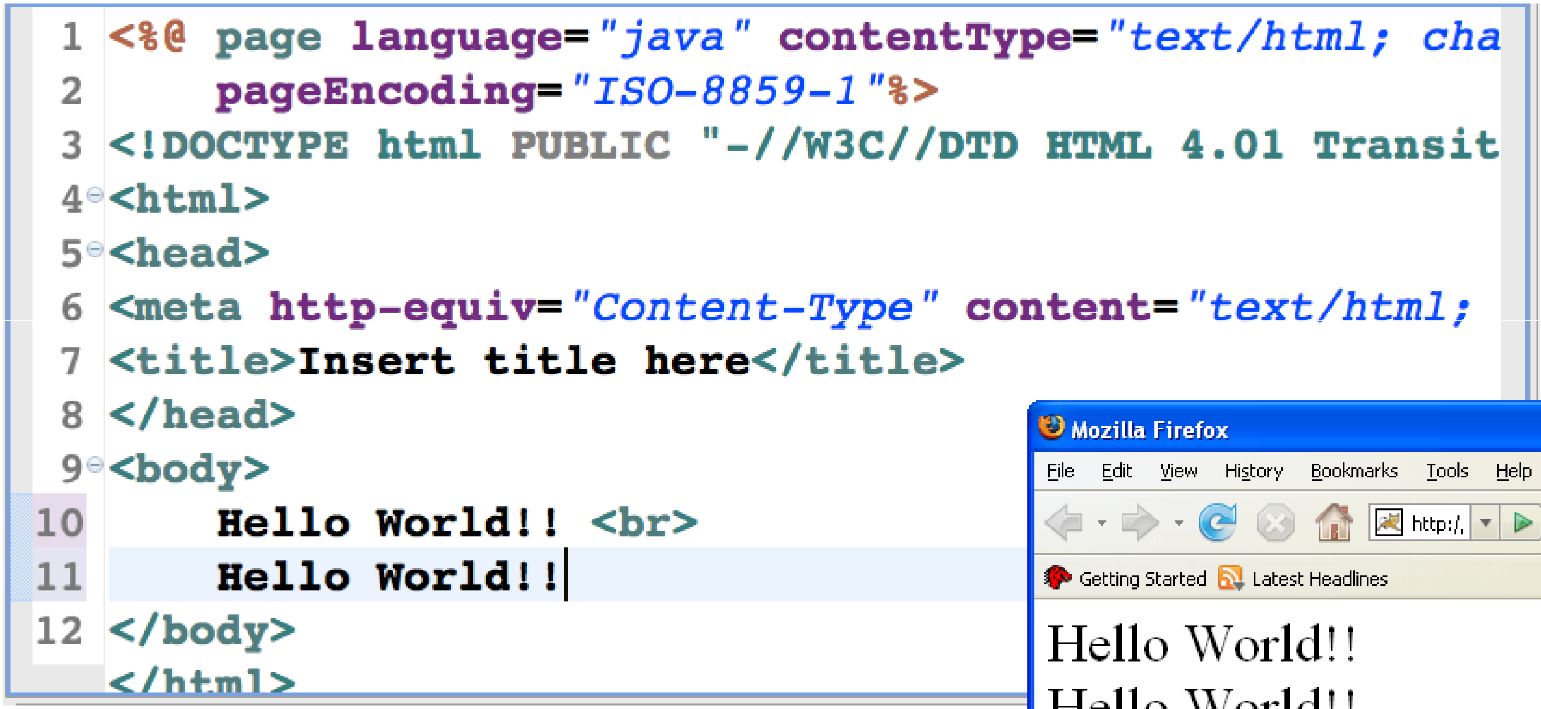
JSP Page Translation Procedure



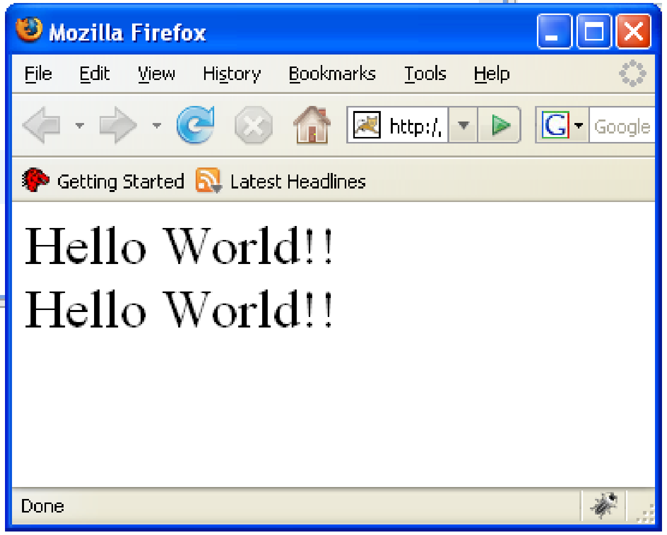
What are the Advantages of JSP?

- Content and display logic are separated.
- Simplify development with JSP, JavaBeans and custom tags.
- Supports software reuse through the use of components.
- Recompile automatically when changes are made to the source file.
- Easier to author.
- Platform-independent

helloWorld.jsp



```
<%@ page language="java" contentType="text/html; cha
    pageEncoding="ISO-8859-1"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transit
<html>
<head>
<meta http-equiv="Content-Type" content="text/html;
<title>Insert title here</title>
</head>
<body>
    Hello World!! <br>
    Hello World!!
</body>
</html>
```



```
Hello World!!
Hello World!!
```

What does a JSP look like?

- Three main JSP constructs:
  - **Directives**
    - Allows one to control structure of the servlet
  - **Scripting Elements**
    - Used to include Java code
  - **Actions**
    - Specific tags that affect the runtime behavior of the JSPs

An Example

Fixed Template data

```
<HTML>
<HEAD><TITLE>MyFirstProgram.jsp</TITLE></HEAD>
<BODY>
<!-- MyFirstProgram.JSP -->

<%@ page import = "java.util.Date" %>

<% out.println("Hello there!"); %><BR>

<jsp:useBean id="name" scope="page"
  class="myBean.SimpleBean" />
</BODY>
</HTML>
```

JSP Directive

JSP Scripting

JSP Action

Directives

- JSP page have the following three types of directives:
  - page
  - include
  - taglib
- All three directive types must be declared between **`<%@`** and **`%>`** directive delimiters and take the following form:

```
<%@ directive {attribute="value"}* %>
```

The `page` Directive

- The **`page`** directive is a JSP tag that you will use in almost every JSP source file.
- The **`page`** directive gives instructions to the JSP container that apply to the entire JSP source file.
- **`page`** might specify the scripting language used in the JSP source file, packages the source file would import, or the error page called if an error or exception occurs.
- You can use the **`page`** directive anywhere in the JSP file, but it's good coding style to place it at the top of the file.

Attributes of The `page` Directive

- `language`
  - Currently, JSP 2.0 only supports a value of `"Java"`
- `extends`
  - Java class that will form the superclass of the JSP page’s servlet.
- `import`
  - Indicates the classes available for use within the scripting environment
  - The default import list is as follows:
    - `javax.servlet.jsp.*  and`
    - `javax.servlet.http.*`
- `session`
  - Indicates that the page requires an `HttpSession`
  - The default value is `"true"`

Attributes of The **`page`** Directive (cont.)

- **`buffer`**
  - **Specifies the buffering model for `JspWriter` used to handle the response**
  - **The default value isn't less than 8 KB.**
- **`autoFlush`**
  - **Indicate whether the output buffer should be flushed automatically when full**
  - **The default value is `"true"`**
- **`isThreadSafe`**
  - **Indicates the threading model to be used by the JSP**
  - **The default value is `"true"`**
- **`info`**
  - **Can be used to provide any arbitrary string**
  - **Returned via a call to the page servlet's `getServletInfo()` method**

Attributes of The `page` Directive (cont.)

- `isErrorPage`
  - Indicates whether or not the current JSP is intended to be an error page
  - The default value is `"false"`
- `errorPage`
  - Defines the URL that any throwable objects are forwarded for error processing
- `contentType`
  - Defines the character encoding for the JSP page and MIME type of the response
  - The default value for the type is `"text/html"` and for the charset it’s `"ISO-8859-1"`

Attributes of The `page` Directive (cont.)

- `pageEncoding`
  - Defines the character encoding for the JSP page
  - The default value is `"ISO-8859-1"`

- `isELignored`
  - Defines whether the EL (Expression Language) expressions are evaluated for the JSP page
  - The default value is `"false"`

Examples of The **page** Directive

```
<%@ page language="java" %>
<%@ page import="java.util.*, java.text.*" %>
<%@ page extends="Servlet1" %>
<%@ page session="true" %>
<%@ page buffer="16kb" %>
<%@ page autoFlush="true" %>
<%@ page isThreadSafe="false" %>
<%@ page info="First JSP" %>
<%@ page isErrorPage="false" %>
<%@ page errorPage="ErrorPage.jsp" %>
<%@ page contentType="text/html" %>
<%@ page pageEncoding="ISO-8859-1" %>
<%@ page isELIgnored="false" %>
```

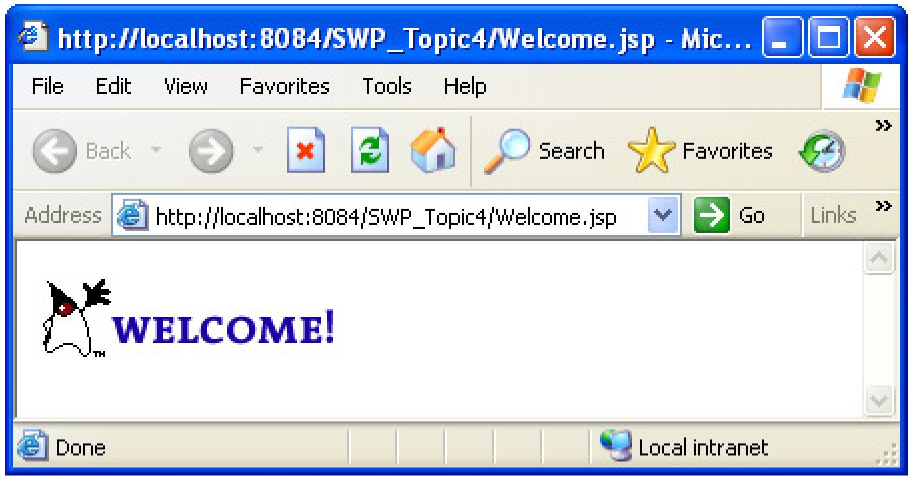
The **`include`** Directive

- The **`include`** directive inserts the contents of another file in the main JSP file, where the directive is located.
- It's useful for including copyright information, scripting language files, or anything you might want to reuse in other applications.
- The included file does not contain <html> or <body> tags, because these tags would conflict with the same tags in the calling JSP file.

Example

```
<!-- dukeBanner.html -->
<img src="Duke.png">
```

```
<%-- welcome.jsp --%>
<%@ include file="dukeBanner.html" %>
```


http://localhost:8084/SWP_Topic4/Welcome.jsp - Mic...
File Edit View Favorites Tools Help
Back
Search
Favorites
Address
http://localhost:8084/SWP_Topic4/Welcome.jsp
Go
Links
WELCOME!
Done
Local intranet

Scripting Elements

- Comments
- Declarations
- Scriptlets
- Expressions

Comments

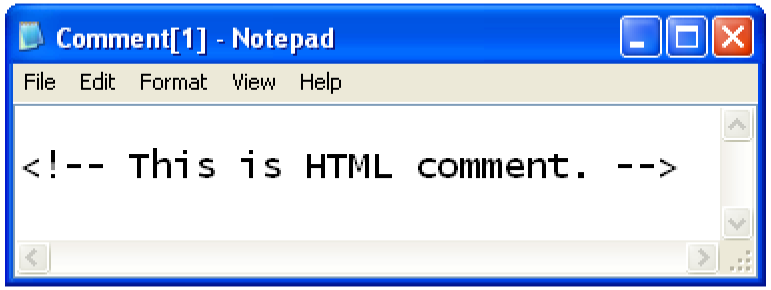
- JSP comments may be declared inside a JSP as follows:

```
<%-- This is JSP comment. --%>
```

- HTML comments

```
<!-- This is HTML comment. -->
```

- They are the same?



```
<!-- This is HTML comment. -->
```

Declarations

- Declarations are the definition of class-level variables and methods that are used in a JSP.
- The general syntax is as follows:

```
<%! Declaration; %>
```

- Example:
  - `<%! private int count; %>`
  - `<%! Date now = new Date(); %>`
  -
```
<%!
        private int sum(int a, int b) {
         …return a+b;
        }
%>
```

JSP Scripting Elements

Points to remember about declarations:

- Declarations do not produce any printable results.
- Variables are initialized when the JSP is initialized.
- Such variables are available to other declarations, expressions, and scriptlets.
- Variables created through declarations become **instance variables**.
- Simultaneous users of the JSP share the same instance of the variable.
- Class variables and methods can be declared in a declaration block.

