Course : Diploma in Multimedia & Infocomm Technology (EGDF15)

Module : Java Enterprise Development (EG3752)

|  |  |
| --- | --- |
| Laboratory : | Lab 1 – Introduction to NetBeans IDE and Java EE Programming  *(Partial Self-Directed Learning Modules available on BlackBoard)* |
| Objectives : | This lab will guide you through the setting up of a development environment on your laptop. Once setup, the lab will take you through the basics of building and testing the simplest possible Web application using NetBeans.  At the end of this session, you should be able to:   * Setup a development environment with JDK 8.0 and, Netbeans IDE 8.0 with GlassFish 4.0 bundle * Develop a simple web application using Java EE 7 – using HTML, JSP and Servlets. * Test your web applications using three approaches with the NetBeans IDE. * Apply common troubleshooting techniques for NetBeans. * Familiarise yourself with the structure of a WAR file (Optional). |
| Software Used : | Java Standard Development Toolkit (JDK™) 8.0  NetBeans IDE 8.0 with GlassFish Server 4.0 bundle |

**Read This First: Compatibility between Java EE, Java SE and NetBeans**

When setting up the development environment for your computer for the first time, please pay attention to the version number of the software you are using.

In this module, we will be coding using the Java Platform, Enterprise Edition 7 (Java EE 7) standard. For the Java EE 7 web applications to run correctly, the reference implementation server (GlassFish Server 4.0) must be used.

*\*Sidenote: There are known bugs with GlassFish 4.1 hence we will not be using it even though it is a newer version.*

Finally, the JDK that accompanies Java EE 7 will be Jave SE Development Kit 8. NetBeans 8 will refuse to install without JDK 8 so it will be hard to get this wrong.

**Preparatory Task: Setting up your Development Environment**

**Installing Java SE Development Kit 8**

P.1 Download and install the latest Java SE Development Kit 8 from the following website:

<http://www.oracle.com/technetwork/java/javase/downloads/jdk8-downloads-2133151.html>

IMPORTANT: Take note of the installation path.

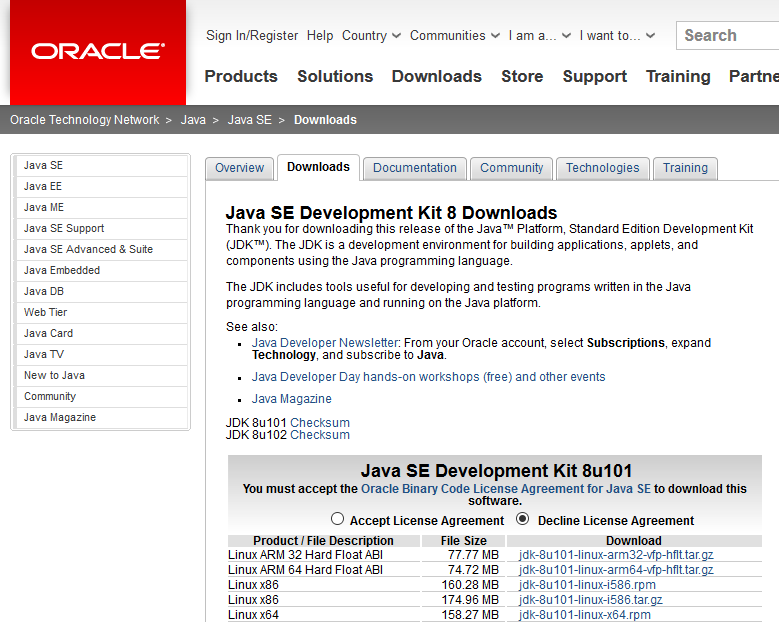


Figure P.1: Website for Java SE Development Kit Download

**Setting JAVA\_HOME environment variable**

P.2.1 Search for **System (Control Panel)** using the Windows 10 Search function.

P.2.2 Select **Advanced System Settings** from the left menu

P.2.3 Under the **Advanced** tab, click **Environment Variables** button.

