

Java Server Pages

JSP Life Cycle

JSP Elements

JSP Directives

JSP Implicit Objects

Techniques:

MVC Design Pattern

Dynamic DB Connection

Review

- **Security Mechanisms**

- Authentication (BASIC, DIGEST, FORM)
- Authorization
- Data Integrity
- Confidentiality (CLIENT-CERT or HTTPS CLIENT)
- Applied JAAS and ACLs file
- 2 types: Declarative or Programmatic, or Mixing
- How to implement Security on Web application with supporting Web/Servlet container

Objectives

- **Java Server Pages**
- **JSP Life Cycle**
 - JSP Translation and Execution
 - The JSP Translation Phase
 - The JSP Request/Execution Phase
- **JSP Elements**
 - Anatomy of a JSP Page
 - Template Data
 - Elements of a JSP Page
- **JSP Directives**
 - Directives
- **JSP Implicit Objects**
 - JSP Implicit Objects
- **Techniques: MVC Design Pattern & Dynamic DB Con**

Java Server Pages

JSP

- Java Server Page (JSP) is a **server side script language** running web (application) server (Tomcat, Sun, JBoss ...)
- Saved with **.jsp extension**
- A simple, yet powerful Java technology for **creating and maintaining dynamic-content** webs pages (embedded)
- JSP page **are converted** by the web container **into a Servlet instance**
- It focus on the **presentation logic** of the web application
- JSP page **contains HTML tags**
- JSP page contains tags (standard & custom), which are used to generate dynamic content and invoke the operations on Javabeans components and processing requests.
- **A combination of HTML, XML, Servlet** (extends from Servlet), **and Java Code** to create dynamic Web content.

Java Server Pages

JSP

- Benefits

- Segregation of the work profiles of a Web designer and a Web developer (**separating presentation logic and content/business/processing logic**)
- Emphasizing Reusable Components (**JavaBeans**)
- Simplified Page Development (**easy to use JSP through tag, flexibility, scalability**)
- Access & instantiate JavaBeans component (**support tag element with get/set functions**)
- High secure

- Choosing Servlet or JSP

- **Servlet** are well suited for **handling binary data dynamically**, for example, for uploading files or for creating dynamic images, since they **need not contain any display logic**
- **JSP** is **easy to create the presentation logic with dynamic generated data** combine template and **do not compile before executing or running time**

Java Server Pages

JSP – Example

- **Simple JSP page** displays current date and time

```
<html>
```

```
<head>
```

```
<title>A simple date</title>
```

```
</head>
```

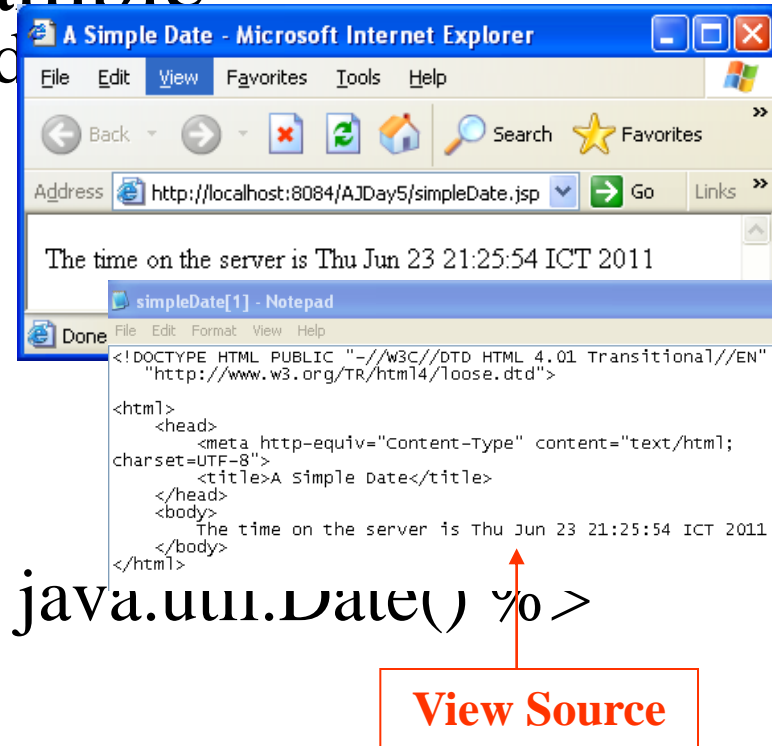
```
<body>
```

```
The time on the server is <%= new java.util.Date() %>
```

```
</body>
```

```
</html>
```

- Server processes JSP components converting static data on **HTML** which can be displayed by Web browser
- To testing JSP page, the JSP page should be copied to the **ROOT** of web server – Tomcat



Java Server Pages

JSP – In Nature

- When the JSP page is requested to server, the JSP page is converted to java file as **filename_jsp.java** (**filename.jsp**)
- The filename_jsp.java file is **compiled** if it is correct syntax
- **Ex:**
 - Omit the “)” of Date function in the simpleDate.jsp
 - Correct above mistake, run the file, then checking the result at
 - C:\Documents and Settings\LoggedUser\.netbeans\6.9\apache-tomcat-6.0.26_base\work\Catalina\localhost
 - C:\Users\LoggedUser\.netbeans\6.9\apache-tomcat-6.0.26_base\work\Catalina\localhost

```
compile:
compile-jsp:
Compiling 1 source file to G:\Laptrinh\JAD\JSPMDL\build\generated\classes
G:\Laptrinh\JAD\JSPMDL\build\generated\src\org\apache\jsp\index_jsp.java:67: ')' expected
    out.print( new java.util.Date( );
    out.print( new java.util.Date( );
1 error
G:\Laptrinh\JAD\JSPMDL\nbproject\build-impl.xml:472: The following error occurred while executing this line:
G:\Laptrinh\JAD\JSPMDL\nbproject\build-impl.xml:224: Compile failed; see the compiler error output for details.
BUILD FAILED (total time: 2 seconds)
```

Java Server Pages

JSP – In Nature

\\Trong Khanh\.netbeans\6.9\apache-tomcat-6.0.26_base\work\Catalina\localhost\



simpleDate_jsp.class
CLASS File
4 KB



simpleDate_jsp.java
Java Source File
3 KB

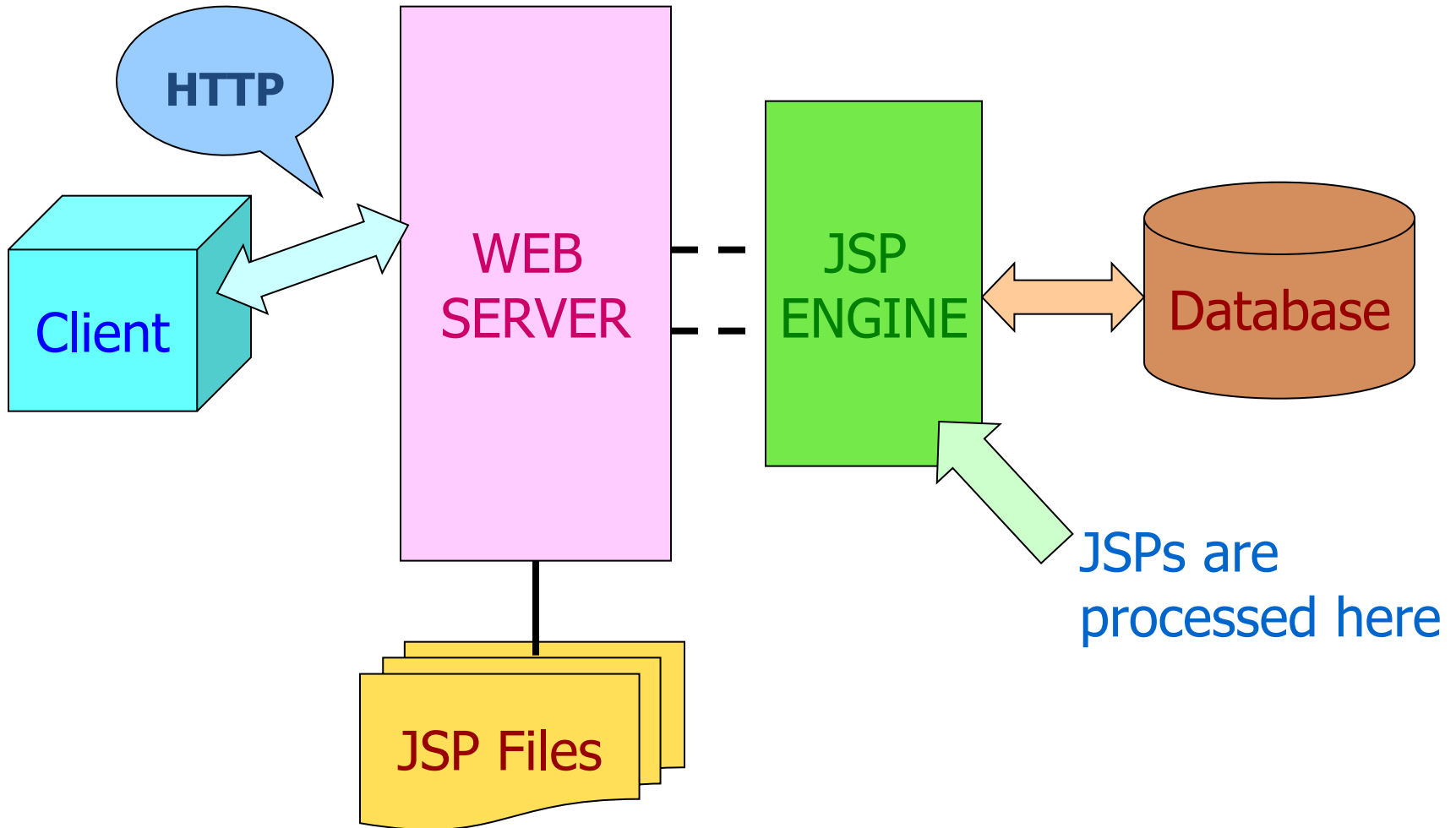
Applying the compile function on file to check JSP file

```
out.write('\n');
out.write('\n');
out.write('\n');
out.write("\n");
out.write("\n");
out.write("<!DOCTYPE HTML PUBLIC \"-//W3C//DTD HTML 4.01 Transitional//EN\"");
out.write("\n");
out.write("<http://www.w3.org/TR/html4/loose.dtd>");
out.write("\n");
out.write("<html>");
out.write("  <head>");
out.write("    <title>A simple date</title>");
out.write("  </head>");
out.write("  <body>");
out.write("    \tThe time on the server is ");
out.print(new java.util.Date());
out.write("\n");
out.write("  </body>");
out.write("</html>");
out.write("\n");
```

```
public void
_jspService(HttpServletRequest
quest request,
HttpServletResponse
response)
throws
java.io.IOException,
ServletException {
```

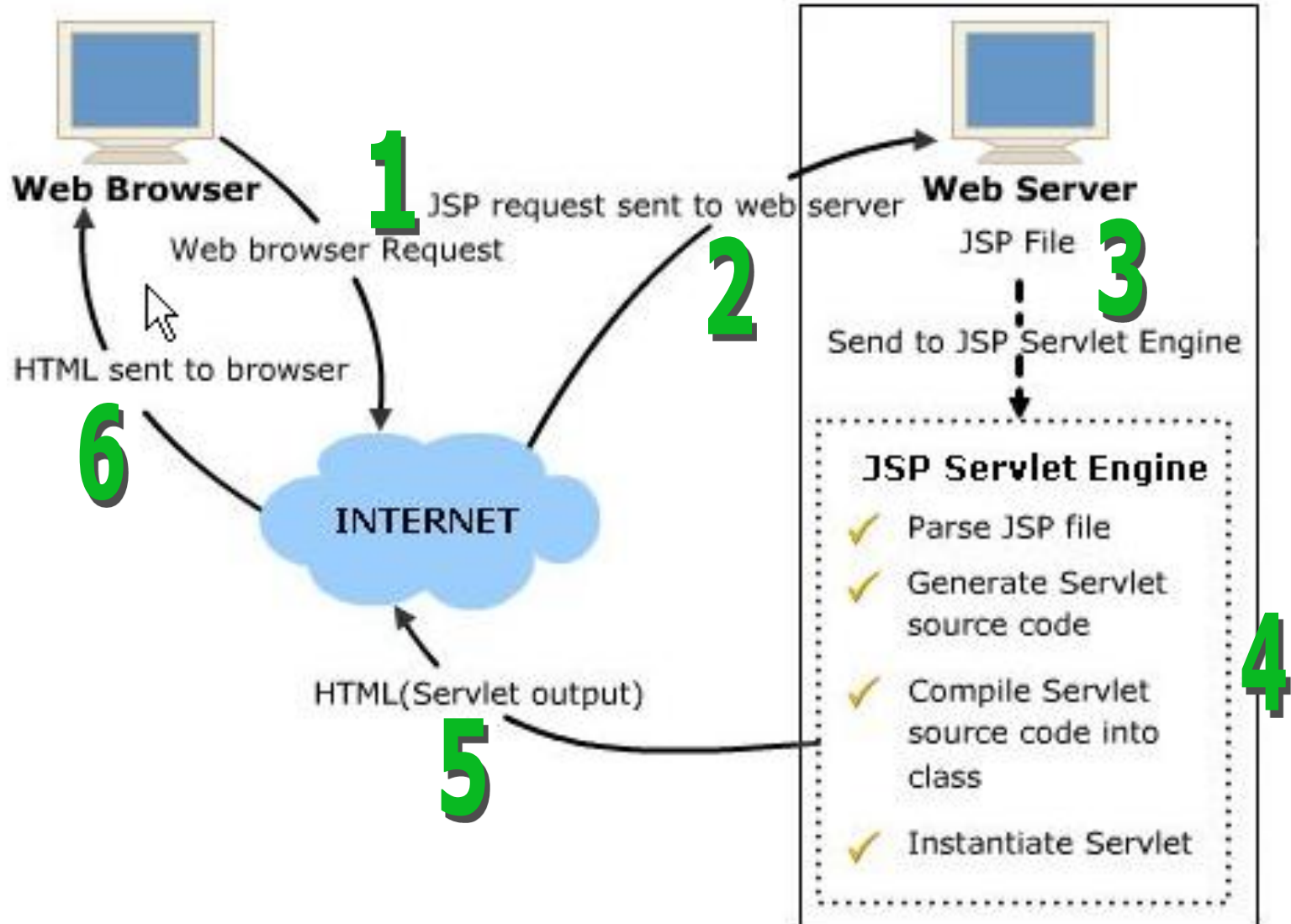

Java Server Pages

JSP Life Cycle



Java Server Pages

JSP Life Cycle



Java Server Pages

JSP Life Cycle

- **Translation**

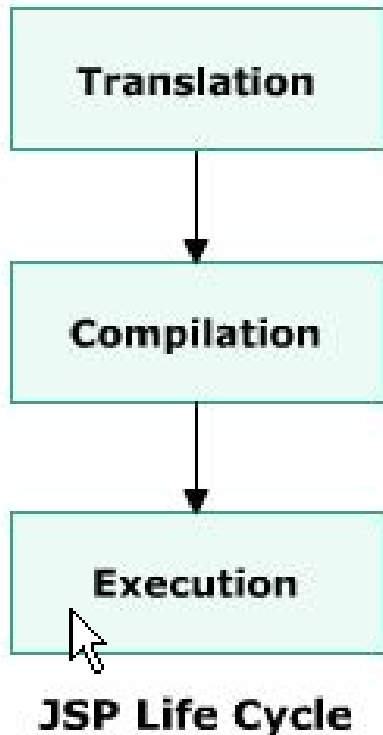
- A servlet code to implement JSP tags is automatically generated, complied and loaded into the servlet container.

