**1. SETUP A JDBC ENVIRONMENT**

Setting up a JDBC (Java Database Connectivity) environment involves several steps to establish a connection between your Java application and a relational database.

**Steps that includes:**

1.Creating a dynamic web project

2.Adding the jar files for MySQL connection for Java

3.Creating an HTML page index.html

4.Creating a DBConnection class to initiate a JDBC connection in code

5.Creating a config.properties file to store JDBC credentials

6.Creating a DemoJDBC servlet

7.Configuring web.xml

8.Checking for servlet-api.jar

9.Building the project

10.Publishing and starting the project

11.Running the project

12.Pushing the code to your GitHub repositories

**Step 1:** **Creating a dynamic web project**

Creating a dynamic web project typically involves using a Java web framework like Java Servlets and JavaServer Pages (JSP) and an Integrated Development Environment (IDE) like Eclipse or IntelliJ IDEA. Here's a step-by-step guide on how to create a dynamic web project using Eclipse IDE:

1: Install Eclipse

If you haven't already, download and install Eclipse IDE. Make sure you have Java Development Kit (JDK) installed on your system as well.

2: Install the Java EE (Enterprise Edition) Tools

In Eclipse, you'll need to install the Java EE tools if you haven't already. You can do this by going to the Eclipse Marketplace:

Click on "Help" in the top menu.

Select "Eclipse Marketplace."

In the search bar, type "Java EE" and install the "Eclipse Java EE Developer Tools."

3: Create a Dynamic Web Project

Open Eclipse.

Go to "File" > "New" > "Dynamic Web Project."

Provide a name for your project in the "Project Name" field.

Choose the target runtime. If you haven't configured a runtime yet, click on "New Runtime" and select a server runtime (e.g., Apache Tomcat) to use.

Set the "Dynamic web module version" to your preferred version (e.g., 4.0).

Configure the source folder (usually named "src") and the WebContent folder (usually named "WebContent") for your project.

Click "Finish."

4:Create Servlets and JSP Pages

You can now start creating Java Servlets and JSP pages within your dynamic web project. Right-click on the "src" folder and choose "New" > "Servlet" to create a new servlet. Similarly, you can create JSP pages in the "WebContent" folder.

5: Configure Deployment Descriptor (web.xml)

If your project requires configuration settings like servlet mappings or filters, you can do so by editing the web.xml file located in the "WebContent/WEB-INF" folder.

6: Configure Server

Before running your dynamic web project, you need to configure the server in Eclipse:

In the "Servers" tab, right-click and select "New" > "Server."

Choose the server type you installed (e.g., Apache Tomcat).

Configure the server runtime environment.

Click "Finish."

7: Deploy and Run

To deploy and run your dynamic web project:

Right-click on your project and select "Run As" > "Run on Server."

Choose the server you configured earlier and click "Finish."

Eclipse will deploy your project to the server and open a web browser with your application.

You can now develop, test, and debug your dynamic web application within Eclipse. Remember to update your servlets, JSP pages, and other web-related components as needed.

**Step 2:** **Adding the jar files for MySQL connection for Java**

To add the MySQL JDBC driver JAR file to your Java project for establishing a database connection, follow these steps:

1.Download the MySQL JDBC Driver

Visit the official MySQL website or the MySQL Maven Repository (https://mvnrepository.com/artifact/mysql/mysql-connector-java) to download the MySQL JDBC driver (Connector/J).

Select the version of the driver you want to use and download the JAR file. The file typically has a name like mysql-connector-java-X.Y.Z.jar, where X.Y.Z represents the version number

2.Add the JAR File to Your Java Project

Now, you need to add the downloaded JAR file to your Java project. You can do this in various Integrated Development Environments (IDEs) like Eclipse, IntelliJ IDEA, or by manually adding the JAR to your classpath.

Using Eclipse (as an example):

Open your Java project in Eclipse.

Create a "lib" folder in your project if it doesn't already exist. You can right-click on your project, select "New" > "Folder," and name it "lib."

Copy the downloaded MySQL JDBC driver JAR file into the "lib" folder.

Right-click on your project and select "Build Path" > "Configure Build Path."

In the "Java Build Path" dialog, go to the "Libraries" tab.