Scriptlets

- Scriptlets are small blocks of source code contained within the **<%** and **%>**
- The general syntax is as follows:

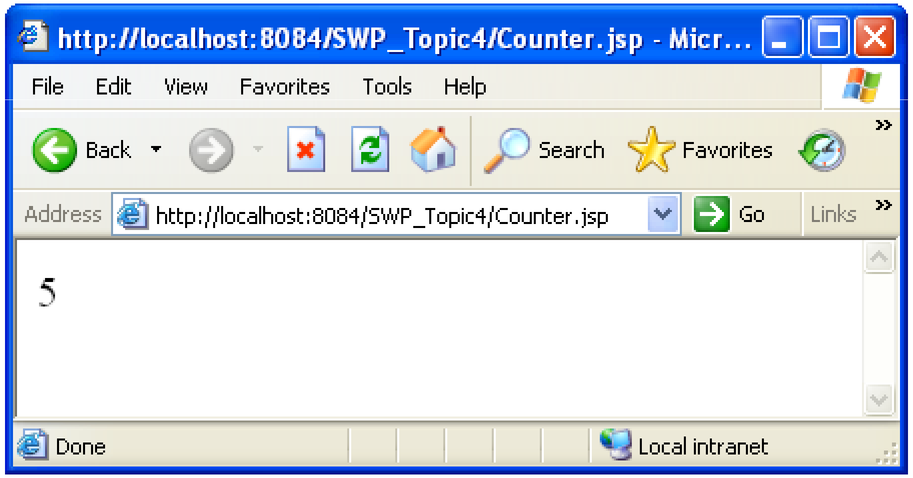
```
<% scriptlet_source; %>
```

- Example

```
<% out.println(++count); %>
```

Example

```
<%! int count; %>
<% out.println(++count); %>
```


http://localhost:8084/SWP_Topic4/Counter.jsp - Micr...
File Edit View Favorites Tools Help
Back Search Favorites
Address http://localhost:8084/SWP_Topic4/Counter.jsp Go Links
Done Local intranet

5

```
<% int count = 0; %>
<% out.println(++count); %>
```

They are the same?

Implicit Objects

- Server-side objects are defined by the JSP container
- They are always available in a JSP, without being declared
- These objects determine:
  - how to accept the request from the browser
  - how to send the response from the server
  - how session tracking can be done

Implicit Objects (cont.)

Objects	class	scope
request	javax.servlet.http.HttpServletRequest	request
response	javax.servlet.http.HttpServletResponse	page
session	javax.servlet.http.HttpSession	session
application	javax.servlet.ServletContext	application
out	javax.servlet.jsp.JspWriter	page
PageContext	javax.servlet.jsp.PageContext	page
config	javax.servlet.ServletConfig	page
page	java.lang.Object	page
exception	java.lang.Throwable	application

Expressions

- Evaluate a regular Java expression and return a `String` or be convertible to a `String` result
- The general syntax is as follows:

```
<%= expression %>
```

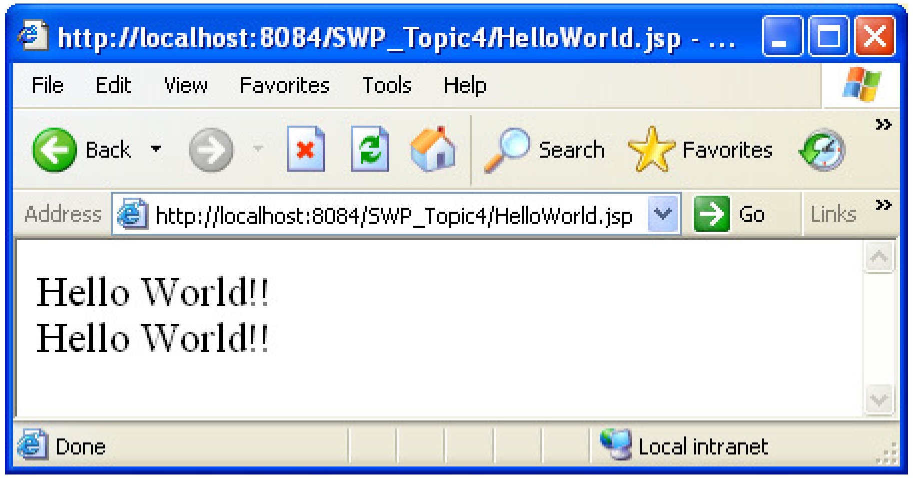
- Example

```
<%= ++count %>
```

- JSP expressions don't require a closing semicolon.

A Simple Web Application

```
</head>
<body>
    <% out.print("Hello World!!<br>"); %>
    <%= "Hello World!!" %>
</body>
</html>
```


http://localhost:8084/SWP_Topic4/HelloWorld.jsp - ...
File Edit View Favorites Tools Help
Back
Search
Favorites
Address
http://localhost:8084/SWP_Topic4/HelloWorld.jsp
Go
Links
Done
Local intranet

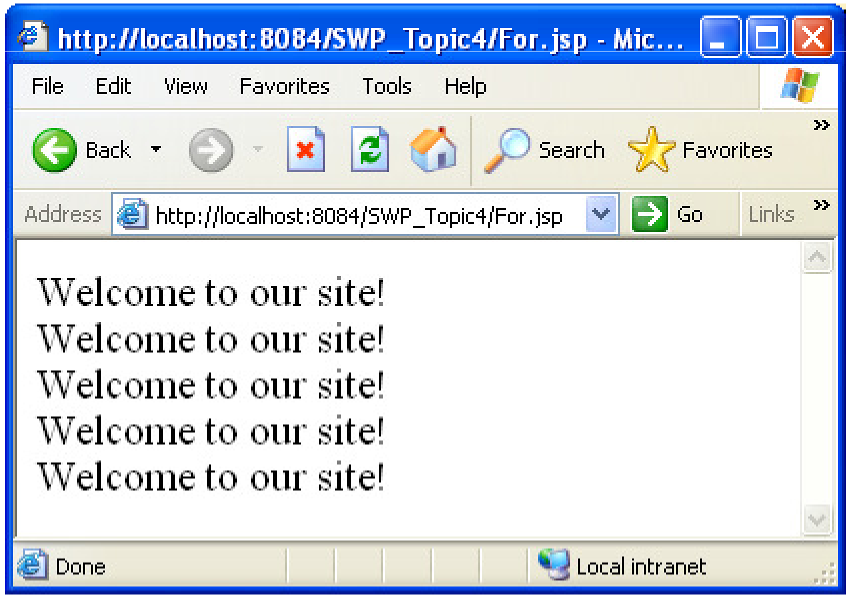
Hello World!!
Hello World!!

Conditional Statements

```
<%! int count; %>

This page has been accessed <%= ++count %> time(s). <BR>

<% if (count == 1) { %>
     Welcome, you are the first visitor to our site.<BR>
<% } else { %>
     You are not the first visitor to our site.<BR>
<% } %>
   Please do visit us again. Thank you!
```

Iterative Statements (For)

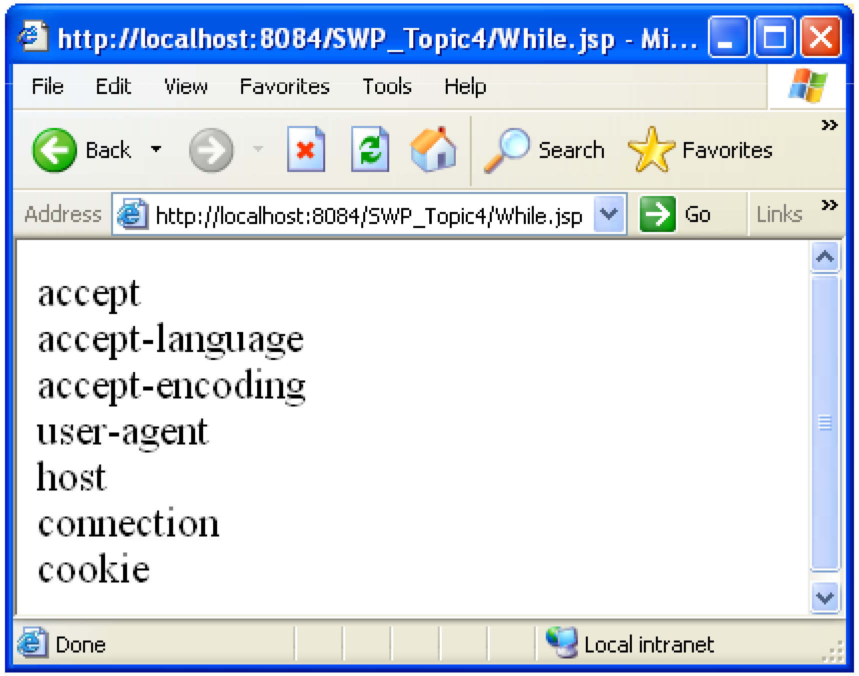
```
<% for (int i=0; i<5; i++) { %>
     Welcome to our site! <BR>
<% } %>
```


http://localhost:8084/SWP_Topic4/For.jsp - Mic...
File Edit View Favorites Tools Help
Back
Search
Favorites
Address
http://localhost:8084/SWP_Topic4/For.jsp
Go
Links
Done
Local intranet

```
Welcome to our site!
Welcome to our site!
Welcome to our site!
Welcome to our site!
Welcome to our site!
```

Iterative Statements (While)

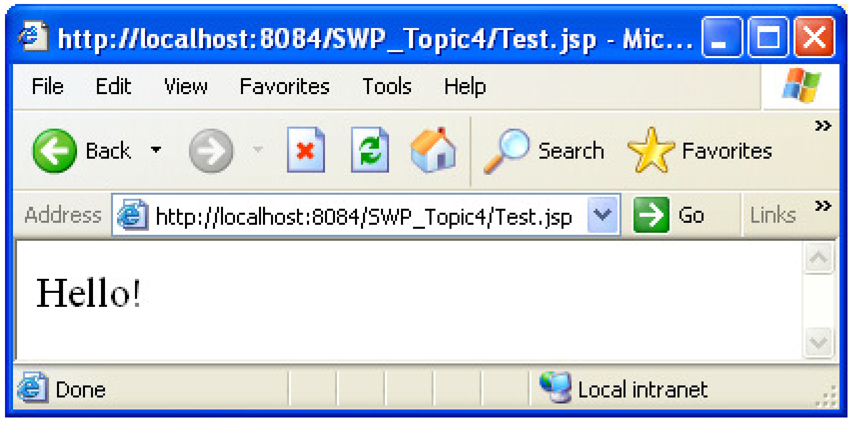
```
<% java.util.Enumeration enum1 = request.getHeaderNames(); %>
<% while (enum1.hasMoreElements()) { %>
   <%= enum1.nextElement() %> <BR>
<% } %>
```


http://localhost:8084/SWP_Topic4/While.jsp - Mi...
File Edit View Favorites Tools Help
Back
Search
Favorites
Address http://localhost:8084/SWP_Topic4/While.jsp
Go
Links
Done
Local intranet

```
accept
accept-language
accept-encoding
user-agent
host
connection
cookie
```