- **jspInit()** method is invoked when the JSP page is **initialized** and **requested**

- The **_jspService()** method corresponds to the body of the JSP page and is **defined automatically** by the JSP container and **never** be **defined** by the JSP page

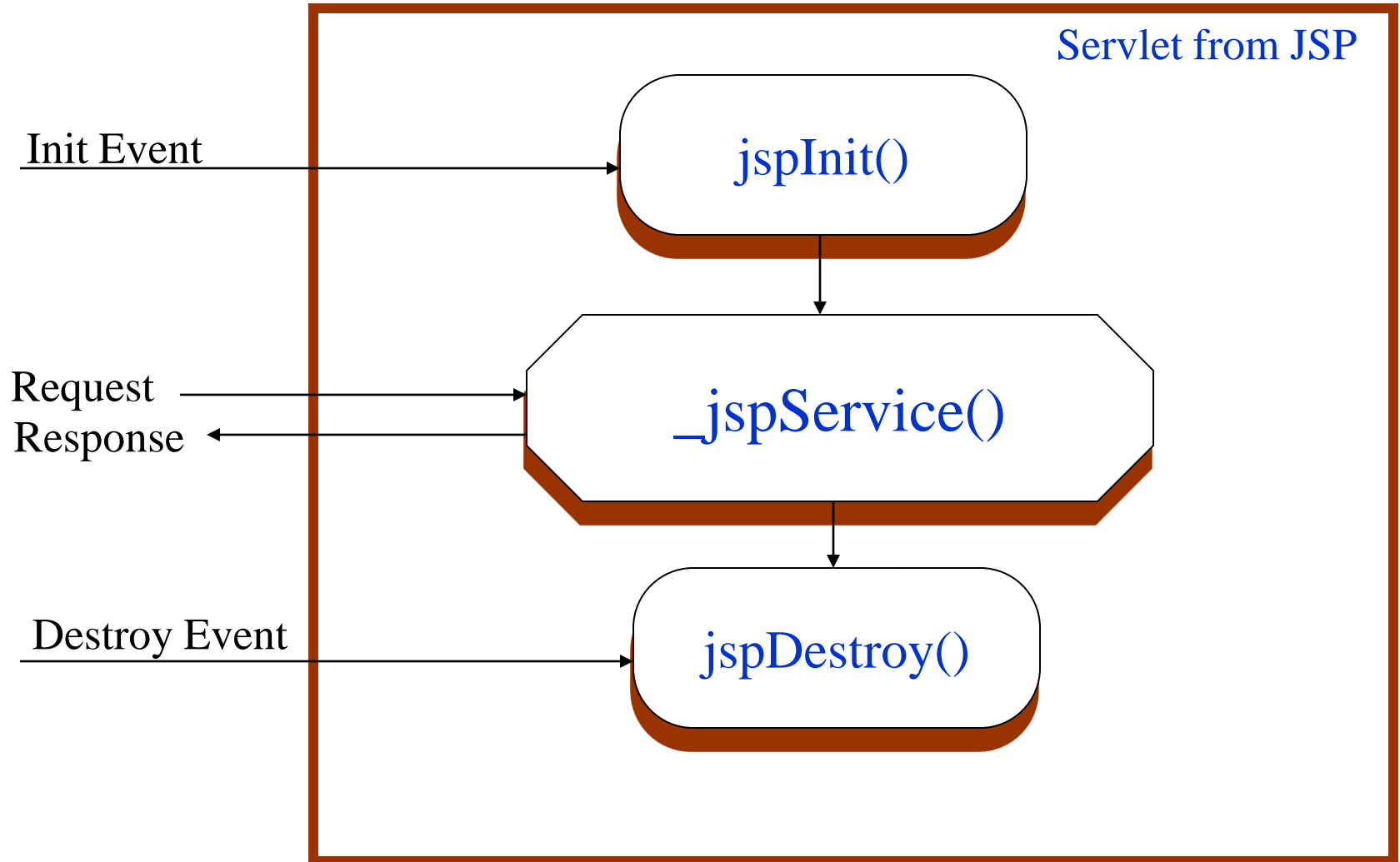
- The **jspDestroy()** method is invoke when the JSP page is going to be **destroyed (requested again)**

- **Notes:** the servlet must implement the **javax.servlet.jsp.HttpJspPage** interface



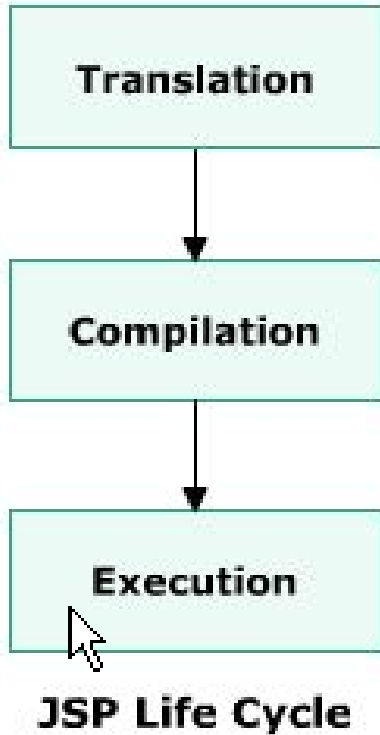
Java Server Pages

JSP Life Cycle



Java Server Pages

JSP Life Cycle



- **Compilation**

- The JSP page is automatically compiled and executed again by JSP/ Servlet Engine

- **Execution**

- Is carried out with the help of page directives controlling various execution parameters and are used for buffering output and handling errors

JSP Elements

Overview

- Enables to **create dynamic JSP pages**
- The JSP server **translates** and **executes** JSP elements

Elements	Description
Root	Classifies standard elements and attributes of namespaces in tag library
Comment	Used in JSP file documentation
Declaration	Declares variables and methods in a scripting language page.
Expression	Includes expression in a scripting language page
Scriptlet	Includes code fragment in a scripting language page
Text	Includes data and text
include Directive	Includes content of one JSP page into the current JSP page

JSP Elements

Overview

Elements	Description
page Directive	Defines attribute of JSP page and passes processing information about JSP page to JSP container
taglib Directive	Defines custom tags in JSP page
<jsp:param>	Adds value of parameter to a request sent to another JSP page using <jsp:include> or <jsp:forward>
<jsp:forward>	Forwards request from a client to the Web server
<jsp:include>	Includes the output from one file into the other
<tagPrefix:name>	Accesses the functions of custom tags
<jsp:setProperty>	Sets the value of a Java Bean
<jsp:getProperty>	Includes the bean property and value into the result set
<jsp:plugin>	Uses a plugin to execute an applet or Bean
<jsp:useBean>	Sets the location and initializes the bean with a specific name and scope

JSP Elements

JSP Tags

- Tags
 - Interface
 - Functional
 - Encapsulation
 - A tag starts with “<” and ends with “>”.
 - The tags contain body and attributes.
- The 4 types of JSP tags
 - Comments
 - Directives
 - Scripting elements
 - Standard actions

JSP Elements

Comments

- Are used for **documenting** JSP code
- Should **not be visible** to the client
- **Explains** the functioning of the code
- Comments **are ignored by the servlet** during compilation
- A JSP page contains **03 types** comments as JSP, HTML, and scripting
 - **HTML comments**
 - Are passed to resulting HTML documents
 - Are not visible in the output but the end user can view them.
 - **JSP comments**
 - The browser cannot view these comments as a part of the source code
 - Are ignored by the JSP to scriptlet translator.
 - **Scripting language comments**
 - Are written scriptlets in a JSP page.
 - The comment should be in the same comment syntax used for scripting language.
 - Are not visible in the output of the JSP page

JSP Elements

Comments

- Syntax
 - **JSP comments:** `<%-- comments --%>`
 - Ex: `<%-- a JSP comment --%>`
 - **HTML comments:** `<!-- comments -->`
 - Ex: `<!-- a HTML comment -->`
 - **Scripting language comment**
 - `<%/* comments */%>`
 - `<%// comments %>`
 - Ex: `<%//It's is a variable declaration %>`

JSP Elements

Scripting Elements

- A way of **performing server-side operations** in a JSP page
- Enable the **code to be directly embedded** in a JSP page
- **Insert** Java code into the JSP page
- **Declarations:**
 - Defines the **variables and methods** for a JSP page
 - Are **inserted** into the servlet, **outside** the `_jspService()` method
 - Are used in **combination** with scriptlets and expressions to display an output.
 - A **single** declaration tag can be used to **define multiple variables**
 - Syntax: `<% ! Declaration; %>`
 - **Ex:** `<% ! String s = "Aptech"; %>`

JSP Elements

Scripting Elements

- **Scriptlets**

- Is used to embed Java code, which is **inserted into the `_jspService()` method** of the servlet within an HTML code
- Refers to **code blocks** executed for every request (a fragment codes)
- Are used to add complex data to an HTML form
- Syntax: `<% scriptlet %>`
- **Ex:** `<% for (int i =0 ; i<n; i++){
System.out.println(i + “.This is scriptlets.”); } %>`

- **Expressions**

- Can be used to **display individual variables or the result** of some calculation
- Contains a Java statement whose value will **be evaluated** and **inserted** into the generated web page
- Refers to **single line codes** executed for every request.
- Provides **dynamic output generation and the result** is converted into a string
- Evaluates at HTTP request
- A declaration block is enclosed between delimiters.
- Syntax: `<%= expression %>`
- **Ex:** `<%= i %>`

JSP Directives

Directives

- **Controls the structure of the servlet** by sending messages from the JSP page to the JSP container.
- The **scope** of directives is the **entire JSP file**
- JSP uses directives for **controlling the processing** of JSP pages.
- **Do not produce any output** and **inform the JSP engine** about the **actions** to be performed on the JSP page
- Specify scripting language used
 - **Ex:** `<%@ page language = "java" ...%>`
- Denote the use of custom tags library (taglib)
 - **Ex:** `<%@ taglib uri = "c:\..." prefix = "abc" %>`
- Include the contents of another JSP page into the current page
 - **Ex:** `<%@ include file = "c:\..." %>`
- Include Java file to Java packages list.
 - **Ex:** `<%@ page import = "java.util.*, java.lang.*" %>`
- Error handle in JSP page and JSP page is caught errors (isErrorPage).
 - **Ex:** `handle error <%@ page ... errorpage = "/error.jsp" ... %>`
`process error <%@ page isErrorPage = "true" %>`

Page Directives

- Is used **to define and manipulate** a number of important attributes that affect the entire JSP page
- Is written **at the beginning** of a JSP page
- A JSP page can contain any number of page directives. All directives in the page are processed together during translation and result is applied together to the JSP page
- **Syntax:** `<% @ page attributes %>`

Attributes	Descriptions
language	Define the scripting language used in the page. Default value is Java
extends	Change the content of the servlet that is generated
import	Include Java files to the Java package import list. Separating uses commas
session	Specify if the JSP page takes part in a HTTP session. Default value is true
buffer	Specify the size of the page buffer. Default value is 8KB
autoflush	Flush the page buffer is filled. Default value is true
isThreadSafe	Define the safety level of threads in the page. The JSP engine queues the requests sent for processing when the value is set to false. Default value is true
info	Describe the page
errorPage	Define the page to display errors occurring in the JSP page
isErrorPage	Indicate the current JSP page if contains the path another error page of JSP
contentType	Set the content type or MIME type and character encoding for JSP. Default value is text/html

JSP Directives

Include & Taglib Directives

- **Include**

- Is used to **physically include** the contents of another file sending to the server. The included file can be a HTML or JSP
- Identify the file through the local URL
- A single JSP file can include multiple include directives
- **Syntax:** `<%@ include file = "URL" %>`

- **Taglib**

- Enables the **use of custom tags** in the JSP page
- Access to all the objects that are available for a JSP page
- Extend the functionality of a JSP page one after the other
- The TLD – Tag Library Descriptor is identified through the URI – Uniform Resource Identifier and prefix describes the prefix string used to define the custom tag
- **Syntax:** `<% @ taglib uri = "URL" prefix = "name" %>`

Example

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

Directives - page

`<html>`

`<head>`

`<title>First JSP program</title>`

`</head>`

HTML Comments

`<body>`

Scriptlet

`<!-- myFirstProgram.jsp -->`

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

`<%= "Current date is " + new Date() %>`

Expression

`<%-- end Program --%>`

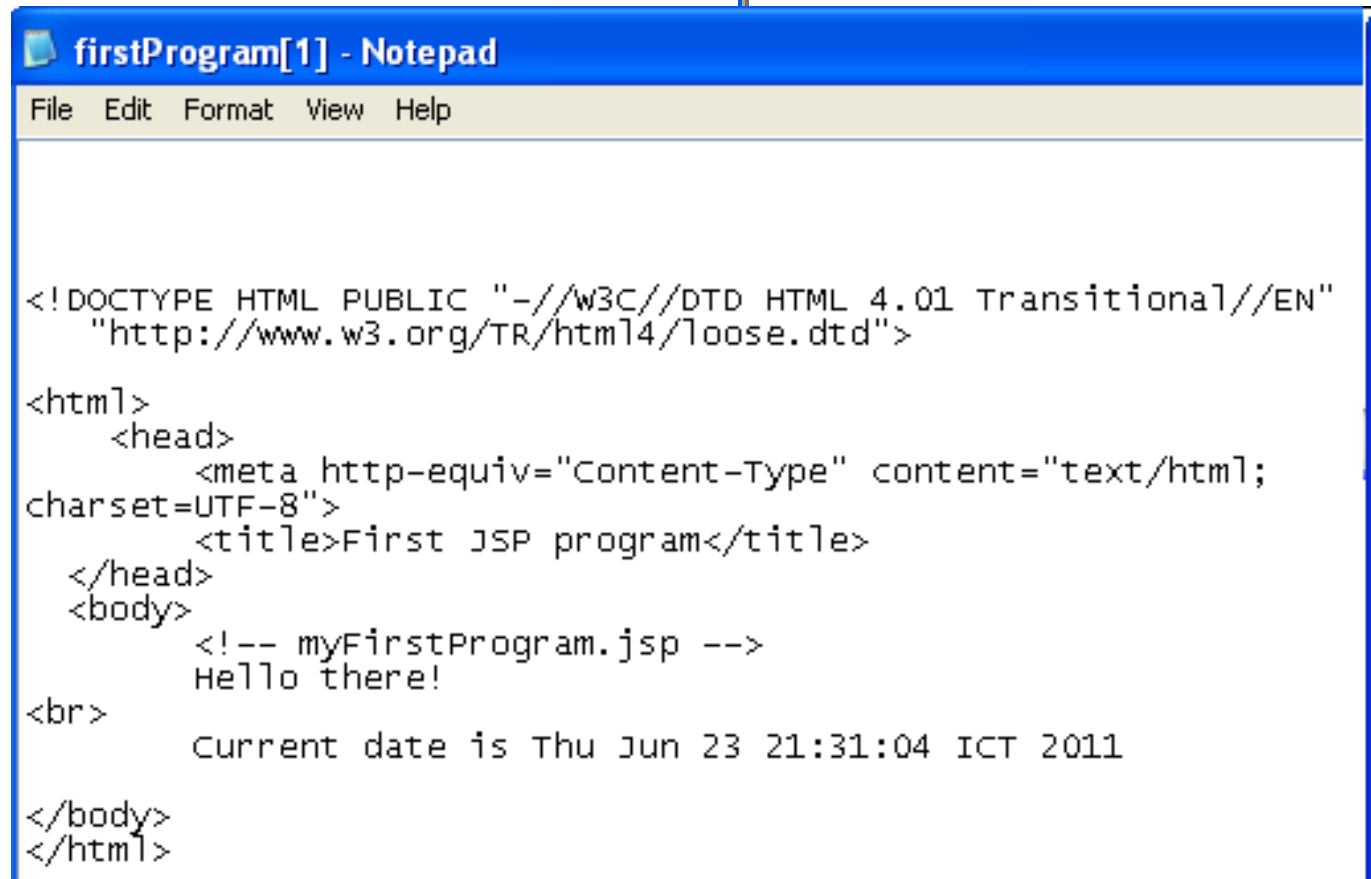
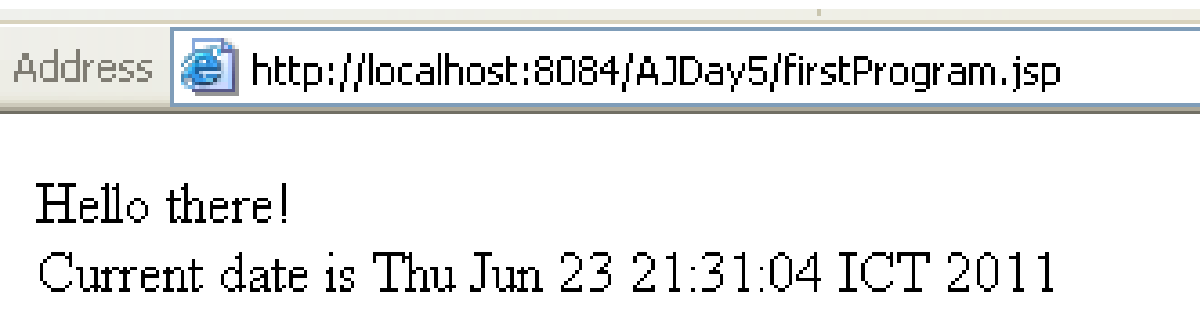
JSP Comments

`</body>`

`</html>`

JSP Elements

Example



JSP Elements

Example

- **Main (main.jsp) file**

```
<html>
<head>
  <title>Directive Includes JSP program</title>
</head>
<body>
  <%-- include use directives --%>
  Current date is
  <% @ include file = "myDate.jsp" %>
</body>
</html>
```

JSP Comments


Directives - include

Directives - page

- **Include (myDate.jsp) file**

```
<% @ page import="java.util.Date" %>
<html>
<head>
  <title>Date JSP program</title>
</head>
<body>
  <%= new Date().toLocaleString() %>
</body>
</html>
```

Expression

Address

http://localhost:8084/AJDay5/main.jsp

Current date is 21:34:21 23-06-2011

c:\Documents and Settings\Trong Khanh\.netbeans\6.9\apache-tomcat-6.0.26_base\work\Cat

Name	Ext	Size	
..	<DIR>	2	
main.jsp	class	4.510	2
main.jsp	java	3.857	2

main[1] - Notepad
File Edit Format View Help

```

<!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=UTF-8">
    <title>Directive Includes JSP program</title>
  </head>
  <body>

    Current date is

  </body>
</html>

```

```

<!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=UTF-8">
    <title>Date JSP program</title>
  </head>
  <body>
    21:34:21 23-06-2011
  </body>
</html>

```

JSP Elements

Example

```
out.write("\n");
...
out.write("<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN\"");
out.write(" \"http://www.w3.org/TR/html4/loose.dtd\">\n");
out.write("\n");
out.write("<html>\n");
out.write(" <head>\n");
out.write(" <title>Directive Includes JSP program</title>\n");
out.write(" </head>\n");
out.write(" <body>\n");
out.write(" \t");
out.write("\n");
out.write(" \tCurrent date is\n");
out.write(" \t");
out.write("\n");
...
out.write("<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN\"");
out.write(" \"http://www.w3.org/TR/html4/loose.dtd\">\n");
out.write("\n");
out.write("\n");
out.write("<html>\n");
out.write(" <head>\n");
out.write(" <title>Date JSP program</title>\n");
out.write(" </head>\n");
out.write(" <body>\n");
out.write(" \t");
out.println( new Date().toLocaleString() );
out.write("\n");
out.write(" </body>\n");
out.write("</html>");
out.write("\n");
out.write(" </body>\n");
out.write("</html>");
```

