

Chapter 9

Java Server Pages (JSP)

Lesson Objectives

At the end of this lesson, you should be able to

- Create and use JavaServer Pages (JSPs).
- Use JavaBeans components in JSP.
- Forward requests from one JSP to another.

Introduction

Servlets can be used to generate dynamic Web content but has a drawback

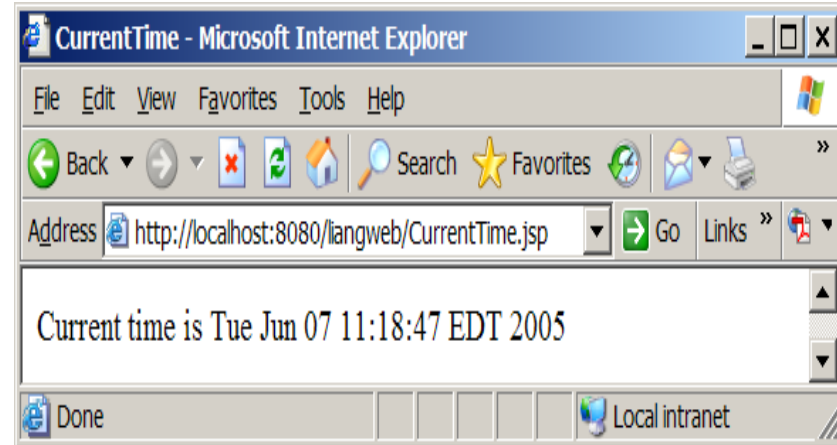
- you have to embed HTML tags and text inside the Java source code
- the program is difficult to read and maintain.

JavaServer Pages (JSP) was introduced to overcome this drawback.

- JSP enables you to write HTML script in the **normal way**
- embed Java code to produce dynamic content.

A Simple JSP

```
<!-- CurrentTime.jsp -->
<HTML>
<HEAD>
<TITLE>
CurrentTime
</TITLE>
</HEAD>
<BODY>
Current time is <%= new java.util.Date() %>
</BODY>
</HTML>
```



Demo:

CurrentTime.jsp

Note:

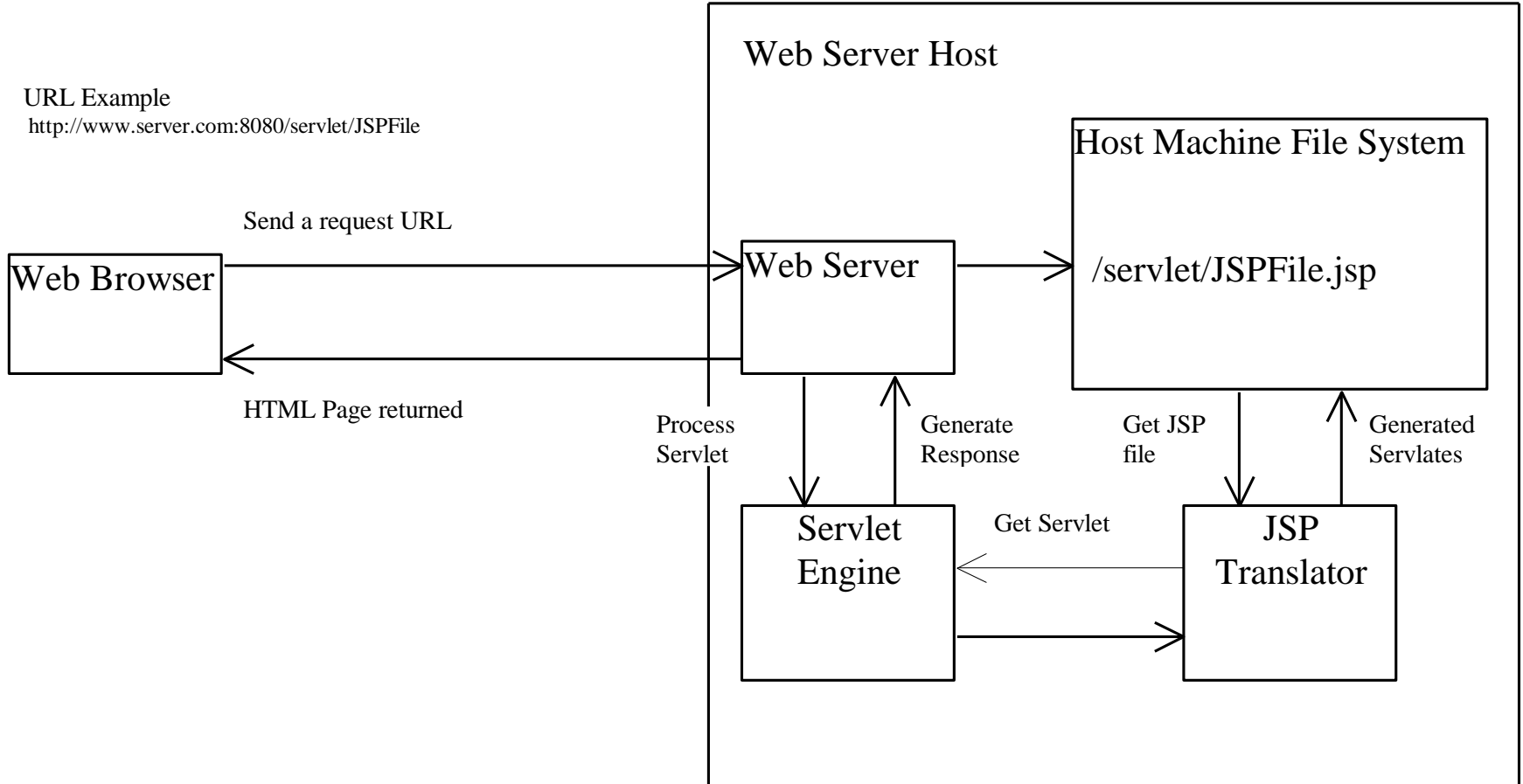
JSP pages must be stored in a file with a **.jsp** extension



