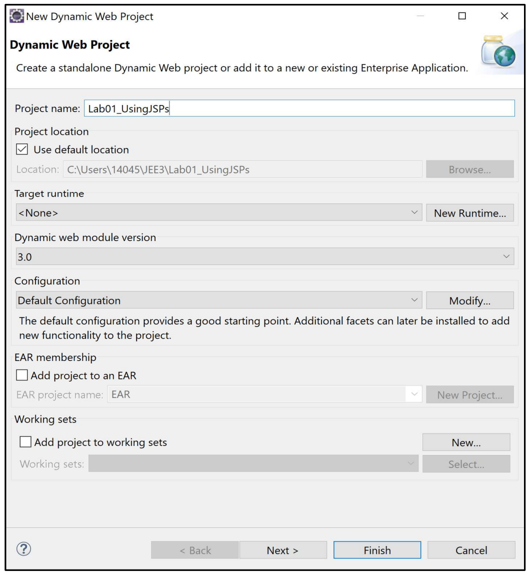
Lab 1 - Using JSPs

- **Launch Eclipse, selecting a default workspace**
- **Create a new Dynamic Web Project**
  - Use the menu bar to select a new project (Files>New>Project…)
  - Search for Dynamic Web Project in the search bar, select the Dynamic Web Project and then "Next >"
  - Enter "Lab01_UsingJSPs" as the Project Name and then select "Finish >" [1]



New Dynamic Web Project

Dynamic Web Project

Create a standalone Dynamic Web project or add it to a new or existing Enterprise Application.

Project name: Lab01_UsingJSPs

Project location

☑ Use default location

Location: C:\Users\14045\JEE3\Lab01_UsingJSPs

Target runtime: <None>

Dynamic web module version: 3.0

Configuration: Default Configuration

The default configuration provides a good starting point. Additional facets can later be installed to add new functionality to the project.

EAR membership

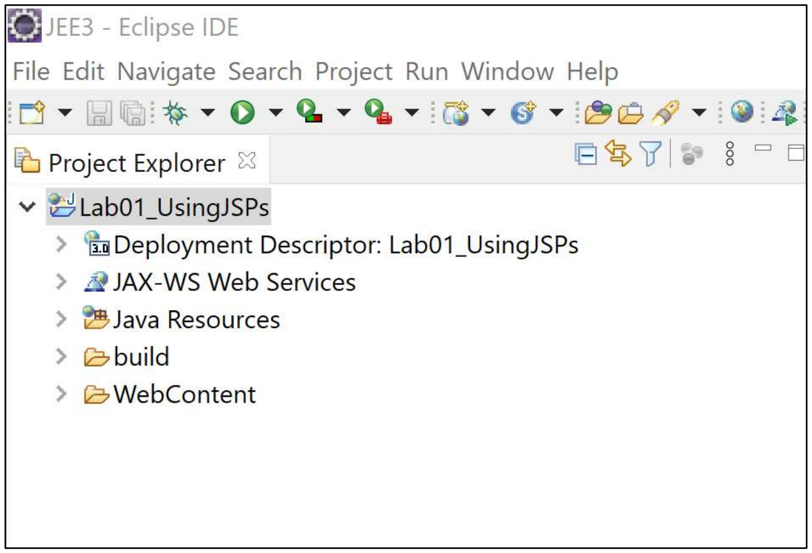
☐ Add project to an EAR

EAR project name: EAR

Working sets

☐ Add project to working sets

Working sets:



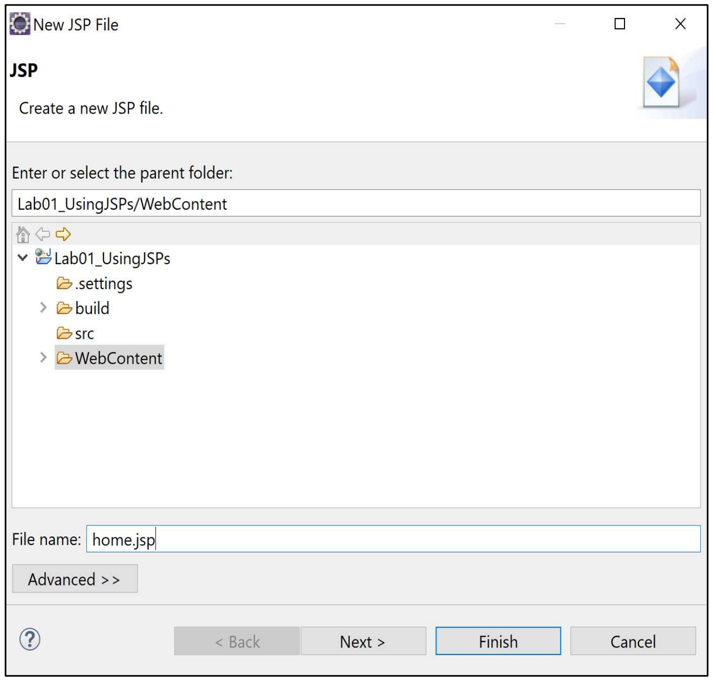
JEE3 - Eclipse IDE

- Lab01_UsingJSPs
  - Deployment Descriptor: Lab01_UsingJSPs
  - JAX-WS Web Services
  - Java Resources
  - build
  - WebContent

  - Open the EE perspective if prompted (otherwise select Window>Perspective>Open Perspective>Java EE to open the EE perspective)

Lab 1 - Using JSPs

- **Right click on WebContent and create a new JSP file**
- Name it home.jsp
  - Click "Finish"


New JSP File

JSP

Create a new JSP file.