JSP Elements Example

```

1  <%-- ... --%>
2
3  <%@page contentType="text/html" pageEncoding="UTF-8"%>
4  <!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
5    "http://www.w3.org/TR/html4/loose.dtd">
6
7  <html>
8  <head>
9    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8"%>
10    <title>A Simple Date</title>
11  </head>
12  <body>
13    The time on the server is <%= new Date() %>
14  </body>
15 </html>

```

Apache Tomcat/6.0.26 - Error report - Microsoft Internet Explorer

File Edit View Favorites Tools Help

Back Forward Stop Home Search Favorites

Address <http://localhost:8084/AJDay5/> Go Links

HTTP Status 500 -

type Exception report

message

description The server encountered an internal error () that prevented it from fulfilling this request.

exception

org.apache.jasper.JasperException: Unable to compile class for JSP:

An error occurred at line: 17 in the jsp file: /simpleDate.jsp
Date cannot be resolved to a type

```

14:      <title>A Simple Date</title>
15:    </head>
16:    <body>
17:      The time on the server is <%= new Date() %>
18:    </body>
19: </html>

```

Done Local intranet

JSP Elements Example

```
<h1>JSP Life Cycle Demo</h1>
```

```
<%!
    int num;
    public void jspInit() {
        System.out.println("jspInit is invoked!!!");
        num = 10;
    }

    public void jspDestroy() {
        System.out.println("jspDestroy is invoked!!!!");
        num = 0;
    }

    public int add (int n) {
        System.out.println("add is called!!!");
        num += n;

        return num;
    }
%>
```


jspInit

Declarations

jspDestroy

```
Init number <%= num %><br/>
```

```
<% out.print("Result of add is " + add(5)); %>
```

Address  http://localhost:8084/AJDay5/declaration.jsp


JSP Life Cycle Demo

Init number 10
Result of add is 15

Output

Apache Tomcat 6.0.26 Log x Apache Tomcat 6.0.26 x AJDay5 (run) x

```
INFO: Container org.apache.catalina.core.ContainerBase.[Catalina
jspInit is invoked!!!
add is called!!!
```

Address  http://localhost:8084/AJDay5/declaration.jsp

JSP Life Cycle Demo

Init number 15
Result of add is 20

Output

Apache Tomcat 6.0.26 Log x Apache Tomcat 6.0.26 x AJDay5 (run) x

```
jspInit is invoked!!!
add is called!!!
add is called!!!
```

JSP Elements

Example

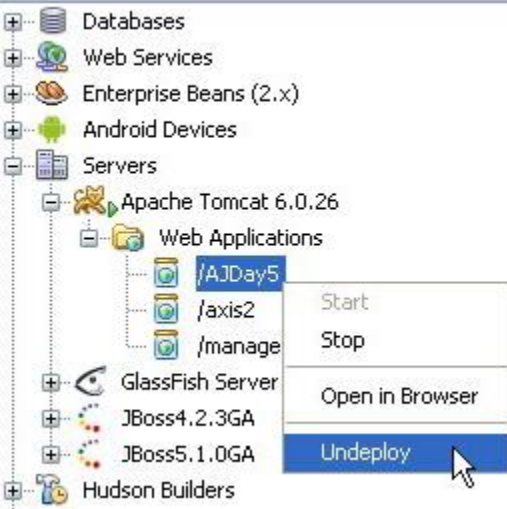
Output

```

Apache Tomcat 6.0.26 Log x Apache Tomcat 6.0.26 x AJDay5 (run) x
INFO: Deploying configuration descriptor /AJDay5.xml
23-06-2011 21:51:17 org.apache.catalina.core.StandardContext start
INFO: Container org.apache.catalina.core.ContainerBase.[Catalina].[localhost].[/AJDay5] has already been started
jspInit is invoked!!!
add is called!!!
add is called!!!
jspDestroy is invoked!!!!
23-06-2011 21:51:56 org.apache.catalina.startup.HostConfig checkResources
INFO: Undeploying context [/AJDay5]

```

Services



Output

```

Apache Tomcat 6.0.26 Log x Apache Tomcat 6.0.26 x AJDay5 (clean,dist) x
23-06-2011 21:46:36 org.apache.catalina.core.StandardContext start
INFO: Container org.apache.catalina.core.ContainerBase.[Catalina].[localhost].[/AJ
jspInit is invoked!!!
add is called!!!
add is called!!!
jspDestroy is invoked!!!!
23-06-2011 21:49:39 org.apache.catalina.startup.HostConfig checkResources
INFO: Undeploying context [/AJDay5]

```

Clean and Build Application


```

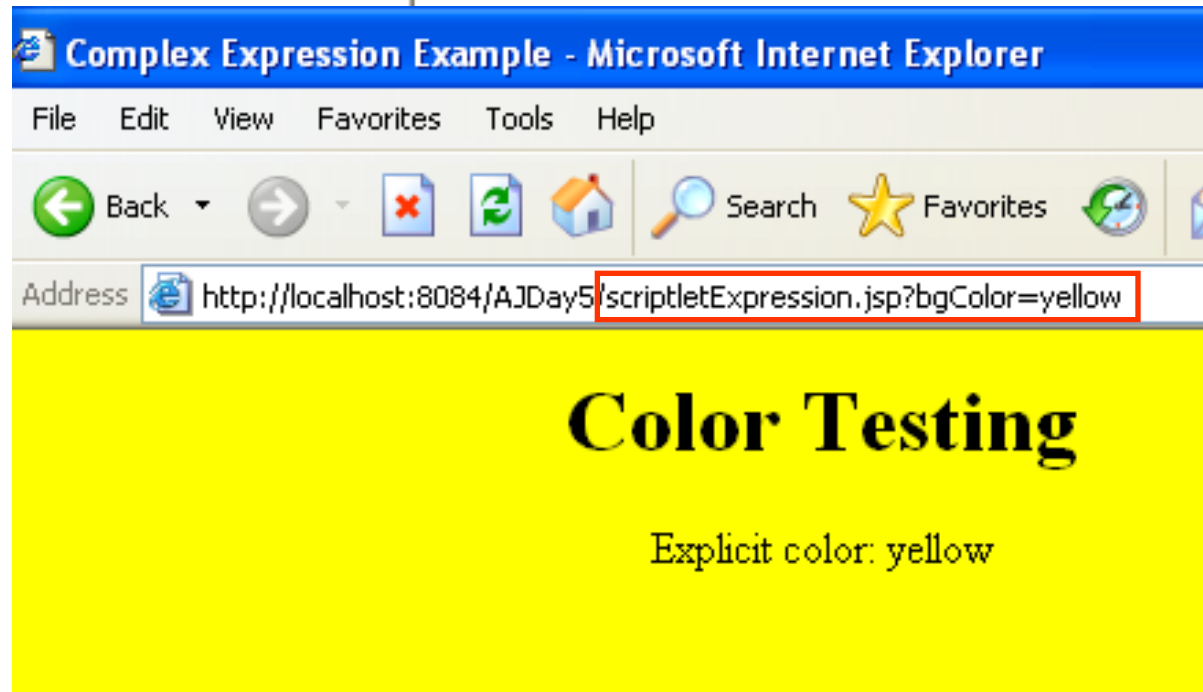
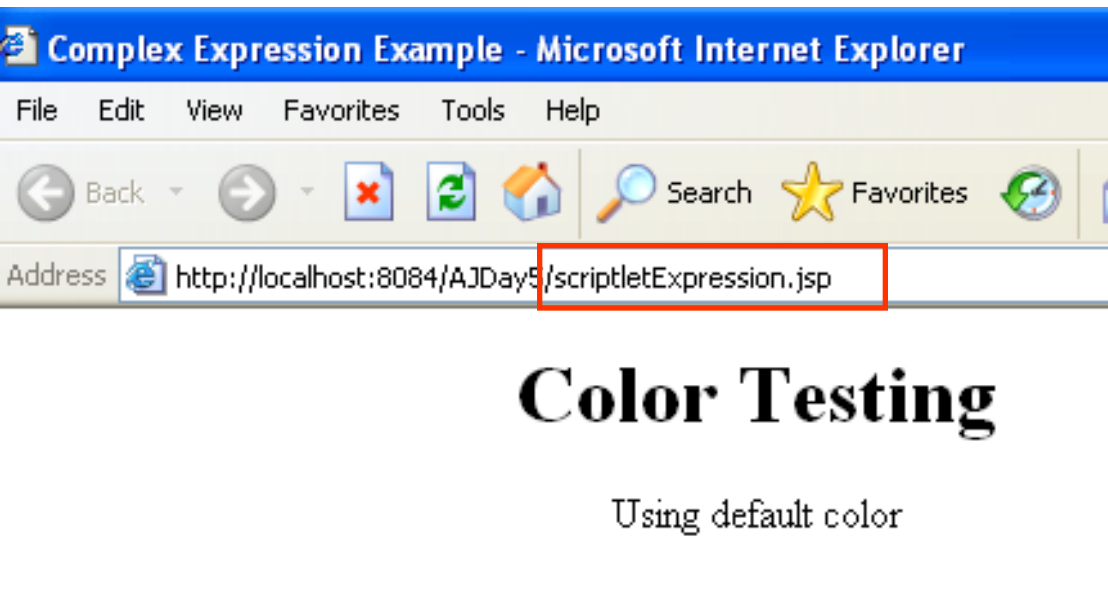
scriptletExpression.jsp
8  <!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
9    <http://www.w3.org/TR/html4/loose.dtd">
10
11 <html>
12   <head>
13     <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
14     <title>Complex Expression Example</title></head>
15   <%
16       String bgColor = request.getParameter("bgColor");
17       boolean hasExplicitColor;
18       if (bgColor != null) {
19         hasExplicitColor = true;
20       } else {
21         hasExplicitColor = false;
22         bgColor = "white";
23       }
24   %>
25   <body bgcolor="<%= bgColor%>">
26     <center>
27       <h1>Color Testing</h1>
28     <%
29       if (hasExplicitColor) {
30         out.println("Explicit color: " + bgColor);
31       } else {
32         out.println("Using default color");
33       }
34     %>
35   </center>
36 </body>
37 </html>

```

Expression!!!

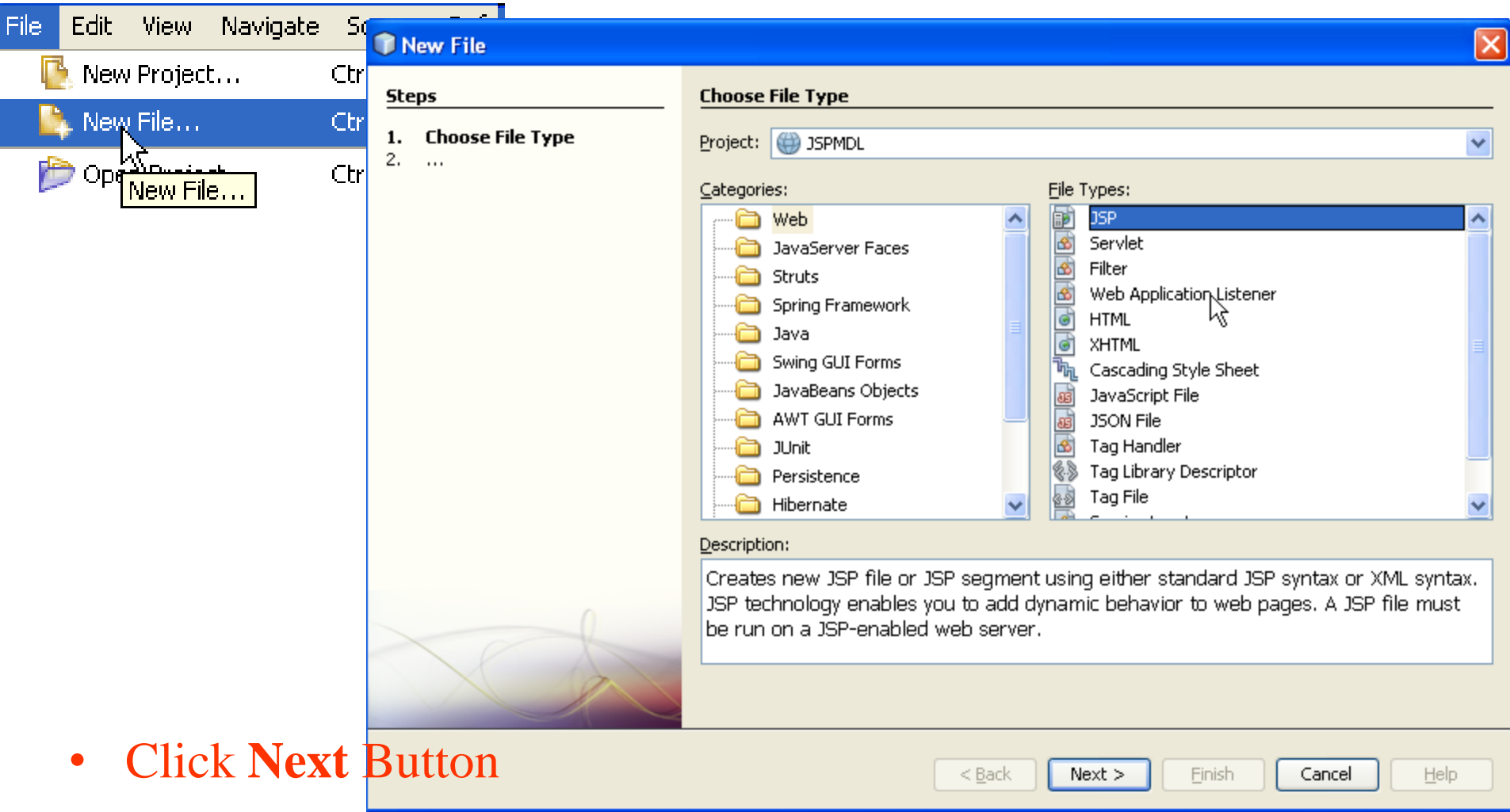
- Run jsp page without parameters
- Run with parameters as yellow, blue, red ...

JSP Elements Example



Additional

Create the JSP Page on NetBeans



Additional

Create the JSP Page on NetBeans

New JSP File

Steps

1. Choose File Type
2. **Name and Location**

Name and Location

JSP File Name:

Project:

Location:

Folder:

Created File:

Options: ☒ JSP File (Standard Syntax) ☐ JSP Document (XML Syntax)

☐ Create as a JSP Segment

Description:

< Back Next > Finish Cancel Help

Fill your file name without .jsp

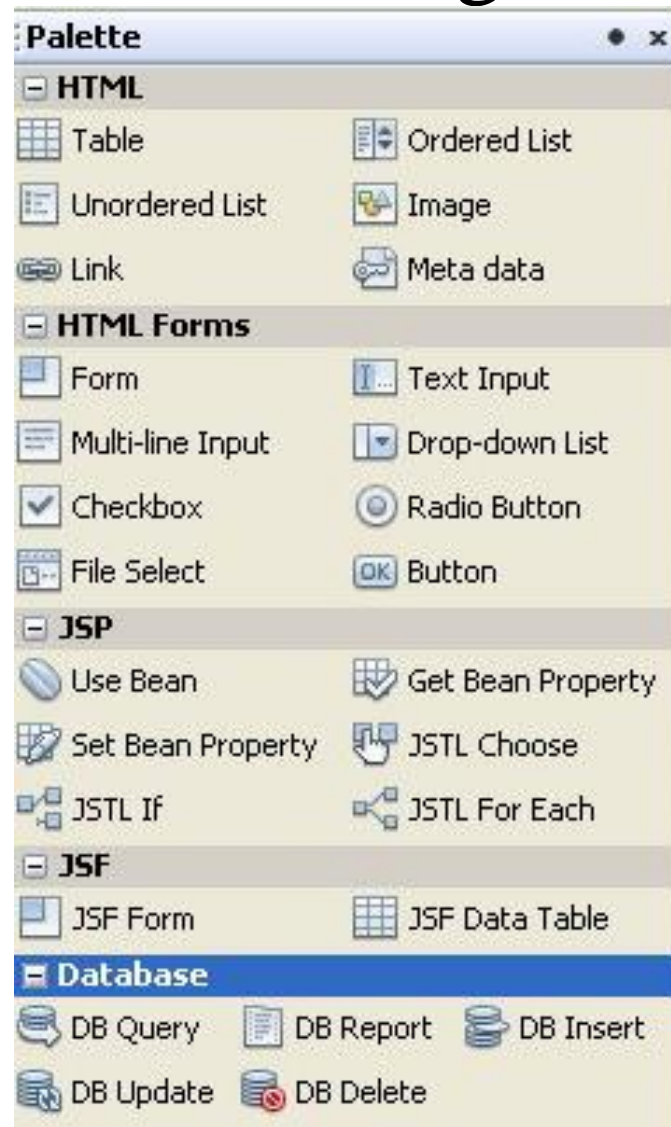
Locate the directory store jsp file in current project

Using JSP File

- Click **Finish** Button

Additional

Create the JSP Page on NetBeans



JSP Elements

Example – Exception

```

14      <title>Scripting</title>
15  </head>
16  <body>
17      <h1>Scripting Demo</h1>
18      <%
19          String s = "declaration";
20
21          public void test() {
22              System.out.println("This is a test function!!!");
23          }
24      %>
25
26      <%
27          String s = "scriptlet";
28
29          public void test1() {
30              System.out.println("This is a function in scriptlet");
31          }
32      %>
33  </body>
34  </html>
  
```

Compiling 1 source file to F:\Laptrinh\Servlet\ClassF21005_NB691\JSP\Day4JSP1005Y2\build\generated\classes

F:\Laptrinh\Servlet\ClassF21005_NB691\JSP\Day4JSP1005Y2\build\generated\src\org\apache\jsp\testDemo_jsp.java:72: illegal start of expression

```

    public void test1() {
    ^
  