P.2.4 Under **System Variables**, click New

P.2.5 Enter **JAVA\_HOME** at the variable name and the **installation path** for the Java Development Kit which you have noted down in Task P.1 as the variable value

P.2.6 Click **OK**, follow by **OK**, follow by **Apply**

P.2.7 Ensure that JAVA\_HOME has been set correctly to your JDK installed directory. You can do this by typing **set JAVA\_HOME** in your Command Prompt shell. The shell can be brought up by searching for **cmd** in the instant search of your Windows 10 OS.

If not, try restarting windows.

**Installing NetBeans 8.0 in your NoteBook**

P.3.1 Download NetBeans IDE 7.4 Installer at <http://netbeans.org/downloads/8.0/> . Choose the JavaEE Bundled with GlassFish 4.0 and Apache Tomcat

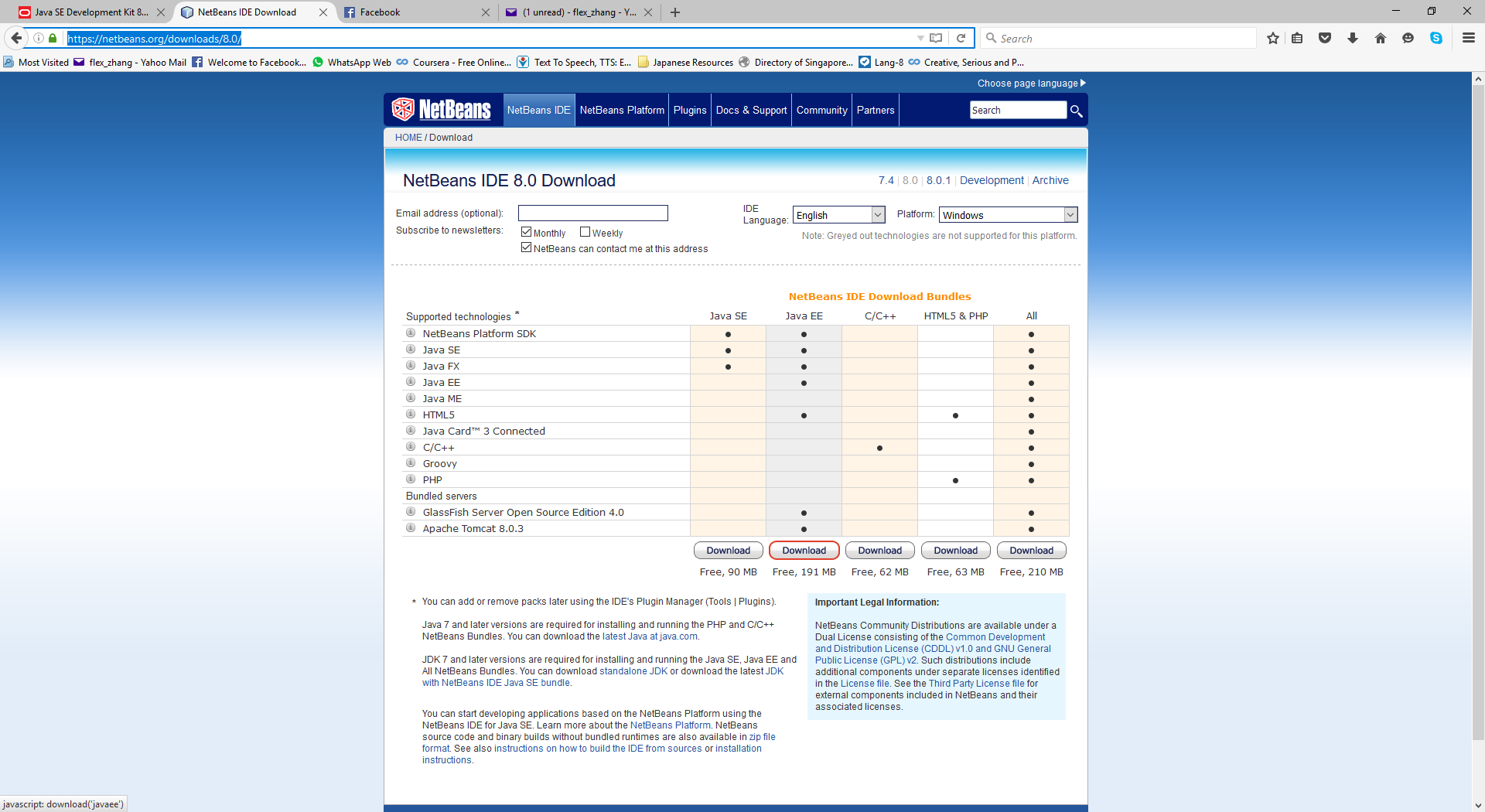


Figure P.3.1: NetBeans IDE 8.0 Download

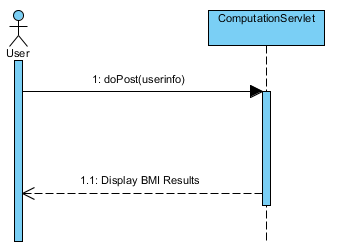
P.3.2 Install NetBeans by double clicking on the exe file that you have downloaded. Include **GlassFish Server Open Source Edition** in the installation.

**Overview of the Simple Registration Web Application**

The simple registration web application will allow users to input some profile information and outputs their BMI. The full requirement specifications are listed in the table below:

|  |  |
| --- | --- |
| **Requirement Reference** | **Requirement Specifications** |
| 1.1 Registration | The system shall provide the functionality for users to submit the following information: Name, Admin Number, Email Address, Gender (Male or Female), Choice of 3rd Year Specialisation (IS, IM, CNet or ITSM), Height (in cm) and Weight (in kg). |