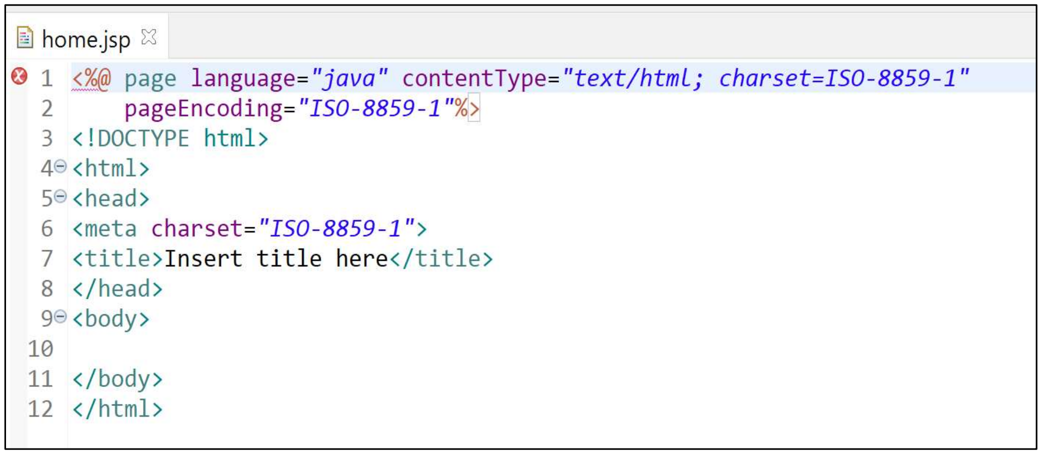
Enter or select the parent folder:

Lab01_UsingJSPs/WebContent

```
Lab01_UsingJSPs
    .settings
  > build
    src
  > WebContent
```

File name: home.jsp

- **Don't worry about the error on line one, we'll correct it shortly**

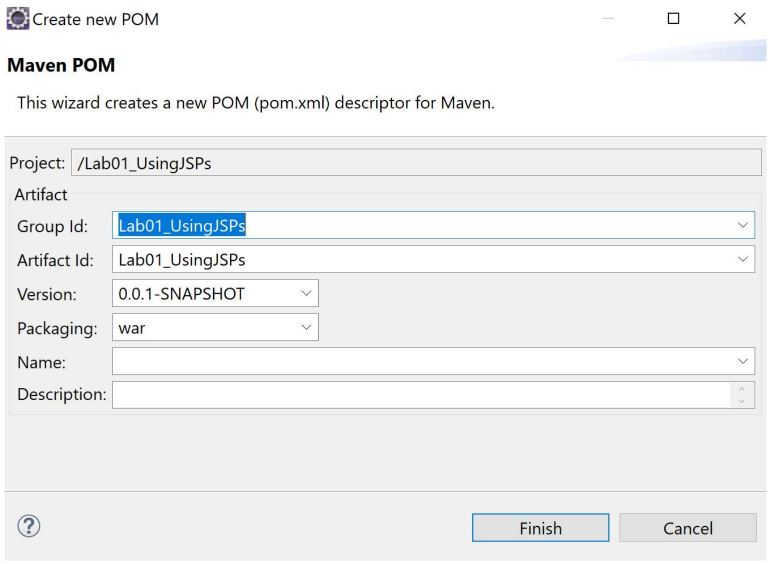

home.jsp

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>
<!DOCTYPE html>
<html>
<head>
<meta charset="ISO-8859-1">
<title>Insert title here</title>
</head>
<body>

</body>
</html>
```

Lab 1 - Using JSPs

Convert the project to a Maven project

- Within the Project Explorer, configure the project as a Maven project (right click on the project> Configure>Convert to Maven Project)
- Within the Create new POM window, select Finish to convert the project


Create new POM

Maven POM

This wizard creates a new POM (pom.xml) descriptor for Maven.

Project: /Lab01_UsingJSPs

Artifact

Group Id:	Lab01_UsingJSPs
Artifact Id:	Lab01_UsingJSPs
Version:	0.0.1-SNAPSHOT
Packaging:	war
Name:	
Description:	

Lab 1 - Using JSPs

Add the Servlet API maven dependency to the pom.xml file

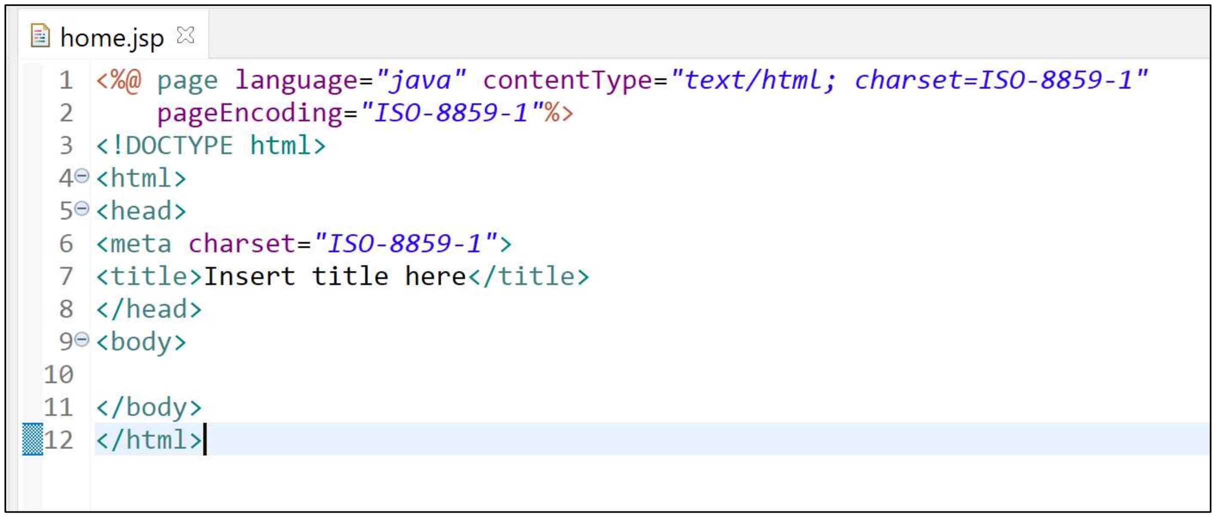
- Open the pom.xml file and add the following (immediately before the build tag):