```

F:\Laptrinh\Servlet\ClassF21005_NB691\JSP\Day4JSP1005Y2\build\generated\src\org\apache\jsp\testDemo_jsp.java:72: illegal start of expression

```

    public void test1() {
    ^
  
```

F:\Laptrinh\Servlet\ClassF21005_NB691\JSP\Day4JSP1005Y2\build\generated\src\org\apache\jsp\testDemo_jsp.java:72: ';' expected

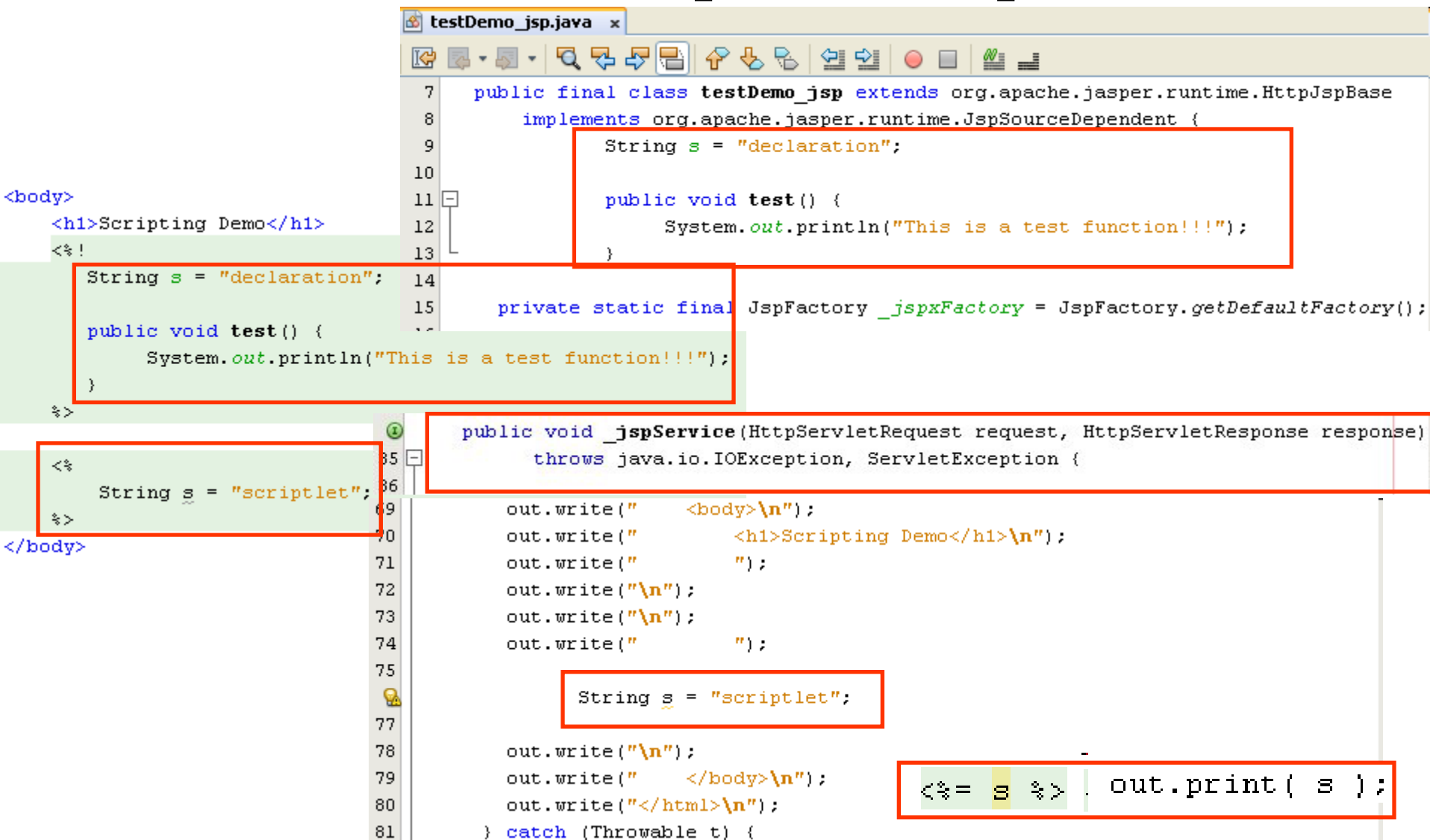
```

    public void test1() {
    ^
  
```

3 errors

JSP Elements

Example – Exception



```

<body>
<h1>Scripting Demo</h1>
<%!
String s = "declaration";

public void test() {
    System.out.println("This is a test function!!!");
}
%>

<%
String s = "scriptlet";
%>
</body>

```

```

testDemo_jsp.java
7 public final class testDemo_jsp extends org.apache.jasper.runtime.HttpJspBase
8 implements org.apache.jasper.runtime.JspSourceDependent {
9     String s = "declaration";
10
11     public void test() {
12         System.out.println("This is a test function!!!");
13     }
14
15     private static final JspFactory _jspxFactory = JspFactory.getDefaultFactory();
16
17     public void _jspService(HttpServletRequest request, HttpServletResponse response)
18         throws java.io.IOException, ServletException {
19
20         out.write("    <body>\n");
21         out.write("        <h1>Scripting Demo</h1>\n");
22         out.write("    ");
23         out.write("\n");
24         out.write("\n");
25         out.write("\n");
26         out.write("    ");
27
28         String s = "scriptlet";
29
30         out.write("\n");
31         out.write("    </body>\n");
32         out.write("</html>\n");
33
34         <%= s %> out.print( s );
35     }
36 }
37 catch (Throwable t) {

```

JSP Implicit Objects

Implicit Objects

- **Does not initialize or declare**
- **Are loaded** by the Web Container **automatically and maintains** them in a JSP page (**Available for scriptlets or expressions**)
- Created using directives and accessible according to the **specified scopes**
- The names of the implicit objects **are reserved words** of JSP
- The scopes for the IB in JSP page including **page, request, session, and application**
- **Access dynamic content** using JavaBeans
- **Syntax:** **ImplicitObject.method(params)**
- **Types of Implicit Objects**
 - *Input & Output Objects*
 - The objects control page input and output
 - Are request, response and out
 - *Scope Communication Objects*: provide access to all objects available in the given scope
 - *Servlet Objects*
 - Provides information about the page context
 - Processes request objects from a client and sends the response objects back to the client

JSP Implicit Objects

- Types of Implicit Objects (cont)
 - The Error Objects
 - The object handles errors in a JSP page (exception)
 - Can access this object by declaring your JSP page as an error page
`<% @page isErrorPage="true" %>`

Object	Class / Interface
page	javax.servlet.jsp.HttpJspPage – variable synonym for this object
config	javax.servlet.ServletConfig
request	javax.servlet.http.HttpServletRequest
response	javax.servlet.http.HttpServletResponse
out	javax.servlet.jsp.JspWriter
session	javax.servlet.http.HttpSession
application	javax.servlet.ServletContext
pageContext	javax.servlet.jsp.PageContext
exception	java.lang.Throwable

JSP Implicit Objects

Input & Output Objects

Objects	Descriptions
request	<ul style="list-style-type: none"> - Refer to the current request made by the client that is being processed by JSP container. The container passed the request IB to JSP page as a parameter to the <code>_jspService()</code>. - Implement the <code>javax.servlet.http.HttpServletRequest</code> interface - Syntax: <code>request.method(params)</code> - Scope: request - Ex: <code>request.getParameter("username");</code>
response	<ul style="list-style-type: none"> - Refers the result that is returned to the user after a JSP processed - Implement the <code>javax.servlet.http.HttpServletResponse</code> interface - Syntax: <code>response.method(params)</code> - Scope: page - Ex : <code>response.addCookie(cookie)</code>
out	<ul style="list-style-type: none"> - Represent the output stream for the JSP page (send to client) - Implement the <code>javax.servlet.jsp.JspWriter</code> interface (the buffer size is supported by Servlet) - Syntax: <code>out.method(params)</code> - Scope: page - Ex: <code>out.println("output stream")</code>

JSP Implicit Objects

Input & Output Objects – Example

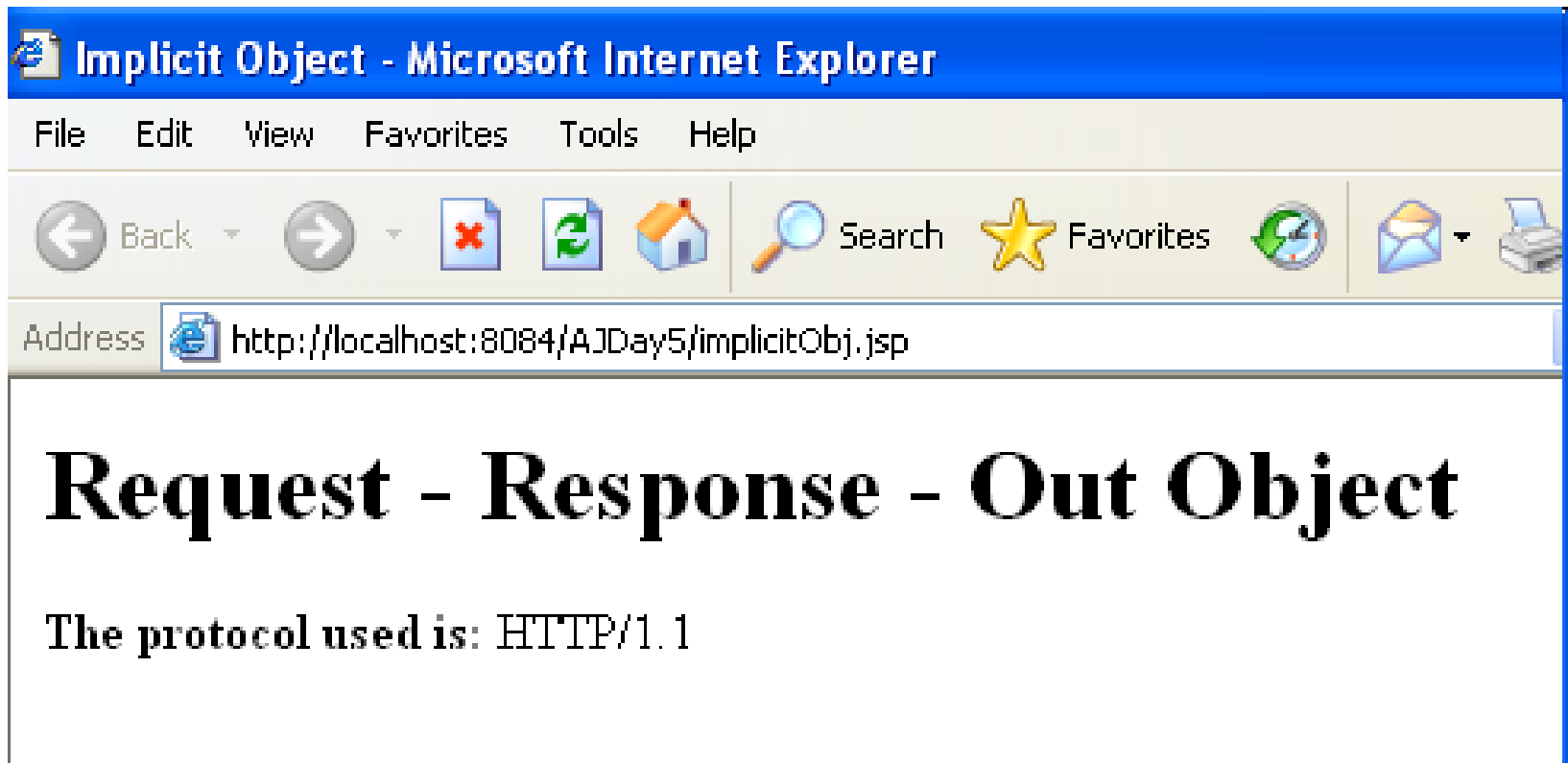
```

1  <%--...--%>
6
7  <%@page contentType="text/html" pageEncoding="UTF-8"%>
8  <!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
9    "http://www.w3.org/TR/html4/loose.dtd">
10
11 <html>
12   <head>
13     <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
14     <title>Implicit Object</title>
15   </head>
16   <body>
17
18     <h1>Request - Response - Out Object</h1>
19
20     <%
21       response.setDateHeader ("Expires", 0);
22       response.setHeader ("Pragma", "no-cache");
23       if (request.getProtocol ().equals ("HTTP/1.1")){
24         response.setHeader ("Cache-Control", "no-cache");
25         out.println ("<b>The protocol used is:</b> " + request.getProtocol ());
26       }
27     %>
28   </body>
29 </html>

```

JSP Implicit Objects

Input & Output Objects – Example



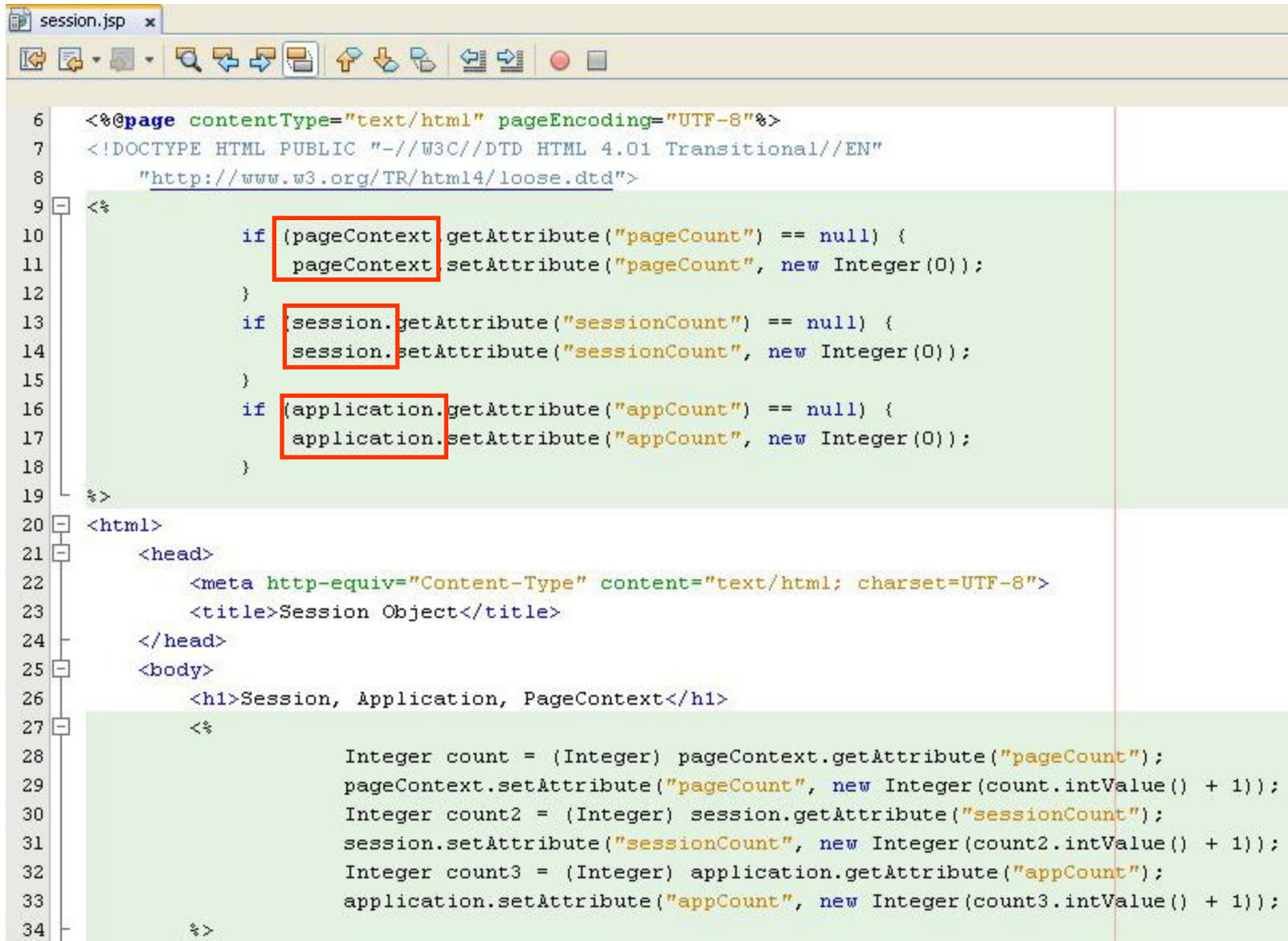
JSP Implicit Objects

Scope Communication Objects

Objects	Descriptions
session	<ul style="list-style-type: none">- Specify data and store information in the current session.- Implement the javax.servlet.http.HttpSession interface- Syntax: session.method(params)- Scope: session- Ex: session.setAttribute("username", "Aptech");
application	<ul style="list-style-type: none">- Represent the application of the required JSP page and represent the servlet context about Web Application in which it is running.- Implement the javax.servlet.ServletContext interface- Syntax: application.method(params)- Scope: application- Ex: application.setAttribute("username", "Aptech");
pageContext	<ul style="list-style-type: none">- An instance of Pages (javax.jsp.PageContext)- Enable access to the JSP page and the attributes associated with that page- provides following fields to find the scope or specify the scope of the objects (PAGE, REQUEST, SESSION, and APPLICATION)- Syntax: pageContext.method(params)- Ex: pageContext.getAttributes("username");