| 1.2 BMI Computation | The system will compute the BMI of the user based on the height and weight that he has keyed in. |
| 1.3 Display | The system will display all information that is keyed in by the user for his verification.  The system will also display a customised message based on the computed BMI. The message to be displayed is as follows:   * BMI < 20 : You are considered underweight. * 20 <= BMI <= 25 : You are just right. * BMI > 25 : You are considered overweight. |

The flow and components of the web application is represented by the sequence diagram shown below:



Creating a Project in NetBeans IDE

***(Self-Directed Learning version available on BlackBoard)***

1.1 Start **NetBeans IDE.**

1.2 Create a new project by selecting the **** icon.

1.3 Select **JavaWeb** under *Categories* follow by **Web Application** under *Projects.*

1.4 Click **Next.**

1.5Type **SimpleRegistration** as the *Project Name*.

1.6Click **Next.**

1.7 Select **GlassFish Server 4.0** for the *Server.* Select **Java EE 7 Web** for the *Java EE Version*.

1.8 Click **Next.**

1.9 Click **Finish**.

1.10 Notice that the **Lab1** project node appears on the left-hand-side and the **index.html**[[1]](#footnote-1) file generated by IDE displayed on the right-hand-side.

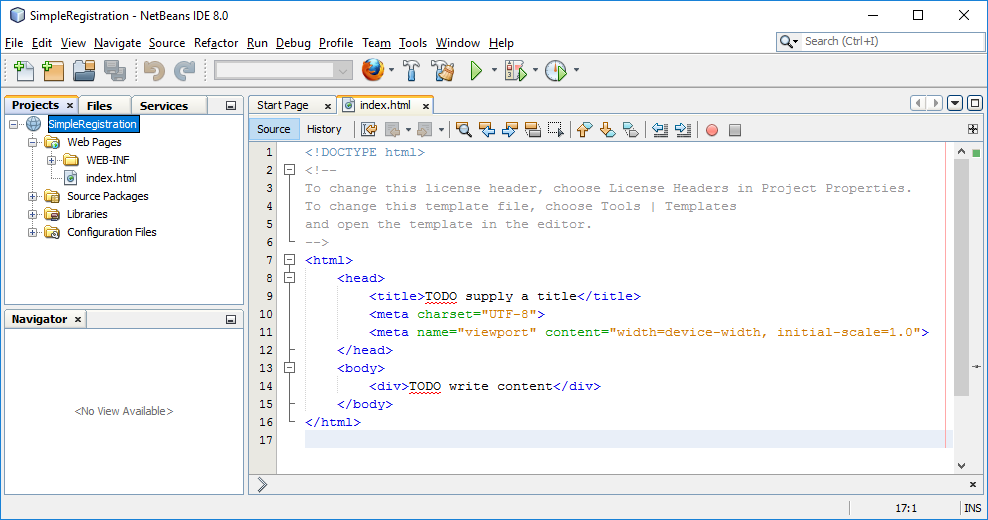


Figure 1.10: Screenshot of the IDE

Testing a Java EE Web Application using NetBeans

***(Self-Directed Learning version available on BlackBoard)***

There are three main methods which you can test your Java EE application. We will learn about the methods while working on our registration page.