Create a JSP Using NetBeans

1. Create a new Web project .
2. Right-click the Web project node in the project pane and choose **New > JSP** to display the New JSP dialog box.
3. Enter **CurrentTime** in the **JSP File Name** field and click *Finish*. You will see **CurrentTime.jsp** appearing under the Web Pages node in the Web project.
4. Complete the code for **CurrentTime.jsp**.
5. Right-click **CurrentTime.jsp** in the project pane and choose **Run File**. The JSP page will be displayed in a Web browser.

How is a JSP Processed?



JSP Page Processing

A JSP page must first be processed by a Web server before it can be displayed in a Web browser. The Web server performs the following for a `.jsp` file:

1. Translate the `.jsp` file into a Java servlet.
2. Compile the servlet.
3. Execute the servlet. The result of the execution is sent to the browser for display.

Note:

A JSP page is translated into a servlet when the page is requested for the first time. It is not retranslated if the page is not modified.

JSP Scripting Constructs

There are three types of scripting constructs you can use to insert Java code into the resultant servlet:

- *expressions*
- *scriptlets*
- *declarations*

JSP Expression

- A JSP expression is used to insert a Java expression directly into the output. It has the following form:

<%= Java-expression %>

- The **expression is evaluated, converted into a string**, and sent to the output stream of the servlet.

Note:

- **There is no semicolon at the end of a JSP expression. For example, <%= i; %> is incorrect.**

JSP Scriptlet

A JSP scriptlet enables you to insert a Java statement into the servlet's **jspService** method, which is invoked by the **service** method. A JSP scriptlet has the following form:

```
<% Java statement %>
```

JSP Declaration

A JSP declaration is for declaring methods or fields into the servlet. It has the following form:

<%! Java method or field declaration %>

JSP Comment

HTML comments have the following form:

<!-- HTML Comment -->

If you don't want the comment appear in the resultant HTML file, use the following comment in JSP:

<%-- JSP Comment --%>

Demo: Factorial.jsp

<HTML>

<HEAD>

<TITLE>Factorial</TITLE>

</HEAD>

<BODY>

<% for (int i = 0; i <= 10; i++) { %>

Factorial of <%= i %> is

<%= computeFactorial(i) %>

<% } %>

<%! private long computeFactorial(int n) {

if (n == 0)

return 1;

else

return n * computeFactorial(n - 1);

}

%>

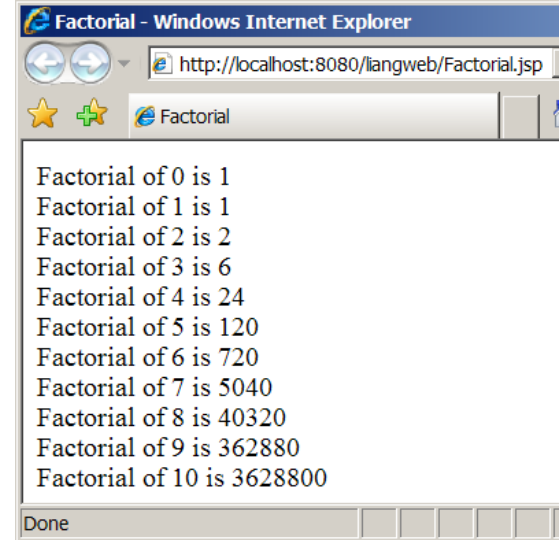
</BODY>

</HTML>

JSP scriptlet

JSP expression

JSP declaration



Note: For JSP loop body, even though it contains a single statement, must be placed inside braces.