```
<dependencies>
        <dependency>
                <groupId>org.apache.tomcat</groupId>
                <artifactId>tomcat-servlet-api</artifactId>
                <version>9.0.1</version>
        </dependency>
</dependencies>
```

- Save the pom.xml file
- Update the Maven Project
  - ✓ Within the Explorer window, right click on the project > Maven > Update Project

Lab 1 - Using JSPs

At this point the error in your jsp should be resolved



home.jsp

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>
<!DOCTYPE html>
<html>
<head>
<meta charset="ISO-8859-1">
<title>Insert title here</title>
</head>
<body>

</body>
</html>
```

Lab 1 - Using JSPs

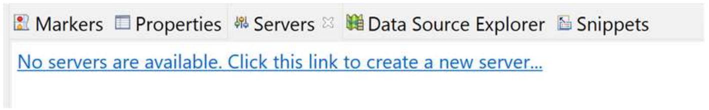
Add the following code to the body of your home.jsp

```
<body>

  <h3>Welcome Home</h3> <br/><hr/>

  The protocol is: </p> <% out.println(request.getProtocol()); %> <br/>
  The port is: <% out.println(request.getServerPort()); %> <br/>
  The remote address is: <% out.println(request.getRemoteAddr()); %> <br/>
  The context path is: <% out.println(request.getContextPath()); %> <br/>

</body>
```

- Don't worry about the strange <% %> tags, we'll explain them soon
- Be careful to add semi-colons and parentheses exactly shown
- Can you predict the output of this page?

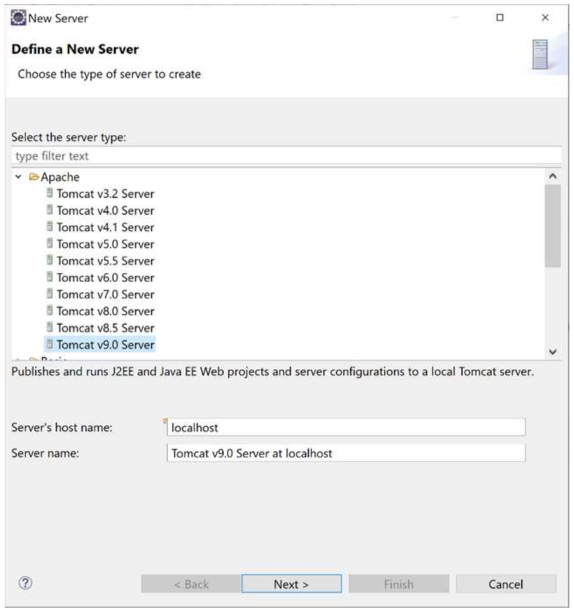
Lab 1 - Using JSPs

- **Add a Server**
  - Within the server tab, click on "Click this link to create a new server…" (if this window isn't visible, then use the menu bar to make it visible Window>Show Window>Servers…



No servers are available. Click this link to create a new server...

  - Within the New Server window, select Apache>Tome v9.0 Server and then select "Next >"



Define a New Server

Choose the type of server to create

Select the server type:

type filter text

- Apache
  - Tomcat v3.2 Server
  - Tomcat v4.0 Server
  - Tomcat v4.1 Server
  - Tomcat v5.0 Server
  - Tomcat v5.5 Server
  - Tomcat v6.0 Server
  - Tomcat v7.0 Server
  - Tomcat v8.0 Server
  - Tomcat v8.5 Server
  - Tomcat v9.0 Server

Publishes and runs J2EE and Java EE Web projects and server configurations to a local Tomcat server.