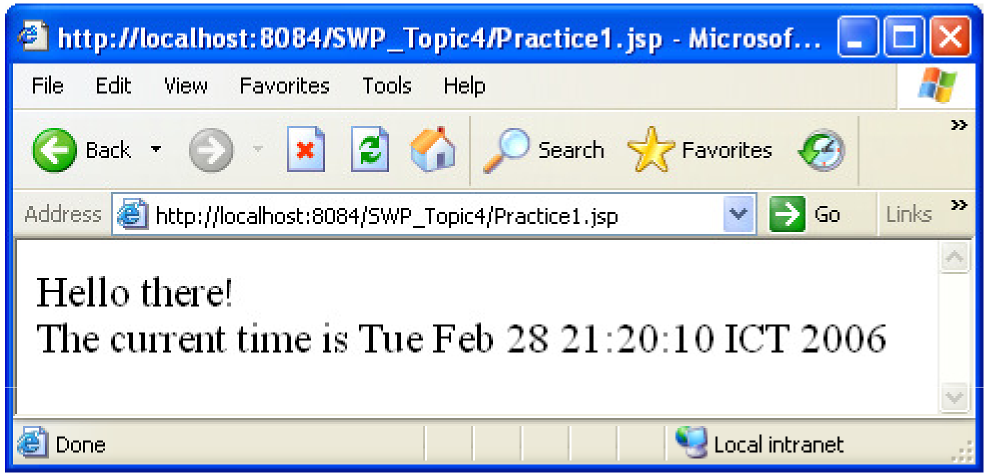
Using Method

```
<%! public String methodA() {
        return "Hello!";
    }
%>
<% out.println(methodA()); %>
```


http://localhost:8084/SWP_Topic4/Test.jsp - Mic...
File Edit View Favorites Tools Help
Back
Search
Favorites
Address http://localhost:8084/SWP_Topic4/Test.jsp
Go
Links
Done
Local intranet

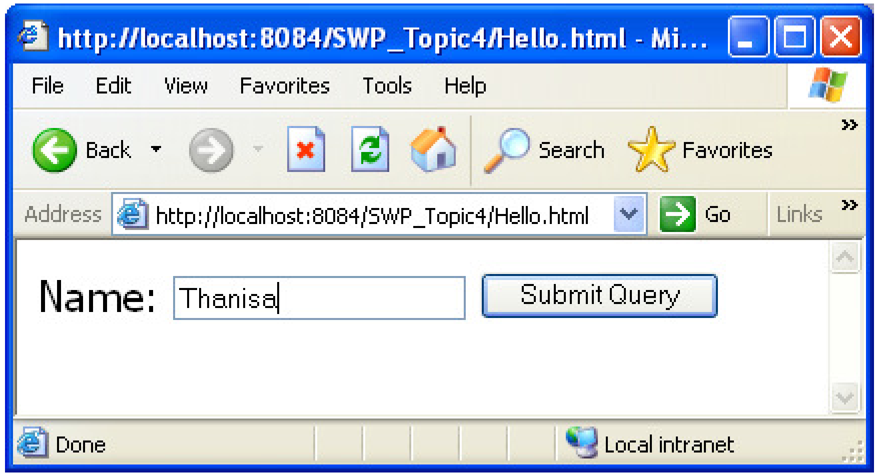
```
Hello!
```

Practice



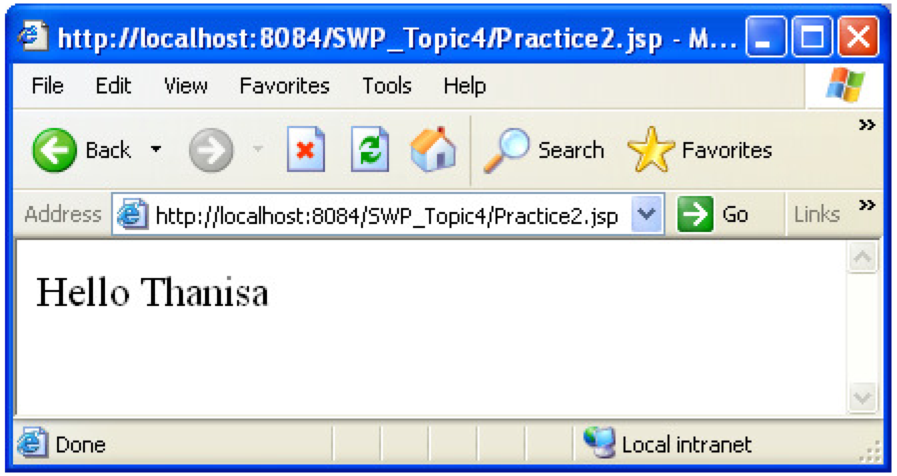
http://localhost:8084/SWP_Topic4/Practice1.jsp

Hello there!
The current time is Tue Feb 28 21:20:10 ICT 2006



http://localhost:8084/SWP_Topic4/Hello.html

Name: Thanisa



http://localhost:8084/SWP_Topic4/Practice2.jsp

Hello Thanisa

Exercise

calculator.html

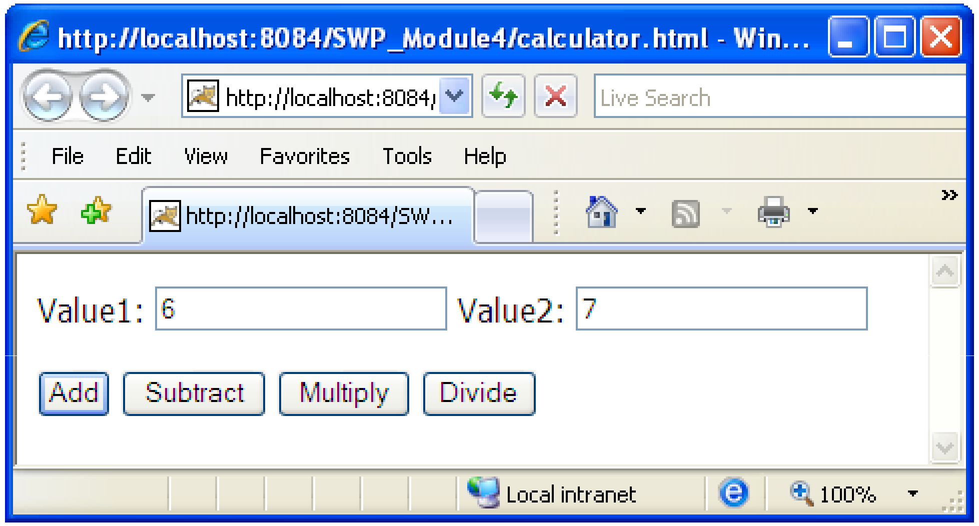
```
<form action=“calculator.jsp">
  Value1: <input name="value1">
  Value2: <input name="value2"><BR><BR>
  <input type="submit" name="cmd" value="Add">
  <input type="submit" name="cmd" value="Subtract">
  <input type="submit" name="cmd" value="Multiply">
  <input type="submit" name="cmd" value="Divide">
</form>
```

Exercise (cont.)

calculator.jsp

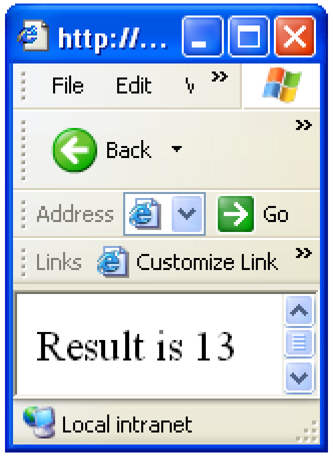
```
:
<% try {
      double value1 = Double.parseDouble(request.getParameter("value1"));
      double value2 = Double.parseDouble(request.getParameter("value2"));
      :
   } catch (NumberFormatException e) { %>
      Please enter valid numbers!!!
<% } %>
```

Result

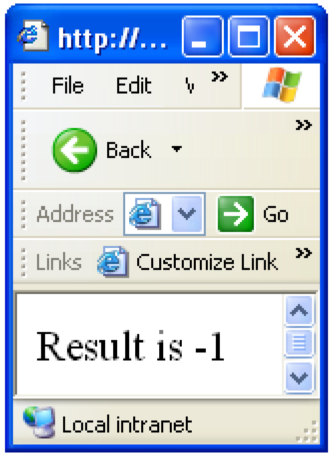


http://localhost:8084/SWP_Module4/calculator.html - Win...

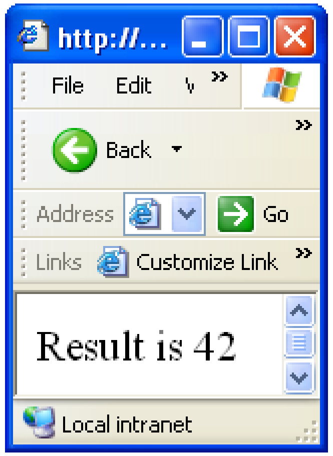
Value1: 6 Value2: 7



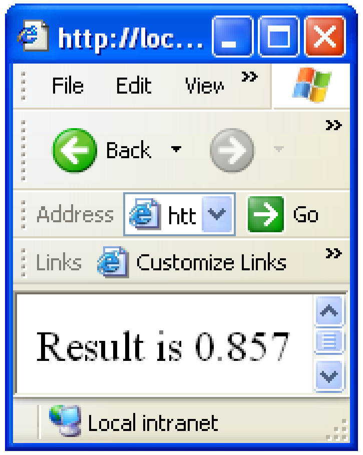
Result is 13



Result is -1



Result is 42



Result is 0.857

JSP Standard Actions

- **A JSP action directive provides an easy method to encapsulate common tasks. These typically create or act on objects, usually JavaBeans.**
  - `<jsp:param>`
  - `<jsp:include>`
  - `<jsp:forward>`
  - `<jsp:plugin>`
  - `<jsp:useBean>`
  - `<jsp:setProperty>`
  - `<jsp:getProperty>`

JSP Standard Actions

- **<jsp:param>**
  - Is used to provide the tag/value pairs of information
  - Included as sub-attributes of jsp:include, jsp:forward, and jsp:plugin actions
  - The syntax is as follows:

```
<jsp:param name = "pName" value = "pValue"/>
```

```
<jsp:param name = "pName" value = "pValue">
</jsp:param>
```

JSP Standard Actions

- **<jsp:include>**
  - Provides a mechanism for including additional static and **dynamic** resources in the current JSP
  - The syntax is as follows:

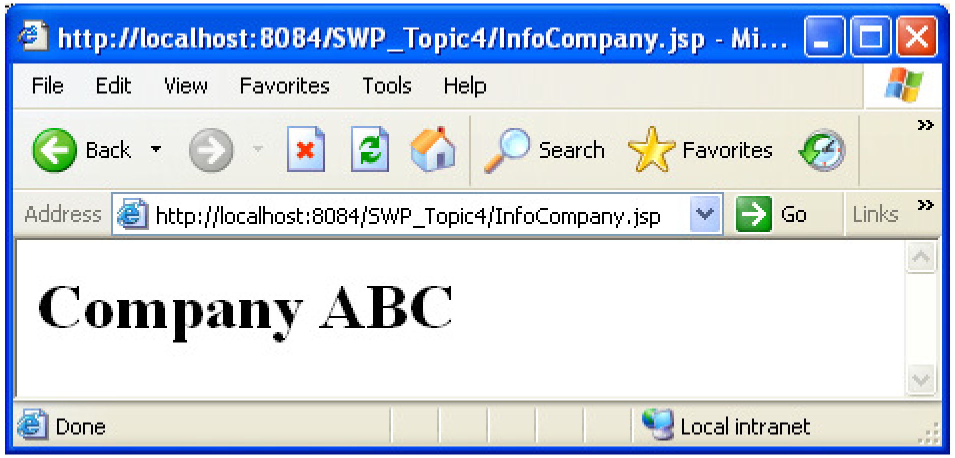
```
<jsp:include page = "urlSpec" flush = "true"/>
```

```
<jsp:include page = "urlSpec" flush = "true">
  <jsp:param .../>
</jsp:include>
```