JSP Implicit Objects

Scope Communication Objects – Example



```

6  <%@page contentType="text/html" pageEncoding="UTF-8"%>
7  <!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
8    "http://www.w3.org/TR/html4/loose.dtd">
9  <%
10      if (pageContext.getAttribute("pageCount") == null) {
11          pageContext.setAttribute("pageCount", new Integer(0));
12      }
13      if (session.getAttribute("sessionCount") == null) {
14          session.setAttribute("sessionCount", new Integer(0));
15      }
16      if (application.getAttribute("appCount") == null) {
17          application.setAttribute("appCount", new Integer(0));
18      }
19  %>
20  <html>
21  <head>
22      <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
23      <title>Session Object</title>
24  </head>
25  <body>
26      <h1>Session, Application, PageContext</h1>
27      <%
28          Integer count = (Integer) pageContext.getAttribute("pageCount");
29          pageContext.setAttribute("pageCount", new Integer(count.intValue() + 1));
30          Integer count2 = (Integer) session.getAttribute("sessionCount");
31          session.setAttribute("sessionCount", new Integer(count2.intValue() + 1));
32          Integer count3 = (Integer) application.getAttribute("appCount");
33          application.setAttribute("appCount", new Integer(count3.intValue() + 1));
34      %>

```


JSP Implicit Objects

Scope Communication Objects – Example

```

35      <b>Page Count =</b>
36      <%= pageContext.getAttribute("pageCount") %><br/>
37
38      <b>Session Count =</b>
39      <%= session.getAttribute("sessionCount") %><br/>
40
41      <b>Application Count =</b>
42      <%= application.getAttribute("appCount") %><br/>
43
44      <b>Time =</b>
45      <%= new java.sql.Time(System.currentTimeMillis()) %><br/>
46  </body>
47  </html>

```

Address  http://localhost:8084/AJDay5/session.jsp

Session, Application, PageContext

Page Count = 1
Session Count = 1
Application Count = 1
Time = 22:10:01

Address  http://localhost:8084/AJDay5/session.jsp

Session, Application, PageContext

Page Count = 1
Session Count = 2
Application Count = 2
Time = 22:10:23

JSP Implicit Objects

Servlet Objects

Objects	Descriptions
page	<ul style="list-style-type: none">- Represents the servlets and the initialization parameters of the servlet are stored in the config IB- Use “this” reference and the page IB represents it.- Implements the javax.lang.object interface- Syntax: <code><%@ page info = “information” %></code>- Scope: page
config	<ul style="list-style-type: none">- Represent the configuration of the servlet data- Implement the javax.Servlet.ServletConfig interface- Access objects through <code>config.getInitParameter(“par”)</code>- Scope: page

JSP Implicit Objects

Servlet Objects – Example

servletObj.jsp



```

6
7 <%@page import="java.util.Date"%>
8 <%@page import="java.util.Calendar"%>
9 <%@page contentType="text/html" pageEncoding="UTF-8"%>
10 <!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
11   "<a href='\"http://www.w3.org/TR/html4/loose.dtd\"'>http://www.w3.org/TR/html4/loose.dtd">
12
13 <html>
14   <head>
15     <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
16     <title>ServletObj</title>
17   </head>
18   <body>
19     <h1>Welcome to TimeZone</h1>
20     <% Calendar cal = Calendar.getInstance();%>
21     Today's Date and Time is: <%= new Date()%><br/>
22     The JSP Page is created by <%= ((HttpServlet) page).getServletInfo()%><br/>
23   </body>
24 </html>

```

Address  http://localhost:8084/AJDay5/servletObj.jsp

Welcome to TimeZone

Today's Date and Time is: Thu Jun 23 22:12:51 ICT 2011
The JSP Page is created by Jasper JSP 2.1 Engine

JSP Implicit Objects

Error Objects

Objects	Descriptions
exception	<ul style="list-style-type: none">- Refer to the runtime exception in an error page- Is available only on pages that are assigned as error page using the isErrorPage attribute of the page directive.- Implement the javax.lang.Throwable interface- Exception methods are supported<ul style="list-style-type: none">+ getMessage() : return the error message associated with the exception+ toString() : Return a string with the class name of the exception within the error message.+ printStackTrace(): prints the execution stack in effect when the exception was thrown to the designated output stream

JSP Implicit Objects

Error Objects – Example

- A JSP page traps errors occurred

```
<%@ page errorPage="error.jsp" %>
```

```
<html>
```

```
<head> <title>Tag Example</title> </head>
```

```
<body>
```

```
<%! String name; %>
```

```
<%      name = request.getParameter("name");
```

```
    if(name==null) name="World"; %>
```

```
<h1>Hello, <%= name %></h1>
```

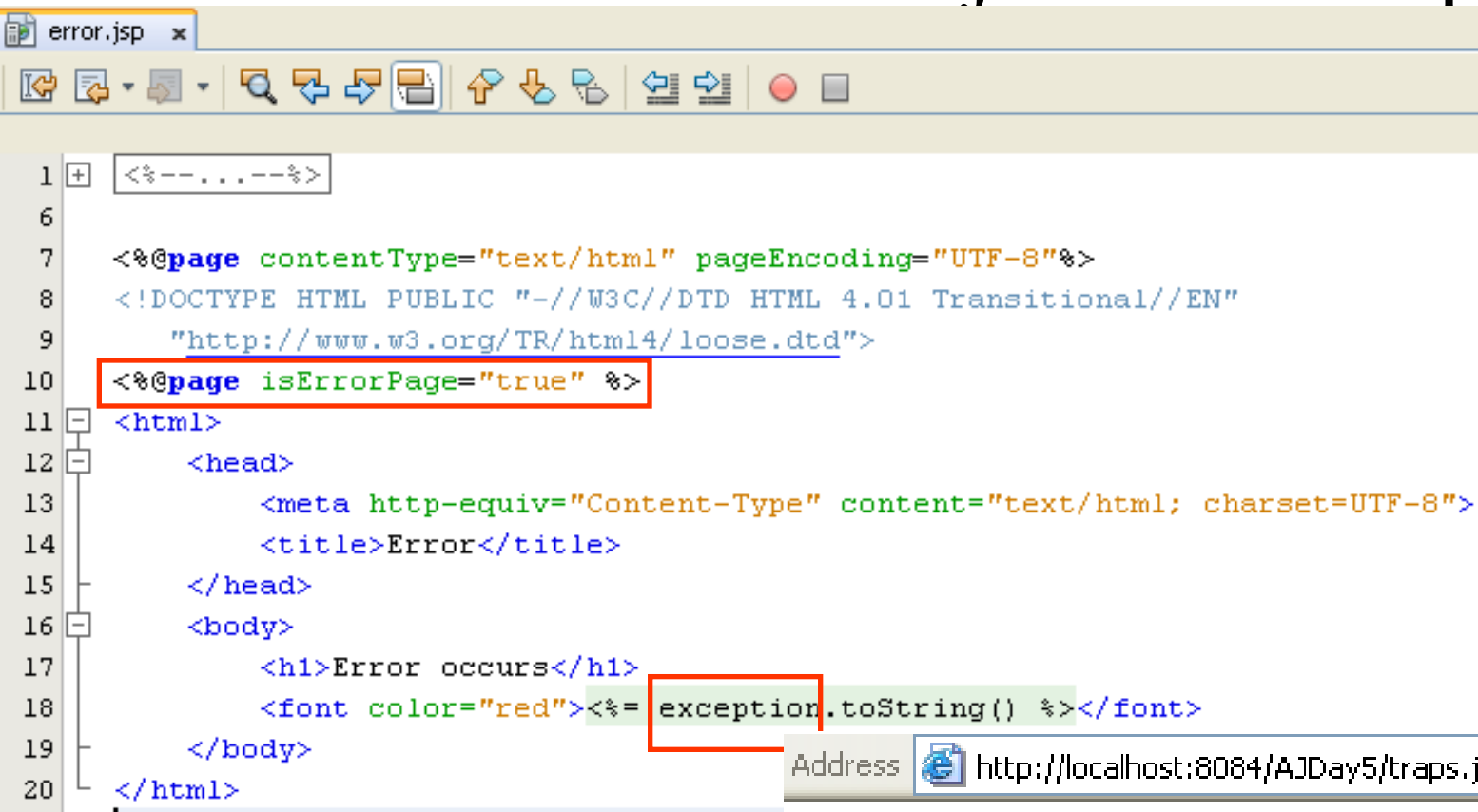
```
<% out.println("<H1>Hello " + name + "</H1>"); %>
```

```
<% int num=Integer.parseInt("a"); %> </body>
```

```
</html>
```


JSP Implicit Objects

Error Objects – Example



```

1  <%--...--%>
6
7  <%@page contentType="text/html" pageEncoding="UTF-8"%>
8  <!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
9    "http://www.w3.org/TR/html4/loose.dtd">
10 <%@page isErrorPage="true" %>
11 <html>
12 <head>
13     <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
14     <title>Error</title>
15 </head>
16 <body>
17     <h1>Error occurs</h1>
18     <font color="red"><%= exception.toString() %></font>
19 </body>
20 </html>
  
```

Address  http://localhost:8084/AJDay5/traps.jsp

Error occurs

java.lang.NumberFormatException: For input string: "a"

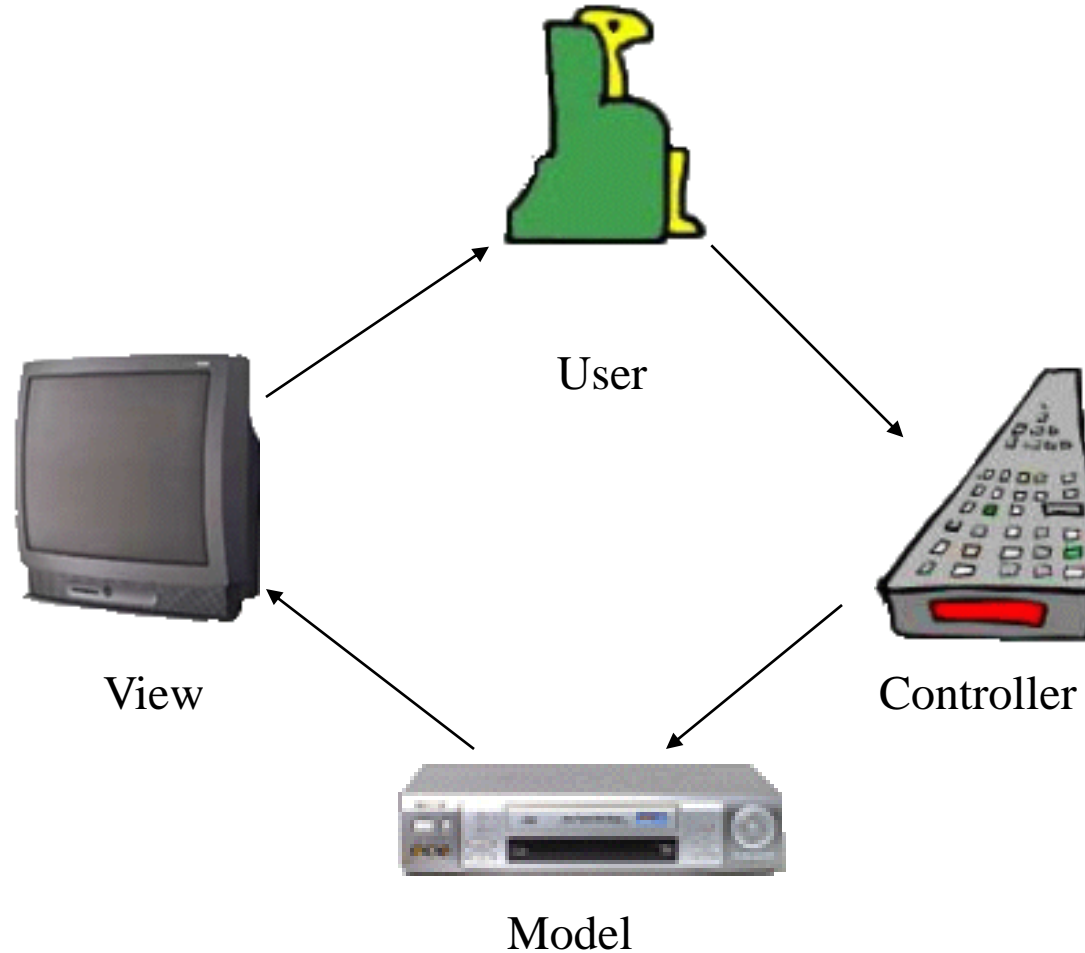
MVC Design Pattern

Model – View – Controller

- **Main concern** of MVC is to **separate** the **data (model)** and the **user interface (view)**
 - Helps the developer **changing** to the **user interface** can be made **without affecting** the **underlying data** handling logic.
 - The data can be **reorganized without changing** the user interface.
- **Separation** is **achieved** by introducing an **controller** component
 - **Controller** is an **intermediate component**
 - Controller **defines** as how the **user interface** should **react** to a **user input**

MVC Design Pattern

Model – View – Controller



This is a MVC Model

MVC Design Pattern

Model – View – Controller

- **Relationships** between components
 - **View and Controller:** Controller is responsible for creating or selecting view
 - **Model and View**
 - View depends on Model
 - If a change is made to the model then there might be required to make parallel changes in the view
 - **Model and Controller**
 - Controller depends on model
 - If a change is made to the model then there might be required to make parallel changes in the Controller
- **Logical Layers** in Web application
 - **Model** [Business Process Layer]
 - **View** [Presentation Layer]
 - **Controller** [Control Layer]

MVC Design Pattern

Model – View – Controller

- **Model**

- Models data and behavior behind business process
- Manages information (**access, modify, and represent application's data**) and notifies observers whenever the information changes
- Contains data and related functionality that are related by a common purpose
- Maps Real-World Entities (**implement business logic, workflow**)
- Performing DB Queries
- Calculating Business Process
- **Encapsulates Domain Logic** (a Java Object – **Java Bean**) which are independent of Presentation

- **View**

- Obtains data from model & presents to the user
- **Represents** Output/Input of the application (**GUI – JSP & HTML ...**)
- **Display** results of Business Logic
- Free Access to Model, but should not change the state of the model.
- **Reads** Data from Model – Using Query Methods

MVC Design Pattern

Model – View – Controller

- **Controller**

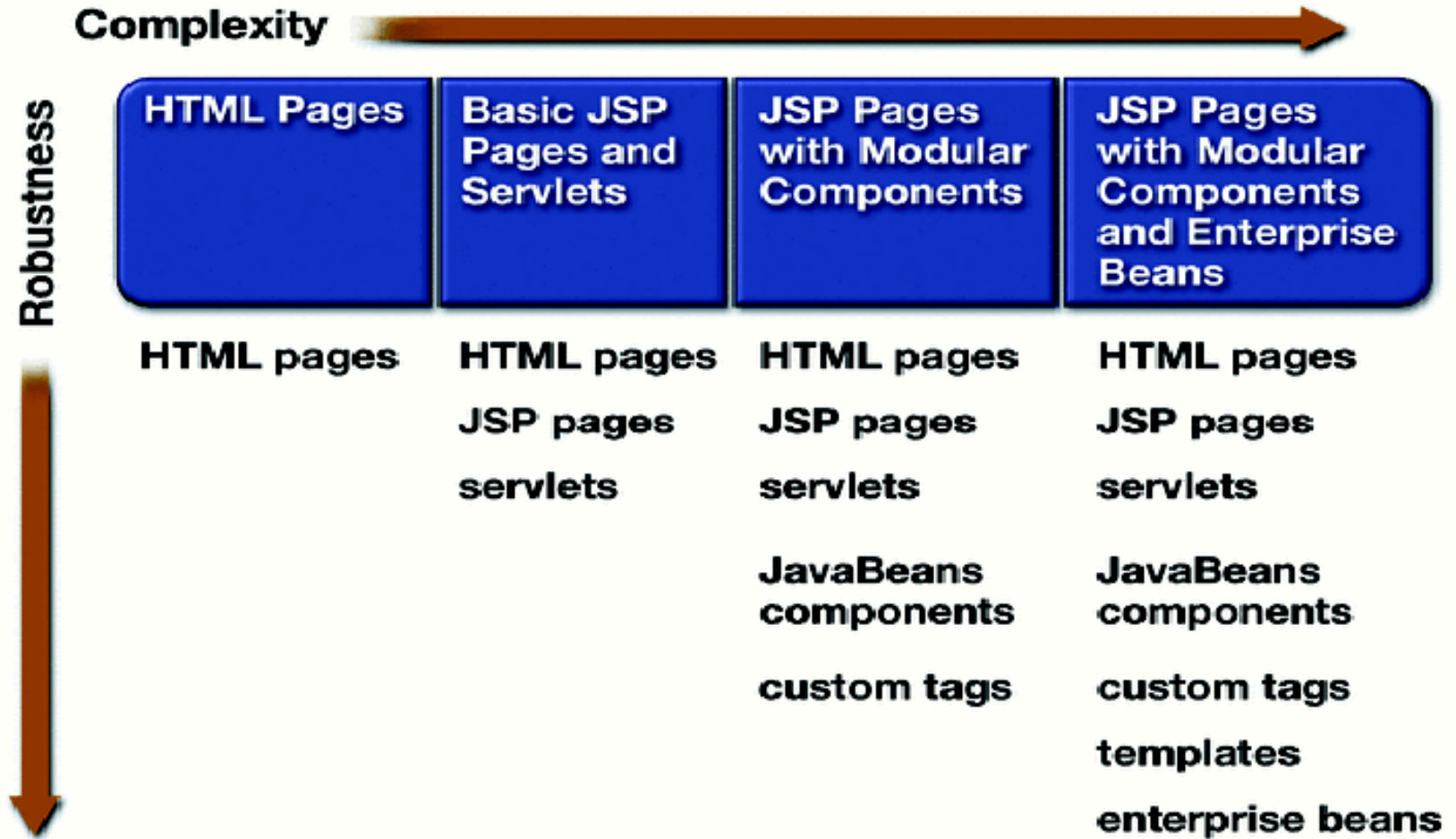
- Serves logical connection between user's interaction and the business process
- It receives and translates input to request on model or view
- Input from user and instructs the model and view to perform action
- Responsible for making decision among multiple presentation
- Maps the end-user action to the application response
- Is responsible for calling methods on the model that changes the state of the model
- **Updates the state of the Model and generates one or more views (servlets)**