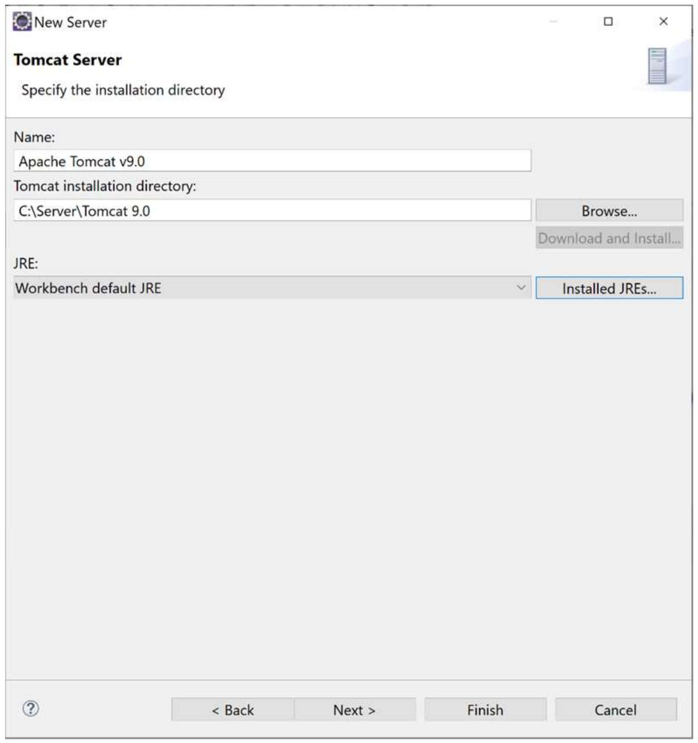
Server's host name: localhost

Server name: Tomcat v9.0 Server at localhost

Lab 1 - Using JSPs

Add a Server

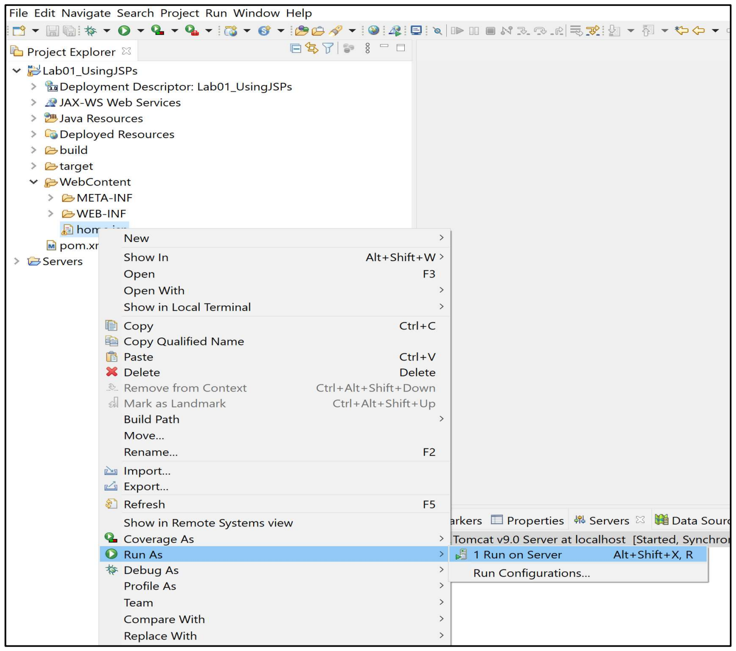
- Within the New Server window, use Browse… to locate and set the Tomcat Installation directory
- Select "Finish" to create the Tomcat Server

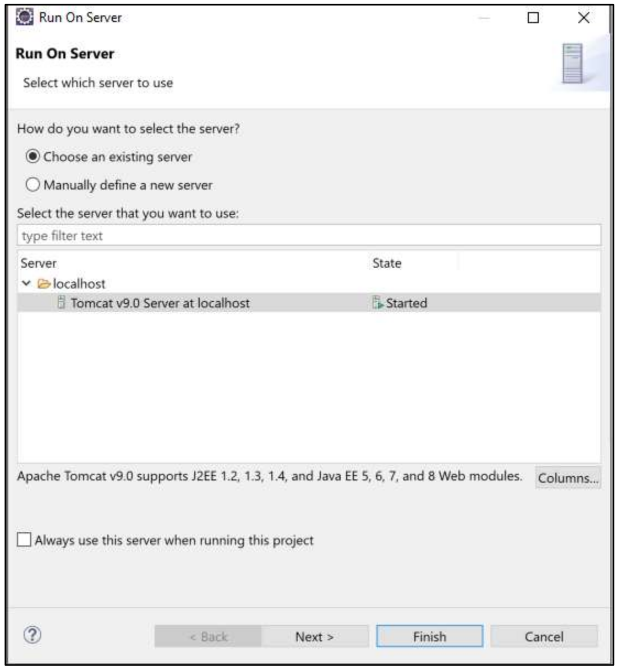


Name:
Apache Tomcat v9.0
Tomcat installation directory:
C:\Server\Tomcat 9.0
JRE:
Workbench default JRE

Lab 1 - Using JSPs

- **Use an external Web Browser**
  - On the menu bar select Window>Web Browser > Chrome if it exists or anything other than Internal Web Browser
- **Run the application**
  - Within the Project Explorer window, right click on the welcome.html file and select Run> Run As > Run on Server
  - Confirm that Tomcat v9.0 Server at localhost is selected and then select "Finish"





Run On Server
Run On Server
Select which server to use
How do you want to select the server?
Choose an existing server
Manually define a new server
Select the server that you want to use:
type filter text
Server
State
localhost
Tomcat v9.0 Server at localhost
Started
Columns...
Always use this server when running this project
< Back
Next >
Finish
Cancel

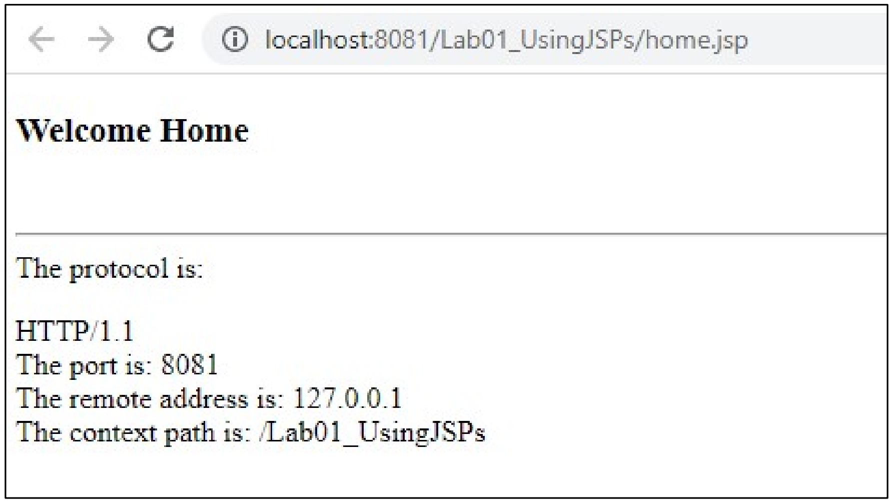
Apache Tomcat v9.0 supports J2EE 1.2, 1.3, 1.4, and Java EE 5, 6, 7, and 8 Web modules.