Using <jsp:include> - An Example

```
<%-- header.jsp --%>
<h2> Company <%= request.getParameter("name")%></h2>
```

```
<%-- infoCompany.jsp --%>
<jsp:include page="header.jsp" >
    <jsp:param name="name" value="ABC" />
</jsp:include>
```


http://localhost:8084/SWP_Topic4/InfoCompany.jsp - Mi...
File Edit View Favorites Tools Help
Back
Search
Favorites
Address http://localhost:8084/SWP_Topic4/InfoCompany.jsp
Go
Links
Done
Local intranet

Company ABC

JSP Standard Actions

- **<jsp:forward>**
  - Enables the JSP engine to dispatch the current request to a static resource, a servlet, or to another JSP at run-time
  - The syntax is as follows:

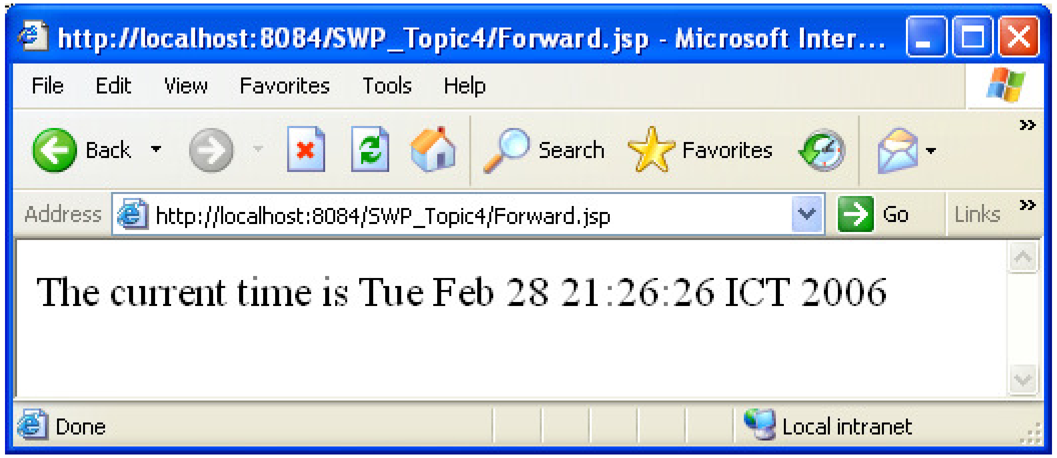
```
<jsp:forward page = "relativeURLSpec" />
```

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

Using <jsp:forward> - An Examples

```
<%-- date.jsp --%>
The current time is <%= new java.util.Date() %>
```

```
<%-- forwardDemo.jsp --%>
<jsp:forward page="date.jsp" />
```


http://localhost:8084/SWP_Topic4/Forward.jsp - Microsoft Inter...
File Edit View Favorites Tools Help
Back Search Favorites
Address http://localhost:8084/SWP_Topic4/Forward.jsp Go Links
Done
Local intranet

The current time is Tue Feb 28 21:26:26 ICT 2006

JSP Standard Actions

- **<jsp:plugin>**
  - Generates HTML that contains the appropriate client-browser dependent constructs, such as OBJECT or EMBED. This will prompt the download of the required Java plugin, and subsequent execution of the applet or bean.
  - The syntax is as follows:

```
<jsp:plugin type="pluginType" code="classfilename">
   <jsp:params>
<jsp:param …/>
   </jsp:params>
</jsp:plugin>
```

JSP Standard Actions

- **<jsp:useBean>**
  - Associates an instance of a pre-defined JavaBean with a given scope and ID
- **<jsp:setProperty>**
  - Sets the value of a bean's property
- **<jsp:getProperty>**
  - Accesses the value of the specified property of a bean instance
  - Converts it to a java.lang.String object
  - Places it in the implicit out object

Difference between a JavaBean and a Java class

A bean is a Java class with the following additional characteristics:

- The class must be instantiable
- It must have a default constructor
- It must be serializable
- It must follow the JavaBeans design patterns
- It must follow the new Java 1.1 AWT delegation event model

Student.java

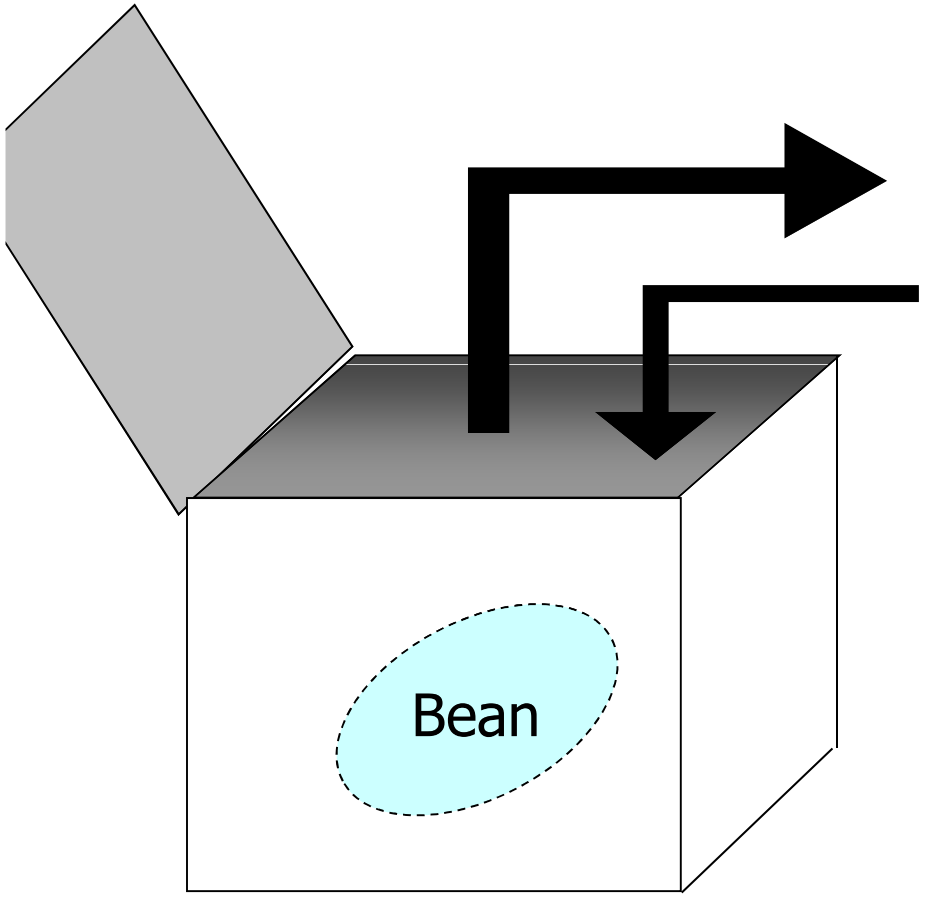
```java
package myBeans;

public class Student implements java.io.Serializable {
    private String name;
    private String id;
    public void setName(String name) {
        this.name = name;
    }
    public String getName() {
        return name;
    }
    public void setId(String id) {
        this.id = id;
    }
    public String getId() {
        return id;
    }
}
```

Property

setProperty()

getProperty()

Using JSP Bean Tags



getProperty_Name()

setProperty_Name(*value*)

The JSP need not know the inner structure of the bean

The Black box approach

Using JSP Bean Tags

- JSP provides three basic bean tags:
  - To find and use the bean – **jsp:useBean**
  - To set one or more properties – **jsp:setProperty**
  - To get a property – **jsp:getProperty**
- These are also known as actions and are specific tags that affect the following:
  - the runtime behavior of the JSP
  - the responses that are sent back to the client

jsp:useBean

- This is used to associate a JavaBean in JSP.
- It ensures that the object can be referenced from JSP, using an ID and scope
- The syntax is as follows:

The bean class must be available to the JSP engine

```
<jsp:useBean id="bean_name"
   scope="page|request|session|application"
   class="bean_class" >
```

Compare the Two

```
<%
  ShoppingCart cart = (ShoppingCart) session.getAttribute("cart");
  // If the user has no cart object as an attribute in Session scope
  // object, then create a new one.  Otherwise, use the existing
  // instance.
  if (cart == null) {
      cart = new ShoppingCart();
      session.setAttribute("cart", cart);
  }
%>
```

versus

```
<jsp:useBean id="cart" class="cart.ShoppingCart" scope="session"/>
```

jsp:setProperty

- It sets the values of simple and indexed properties of a bean in various ways:
  - At request time, using the parameters in the request object
  - At request time, using the result of an evaluated expression
  - From a specified string
- The syntax is as follows:

```
<jsp:setProperty name = "id" property = "*" />
<jsp:setProperty name = "id" property = "propertyName"
 param = "parameterName"/>
<jsp:setProperty name = "id" property = "propertyName"
 value = "propertyValue"/>
```

Setting Bean Properties

- Properties can be set through an HTML form too
- The users can provide values through the form
- A JSP can be used to pass these values to a bean through the `setProperty` tags
- Form values can be passed like this:

```
<jsp:setProperty  name="id" property="formParam" />
```

Compare the Two

```
<jsp:setProperty name="bookDB" property="bookId"/>
```

is same as

```
<%
  //Get the identifier of the book to display
  String bookId = request.getParameter("bookId");
  bookDB.setBookId(bookId);
     ...
%>
```

jsp:getProperty

- This action is complementary to the jsp:setProperty action
- It is used to access the properties of a bean
- It converts the retrieved value to a String
- The syntax is as follows:

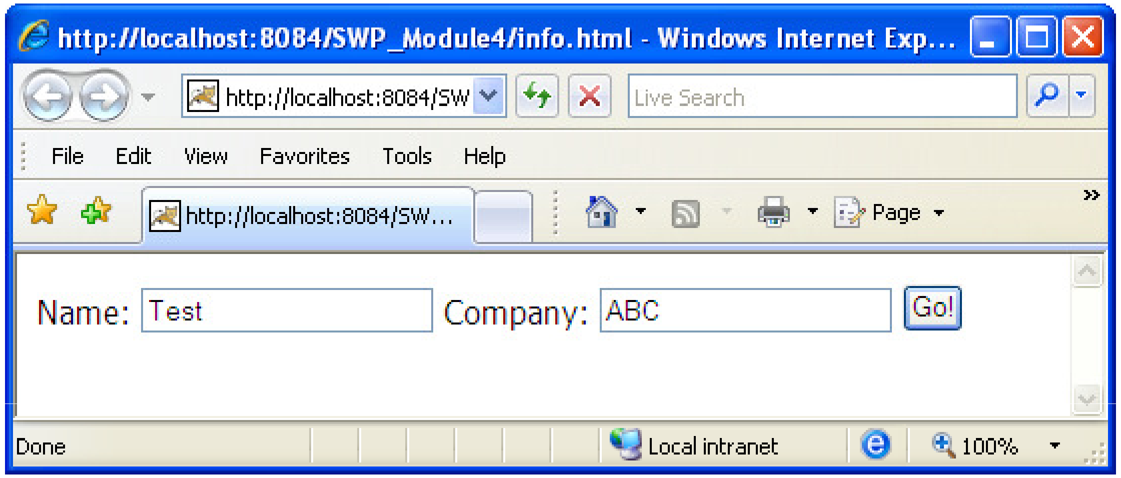
```
<jsp:getProperty name="id" property="propertyName" />
```