Click on the "Add JARs..." or "Add External JARs..." button, depending on where you placed the JAR file.

Navigate to your "lib" folder and select the MySQL JDBC driver JAR file. Click "Open."

Click "Apply and Close" to save the changes.

**Step 3:** **Creating an HTML page index.html**

Creating an index.html page is straightforward and is often the starting point for building a website. Here's a basic example of how to create an index.html page:

Open a Text Editor:

Open a plain text editor like Notepad (on Windows), TextEdit (on macOS), or any code editor of your choice, such as Visual Studio Code, Sublime Text, or Atom.

Create a New File:

Create a new file and save it with the name index.html. Make sure to use the .html file extension.

Write HTML Code:

Add HTML code to your index.html file. Here's a minimal example:

This code includes a basic HTML structure with a title, a heading (<h1>), and a paragraph (<p>).

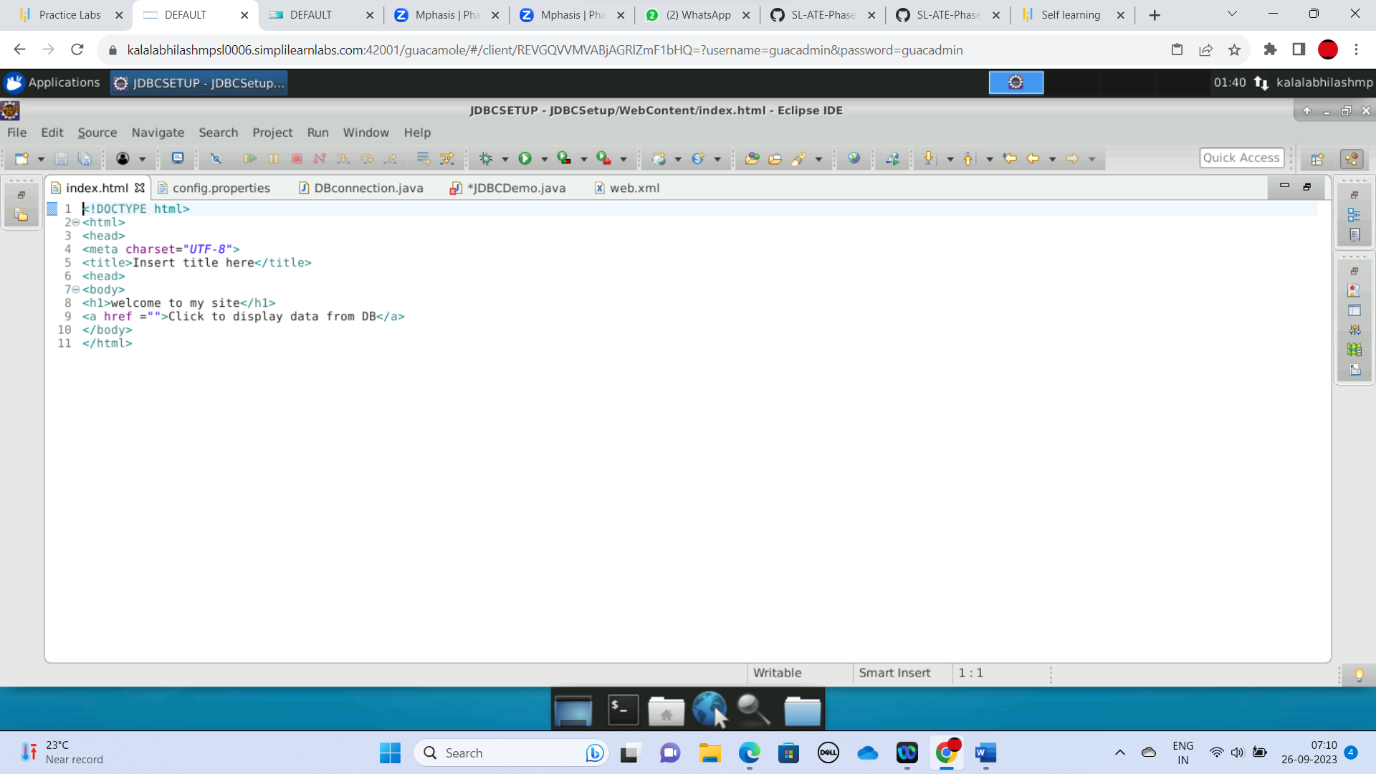
Save the File:

Save your index.html file after adding the HTML code.