Lab 1 - Using JSPs

Test drive the application; did you notice a slight delay when pages first accessed?

- May not be noticeable on a really fast machine
- Can take up to several seconds

Your output should look something like this


localhost:8081/Lab01_UsingJSPs/home.jsp

```
Welcome Home

The protocol is:

HTTP/1.1
The port is: 8081
The remote address is: 127.0.0.1
The context path is: /Lab01_UsingJSPs
```

STOP

Module 1, Lesson 1 - Discussion Question

What’s the benefit of using JSPs just for presentation of content?

Module 1, Lesson 1 – Activity

Arrange the following lifecycle steps and their output in the correct chronological order	
Container translates the file	home_jsp.jsp
Class is loaded and intialized	home.jsp
File is compiled	home_jsp.class
Developer Writes the Code	home_jsp

Module 1, Lesson 1 – Activity SOLUTION

Arrange the following lifecycle steps and their output in the correct chronological order	
Developer Writes the Code	home.jsp
Container translates the file	home_jsp.jsp
File is compiled	home_jsp.class
Class is loaded and intialized	home_jsp

Lesson Questions

See M1L1Q.docx for 3 ungraded questions

Module 1, Lesson 2 Scripting Elements

x

Module 1, Lesson 2 - Description

In this lesson we'll learn how to work with Scriplets, Expressions, and Declarations to create dynamic Java Server Pages.

Learning Objectives

Scripting Elements

Upon Completion of this lesson, learners will be able to:

- Explain the syntax for scripting elements
- Understand the differences between Scriplets, Expression, and Declaration
- Write code that uses scripting elements

Syntax Rules for Scripting Elements

All scripting elements delineated with <% and %>

<% %>	**Scriptlet**
<%-- --%>	**Comment**
<%= %>	**Expression**
<%@ %>	**Directive**
<%! %>	**Declaration**

Follows XML syntax rules

Case sensitive

- Tags must be closed
  - Can use **<tag/>** for empty tags
- Attribute values must be quoted
  - Single or double quotes

Comments

Marked with <%-- --%>

JSPs can use HTML comments (remember, JSP = HTML+)

```
<!-- comment goes here -->
```

- Comments are visible in the browser
- User can see with "View Source"

JSP comments also available

```
<%-- comment goes here --%>
```

- Not returned to the browser

Scriptlets

Marked with <% %>

Fragments of straight Java code

- **Inserted into the JSP's service method**, in-place and as-is
- Executes each time the JSP is accessed

Full power of Java and Java EE at your disposal

- Query a DB, call a Web Service, perform some calculation
- But remember MVC
  - Just because you can, doesn't mean you should
  - JSPs are for presentation!

Scriptlet - Example

Inserted into the generated servlet's service method

```
The current date and time is:
<p>
<%
   // format the date
   java.text.Format formatter = new java.text.SimpleDateFormat();
   String prettyDate = formatter.format(new java.util.Date());
   out.println(prettyDate);
%>
</p>
Formatted dates are nice.
```

The HTML is output automatically by the generated servlet

- Output from the scriptlet must be done via **out.println**
- The variables are local variables in the JSP's service method

Expressions

Marked with <%= %>

Outputs a string value to the client, in-place

- Expression converted to String (if necessary) and output
- An out.println call is generated for you (it has the semicolon)

Inserted in an out.print statement in the generated servlet's service method

```
The current date and time is:
<p>
<%
  // format the date
  java.text.Format formatter = new java.text.SimpleDateFormat();
  String prettyDate = formatter.format(new java.util.Date());
%>
<%= prettyDate %>
</p>
Formatted dates are nice.
```

Notice that the expression does not include a semicolon

```
Your browser is: <%= request.getHeader("User-Agent") %>
```

Declarations

Marked with <%! %>

Declare instance variables and methods in generated servlet

- Which you can then use in scriptlets and expressions

They are processed once, at translation time

- And like directives, can occur anywhere in the JSP file

Should be used rarely, if ever – remember the word of caution!

- If you need fields and methods, just write a class in Java and use it from the JSP
  - Recall that instance variables are not thread-safe
    - But can be safely used in a read-only fashion

Declarations - Example

```
<%@ page import='java.text.Format, java.text.SimpleDateFormat' %>

<%! private Format formatter = new SimpleDateFormat(); %>
The current date and time is:
<p>
<%= getDate() %>
</p>
Formatted dates are nice.

<%!
  private String getDate() {
    return formatter.format(new java.util.Date());
  }
%>
```

Interweaving Scriptlets with HTML

Multiple scriptlets may be included in a JSP

- All are inserted into the JSP's service method, in-place and as-is

Can have HTML tags between scriptlets

- They are simply output by the generated servlet, in-place

Can start a Java statement in one scriptlet and finish it in a later scriptlet

- Be careful – very easy to mess this up

```
<%
  String num = (String) application.getAttribute("NUM_ITERATIONS");
  int numIterations = Integer.parseInt(num);

  for (int i = 0; i < numIterations; i++) {
%>
  <h3>HELLO</h3>
<%
  }
%>
```