P9 Q1

JSP Predefined Variables

You can use variables in JSP. For convenience, JSP provides 8 predefined variables from the servlet environment that can be used with JSP expressions and scriptlets. These variables are also known as *JSP implicit objects*:

- request
- response
- out
- session
- application
- config
- pagecontext
- page

Description of JSP Predefined Variables – 1/3

request

- Represents the client's request, which is an instance of `HttpServletRequest`. You can use it to access request parameters, HTTP headers such as cookies, hostname, etc.

response

- Represents the servlet's response, which is an instance of `HttpServletResponse`. You can use it to set response type and send output to the client.

out

- Represents the character output stream, which is an instance of `PrintWriter` obtained from `response.getWriter()`. You can use it to send character content to the client.

Description of JSP Predefined Variables – 2/3

session

- Represents the **HttpSession** object associated with the request, obtained from **request.getSession()** .

application

- Represents the **ServletContext** object for storing persistent data for all clients.
- The difference between **session** and **application** is
 - session is tied to one client
 - application is for all clients to share persistent data.

config

- Represents the **ServletConfig** object for the page.

Description of JSP Predefined Variables – 3/3

pagecontext

- Represents the `PageContext` object.
- `PageContext` is a class introduced in JSP to give a central point of access to many page attributes .

page

- `page` is an alternative to `this`.

Demo

- **ComputeLoan.html**
 - A HTML page that prompts the user to enter loan amount, annual interest rate and number of years. Clicking the **Compute Loan Payment** button invokes **ComputeLoan.jsp** to compute and display the monthly and total loan payments.
- **ComputeLoan.jsp**
 - Obtains the parameter values using the predefined variable **request** and computes the monthly and total payments.

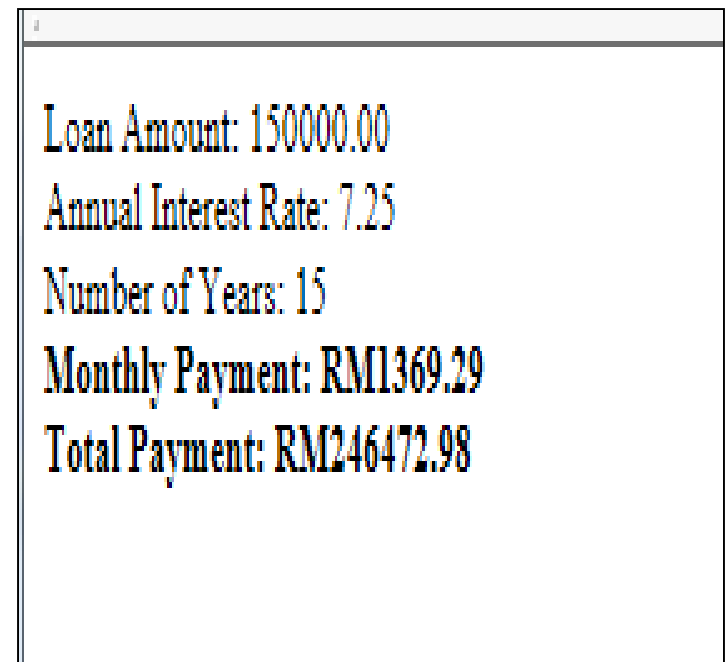


Compute Loan Payment

Loan Amount

Annual Interest Rate

Number of Years



Loan Amount: 150000.00
Annual Interest Rate: 7.25
Number of Years: 15
Monthly Payment: RM1369.29
Total Payment: RM246472.98

JSP Directives

- A JSP directive is a statement that gives the JSP engine information about the JSP page.

E.g., if your JSP page uses a Java class from a package other than the `java.lang` package, you have to use a directive to import this package.

- The general syntax for a JSP directive is as follows:

```
<%@ directive attribute="value" %>, or  
<%@ directive attribute1="value1"  
                attribute2="value2"  
                ...  
                attributen="valuen" %>
```

3 JSP Directives

- **page**
 - Lets you provide information for the page, such as importing classes and setting up content type. The **page** directive can appear anywhere in the JSP file.
- **include**
 - Lets you **insert** a file to the servlet when the page is translated to a servlet. The **include** directive must be placed where you want the file to be inserted.
- **taglib**
 - Lets you define custom tags.

Attributes for page Directives (1/3)

- **import**
 - Specifies the packages to be imported for this page. E.g.,
`<%@page import = "java.util.*, java.text.*" %>`
- **contentType**
 - Specifies the MIME type for the resultant JSP page. The default is `text/html`. (Note: the default content type for servlets is `text/plain`.)
- **session**
 - Specifies a `boolean` value to indicate whether the page is part of the session. By default, `session` is `true`.
- **buffer**
 - Specifies the output stream buffer size. By default, it is 8KB. E.g., `<%@page buffer="10KB" %>`,
`<%@page buffer="none" %>`

Attributes for page Directives (2/3)

- **autoFlush**
 - Specifies a `boolean` value to indicate whether the output buffer should be automatically flushed when it is full or whether an exception should be raised when the buffer overflows. The default is `true`. In this case, the buffer attribute cannot be `none`.
- **isThreadSafe**
 - Specifies a `boolean` value to indicate whether the page can be accessed simultaneously without data corruption. The default is `true`. If it is set to `false`, the JSP page will be translated to a servlet that implements the `SingleThreadModel` Interface.

