Simple Website Setup

This document shows the steps for creating and running the website shown in the Selenium sections of the course.

Exported JAR files.

1. In order to run the basic website application, you need a web server that is capable of hosting a Java servlet. Apache Tomcat is the web server used in this course. All you need to do is download the Tomcat server to a local folder. Navigate to the Tomcat download link and download the **zip file for Windows (64 bit)**. The link for downloading Apache Tomcat is provided below:

https://tomcat.apache.org/download-90.cgi

Please note that Windows installer files and other zip archives are available as installation options for Windows or other operating systems. But the course uses Windows 10 64 bit and Tomcat 9.

2. Unzip the contents of this zipped archive to C:\apache-tomcat-9.0.20.

3. If you do not have Java JDK installed, install the same from the Oracle website. This setup is identical to the steps shown in the software installation steps at the beginning of the course. Make sure you have the system path and JAVA_HOME set up correctly. The download is provided below.

https://www.oracle.com/technetwork/java/javase/downloads/index.html

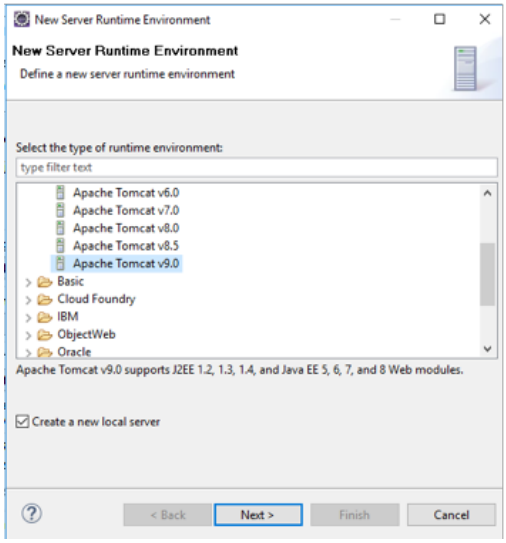
4. Install Eclipse for Enterprise Java Developers.

https://www.eclipse.org/downloads/packages/installer

The Eclipse version for Enterprise Developers has better support for web applications.

5. Once the installation is done, create a new Dynamic Web Project. Call it **BasicWebsite**.

6. For the target runtime, click New Runtime button and select Apache Tomcat v9.0.



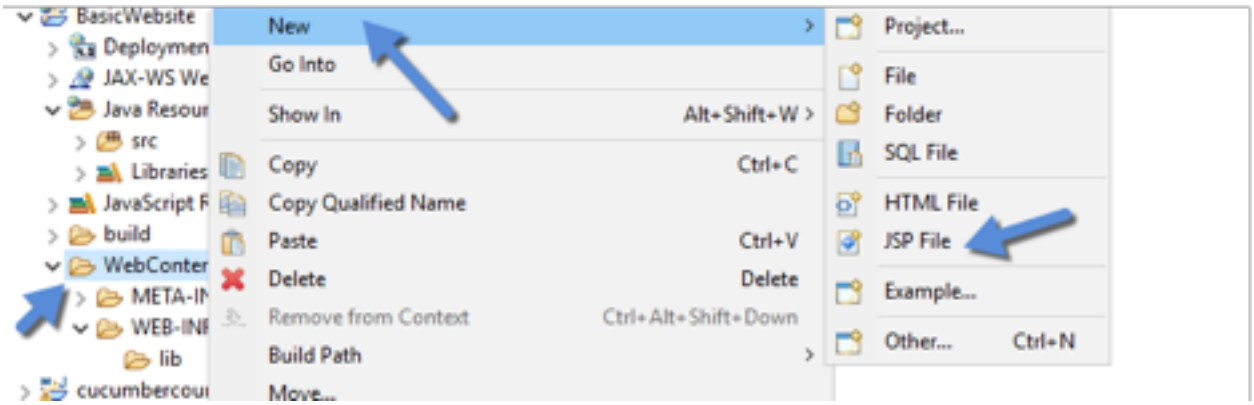
7. For the folder path, supply the path to the folder where Tomcat files were unzipped. Click **Finish**.