Lab 2 – Scripting Elements

Import the Lab02_ScriptingElements_STARTER file and rename it to Lab02_ScriptingElements

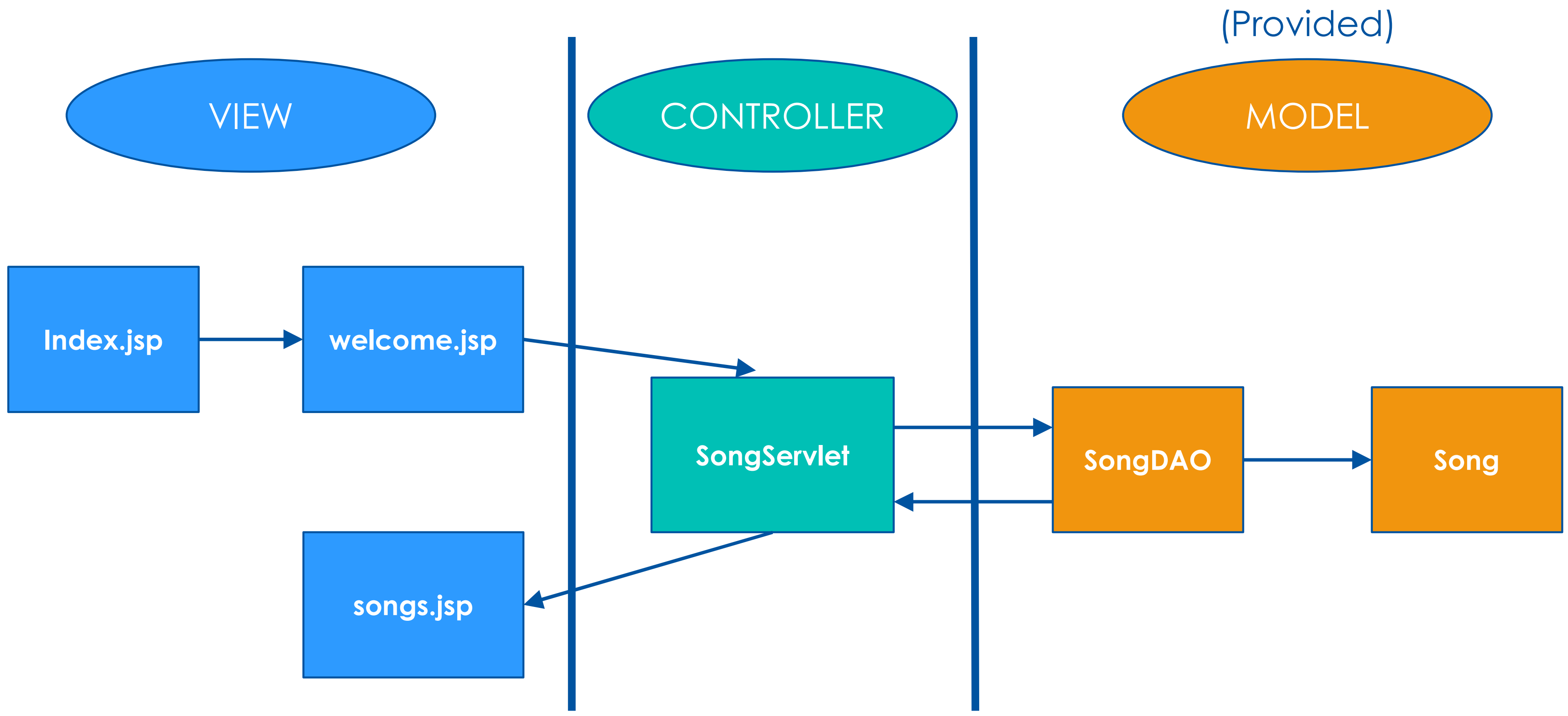
- This is a maven project with the Tomcat dependency already added

- Examine the Song and SongDAO files in the com.music.model package
  - Song is a simple POJO (Plain Old Java Object)
  - SongDAO contains our fake data and access methods

A partially complete songs.jsp file is provided which we'll complete later

Lab 2 – Scripting Elements

Application Overview



Lab 2 – Scripting Elements

- Right click on WebContent > New > JSP File
- Create an index.jsp page and add the code below

```
…
<title>Welcome</title>
</head>
<body>
        <h3>Welcome to Simple Song Service</h3>
        <br>
        <hr>
        <h3>Please enter your user name</h3>
        <form action="welcome.jsp">
        <label   for="username">User Name:</label>
                <input type="text"  name="username"> <br/><br/>
                <input type="submit" value="Next">
        </form>
</body>
```