- **Evolution of MVC Architecture**

- NO MVC
- MVC Model 1 [**Page-centric**] – JSP Model 1
- MVC Model 2 [**Servlet-centric**] – JSP Model 2

MVC Design Pattern

No MVC

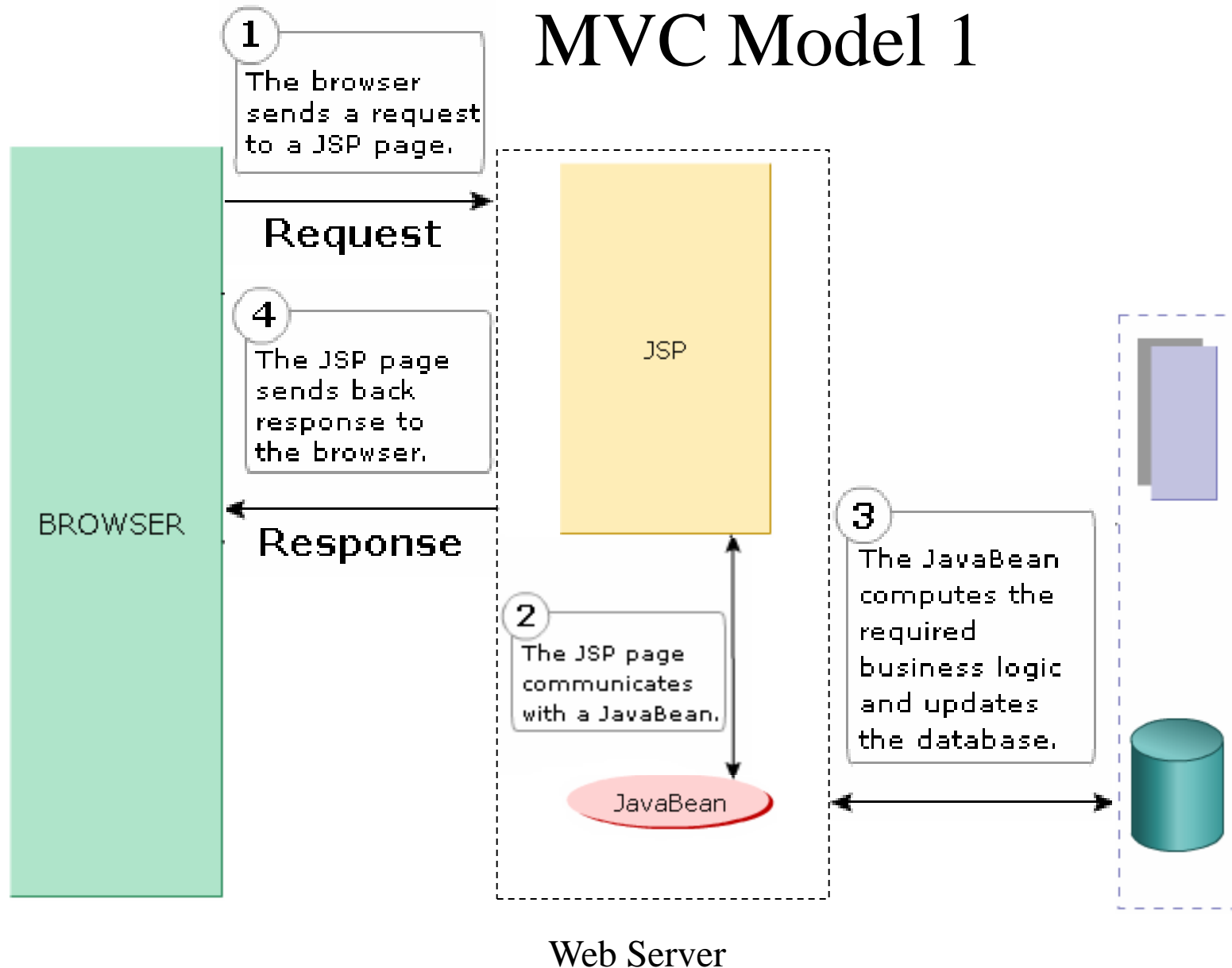


NO MVC

Model 1 Architecture

MVC Design Pattern

MVC Model 1



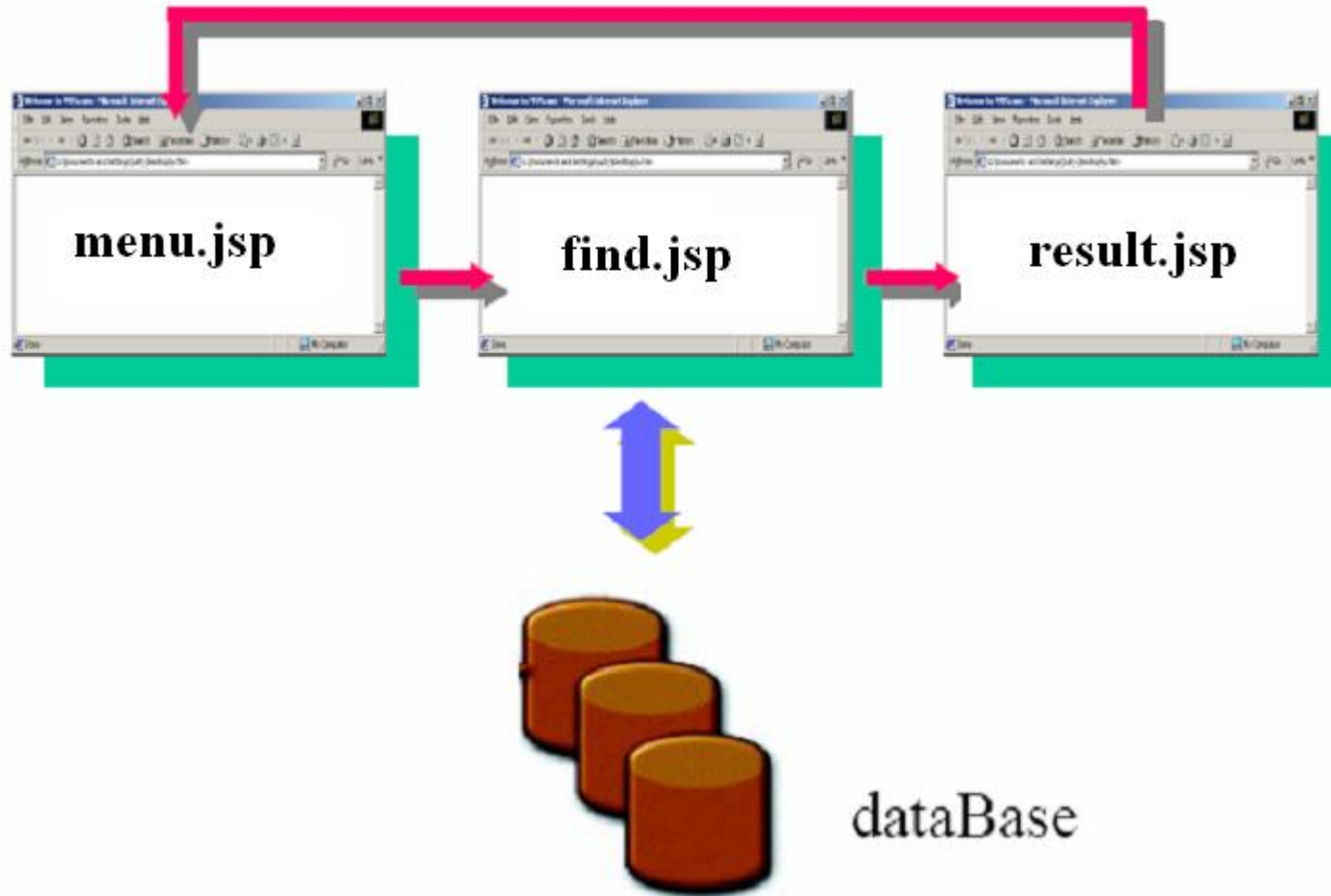
MVC Design Pattern

MVC Model 1

- Composed of a **series of interrelated JSP pages**
- **JSP page handles** the entire request processing mechanism
 - Extract the HTTP request parameters
 - Invoke the business logic (through Java Beans)
 - Process the business logic
 - Handle the HTTP session
- JSP page responsible for displaying the output to the client
- There is no extra Servlet involved in the process.
- **A page centric architecture (Business process logic and control decisions are hard coded inside JSP pages)**
- Next page selection is determined by hyperlink or action of submitting a form. **Ex:**
 - ` Search `
 - `<form action="find.jsp"> ... </form>`

MVC Design Pattern


MVC Model 1



Page-centric application

MVC Design Pattern

MVC Model 1 – Example

Address  http://localhost:8084/AJDay5/menu.jsp

Home page

- [Find](#)
- [Show](#)

 http://localhost:8084/AJDay5/find.jsp

Address  http://localhost:8084/AJDay5/find.jsp

Find Demo

Username

Address  http://localhost:8084/AJDay5/result.jsp

Find Result

username	password	lastname	isAdmin
ajcourse	ajj	Course AJ	

[Back to menu](#)

 http://localhost:8084/AJDay5/menu.jsp

MVC Design Pattern

MVC Model 1 – Example

- menu.jsp

```
<li><a href="find.jsp">Find</a></li>
```

```
<li><a href="result.jsp">Show</a></li>
```

- find.jsp

```
<form method="post" action="result.jsp">
```

```
    Username <input type="text" name="username" value=""  
/><br/>
```

```
    <input type="submit" value="Find" />
```

```
</form>
```

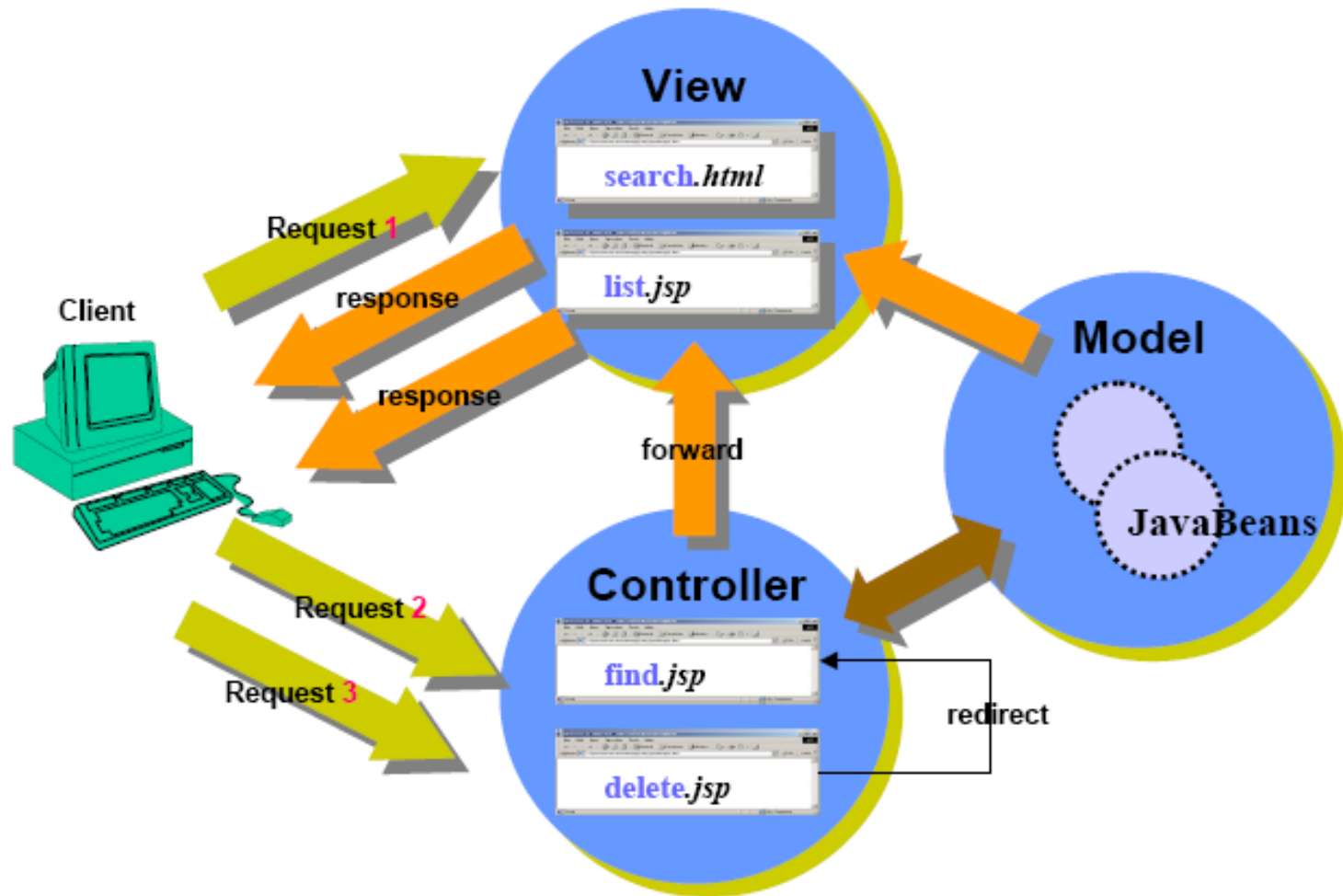
- result.jsp

Processing to DB using JSTL

```
<a href="menu.jsp">Back to menu</a>
```

MVC Design Pattern

MVC Model 1 – Generalization



Page-centric Scenario

MVC Design Pattern

MVC Model 1 – Example

- Display product's information (name, categories)
- **Bean class**

```
public class Products
```

```
{
```

Define the properties,

the get/set methods

```
}
```



Model 1 Demo

Product Name: axe

Product Type: tool

- **JSP**

```
<jsp:useBean id="prod" class="Mod1.Products" scope="session"/>
```

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

```
Product Name:<jsp:getProperty name="prod" property="name"/>
```

```
Product Type:<jsp:getProperty name="prod" property="type"/>
```

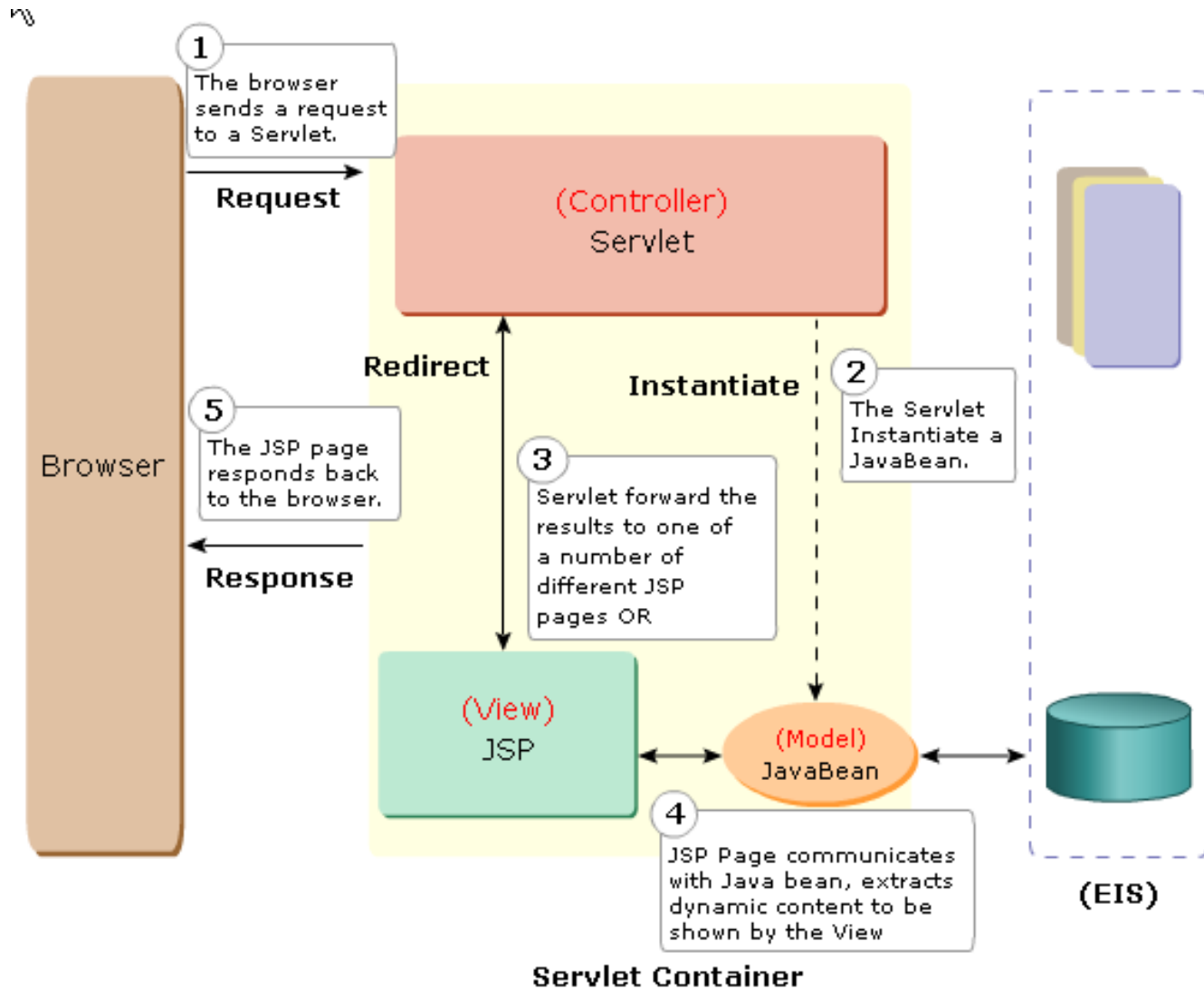
MVC Design Pattern

MVC Model 1 – Generalization

- **Purpose**
 - Separate “business logic” from “presentation logic”
 - Need for **centralized security control** or logging, changes little over time.
 - Apply to applications that have very simple page flow.
- **Advantages**
 - Lightweight design – for small, static application
 - Suitable for small application having very simple page flow, little need for centralized security control/logging
 - Separation of presentation from content
- **Limitations**
 - Navigation Problem – Changing name of JSP file must change in many location
 - Difficult to maintain an application – large java code being embedded in JSP page
 - Inflexible
 - Performance is not high
 - Not scale up over a period time
 - **Not Suitable for large and complex application**