View in a Web Browser:

To view your HTML page, simply double-click on the index.html file. It will open in your default web browser, and you'll see your webpage.

That's it! You've created a basic index.html page. You can further enhance and customize it by adding more HTML elements, styles with CSS, and interactivity with JavaScript as needed for your project.



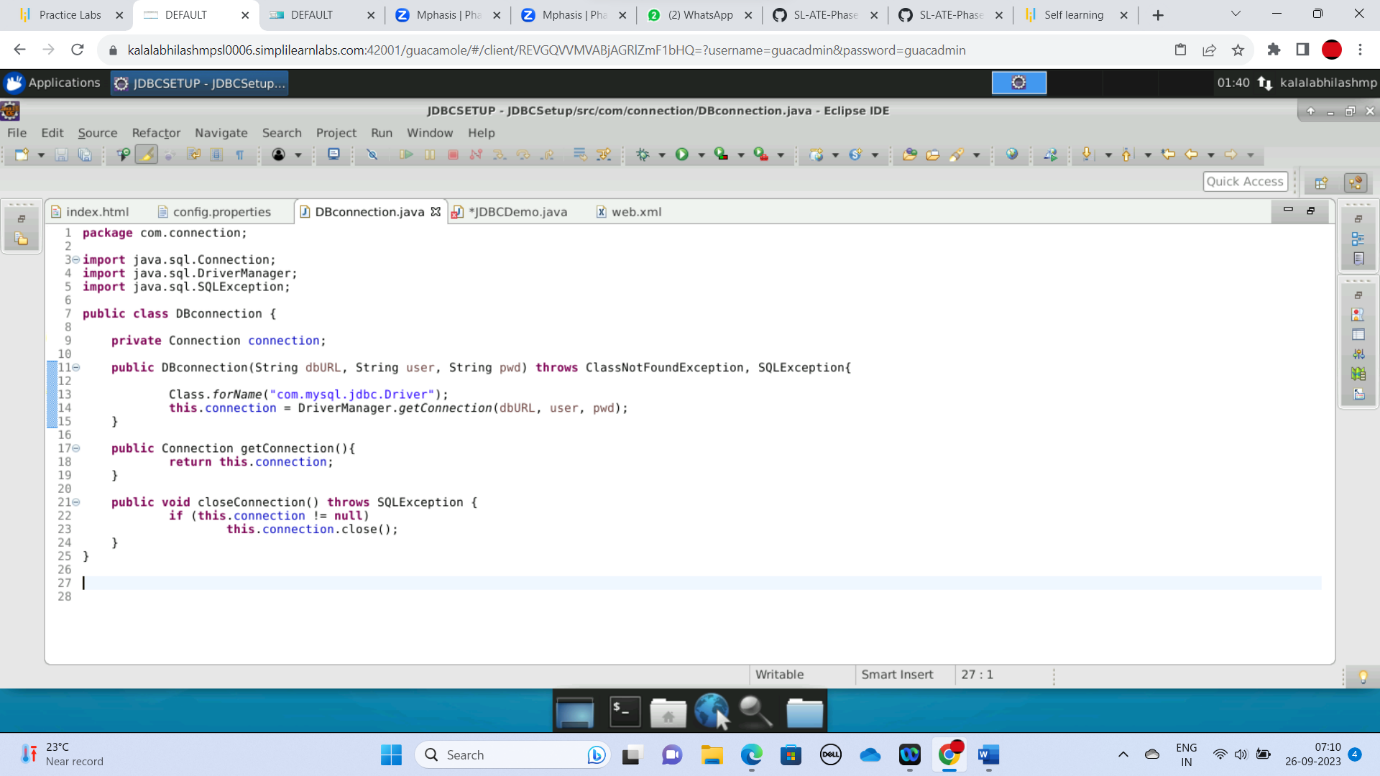
**Step 4: Creating a DBConnection class to initiate a JDBC connection in code**