|  |
| --- |
| **Method 1: Test by Running Project**  2.1 Test the generated web application by right clicking on **Lab1** project node and select **Run**. The IDE will compile any Java source files, and deploy them onto GlassFish Server. |

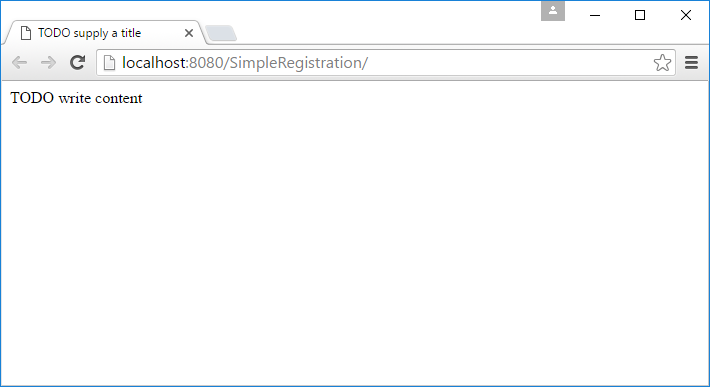


Figure 1.11: Sample output from the SimpleRegistration web application

2.2 Let’s personalise the page by changing the title of the page:

Double-click on **index.html** to open the file if it is not already opened (under*Lab1*->*Web Pages*).

Change the code fragment as highlighted to:

<title>Simple Registration</title>

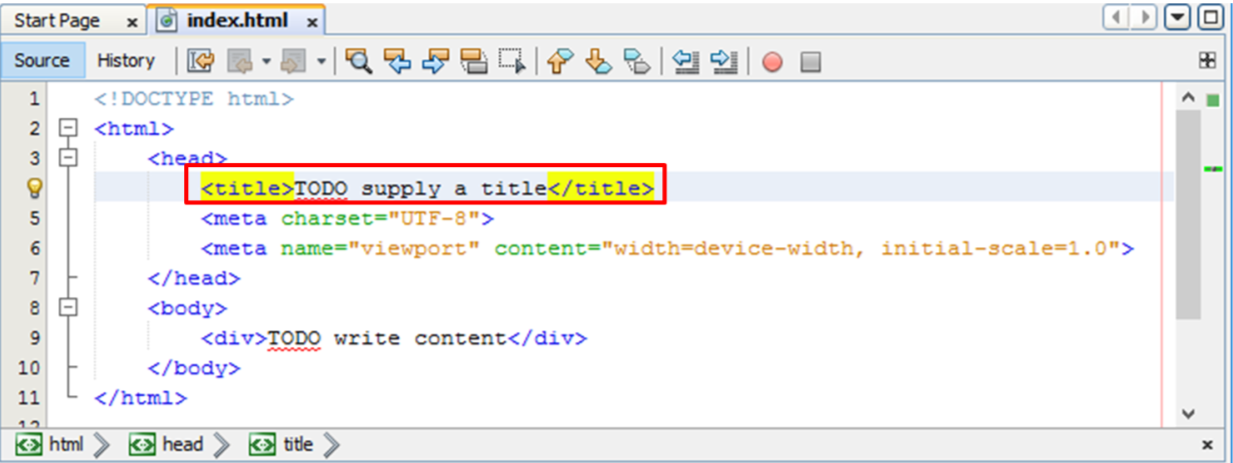


Figure 1.12: Highlighted segment of the code which requires changes

|  |
| --- |
| **Method 2: Test by Running Specific Files**  2.3 After editing, test your page by right clicking on **index.html** and select **Run File**. |

Is the output of your webpage as expected?