MVC Design Pattern

MVC Model 2



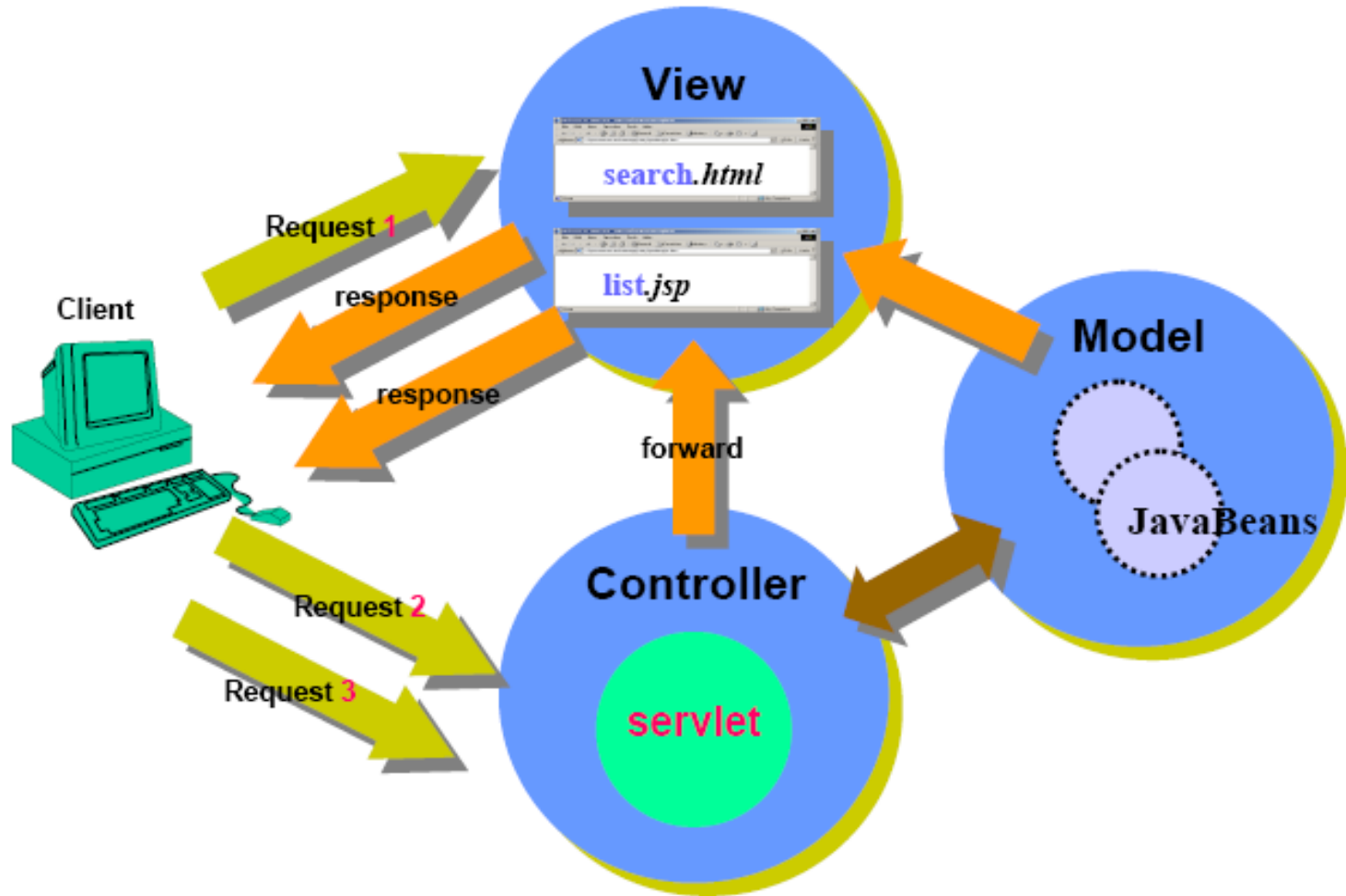
MVC Design Pattern

MVC Model 2

- **Separates** the “**Business Logic**” from the “**Presentation Logic**” and has an additional component – a Controller
- Use Servlet and JSP together
- **JSP pages**
 - Are used only for presentation
 - Retrieve the objects created by the Servlet
 - Extract dynamic content for insertion within a template for display
- **Servlet**
 - Handles initial request, partially process the data, set up beans, select suitable business logic to handle request , then forward the results to one of a number of different JSP pages
 - Serves as a gatekeeper provides common services, such as authentication authorization, login, error handling, and etc
 - Servlet serves as a central controller that act as a state machine or an event dispatcher to decide upon the appropriate logic to handle the request
 - Act as a Controller that controls the way Model and View layer interacts

MVC Design Pattern

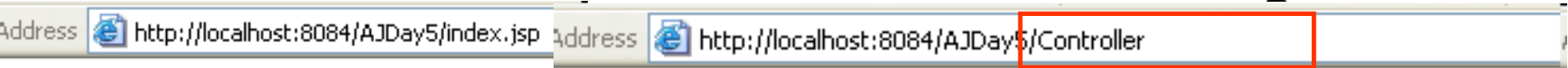
MVC Model 2 – Generalization



Servlet-centric Scenario

MVC Design Pattern

MVC Model 2 – Example



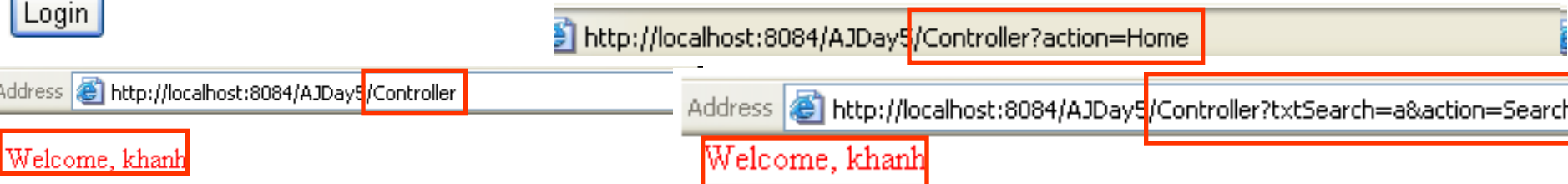
Login Page

Username

Password

Invalid Username or Password

[Click here to try again](#)



Welcome to MVC World!!!

Name

Search Result

No.	Username	Lastname	Roles
1	ajcourse	Course AJ	false
2	hibernate	khanhkt	false
3	jsf2_0	jsftask4	false

MVC Design Pattern

MVC Model 2 – Example

```

index.jsp x
[Icons]

4      Author      : Trong Khanh
5      --%>
6
7      <%@page contentType="text/html" pageEncoding="UTF-8"%>
8      <!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
9          "http://www.w3.org/TR/html4/loose.dtd">
10
11 <html>
12 <head>
13     <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
14     <title>Login</title>
15 </head>
16 <body>
17     <h1>Login Page</h1>
18     <form action="Controller" method="POST">
19         Username <input type="text" name="txtUsername" value="" /><br/>
20         Password <input type="password" name="txtPassword" value="" /><br/>
21         <input type="submit" value="Login" name="btAction" />
22         <input type="reset" value="Reset" />
23     </form>
24 </body>
25 </html>

```

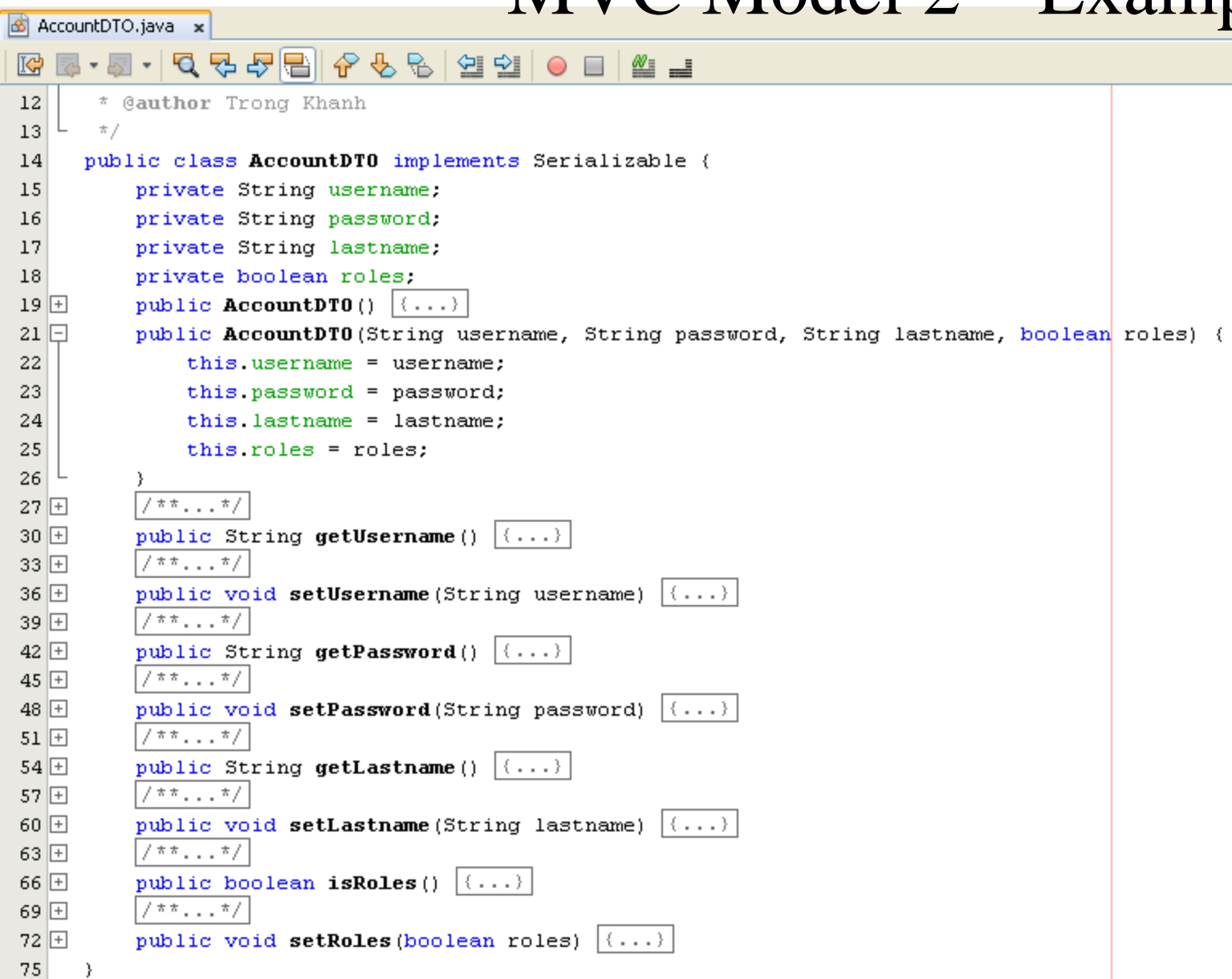
MVC Design Pattern

MVC Model 2 – Example

```
invalid.jsp x
[Icons]
4      Author      : Trong Khanh
5      --%>
6
7      <%@page contentType="text/html" pageEncoding="UTF-8"%>
8      <!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
9          "http://www.w3.org/TR/html4/loose.dtd">
10
11  <html>
12      <head>
13          <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
14          <title>Invalid</title>
15      </head>
16      <body>
17          <h1><font color="red">Invalid username and password!!!!</font> </h1>
18
19          <a href="Controller?action=Home">Click here to try again</a>
20      </body>
21  </html>
```


MVC Design Pattern

MVC Model 2 – Example



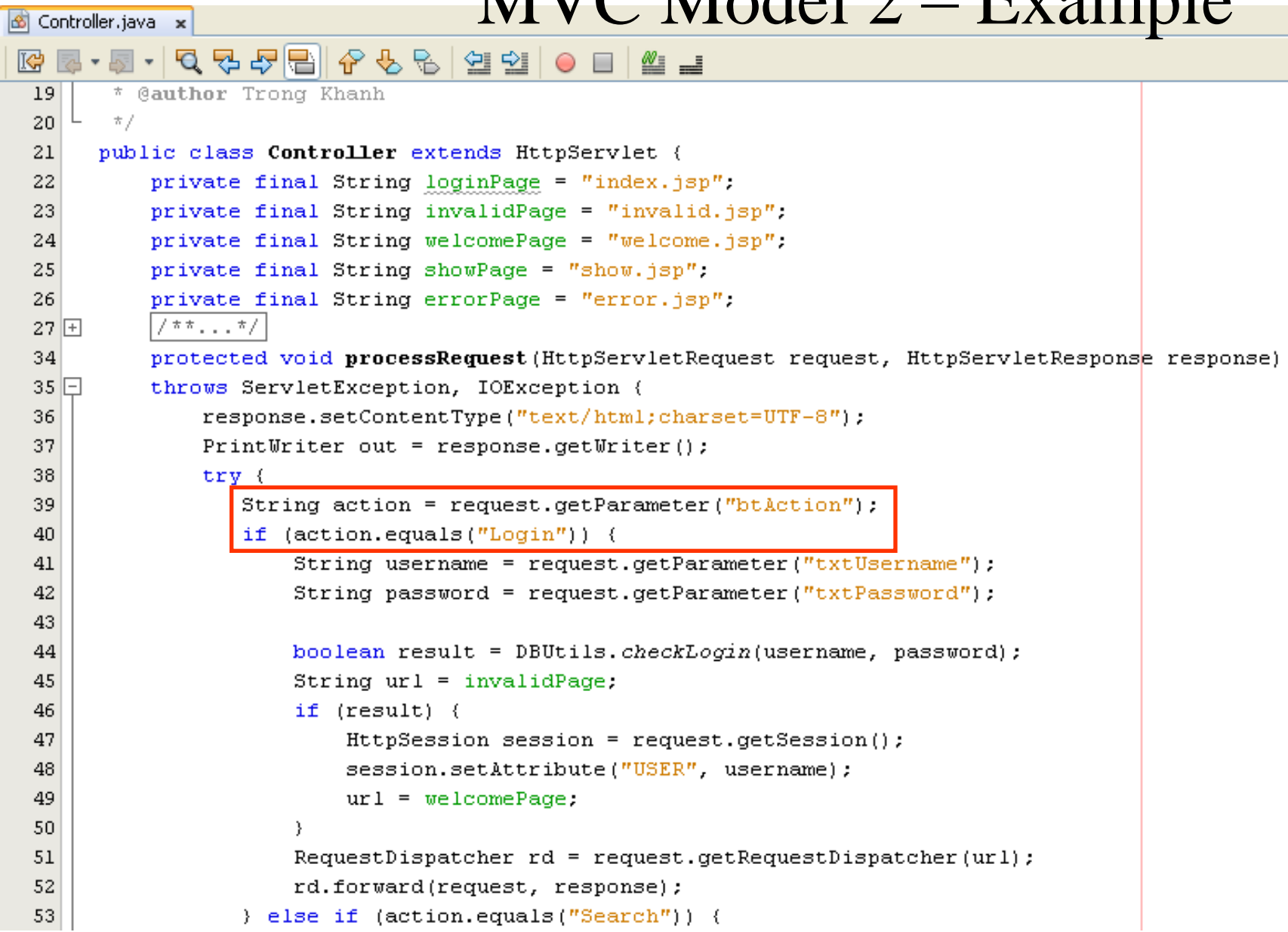
```

12  * @author Trong Khanh
13  */
14  public class AccountDTO implements Serializable {
15      private String username;
16      private String password;
17      private String lastname;
18      private boolean roles;
19      public AccountDTO() { ... }
20
21      public AccountDTO(String username, String password, String lastname, boolean roles) {
22          this.username = username;
23          this.password = password;
24          this.lastname = lastname;
25          this.roles = roles;
26      }
27      /**...*/
30      public String getUsername() { ... }
31
32      /**...*/
33
34      public void setUsername(String username) { ... }
35
36      /**...*/
37
38      public String getPassword() { ... }
39
40      /**...*/
41
42      public void setPassword(String password) { ... }
43
44      /**...*/
45
46      public String getLastName() { ... }
47
48      /**...*/
49
50      public void setLastName(String lastname) { ... }
51
52      /**...*/
53
54      public boolean isRoles() { ... }
55
56      /**...*/
57
58      public void setRoles(boolean roles) { ... }
59  }