8. Click **Next**.

9. Click **Next**.

10. Click **Finish**.

11. Right-click on the **WebContent** folder to add a JSP page called **index.jsp**.



12. Paste the following contents to the **Index.jsp** page:

```
<%@ page language=”java” contentType=”text/html; charset=ISO-8859-1”
    pageEncoding=”ISO-8859-1”%>
<!DOCTYPE html>
<html>
<head>
<meta charset=”ISO-8859-1”>
<title>Eat More Cucumbers Restaurant</title>
</head>
<body>
<h2>Eat More Cucumbers Restaurant Freshville</h2>
 <form action = “Index” method = “POST” target = “”>
 <table>
 <tr>
 <td>Bill Amount: </td>
 <td><input type = “text” id=”billamount” name = “billamount”/></td>
</tr>
 <tr>
 <td>Tax Rate: </td>
 <td><input type = “text” id=”taxrate” name = “taxrate”/>
 </td>
 </tr>
 <tr>
 <td></td>
 <td>
 <br> <input type = “submit” id=”mybutton” value = “Calculate Final Bill” />
  </td>
  </tr>
  </table>
      </form>
</body>
</html>
```

13. Notice that the form post action maps to Index, which will be a servlet we will author.

```
<form action = "Index" method = "POST" target = "">
```

14. Right-click **Java Resources** folder and add a new servlet file called **Index.java**.
15. Call this file Index.
16. Copy and paste the following code to the servlet file.

```
package BasicWebsite;

import java.io.IOException;
import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

import linkedinlearning.cucumbercourse.BillCalculationHelper;

/**
 * Servlet implementation class Index
 */
@WebServlet("/Index")
public class Index extends HttpServlet {
	private static final long serialVersionUID = 1L;

	/**
	 * @see HttpServlet#HttpServlet()
	 */
	public Index() {
		super();
		// TODO Auto-generated constructor stub
	}

	/**
	 * @see HttpServlet#doGet(HttpServletRequest request, HttpServletResponse
	 *      response)
	 */
	protected void doGet(HttpServletRequest request, HttpServletResponse response)
			throws ServletException, IOException {
	}

	/**
	 * @see HttpServlet#doPost(HttpServletRequest request, HttpServletResponse
	 *      response)
	 */
	protected void doPost(HttpServletRequest request, HttpServletResponse response)
			throws ServletException, IOException {
```

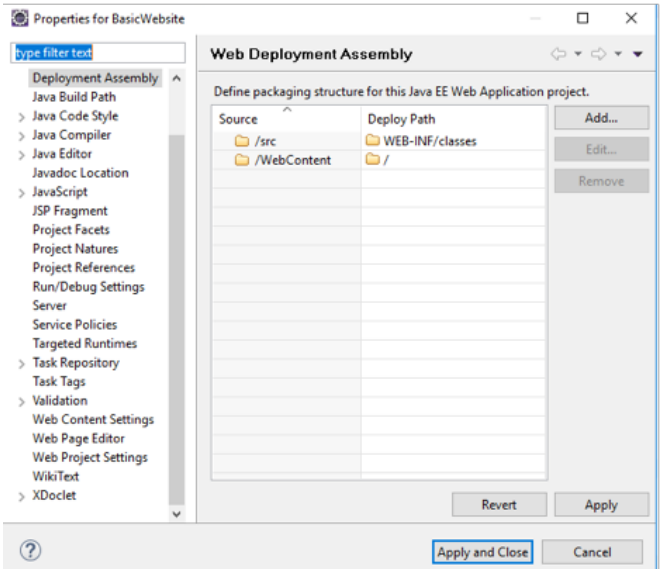
```
try {
                //read the initial bill amount as float
                    double BillAmount = Double.parseDouble(request.
getParameter("billamount"));

                    //read the tax rate
                    Double TaxRate = Double.parseDouble(request.getParameter("taxrate"));

                    //get the final bill amount
                    double FinalBillAmount = BillCalculationHelper.
CalculateBillForCustomer(BillAmount, TaxRate);


                    response.getWriter().write("<html><body><h1>"
                                + "Initial Bill is: $" + BillAmount + "</br>"
                                + "Tax Rate is: " + TaxRate + "%" + "</br>"
                                + "Final Bill Amount is: $" + FinalBillAmount + "</br>"
                                + "</h1></body></html>");
            }
            catch (Exception ex) {
                    response.getWriter().write("Unknown Error: " + ex.getMessage());
            }
      }
}
```

The following import statement line will give you compilation error.