Lab 2 – Scripting Elements

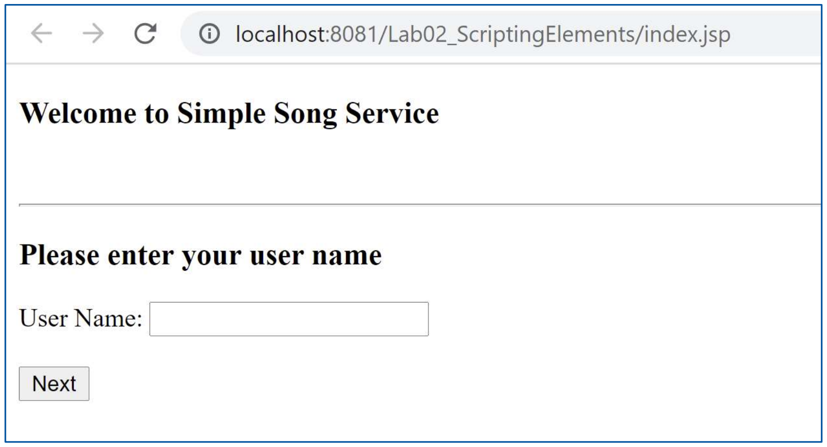
- Next, create a welcome.jsp file
  - Set the title to “Simple Song Service”
  - Add an Expression to get the user name from the request and welcome them
  - Add a link to invoke the get method of SongServlet (which we’ll write next)

```
…
<title>Simple Song Site</title>
</head>
<body>
  <h3>Welcome to Simple Song Service</h3><br><hr>
  <p><%= "Welcome " + request.getParameter("username") %></p>
  <a href="SongServlet">Show All Songs</a>
</body>
```

Lab 2 – Scripting Elements

- You should be able to run the application on the server and see the following output
  - Note the “Show All Songs” link doesn’t work yet

index.jsp



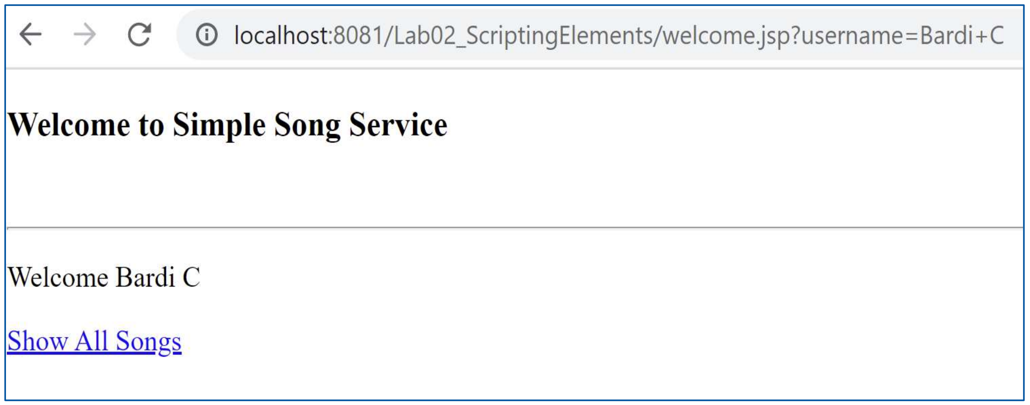
localhost:8081/Lab02_ScriptingElements/index.jsp

Welcome to Simple Song Service

Please enter your user name

User Name:

welcome.jsp



localhost:8081/Lab02_ScriptingElements/welcome.jsp?username=Bardi+C

Welcome to Simple Song Service

Welcome Bardi C

Show All Songs

Lab 2 – Scripting Elements

- Create a new Servlet to serve as our controller
  - Name it SongServlet
  - Put it in the com.music.controller package
  - Override the doGet method to do the following:
    - Create a new SongDAO and call the findAllSongs method
    - Set a session attribute with the key “Songs”. The value will be the list returned from findAllSongs
    - Forward the request to songs.jsp (which we’ll write next)

```
protected void doGet(HttpServletRequest request, HttpServletResponse response)
                                        throws ServletException, IOException {
                          List<Song> allSongs = new SongDAO().findAllSongs();
                          HttpSession session = request.getSession();
                          session.setAttribute("Songs", allSongs);
                          request.getRequestDispatcher("/songs.jsp").forward(request,response);
}
```