Attributes for page Directives (3/3)

- **errorPage**

- Specifies a JSP page that is processed when an exception occurs in the current page. E.g.,

`%@page errorPage = "HandleError.jsp" %`

specifies that HandleError.jsp is processed when an exception occurs.

- **isErrorPage**

- Specifies a **boolean** value to indicate whether the page can be used as an error page. The default is **false**.

Demo

- `\domain\Loan.java`
- `ComputeLoan2.html`
- `ComputeLoan2.jsp`
 - Uses the `Loan` class to simplify the code.



P9 Q2

JavaBeans Component in JSP

A class is a JavaBeans component if it has the following 3 features:

- The class is public.
- The class has a public no-arg constructor.
- The class is serializable.

To enable an object to be **shared among different pages**, its class must be a JavaBeans component.

Using JavaBeans in JSP

To create an instance for a JavaBeans component, use

```
<jsp:useBean id="objectName"  
  scope="scopeAttribute"  
  class="ClassName"/>
```

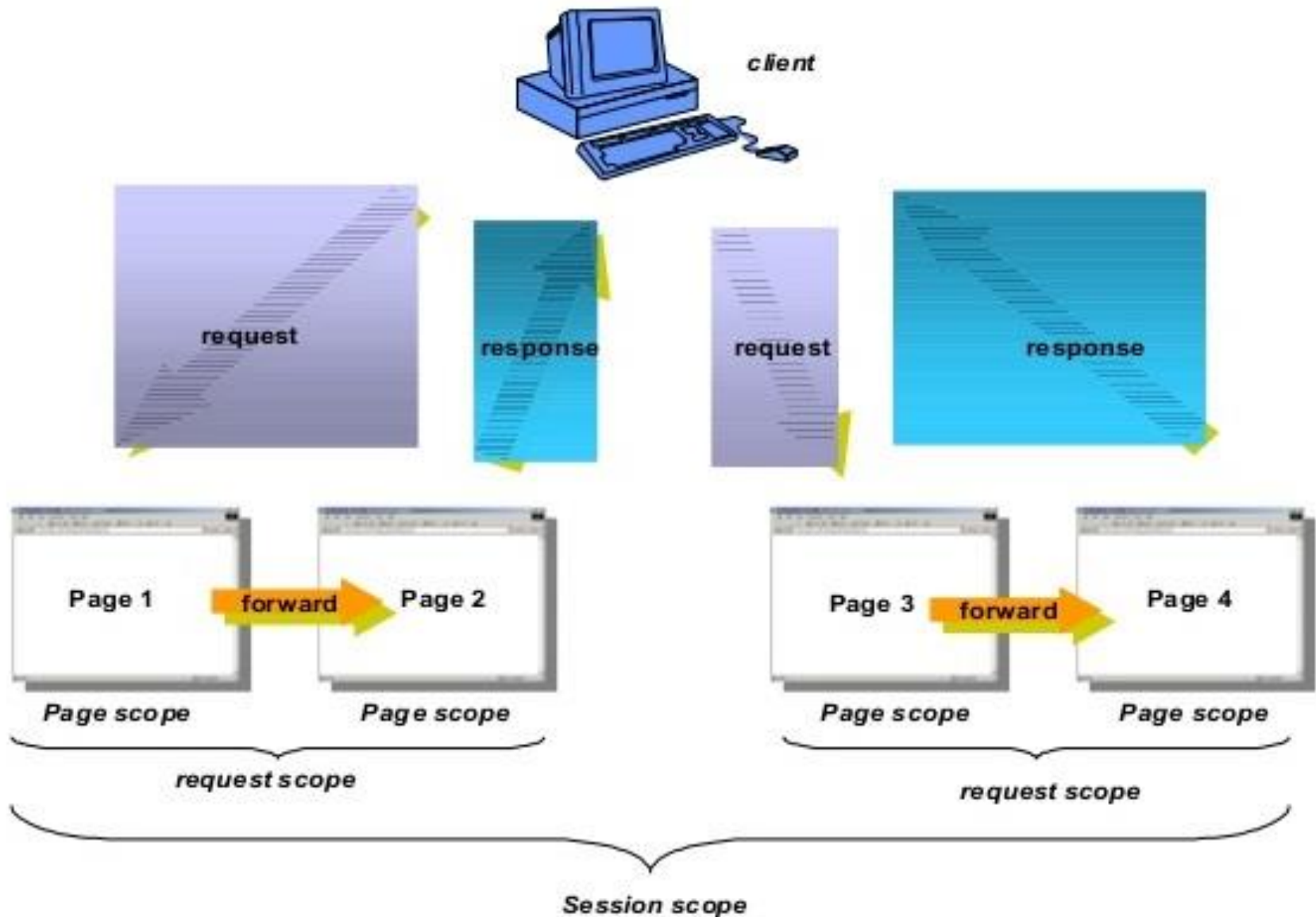
This syntax is equivalent to

```
<% ClassName objectName=new ClassName() %>
```

except that the **scope** attribute is missing.