info.html and useBeanDemo.jsp

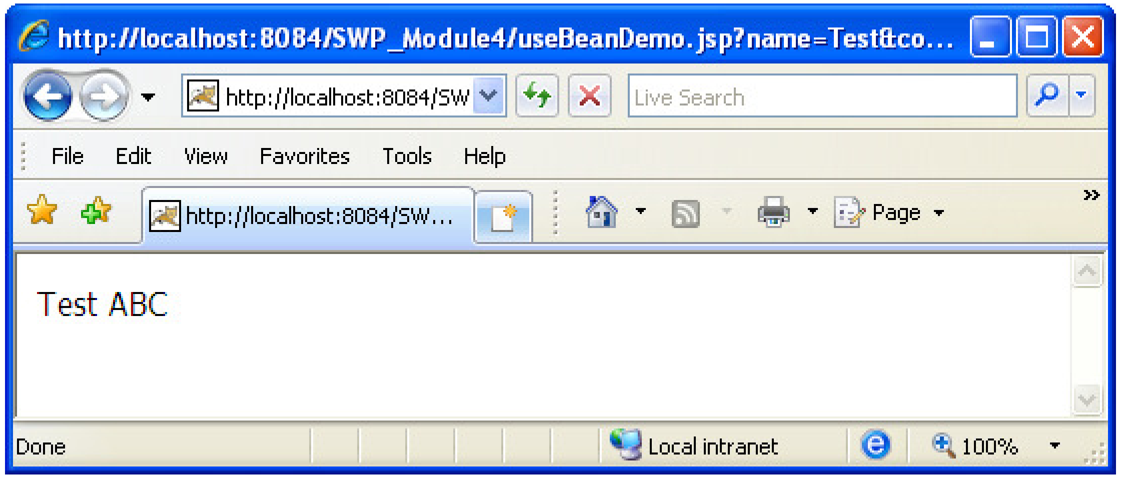
```
<!-- info.html -->
<form action=“useBeanDemo.jsp”>
  Name: <input name=“name”>  Company: <input name=“company”>
  <input type=“submit” value=“Go!”>
</form>
```

```
<%-- useBeanDemo.jsp --%>
<jsp:useBean id="nameID" class="myBeans.Staff" />
<jsp:setProperty name="nameID" property="*" />
<jsp:getProperty name="nameID" property="name" />
<jsp:getProperty name="nameID" property="company" />
```

Result


http://localhost:8084/SWP_Module4/info.html - Windows Internet Exp...

Name: Test Company: ABC Go!


http://localhost:8084/SWP_Module4/useBeanDemo.jsp?name=Test&co...

Test ABC

An Example

The Bean – CounterBean.java

```
package myBeans;

import java.io.*;

public class CounterBean implements Serializable {
    private int count;
    public int getCount() {
 return ++count;
    }
    public void setCount(int c) {
        count = c;
    }
}
```

Property

get Method

set Method

An Example (cont.)

The JSP that uses the bean

Instantiate the Counter bean

```
<HTML> . . .

<jsp:useBean id="id_counter" scope="application"
 class="myBeans.CounterBean" />
<jsp:setProperty name="id_counter" property="count" />
<jsp:getProperty name="id_counter" property="count" />

. . . </HTML>
```

Scope of Beans

- By default, the objects created with **`jsp:useBean`** were bound to local variables in the **`_jspService`** method
- The beans are also stored in one of four different locations, depending on the value of the optional **`scope`** attribute of **`jsp:useBean`**

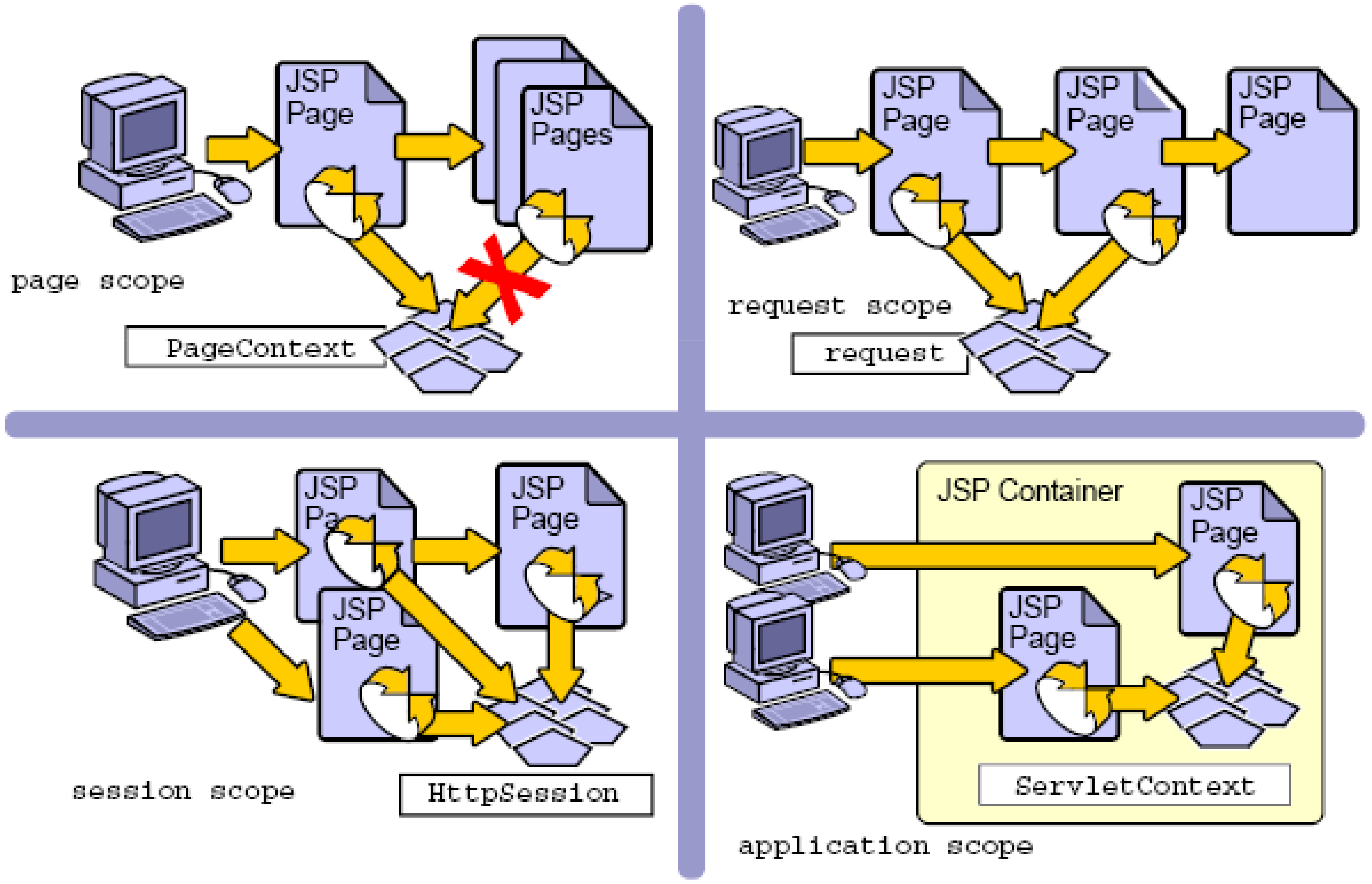
Scope of Beans (cont.)

- **`page`**
  - This is the default value
  - References to this object will only be released after the response is sent back to the client
- **`request`**
  - The bean instance will be available as long as the request object is available
  - References to this object will be released only after the request is processed completely

Scope of Beans (cont.)

- `session`
  - The instance will be available across a particular session between the browser of a particular machine and the Web server
  - References to this object will be released only after the associated sessions end
- `application`
  - The instance will be available to all users and pages of the application
  - It is released only when the run-time environment reclaims the `ServletContext`

Scope of Beans (cont.)



An Example

The Form – calculator1.html

```
<HTML> . . .

Enter any two numbers and click on the
<B>Calculate</B> Button.

<FORM ACTION = “useBean1.jsp">

Value1: <INPUT NAME = "value1">
Value2: <INPUT NAME = "value2">
<INPUT TYPE = "SUBMIT" VALUE = "Calculate">

. . . </HTML>
```

An Example (cont.)

The JSP – useBean1.jsp

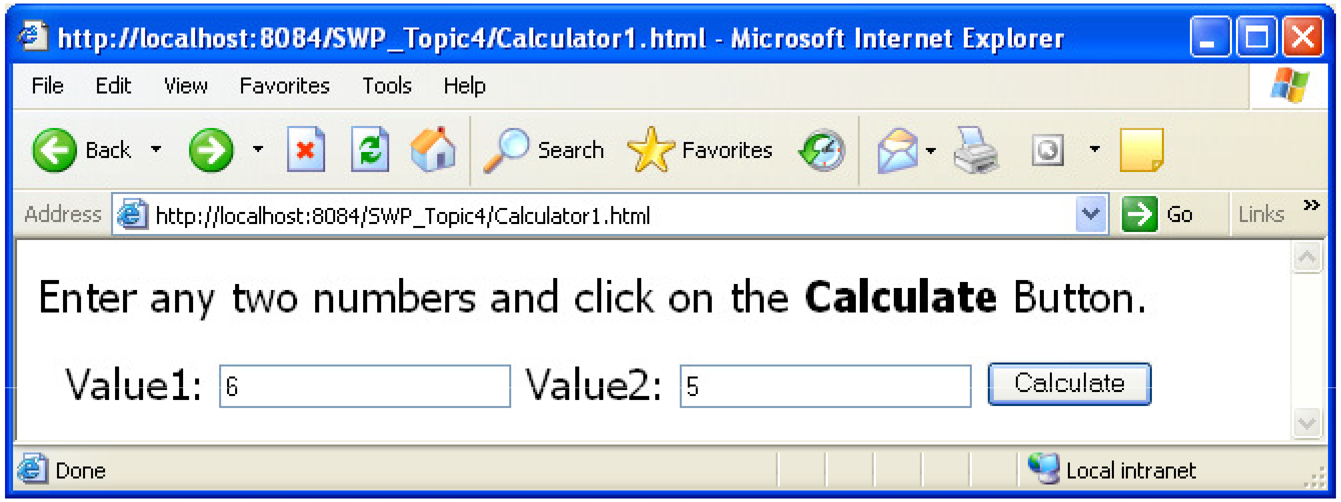
```
<jsp:useBean  id="calc"  class="myBeans.CalcBean" />
<jsp:setProperty  name="calc"  property="*" />

<B> The sum of the two numbers is
<jsp:getProperty  name="calc"  property="sum" /></B>
```

An Example (cont.)

The Bean – CalcBean.java

```
public class CalcBean implements java.io.Serializable {
  private int value1;
  private int value2;
  private int sum;

  public void setValue1(int num1) {
    value1 = num1;
  }

  public void setValue2(int num2) {
    value2 = num2;
  }

  public int getSum() {
    return value1 + value2;
  }
}
```

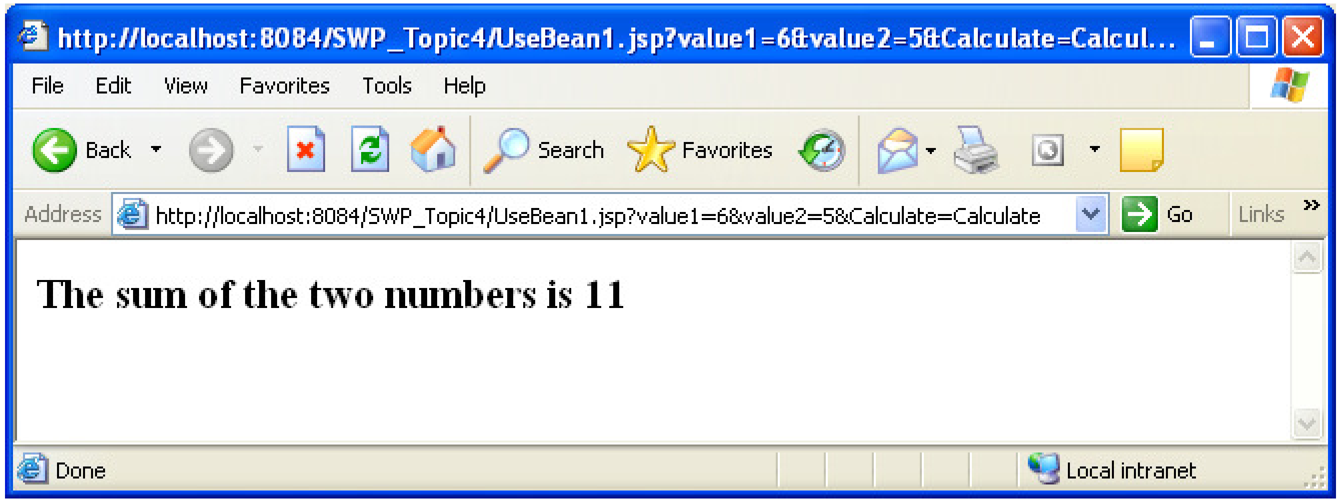
Result


http://localhost:8084/SWP_Topic4/Calculator1.html - Microsoft Internet Explorer

Address http://localhost:8084/SWP_Topic4/Calculator1.html

Enter any two numbers and click on the **Calculate** Button.