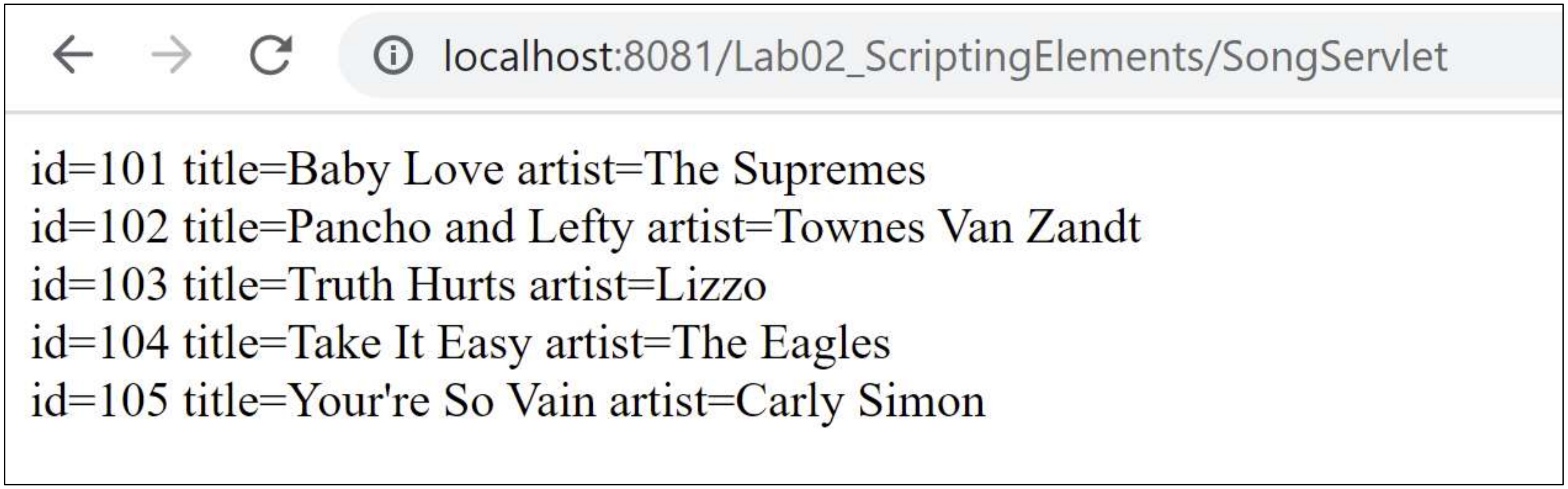
Lab 2 – Scripting Elements

- Open the provided songs.jsp file
- Add a Scriptlet (think <% %>), that does the following:
  - Gets the list of songs from session attribute set in our SongServlet doGet method
  - Loop through the list and print out each song (including id, title, and artist)

```
<%
  List<Song> songs = (List<Song>)session.getAttribute("Songs");
  for(Song _song : songs){
            out.print(_song);
            out.print("<br/>");
  }
%>
```

Lab 2 – Scripting Elements

Run the application. Your output should look something like this



localhost:8081/Lab02_ScriptingElements/SongServlet

```
id=101 title=Baby Love artist=The Supremes
id=102 title=Pancho and Lefty artist=Townes Van Zandt
id=103 title=Truth Hurts artist=Lizzo
id=104 title=Take It Easy artist=The Eagles
id=105 title=Your're So Vain artist=Carly Simon
```

Lab 2 – Scripting Elements

Let's add a declaration (<%! %>) and one more expression (<%= %>)

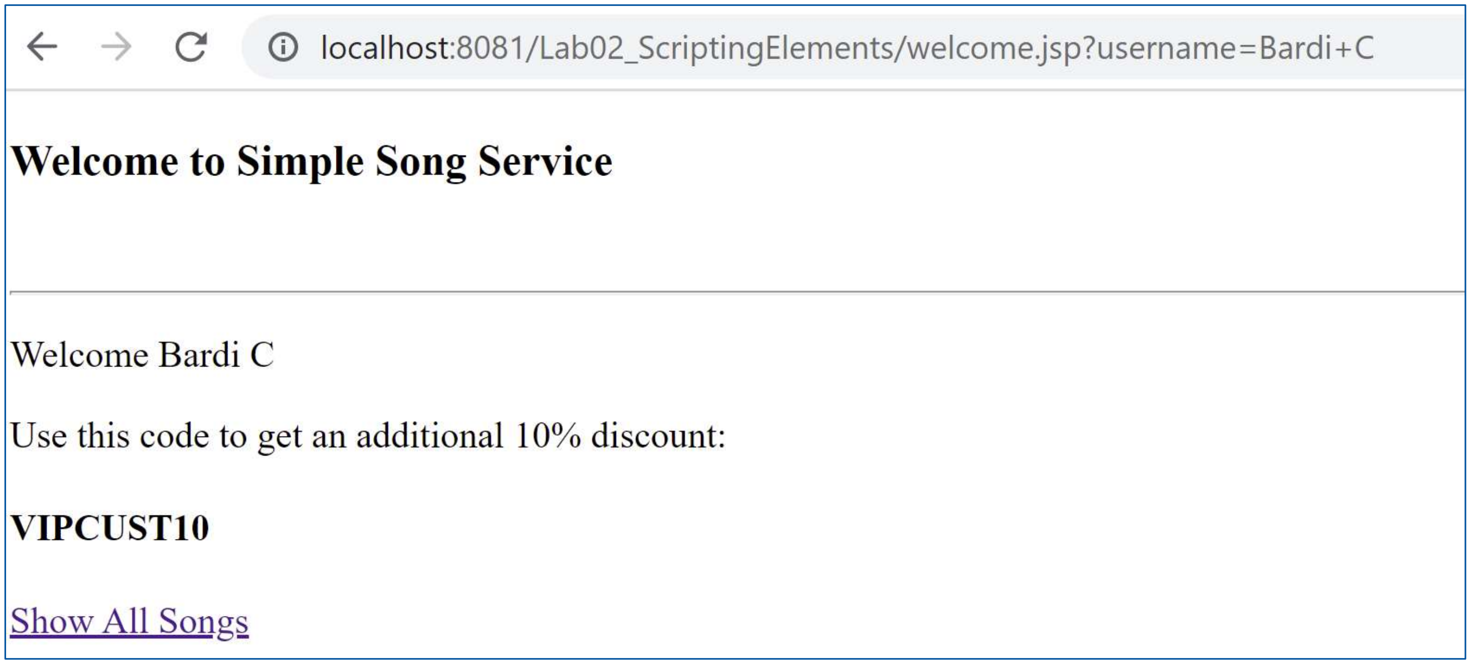
- Work in your welcome.jsp file
- Declare a private variable for storing a discount code and a private method to get the discount code
- Add a message and an expression that calls the get discount code method

```
…
<%!private String discountCode = "VIPCUST10";
private String getDiscountCode() {
        return discountCode;
}%>
<div>
        Use this code to get an additional 10% discount:
        <h4><%=getDiscountCode()%></h4>
</div>
<a href="SongServlet">Show All Songs</a>
```

Lab 2 – Scripting Elements

Run the application again, starting at index.jsp

This time, your welcome.jsp should look like the solution below



localhost:8081/Lab02_ScriptingElements/welcome.jsp?username=Bardi+C

Welcome to Simple Song Service

Welcome Bardi C

Use this code to get an additional 10% discount:

VIPCUST10

Show All Songs