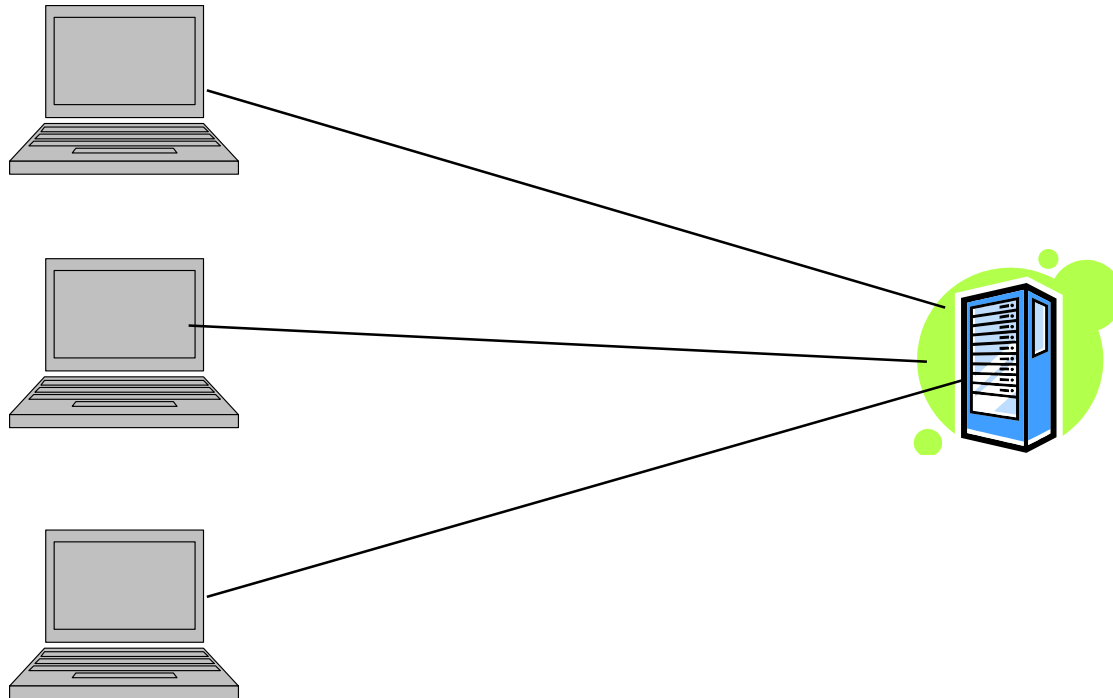
- The **scope** attribute specifies the scope of the object and the object is not recreated if it is already within the scope.
- 4 types of attribute scope: page, request, session and application

Attribute scopes



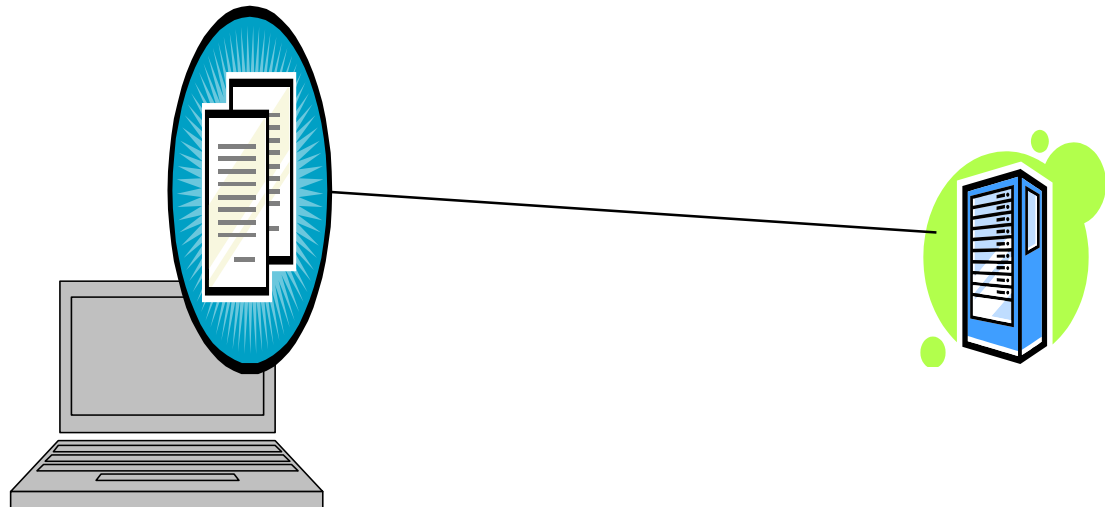
Scope Attributes: **application**

- A web application is a collection of web pages, scripts, servlets, etc. that form a web site.
- Specifies that the object is bound to the application.
The object can be **shared by all sessions** of the application.