In the Project Explorer, expand **JDBCSetup->Java Resources**

Right click on **src** and choose **New->Class**

In **Package**, enter **com.ecommerce** and in **Name** enter **DBConnection** and click on**Finish**

Enter the following code:



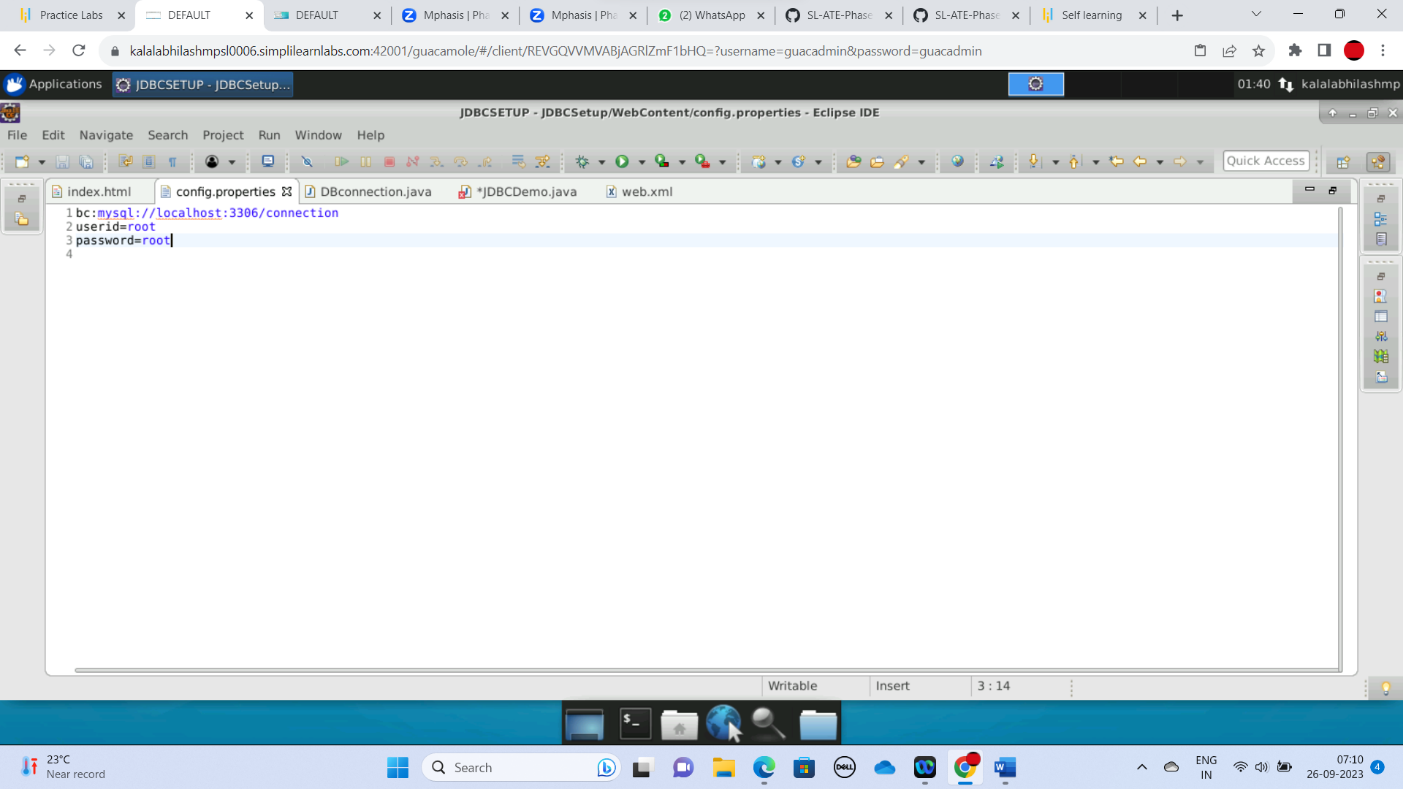
**Step 5:** Creating a config.properties file to store JDBC credentials

In the Project Explorer, expand the project **JDBCSetup**

Expand **WebContent**. Right click on **WebContent**. Choose **New->File**

Enter the filename as config.properties and click on **Finish**

Enter the following data:



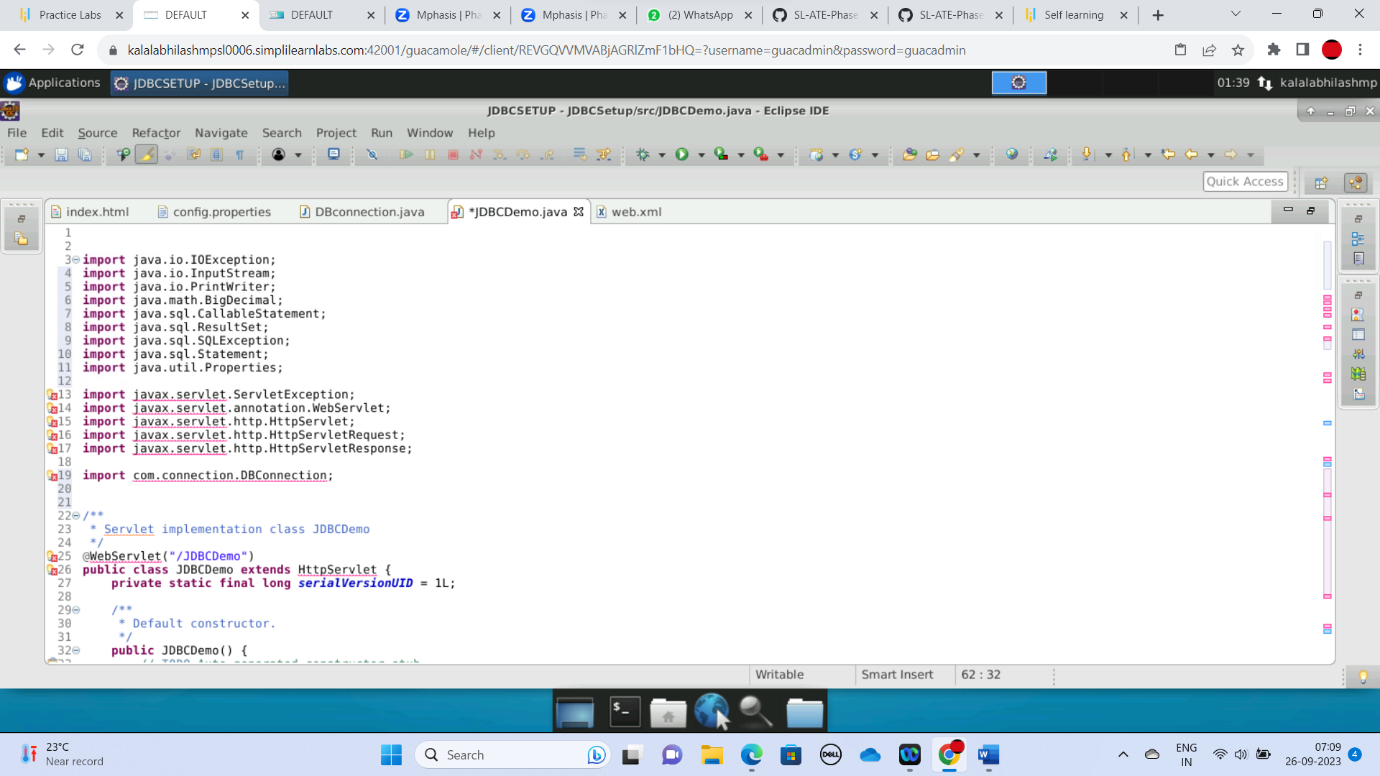
**Step 6:** Creating a DemoJDBC servlet