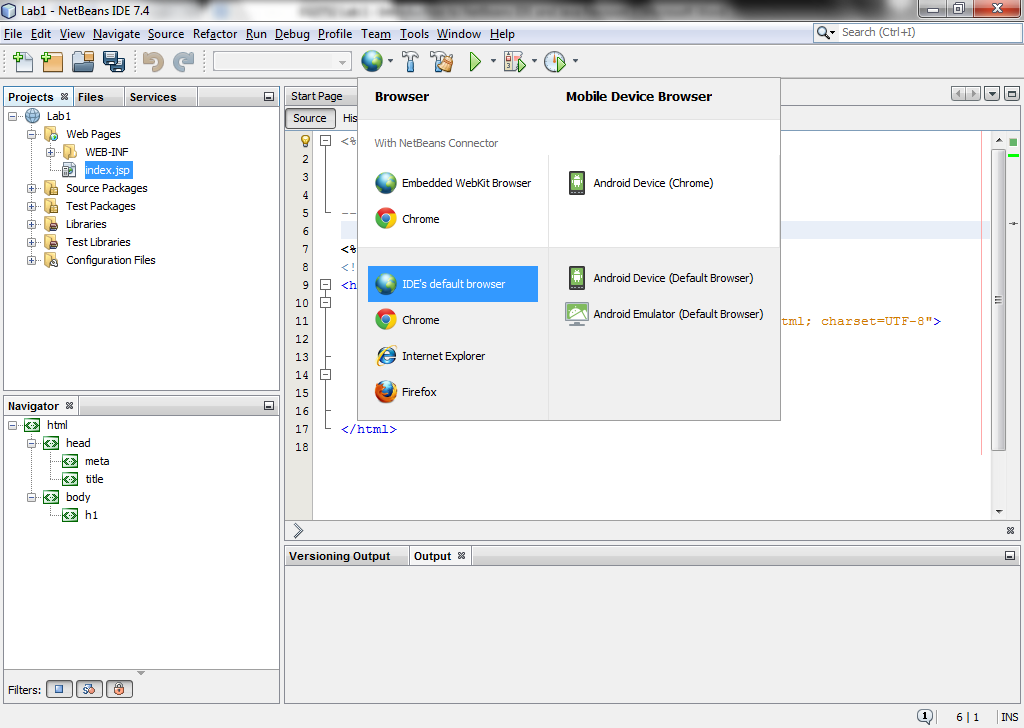
2.4 Once you are done with the testing, do **NOT** close your browser. If you have, please repeat Step 2.3.

2.5 Put together a registration page that has the following fields for the user to key in based on **Requirement Reference 1.1** by copying and replacing the body of the html file with **Code Snippet Lab 1 P 2.5** which can be found on BlackBoard.

**IMPORTANT NOTE:** Yes! You are allowed to copy some of the codes into your lab on one condition:

Read through and understand the codes that was written. Ask the instructor to go through the codes with you where you are unsure. Now go ahead and look for **Code Snippet Lab 1 P 2.5** in BlackBoard and copy.

|  |
| --- |
| **Method 3: Test by Hot Deployment**  2.6 Instead of running the project or the file, save your changes (**Ctrl-S**) and refresh the browser.  Notice that your changes have been made directly to the server after you save the file. The NetBeans IDE allow you to do **“hot” deployment** for ease of testing during the development stage. The hot deployment should work in most cases when working with servlets and JSPs. |

2.7 You can also test your application against different web browsers that are installed on your computer. The default start-up browser can be changed by clicking on the Browser dropdown button as shown in the adjacent figure.

Creating Your First Servlet

***(Self-Directed Learning version available on BlackBoard)***

We will be adding a servlet to compute

3.1 To add a servlet, right click on **SimpleRegistration** project node and select **New -> Java Class…** .

3.2 Under *Name and Location*:

Type **ComputationServlet** for *Class Name* field.

Type **sg.edu.nyp** for *Package* field.