Value1: 6 Value2: 5 Calculate


http://localhost:8084/SWP_Topic4/UseBean1.jsp?value1=6&value2=5&Calculate=Calcul...

Address http://localhost:8084/SWP_Topic4/UseBean1.jsp?value1=6&value2=5&Calculate=Calculate

The sum of the two numbers is 11

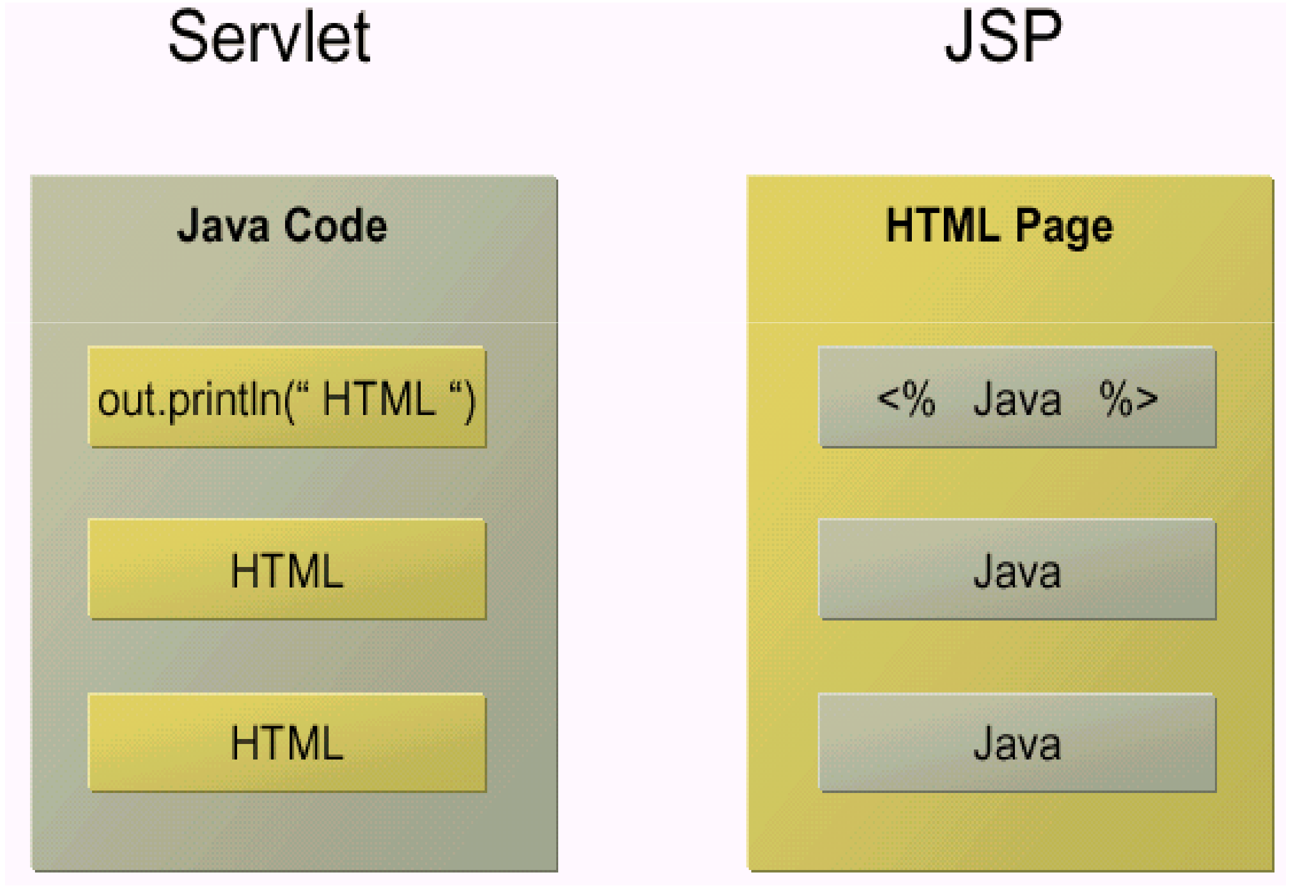
JSP Standard Actions

- JSP actions have the following characteristics:
  - These tags are case-sensitive
  - These are standard set of actions, supported by all containers
  - Attribute values must be enclosed within quotes
  - Tags must have a closing delimiter
  - New actions can be built
  - Existing actions can be extended
  - Actions can be put into the output buffer
  - Actions can access, modify, and create objects on the current server pages.

Comparison of Servlet and JSP

Servlet	**JSP**
HTML code in **Java**	Java code in **HTML**
Any form of Data	Structured Text
Not easy to author	Very easy to author
Underlying semantics	Code is compiled into a servlet

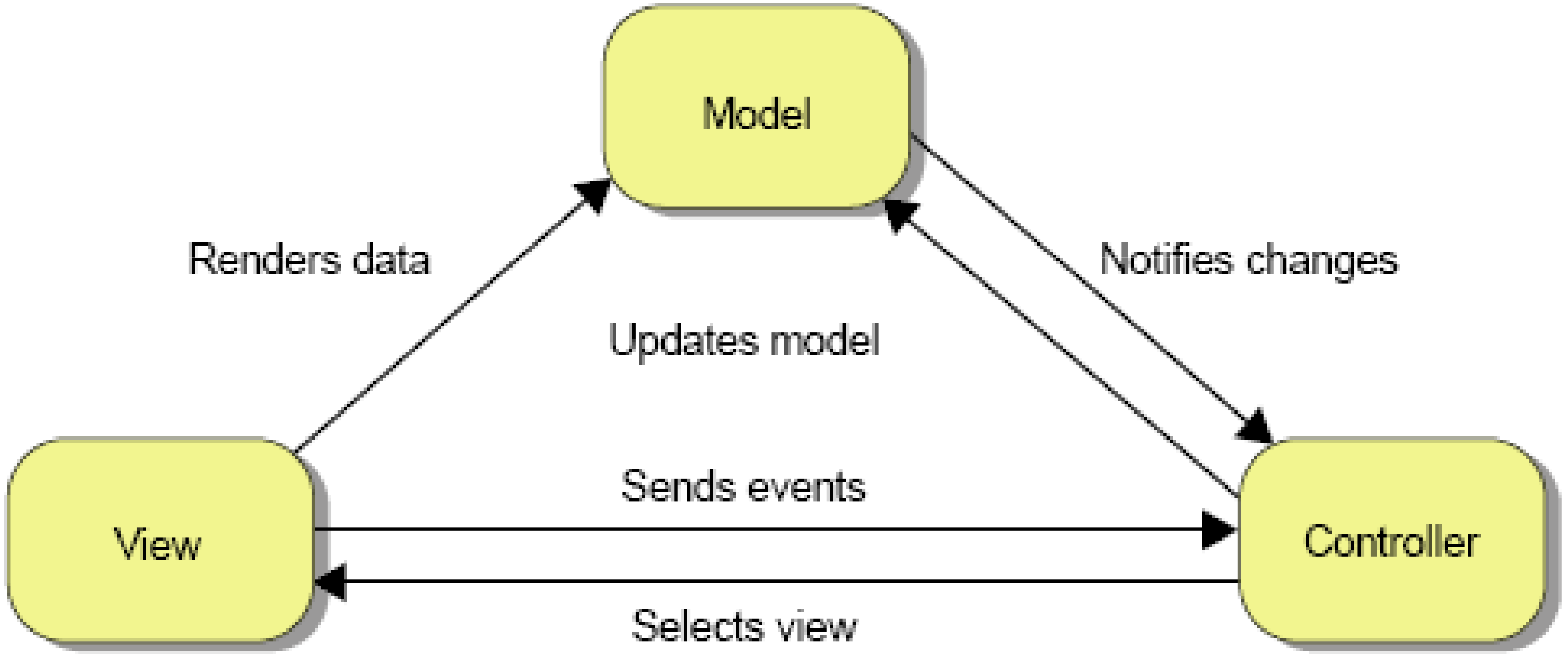
Comparison of Servlet and JSP (cont.)



Recommended Uses of JSP

- Application logic separated from Web content and embedded in components.
- Present dynamic portions of content, which is tailored to a specific use.
- Display repetitive portions of a Web site such as a header, a footer and a navigation bar.

Traditional MVC Architecture

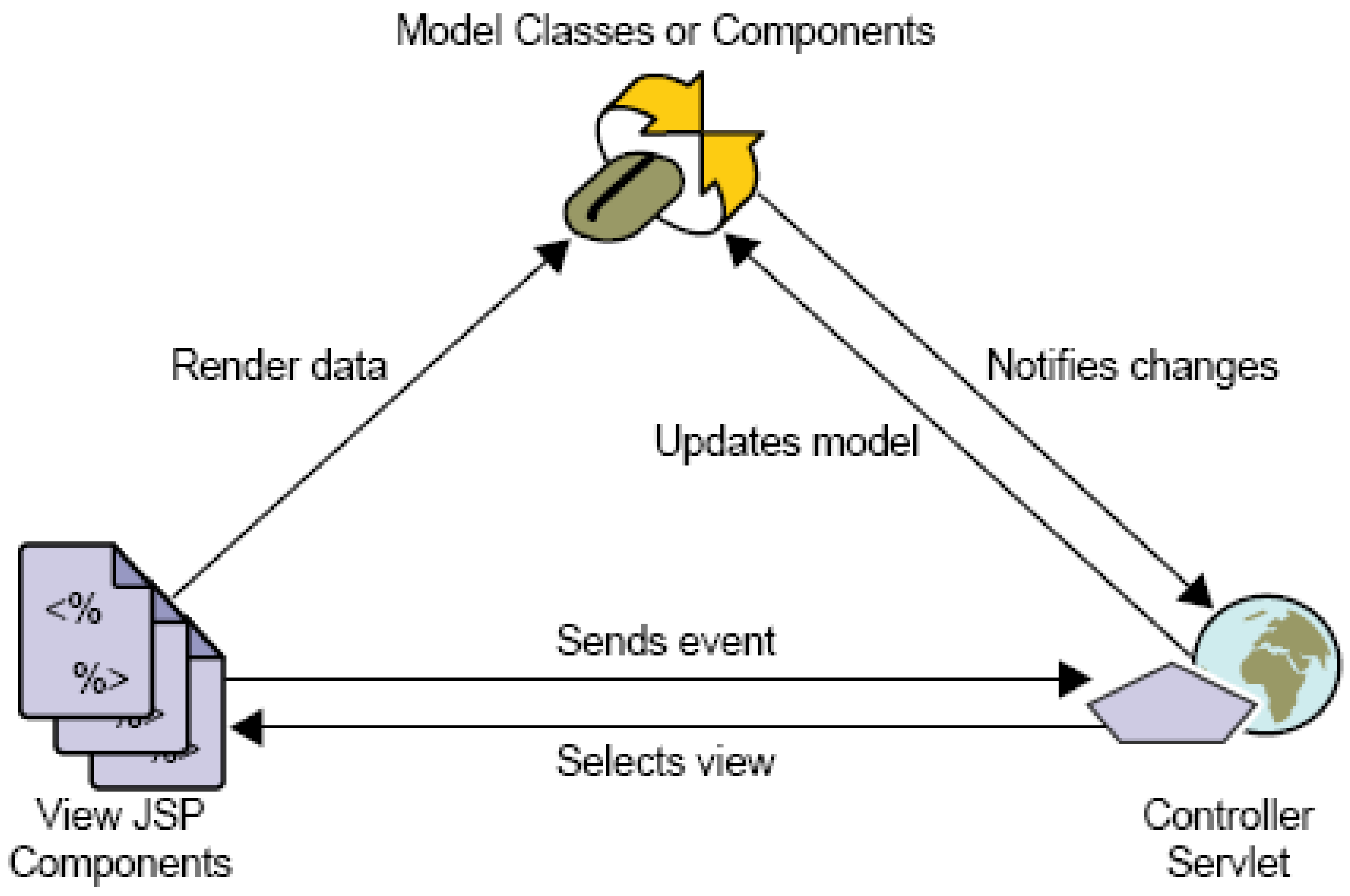


MVC pattern (Model 2)