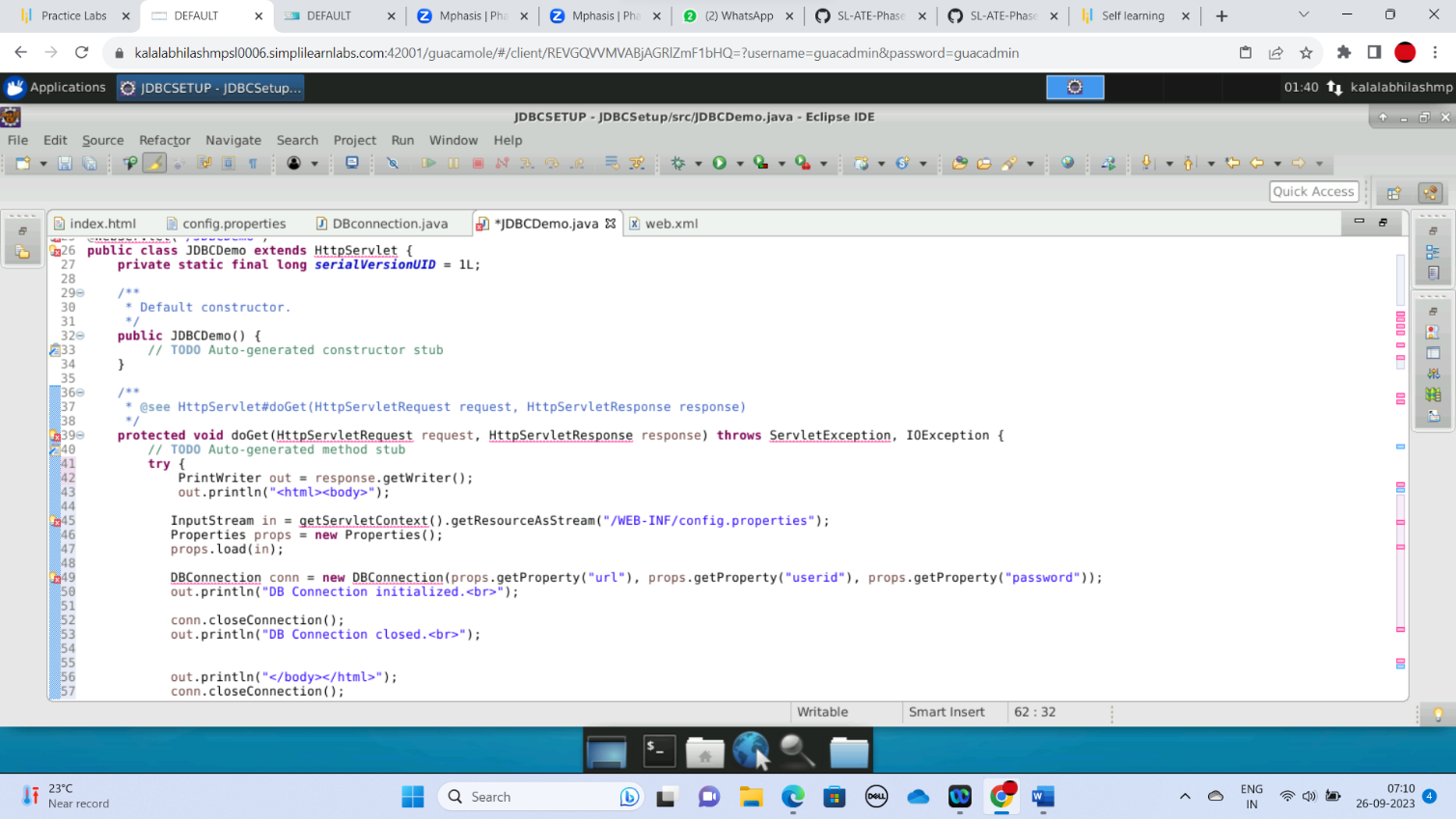
In the Project Explorer, expand **JDBCSetup->Java Resources**