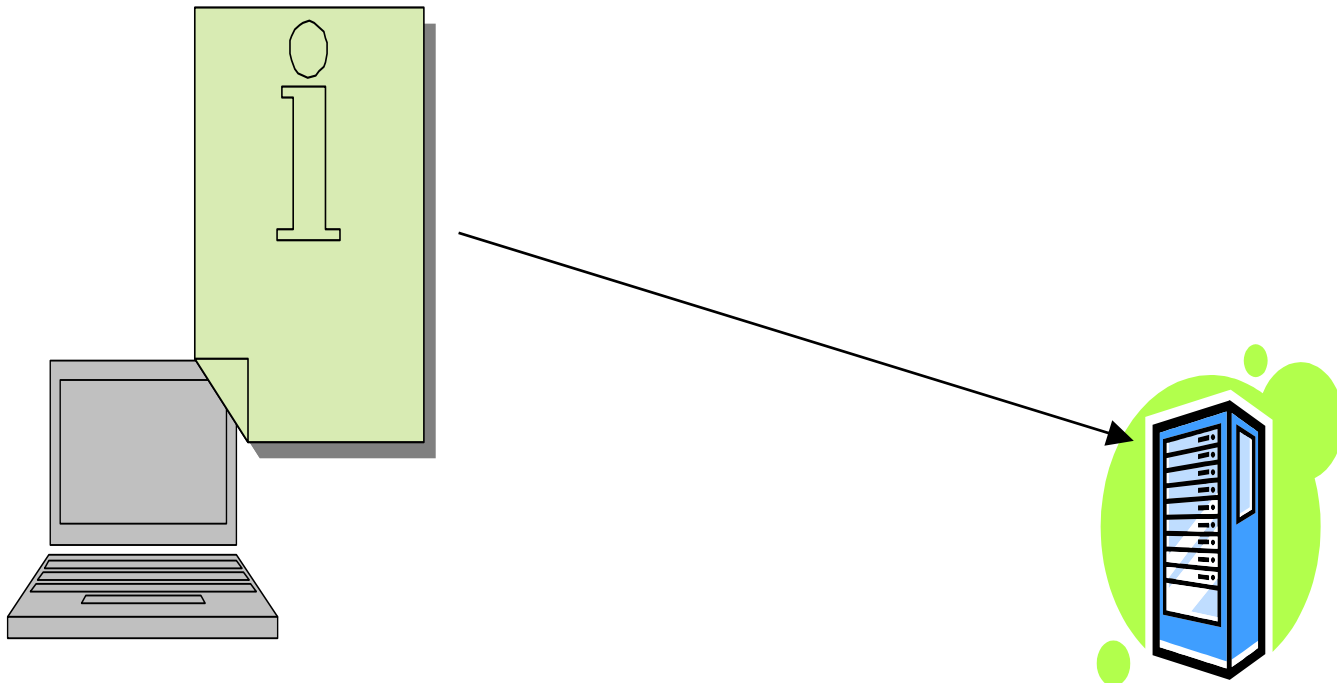
Scope Attributes: **session**

- A session is a series of requests from the same web browser instance to the same web server.
- If the web browser window is closed and a new window opened, a new session is started.
- When a client from the **same browser accesses 2 servlets / JSPs** on the same server, the session is the same.
- Specifies that the object is bound to the client's **session**.