- **Model 1 Architecture**
  - JavaBean (Model) to represent the business logic
  - JSP (View) to handle all the request processing and provide the presentation

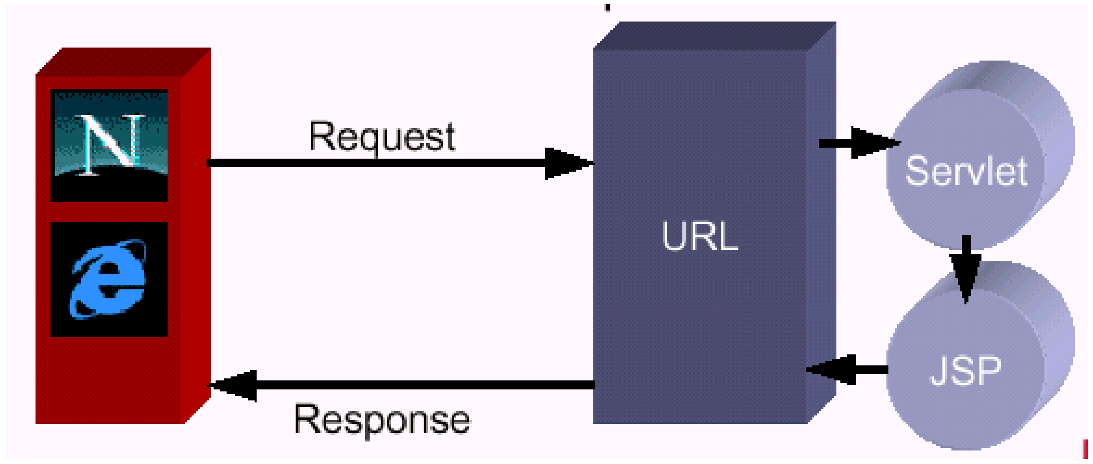
- **Model 2 Architecture (Model-View-Controller)**
  - Often called the "**MVC**" or "Model 2" approach to JSP
  - JavaBean (Model) to represent the business logic
  - JSP (View) to provide the presentation
  - Servlet (Controller) to handle all the request processing

Model 2 Architecture as MVC



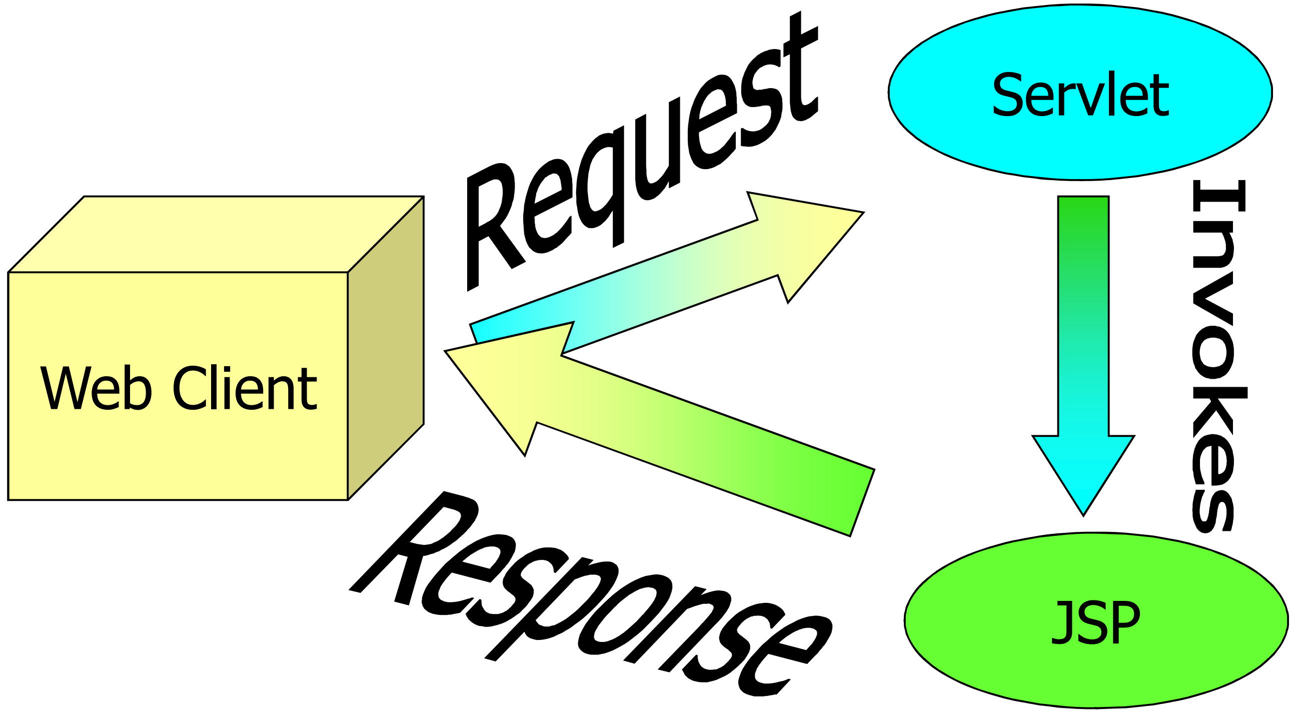
Best practice suggestion

- For complex pages, use both
  - Servlet get request, validate and process
  - `forward()` to JSP for dynamic generated response
  - pass information as request attributes

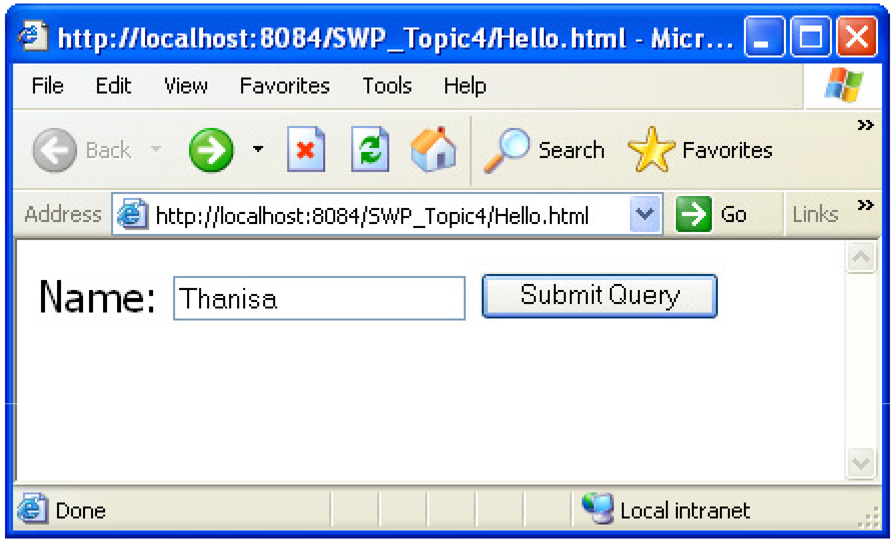


A Typical Web Application

A flexible application with Java Servlet



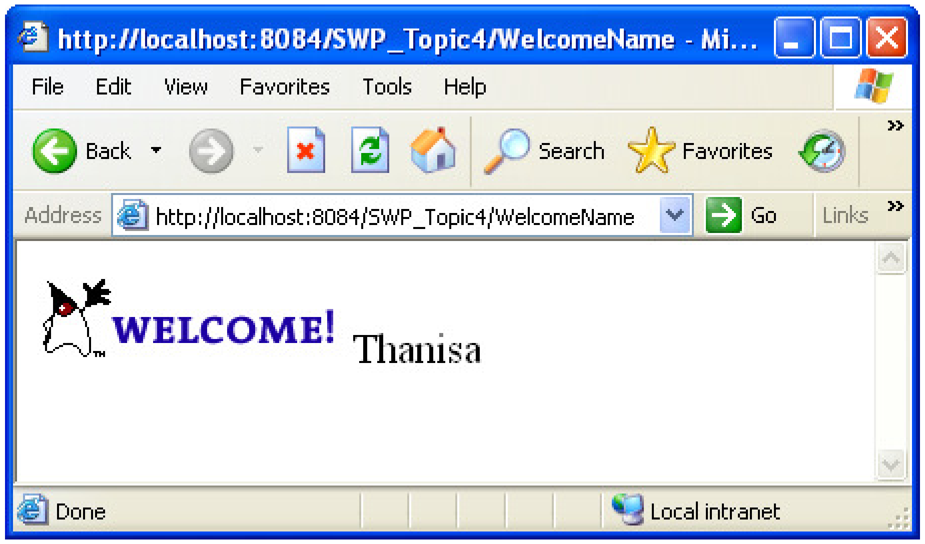
Servlet : `WelcomeName.java`


http://localhost:8084/SWP_Topic4/Hello.html - Micr...
File Edit View Favorites Tools Help
Back Search Favorites
Address http://localhost:8084/SWP_Topic4/Hello.html Go Links
Name: Thanisa Submit Query
Done Local intranet

```
    :
String name = request.getParameter("username");
request.setAttribute("name", name);

RequestDispatcher rd = request.getRequestDispatcher("welcomeName.jsp");
rd.forward(request, response);
    :
```

JSP : **welcomeName.jsp**

```
<img src="Duke.png">
<%= request.getAttribute("name") %>
```


http://localhost:8084/SWP_Topic4/WelcomeName - Mi...