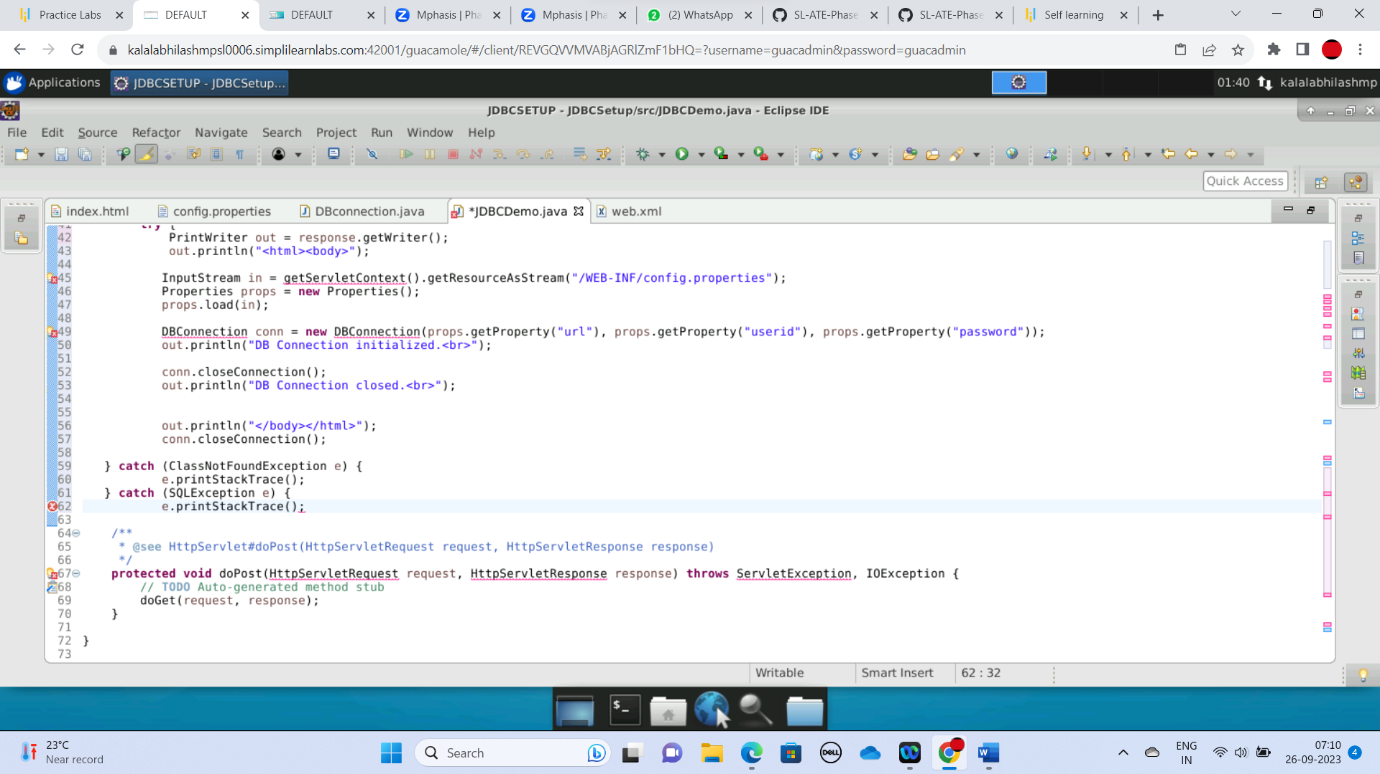
Right click on **src** and choose **New->Servlet**

In **Class Name,** enter **DemoJDBC** and click on **Finish**

Enter the following code:





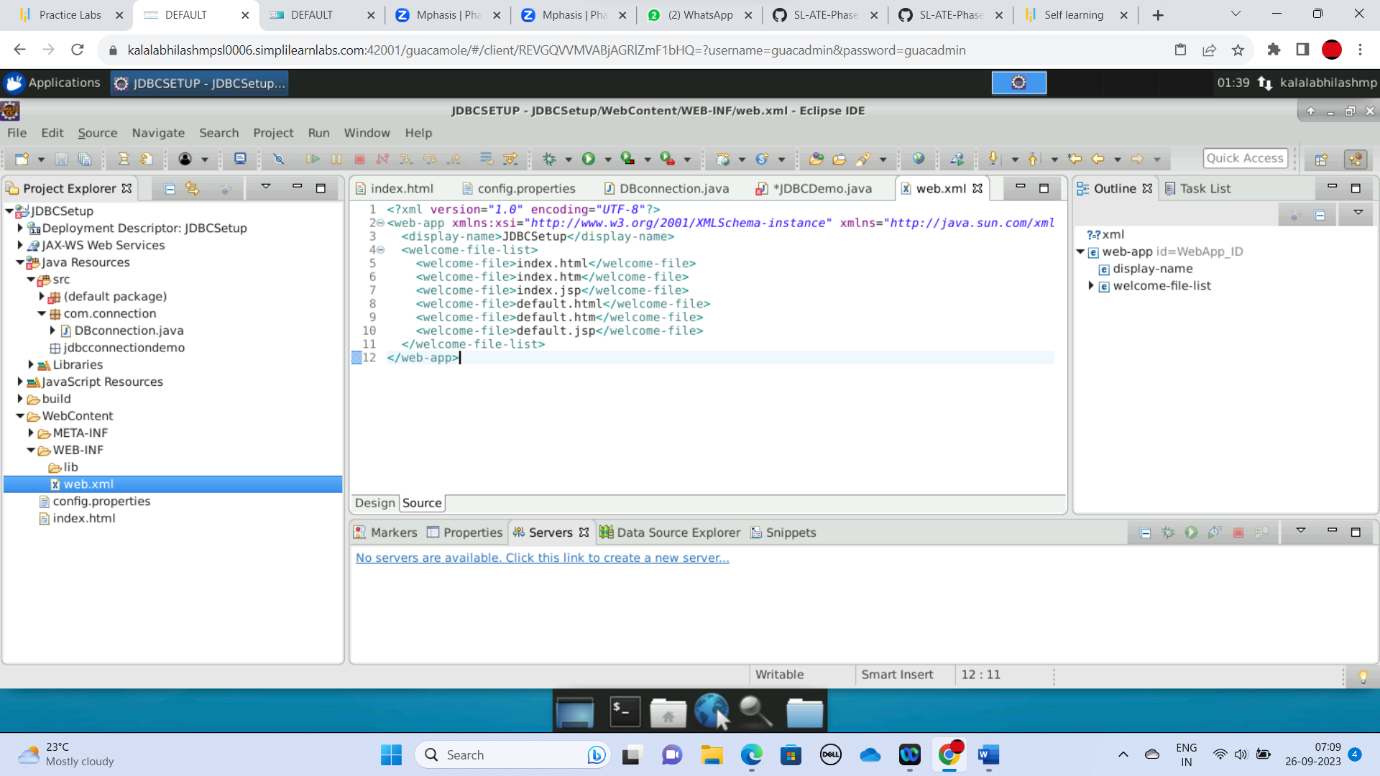


**Step 7:** Configuring web.xml

In the Project Explorer, expand **JDBCSetup->WebContent->WEB-INF**

Double click on **web.xml** to open it in the editor

Enter the following script:



**Step 8:** Checking for servlet-api.jar

Before building the project, we need to confirm that **servlet-api.jar** has been added to the project.

In the Project Explorer, right click on **JDBCSetup** and choose **Properties.**

Select **Java Build Path** from the options on the left.

Click on **Libraries** tab on the right.

Under **ClassPath,** expand the node that says **Apache Tomcat.**

If there is an existing entry for **servlet-api.jar,** then click on **Cancel** and exit the window.

If it is not there, then click on **Classpath** entry and click on **Add External JARs** button on the right.

From the **file** list, select **servlet-api.jar** file and click on **Ok.**

Click on **Apply and Close.**

**Step 9:** Building the project

From the **Project** menu at the top, click on **Build**

If any compile errors are shown, fix them as required

**Step 10:** Publishing and starting the project

If you do not see the **Servers** tab near the bottom of the IDE, go to **Window** menu and click on **Show View->Servers**

Right click the **Server** entry and choose **Add and Remove**

Click the **Add** button to move **JDBCSetup** from the **Available** list to the **Configured** list

Click on **Finish**

Right click the **Server** entry and click on **Publish**

Right click the **Server** entry and click on **Start**

This will start the server

**Step 11:** Running the project

To run the project, open a web browser and type: [**http://localhost:8080/**](http://localhost:8080/ServletConcept)**JDBCSetup**

**Step 12: :** Pushing the code to your GitHub repositories

Open your command prompt and navigate to the folder where you have created your files

cd <folder path>

Initialize your repository using the following command:

git init

Add all the files to your git repository using the following command:

git add .

Commit the changes using the following command:

git commit . -m “Changes have been committed.”

Push the files to the folder you initially created using the following command:

git push -u origin master