```

MVC Design Pattern

MVC Model 2 – Example



```

19  * @author Trong Khanh
20  */
21  public class Controller extends HttpServlet {
22      private final String loginPage = "index.jsp";
23      private final String invalidPage = "invalid.jsp";
24      private final String welcomePage = "welcome.jsp";
25      private final String showPage = "show.jsp";
26      private final String errorPage = "error.jsp";
27      /**...*/
34  protected void processRequest(HttpServletRequest request, HttpServletResponse response)
35  throws ServletException, IOException {
36      response.setContentType("text/html;charset=UTF-8");
37      PrintWriter out = response.getWriter();
38      try {
39          String action = request.getParameter("btAction");
40          if (action.equals("Login")) {
41              String username = request.getParameter("txtUsername");
42              String password = request.getParameter("txtPassword");
43
44              boolean result = DBUtils.checkLogin(username, password);
45              String url = invalidPage;
46              if (result) {
47                  HttpSession session = request.getSession();
48                  session.setAttribute("USER", username);
49                  url = welcomePage;
50              }
51              RequestDispatcher rd = request.getRequestDispatcher(url);
52              rd.forward(request, response);
53          } else if (action.equals("Search")) {
  
```

MVC Design Pattern

MVC Model 2 – Example

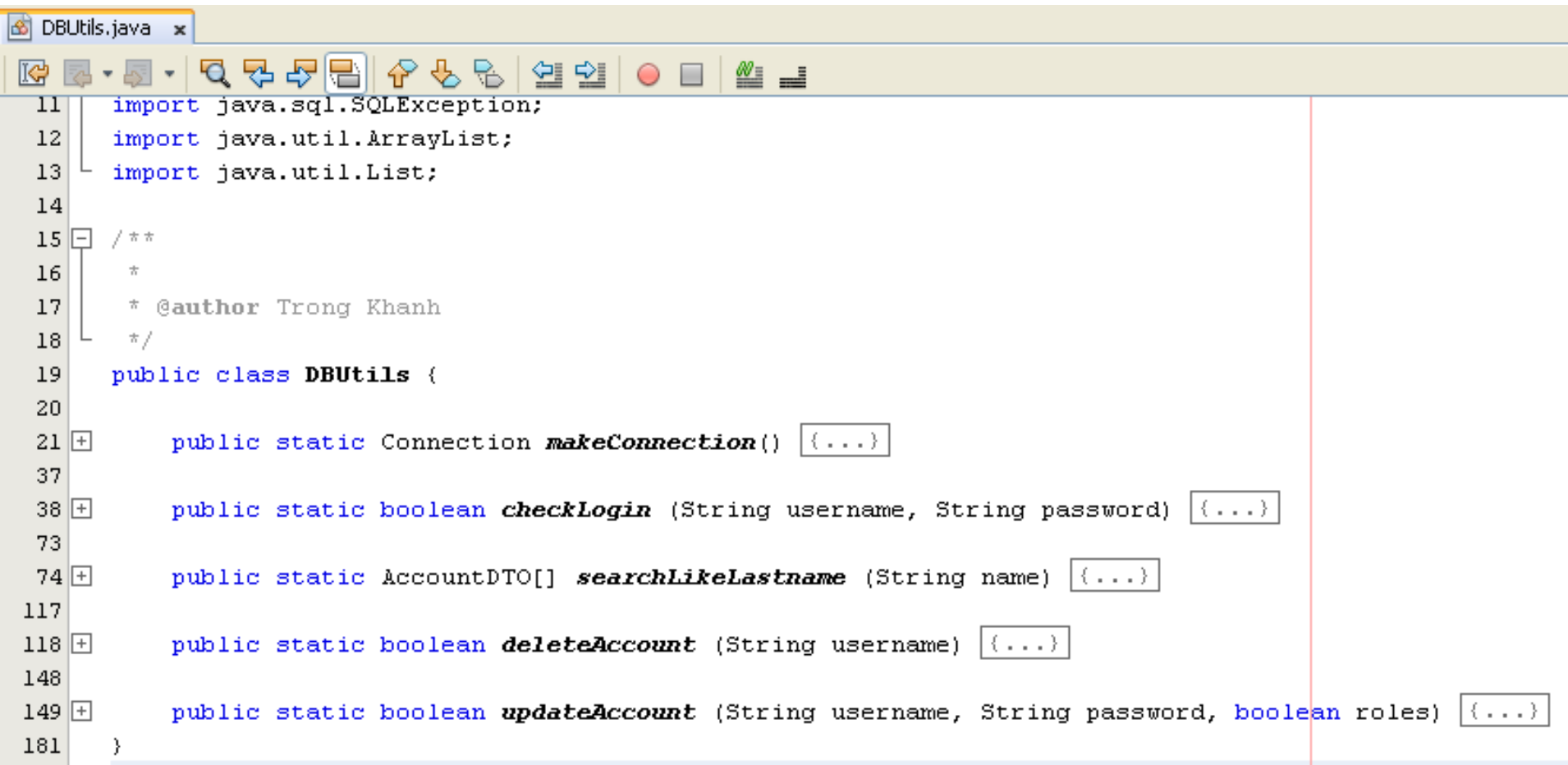
```

53 } else if (action.equals("Search")) {
54     String name = request.getParameter("txtName");
55
56     AccountDTO[] result = DBUtils.searchLikeLastname(name);
57     request.setAttribute("INFO", result);
58
59     RequestDispatcher rd = request.getRequestDispatcher(showPage);
60     rd.forward(request, response);
61 } else if (action.equals("Home")) {
62     RequestDispatcher rd = request.getRequestDispatcher(loginPage);
63     rd.forward(request, response);
64 } else if (action.equals("Delete")) {
65     String username = request.getParameter("txtUsername");
66     String name = request.getParameter("txtSearch");
67
68     boolean result = DBUtils.deleteAccount(username);
69     String url = errorPage;
70     if (result) {
71         url = "Controller?btAction=Search&txtName=" + name;
72     }
73     response.sendRedirect(url);
74 }
75 } finally {
76     out.close();
77 }
78 }

```

MVC Design Pattern

MVC Model 2 – Example



```
11 import java.sql.SQLException;
12 import java.util.ArrayList;
13 import java.util.List;
14
15 /**
16  *
17  * @author Trong Khanh
18  */
19 public class DBUtils {
20
21     public static Connection makeConnection() {...}
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37     public static boolean checkLogin (String username, String password) {...}
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73     public static AccountDTO[] searchLikeLastname (String name) {...}
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108     public static boolean deleteAccount (String username) {...}
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139     public static boolean updateAccount (String username, String password, boolean roles) {...}
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181 }
```

MVC Design Pattern

MVC Model 2 – Example

```

welcome.jsp x
[Icons]

4      Author      : Trong Khanh
5      --%>
6
7      <%@page contentType="text/html" pageEncoding="UTF-8"%>
8      <!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
9          "http://www.w3.org/TR/html4/loose.dtd">
10
11  <html>
12  <head>
13      <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
14      <title>Home</title>
15  </head>
16  <body>
17      <font color="red">Welcome, <%= session.getAttribute("USER") %></font>
18      <h1>Welcome to MVC world</h1>
19      <form action="Controller">
20          Name <input type="text" name="txtName" value="" /><br/>
21          <input type="submit" value="Search" name="btAction" />
22      </form>
23  </body>
24  </html>
  
```

MVC Design Pattern

MVC Model 2 – Example

```

show.jsp x
[Icons]
4 | Author      : Trong Khanh
5 | ---%>
6 |
7 | <%@page import="sample.web.AccountDTO"%>
8 | <%@page contentType="text/html" pageEncoding="UTF-8"%>
9 | <!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
10 |   "<a href='\"http://www.w3.org/TR/html4/loose.dtd\"'>http://www.w3.org/TR/html4/loose.dtd">
11 | <html>
12 |   <head>
13 |     <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
14 |     <title>Show</title>
15 |   </head>
16 |   <body>
17 |     <font color="red">Welcome, <%= session.getAttribute("USER")%></font>
18 |     <h1>Show Result Page</h1>
19 |     <%
20 |         AccountDTO[] result = (AccountDTO[]) request.getAttribute("INFO");
21 |         if (result != null) {
22 |             %>
23 |         <table border="1">
24 |             <thead>
25 |                 <tr>
26 |                     <th>No.</th>
27 |                     <th>Username</th>
28 |                     <th>Password</th>
29 |                     <th>Lastname</th>
30 |                     <th>Roles</th>
31 |                     <th>Delete</th>
32 |                 </tr>
33 |             </thead>

```

MVC Design Pattern

MVC Model 2 – Example

```

34 <tbody>
35 <%
36     for (int i = 0; i < result.length; i++) {
37 <%>
38 <form action="Controller">
39 <tr>
40 <td><%= i + 1%></td>
41 <td>
42 <%= result[i].getUsername()%>
43 <input type="hidden" name="txtUsername"
44     value="<%= result[i].getUsername() %>" />
45 </td>
46 <td><%= result[i].getPassword()%></td>
47 <td><%= result[i].getLastName()%></td>
48 <td><%= result[i].isRoles()%></td>
49 <td>
50 <input type="hidden" name="txtSearch"
51     value="<%= request.getParameter("txtName") %>" />
52 <input type="submit" value="Delete" name="btAction" />
53 </td>
54 </tr>
55 </form>
56 <%
57     } //end for
58 <%>
59 </tbody>
60 </table>
61 <%
62     } //end if
63 <%>
  
```

MVC Design Pattern

- **Purpose** MVC Model 2 – Generalization
 - **Separation** of presentation logic and business logic
 - A **Model** represents information specific to a particular domain on which the application operates
 - A **View** is typically a user interface element. A single Model may be presented as multiple views
 - A **controller** component responds to events and processes request and may invoked changes on the Model
- **Advantages**
 - **Easier** to build, maintain and extend
 - Provide **single point** of control (Servlet) for security & logging
 - **Encapsulate** incoming data into a form usable by the backend
 - Can **reusable code**
- **Limitations**
 - Increase Design Complexity

MVC Design Pattern

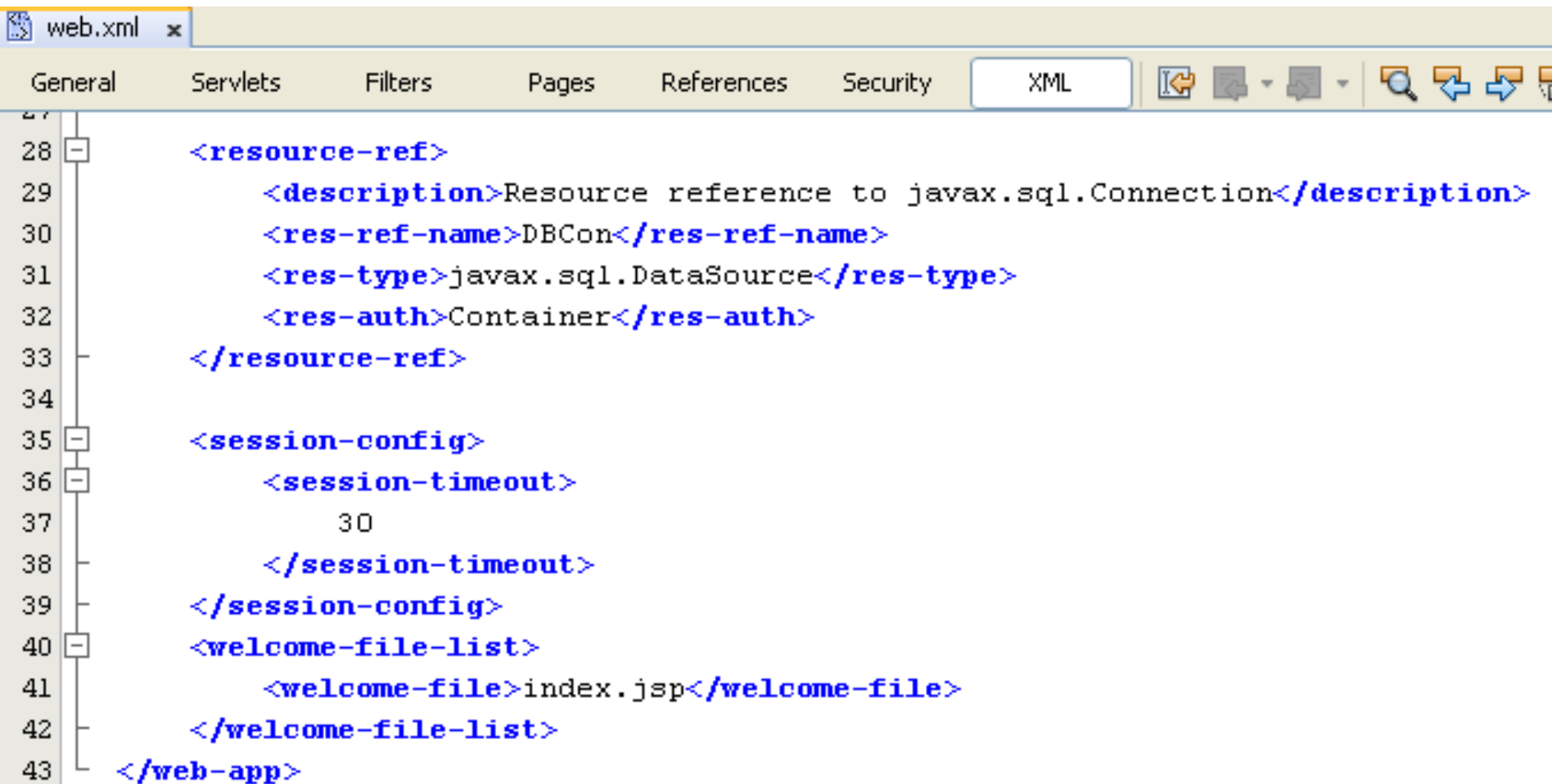
MVC Model 1 & Model 2 Comparison

Criteria	Model 1	Model 2
JSP page responsibility	Processing requests and sending back replies to clients	Are used for the presentation layer
Servlets responsibility	N/A	Are used for processing task
Type of Web application	Simple	Complex
Nature of Developer's Task	Quick prototyping	Developing an application that can be modified and maintained
Who is doing the work?	View and Controller is developed by the same team	View and Controller is developed by the different teams

Additional

Dynamic DB Connection

- Adding and modify the **web.xml** as following



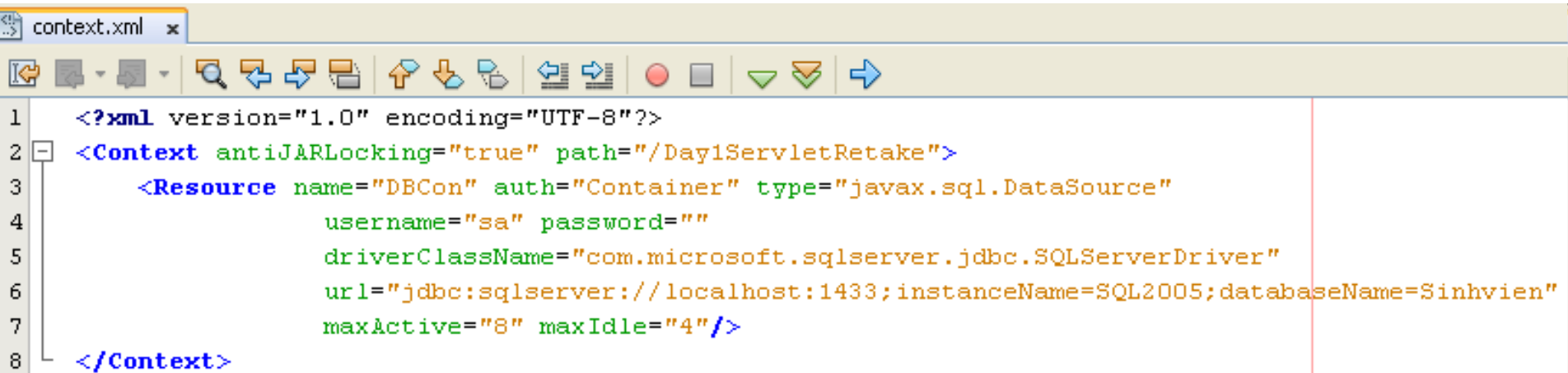
```

web.xml x
General Servlets Filters Pages References Security XML
28 <resource-ref>
29     <description>Resource reference to javax.sql.Connection</description>
30     <res-ref-name>DBCon</res-ref-name>
31     <res-type>javax.sql.DataSource</res-type>
32     <res-auth>Container</res-auth>
33 </resource-ref>
34
35 <session-config>
36     <session-timeout>
37         30
38     </session-timeout>
39 </session-config>
40 <welcome-file-list>
41     <welcome-file>index.jsp</welcome-file>
42 </welcome-file-list>
43 </web-app>
  
```

Additional

Dynamic DB Connection

- Adding and modify the **context.xml** in the **META-INF** directory as following



```

1  <?xml version="1.0" encoding="UTF-8"?>
2  <Context antiJARLocking="true" path="/Day1ServletRetake">
3      <Resource name="DBCon" auth="Container" type="javax.sql.DataSource"
4          username="sa" password=""
5          driverClassName="com.microsoft.sqlserver.jdbc.SQLServerDriver"
6          url="jdbc:sqlserver://localhost:1433;instanceName=SQL2005;databaseName=Sinhvien"
7          maxActive="8" maxIdle="4"/>
8  </Context>
  
```

- Implement code to use

```

Context ctx = new InitialContext();
Context envCtx = (Context) ctx.lookup("java:comp/env");
DataSource ds = (DataSource) envCtx.lookup("DBCon");
Connection con = ds.getConnection();
  
```

Summary

- **Introduction to JSP**
- **JSP Life Cycle**
- **JSP Elements**
- **JSP Directives**
- **JSP Implicit Objects**
- **Techniques: MVC Design Pattern**

Q&A

Next Lecture

- **JSP Standard Actions**
 - JavaBeans
 - Standard Actions
- **Dispatcher Mechanism**
 - Including, Forwarding, and Parameters
 - Vs. Dispatcher in Servlets
- **EL – Expression Languages**
 - What is EL?
 - How to use EL in JSP?