*Question: What is the significance of the package name?*

3.3 Click **Finish**.

3.4 Currently, the generated file is still a typical java class. Type in the following codes to change it to a servlet:

|  |
| --- |
| package sg.edu.nyp;  @WebServlet("/compute")  public class ComputationServlet extends HttpServlet  {    } |

3.5 Take note of the symbol **\compute** found in the Web Servlet annotation. This is known as the url pattern.

3.6 Press **Ctrl-Shift-I** to import all the relevant libraries required for the project automatically.

|  |
| --- |
| **NetBeans Shortcuts**  Some of the most commonly used shortcuts in NetBeans are as follows:  **Ctrl-Shift-I** : Imports unreferenced libraries. If there is more than one possible libraries that could be imported, NetBeans will display a drop down list for you to choose from.  **Alt-Insert** : Inserts codes automatically. One of the most useful applications is to automatically generate “getters” and “setters” for the variables in your class.  **Alt-Shift-F** : Formats codes automatically provided the codes does not contain syntax errors. |

3.7 Go back to **index.html**. Initialise the form **action** attribute to **compute** as such:

<form action=**"compute"**>

Notice that this is the same symbol that was used in the Web Servlet annotation in the **ComputationServlet**.

3.8 Override the **doGet** function in your ComputationServlet class as such. The codes are given as below:

|  |
| --- |
| @WebServlet("/compute")  public class ComputationServlet extends HttpServlet  {  @Override  protected void doGet(HttpServletRequest request,  HttpServletResponse response)  throws ServletException, IOException  {    }  } |

You may use **Ctrl-Shift-I** to import all the relevant libraries required for the project automatically.

Take note that you are using a **GET** method.

3.9 Go back to **index.html**. Initialise the form **method** attribute to **get** as such:

<form action=**"compute"** method=**"get"**>

Notice that this is the same symbol that was used as the name of the function that we have just overridden. The other commonly used symbol is the **POST** method. In this case, the **doPost** method can be overridden.

Retrieving Parameters from the Form in index.html from and Storing addition values in the HttpServletRequest object

***(Self-Directed Learning version available on BlackBoard)***

Input values from a form can be retrieved from **HttpRequest**. The equivalent of this once the request is passed over to the servlet is a **HttpServletRequest**. All values stored as **parameters** as **String** objects in the **HttpServletRequest** object.

Additional values can be stored in the request as an **attribute**.

* 1. To retrieve the height from the form, type out the following codes within the **doGet** method:

String height = request.getParameter("height");

Note that the value argument for the **getParameter** corresponds to the name of the element in the index page.

4.2 Do the same for the **weight** field.

4.3 Complete the rest of the codes to compute the BMI of the person. The pseudocode to compute the BMI is given as follow:

1. Convert the height to integer and weight to double.

2. Compute the BMI. The formula is: **BMI = (10000 / (height \* height)) \* weight**

4.4 Store the bmi in the request as an attribute named “bmi”:

request.setAttribute("bmi", bmi);

Displaying the results using JSP

***(Self-Directed Learning version available on BlackBoard)***

5.1 Create a new JSP named **results.jsp** under the **Web Pages** folder under project tab found on NetBeans IDE.

|  |
| --- |
| **WARNING! Be careful NOT to place web pages in the WEB-INF folder.** |

5.2 Edit the **ComputationServlet** so that it will forward the request to **results.jsp** by filling in the following codes after setting the BMI value into the request as an attribute:

RequestDispatcher rd =

request.getRequestDispatcher("/results.jsp");

rd.forward(request, response);

\*Note: A more in-depth explanation regarding this set of code will be given in Lab 3.

* 1. Test your application to make sure that the forwarding is successful. It should show a “Hello World” message.

|  |
| --- |
| **A Successful Programmer’s Mantra**  One of the critical success factor of the programmer lies in his/hers ability to debug the programme. Often, whether a program compiles and runs correctly or not lays on his/hers ability to spot on the exact line(s) that is/are causing the error.  Three tips:   1. **Avoid syntax errors as far as possible.**   If any, always solve the first syntax error that appears in the file and work your way down. This is because sometimes, solving the first error may solve all the errors below.   1. **Test your programme often.**   If anything goes wrong, the error is often found between when you have last tested your programme correctly and the current test. This reduces the number of lines of codes you need to look through to hunt for ~~Pokemons~~ bugs in your programme.   1. **Format your codes!**   This makes your codes more readable. Enough said!  In short, stop and debug once you run into an error. |

5.4 Edit **results.jsp** such that it will retrieve the appropriate values from the request and display it on screen as such:

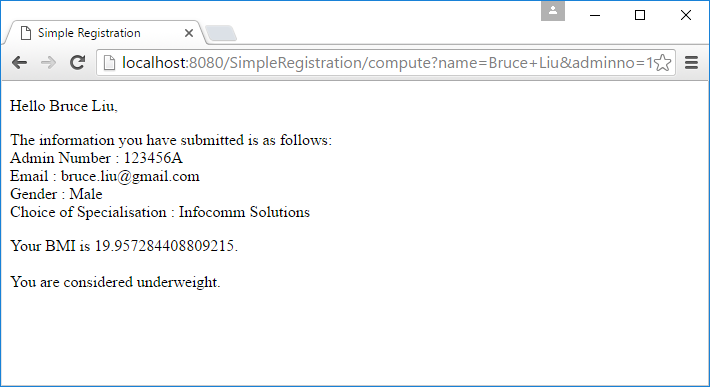


Figure 5.4: Expected output for **results.jsp**

As an example, a code snippet on how you can print out the choice of specialisation is shown below:

|  |
| --- |
| <body>  <p>  **<%=**request.getParameter("specialisation")**%>,**  </p>  </body> |

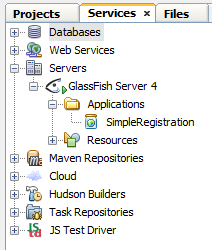
For the last statement that reads “You are considered underweight”, please printout the following statements according to the different values of BMI computed:

* BMI < 20 : You are considered underweight.
* 20 <= BMI <= 25 : You are just right.
* BMI > 25 : You are considered overweight.

**Question to test your understanding:**

In **index.html**, you have asked the user to select the options “IS, IM, CNet and ITSM” from the dropdown box. However, when you print it out in the JSP, it prints out in full. How is this achieved?

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Troubleshooting Web Application using NetBeans

6.1 On the left of the NetBeans IDE, switch to the **Services** tab window.

6.2 Expand the Servers > GlassFish Server > Applications tree (see **Figure 6.2**).

Notice that your web application **SimpleRegistration** is found under the **Applications** node. Whenever you need to test your web applications or enterprise applications, NetBeans will need to deploy your programme onto GlassFish Server. When you run the project, NetBeans will automatically start up GlassFish Server for you if it is not started, and deploy the application for you.

Figure 6.2 Screenshot of the Services Window

6.3 There are times where you might have already made changes to your applications but the changes you made might not have appeared to have taken place. Firstly, check for whether there could be any logical errors in your code. If you are quite sure that it isn’t a problem with your codes, you may try a few of the methods listed in order of time taken and severity of the issues in the table below.

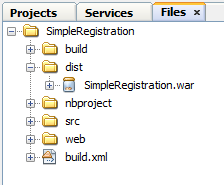
Try the first three techniques listed in the Table to make sure you know how to trigger these techniques where you are faced with problems. Take notes in the space provided if you like to.

|  |  |
| --- | --- |
| **Technique** | **Your Notes** |
| Clear the cache of your Internet Browser (Your application might be stuck in your browser’s cache hence clearing it may help) |  |
| Undeploy and redeploy your web application |  |
| Restart GlassFish Server  (You may find it more useful to undeploy your web application first, restart the GlassFish Server then redeploy the application again) |  |
| Restart NetBeans IDE |  |
| Restart Windows  (If all else fails) |  |
| Start a Whole New Project  (Use this *cheapo* technique if you are unable to recover from any errors at all) |  |

Observing the WAR File (Optional)

7.1 Build a WAR file by right clicking on **SimpleRegistration** project node and selecting **Clean and** **Build**.

7.2 Locate the Lab1.war that you have just built:

* Select the **Files** tab.
* Open the **dist** folder.
* Right click on **SimpleRegistration.war** and select **Properties**.
* Click on the **…** box under the **All Files** field

A dialog box will pop up showing the path to the WAR file.

7.3 Go to the folder where your WAR file is located using your Windows Explorer. Make a copy of **SimpleRegistration.war** .

7.4 Rename the copy you made as **SimpleRegistration.zip**

7.5 Unzip **SimpleRegistration.zip** by right clicking on it and select **Extract** .

7.6 Investigate the directory structure and the files in the extracted folder.

1. In older versions of NetBeans, **index.jsp** gets generated instead. Both the index.html and index.jsp serves the same purpose. [↑](#footnote-ref-1)