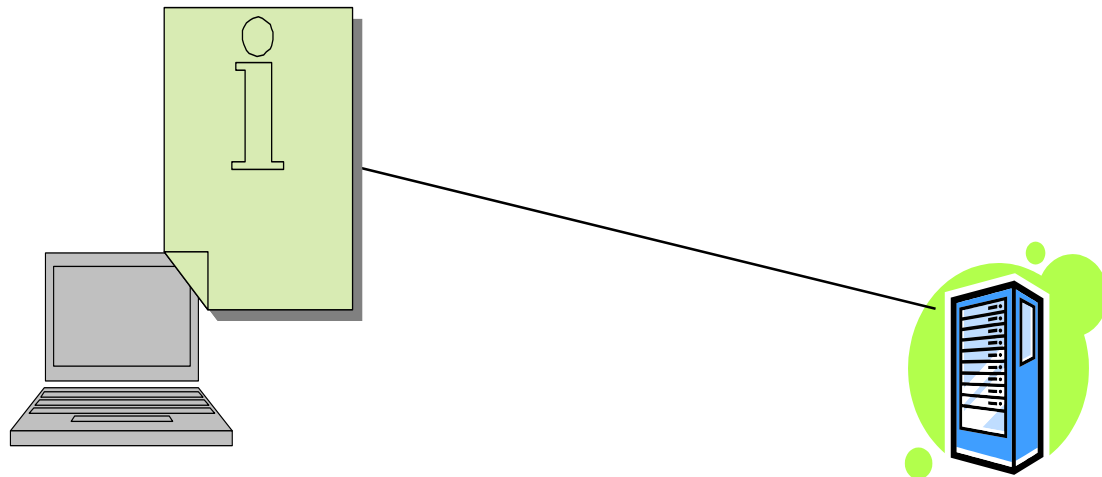
Scope Attributes: `request`

- A request exists from the time the web browser sends it to the web server until the web server has made its response.
- If the request is forwarded to another page, the object is still available but not for a redirect.



Scope Attributes: page

- The **default** scope, which specifies that the object is bound to the page.
- The object is valid only during the processing of the current response; once the response is sent back to the browser, the object is no longer valid. If the request is forwarded to another page or the browser makes another request as a result of a redirect, the object is also lost.



How Does JSP Find an Object

When

```
<jsp:useBean id="objectName"  
    scope="scopeAttribute" class="ClassName"/>
```

is processed, the JSP engine -

1. Searches for the object of the class with the same **id** and **scope**.
2. If found, the pre-existing bean is used;
3. otherwise, a new bean is created.

Alternative Syntax for Creating a Bean

```
<jsp:useBean id="objectName"  
  scope="scopeAttribute" class="ClassName">  
  some statements  
</jsp: useBean>
```

The statements are executed when the bean is created.

If the bean with the same **id** and **class** name already exists, the statements are not executed.

Demo

- **\domain\Count.java**
 - Creates a JavaBeans component name **Count** to be used to count the number of visits to a JSP page.
- **TestBeanScope.jsp**
 - The line **scope="application"** specifies that the bean is alive in the JSP engine and available for all clients to access. The bean can be shared by any client with the directive

```
<jsp:useBean id="count" scope="application" class="Count">
```

- Every client accessing **TestBeanScope.jsp** causes the count to be increased by 1. The first client causes `count` to be created, and subsequent access **TestBeanScope** to uses the same object.

TestBeanScope with different bean scopes

- If the **scope="session"**, the scope of the bean is limited to the session from the same browser. The count will increase only if the page is requested from the same browser.
- If the **scope="page"**, the scope of the bean is limited to the page, and any other page cannot access this bean. The count will always display count 1.
- If the **scope="request"**, the scope of the bean is limited to the client's request, and any other request on the page will always display count 1.
- If the page is destroyed, the count restarts from 0. To keep the latest count value, store the count in a binary file or a database table.

Demo

- `\domain\Count2.java`
 - This class stores and retrieves the count in the **VisitorCount** table in the **collegedb** database.
- `TestBeanScope2.jsp`

Getting and Setting Properties

By convention, a JavaBeans property provides the **set** and **get** methods for reading and modifying its private properties.

You can **get** the property in JSP using:

```
<jsp:getProperty name="beanId" property="age"/>
```

This is equivalent to:

```
<%= beanId.getAge() %>
```

You can **set** the property in JSP as follows:

```
<jsp:setProperty name="beanId" property="age"  
value="30"/>
```

This is equivalent to:

```
<% beanId.setAge(30) ; %>
```

Associating Properties with Input Parameters (1/2)

Often properties are associated with input parameters. To get the value of the input parameter named **score** and set it to the JavaBeans property named **score**:

Method 1:

```
<% double score =  
Double.parseDouble(request.getParameter("score"));  
%>
```

```
<jsp:setProperty name="beanId" property="score"  
value="<%= score %>" />
```

This is equivalent to:

```
<% beanId.setScore(score) >
```


Associating Properties with Input Parameters (2/2)

Method 2 (more convenient syntax):

```
<jsp:setProperty name="beanId" property="score"  
param="score" />
```

Instead of using the **value** attribute, use the **param** attribute to name an input parameter. The value of this parameter is set to the property.

Associating All Properties with Input Parameters

Often the bean property and the parameter (from the form) have the same name.

Use the following statement to associate **all** the bean properties in **beanId** with the parameters that **match** the property names:

```
<jsp:setProperty name="beanId" property="*" />
```

Note:

Simple **type conversion** is performed **automatically** when a bean property is associated with an input parameter. A string input parameter is converted to an appropriate primitive data type or wrapper class for a primitive type.

Demo

- `ComputeLoan3.html`
- `ComputeLoan3.jsp`

```
<jsp:useBean id="loan" class="domain.Loan"
  scope="page" ></jsp:useBean>
```

- Creates a bean named `loan` for the `Loan` class.

```
<jsp:setProperty name="loan" property="*" />
```

- Associates the bean properties `loanAmount`, `annualInterestRate`, and `numberOfYears` with the input parameter values and performs type conversion automatically.