Address http://localhost:8084/SWP_Topic4/WelcomeName

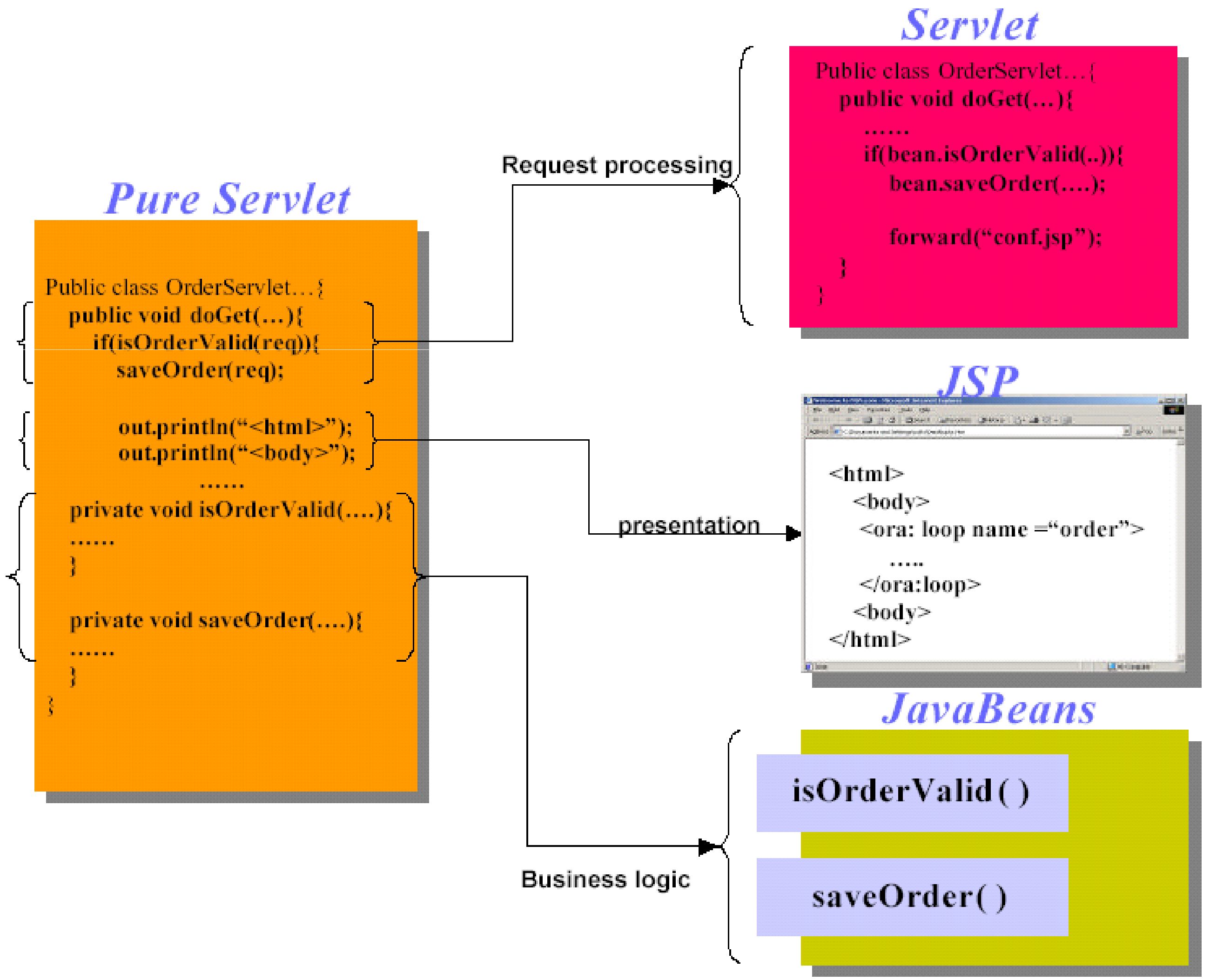
WELCOME! Thanisa

Done

Servlet/JSP Integration

- Joint servlet/JSP process:
  - Original request is answered by a servlet
  - Servlet processes request data, does database lookup,business logic, etc.
  - **Results are placed in beans**
  - Request is forwarded to a JSP page to format result
  - Different JSP pages can be used to handle different types of presentation

Separate Business Logic From Presentation



```
Public class OrderServlet…{
   public void doGet(…){
      if(isOrderValid(req)){
         saveOrder(req);

         out.println("<html>");
         out.println("<body>");
                ……
   private void isOrderValid(….){
   ……
   }

   private void saveOrder(….){
   ……
   }
}
```

```
Public class OrderServlet…{
   public void doGet(…){
      ……
      if(bean.isOrderValid(..)){
         bean.saveOrder(….);

         forward("conf.jsp");
   }
}
```

```
<html>
   <body>
    <ora: loop name ="order">
       …..
    </ora:loop>
   <body>
</html>
```

Servlet : **HelloName.java**

```
      :
  String name = request.getParameter("username");
  myBeans.Person obj = new myBeans.Person();
  obj.setName(name);
  request.setAttribute("user", obj);

  RequestDispatcher rd =
                   request.getRequestDispatcher(“helloName.jsp");
  rd.forward(request, response);
      :
```

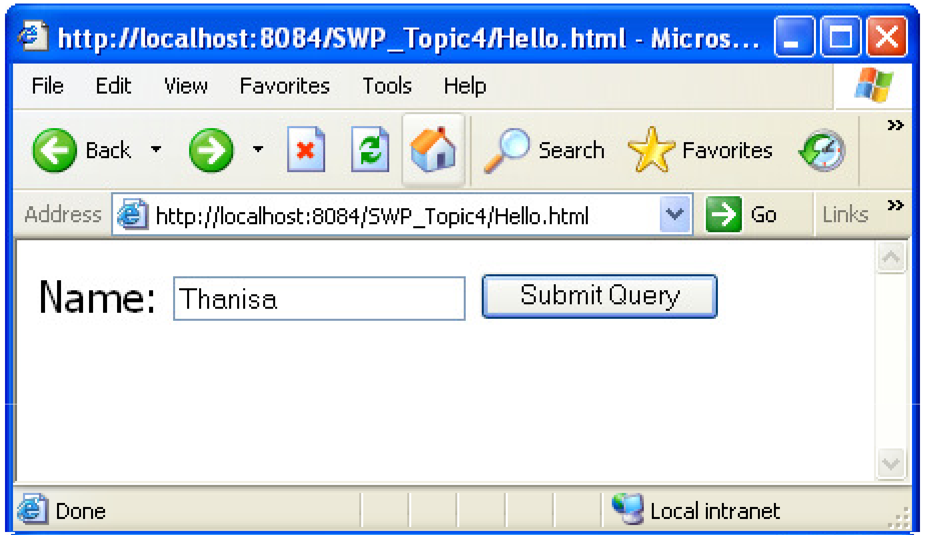
JSP : **helloName.jsp**

```
<img src="Duke.png">
<jsp:useBean id="user" class="myBeans.Person" scope="request" />
<jsp:getProperty name="user" property="name" />
```

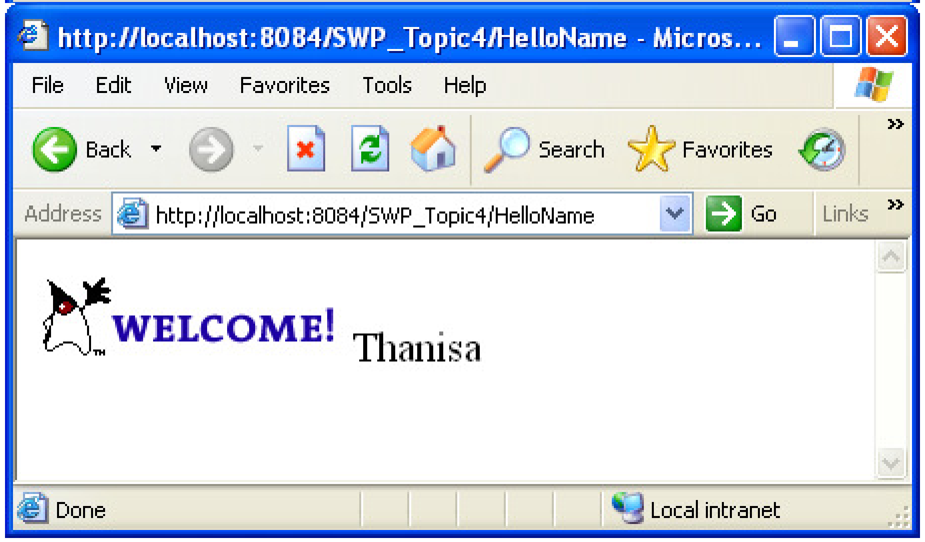
Java Bean : `Person.java`

```
package myBeans;

public class Person {
    private String name;
    public void setName(String name) {
        this.name = name;
    }
    public String getName() {
        return name;
    }
}
```

Servlet, JSP & Java Bean


http://localhost:8084/SWP_Topic4/Hello.html - Micros...
File Edit View Favorites Tools Help
Back Search Favorites
Address http://localhost:8084/SWP_Topic4/Hello.html Go Links
Done
Local intranet

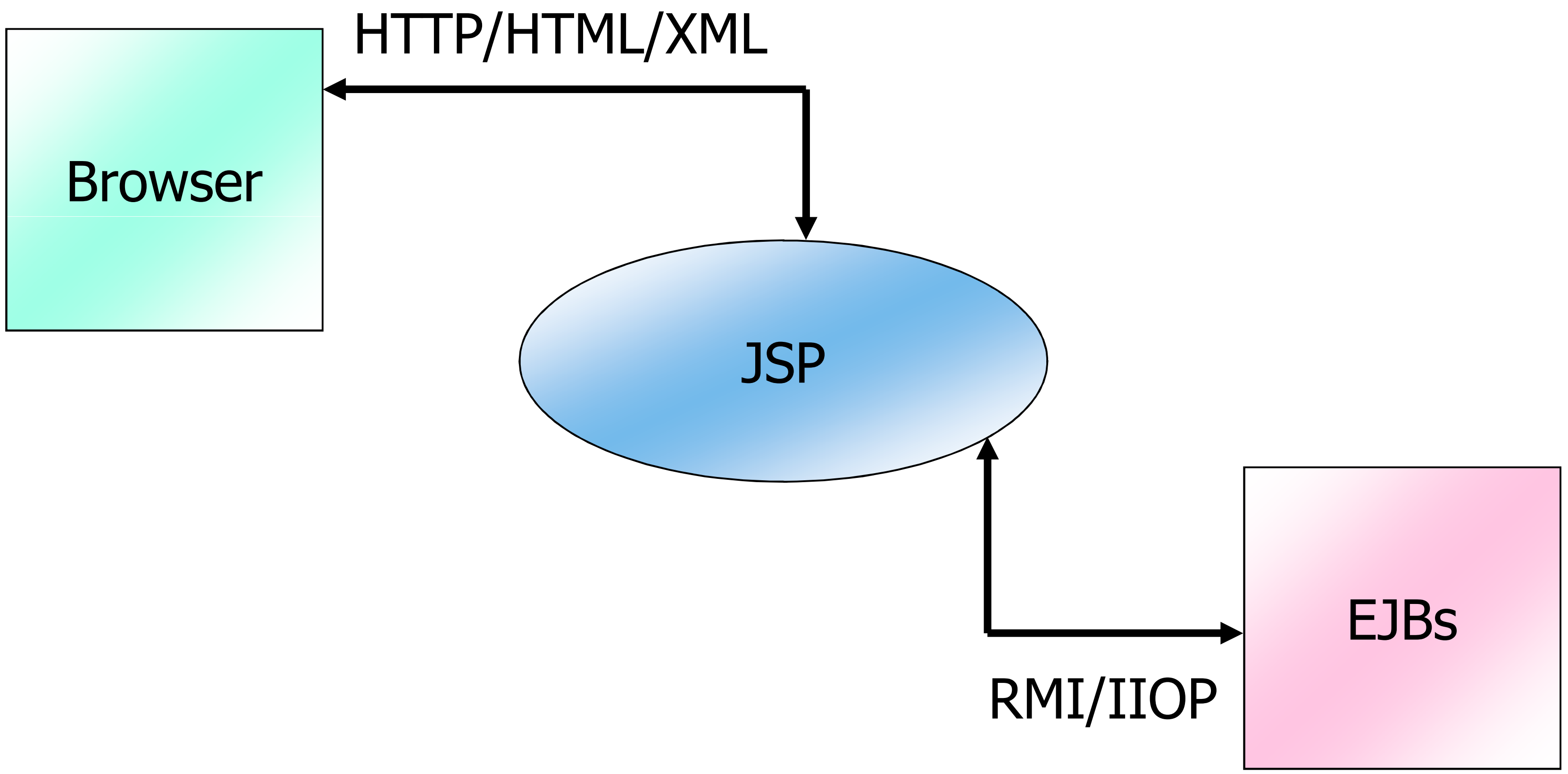
Name: Thanisa Submit Query


http://localhost:8084/SWP_Topic4/HelloName - Micros...
File Edit View Favorites Tools Help
Back Search Favorites
Address http://localhost:8084/SWP_Topic4/HelloName Go Links
Done
Local intranet

WELCOME! Thanisa

A Typical Web Application

JSP combined with Enterprise JavaBeans technology


HTTP/HTML/XML
Browser
JSP
EJBs
RMI/IIOP