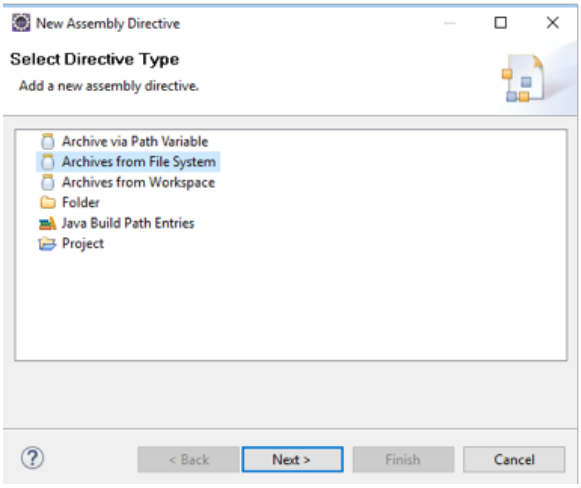
import linkedinlearning.cucumbercourse.BillCalculationHelper;

This is because the Dynamic Web Project has no knowledge of the original Cucumber project. The Cucumber Testing Project was used throughout the course and has a class called BillCalculationHelper, which is needed for our testing. You could duplicate that code as a new class in the website project, but code duplication is not a good idea. So you could export that project and import it into the Basic Website Project. The export option is available in the File menu of Eclipse IDE. This option was used to export the original Cucumber project as cucumbercourse.jar file.

17. Right-click on the **BasicWebsite** project and click **Properties**. **Click Deployment Assembly** option in the left panel and click **Add** button on the right-hand side.

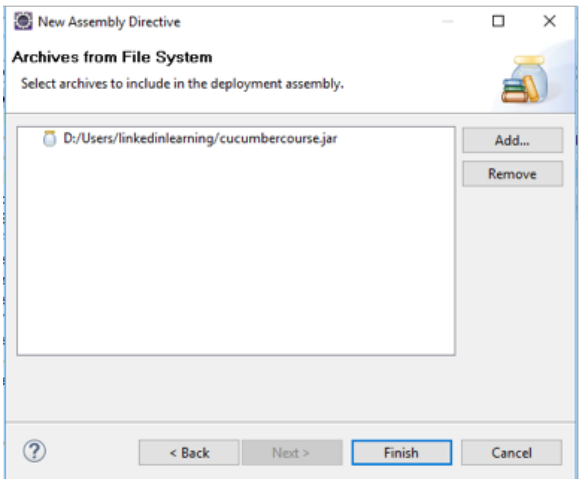


18. Select **Archives** from **File System** menu option. Click **Next**.

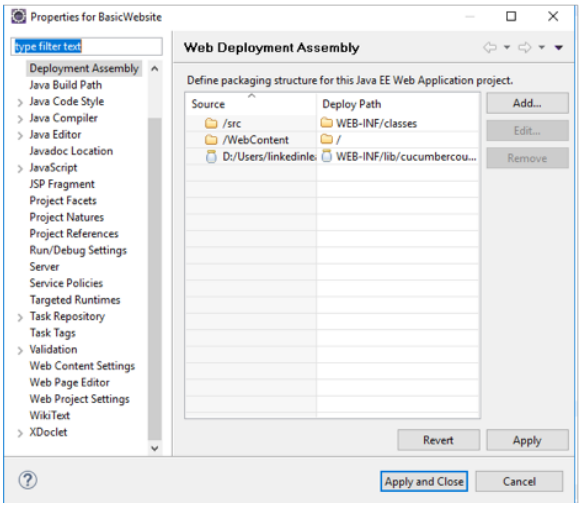


19. Click **Add** button.

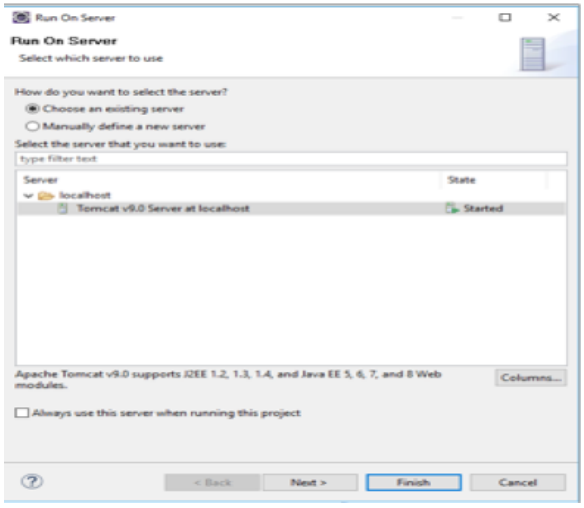
20. Navigate to the cucumbercourse.jar file to add it to the deployment assembly.



21. Click **Apply** and **Close**.



22. Run the project. Right-click on the **BasicWebsite** project and click **Run as -> Run On Server**.



Select **Tomcat** and click Finish.

23. The website will be available.