Lines 10-12: Uses the accessor methods of the loan bean to get the loan amount, annual interest rate and number of years.



Redo **P9 Q2** using the
jsp:setProperty tag

Demo

- `domain\FactorialBean.java`
- `FactorialBean.jsp`
 - The `jsp:useBean` tag creates a bean `factorialBeanId` of the `FactorialBean` class.
 - `<jsp:setProperty name="factorialBeanId" property="*" />` associates all the bean properties with the input parameters that have the same name. In this case, the bean property `number` is associated with the input parameter `number`. When you click the **Compute Factorial** button, JSP automatically converts the input value for `number` from string into `int` and sets it to `factorialBeanId` before other statements are executed.

Design Guide

Mixing a lot of Java code with HTML in a JSP page makes the code difficult to read and to maintain.

You should move the Java code to a **.java** file as much as you can.

Demo:

- **NewFactorialBean.java**
- **NewFactorialBean.jsp**

Note: The java code for formatting the output has been moved to the **.java** class.

Demo : Displaying International Time

- **domain\TimeBean.java**
 - Obtains all the available locales in the array **allLocale** and all time zones in the array **allTimeZone**. The bean properties **localeIndex** and **timeZoneIndex** are defined to refer to an element in the arrays.
- **DisplayTimeForm.jsp**
 - Imports the **TimeBean** class. The bean that is created is used to return all locales and time zones. The scope of the bean is application, so the bean can be shared by all sessions of the application.
- **DisplayTime.jsp**
 - This page is invoked from **DisplayTimeForm.jsp** to display the time with the specified locale and time zone. The character encoding for the page is set to **GB18030** for displaying international characters (default is **UTF-8**). Imports the **TimeBean** class and creates a bean using the same id as in the preceding page. Since the object is already created in the preceding page, the **timeBeanID** in this page and in the preceding page point to the same object.

- **AddNewProgramme.html**
- **domain\Programme.java**
- **da\ProgrammeDA.java**
- **GetProgrammeData.jsp**
 - Creates a bean and obtains the property values from the form (**AddNewProgramme.html**). Then, it displays the data for user confirmation. When the user clicks the Confirm button, it invokes **StoreProgramme.jsp**.
- **StoreProgramme.jsp**
 - For storing the data into the database. The same **programmeId** is shared with the preceding page (**GetProgrammeData.jsp**) within the scope of the same session. A bean for **ProgrammeDA** is created with the scope of application.

Programme Code *

Programme Name *

Faculty

* required fields

You entered the following data

Programme code: BC

Programme name: Biochemistry

Faculty: FASC

Note: Appropriate Scopes

The scope for **programmeId** is *session*, but the scope for **programmeDataId** is *application*.

Why?

- **GetProgrammeData.jsp** obtains programme information, and **StoreProgramme.jsp** stores the information in the same session. So the *session* scope is appropriate for **programmeId**.
- All the sessions access the same database and use the same prepared statement to store data. With the *application* scope for **programmeDataId**, the bean for **ProgrammeDA** needs to be created just once.



P9 Q3

Tips

- Exceptions
 - The **addRecord()** method in **StoreProgramme.jsp** may throw a **java.sql.SQLException**.
 - In JSP, you can omit the try-block for checked exceptions. In case of an **exception, JSP displays an error page**.
- Using *beans*
 - Using beans is an effective way to develop JSP. You should put Java code into a bean as much as you can.
 - The bean not only simplifies JSP programming, but also makes code reusable. The bean can also be used to implement persistent sessions.

Forwarding Requests from JSPs

Web applications developed using JSP generally consist of many pages linked together. JSP provides a forwarding tag in the following syntax that can be used to forward a page to another page.

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

Demo

This demo creates a JSP database application that browses tables. When you start the application, the first page prompts the user to enter the *database name*, *username*, and *password* for a database. After you login to the database, you can select a *table* to browse. Upon clicking the Browse Table Content button, the table content is displayed.

- **da/DBBean.java**
- **DBLogin.html**
- **DBLoginInitialization.jsp**
 - When this page is processed for the first time, an instance of **DBBean** named **dBBeanID** is created. The input parameters *database name*, *username*, and *password* are passed to the bean properties. The **initializaJdbc** method loads the driver and establishes connection to the database. If login fails, the **connection** property is `null` and an error message is displayed. If login succeeds, control is forwarded to **Table.jsp**.
- **Table.jsp**
- **BrowseTable.jsp**
 - Shares **dBBeanID** with **DBLoginInitialization.jsp** and **Table.jsp** in the same session. It retrieves the table contents for the selected table from **Table.jsp**.

Review of Lesson Objectives

You should now be able to

- Create and use JavaServer Pages (JSPs).
- Use JavaBeans components in JSP.
- Forward requests from one JSP to another.