Project 2 Android development and web service programming

This project is worth 10% of your final grade (3% for Part 1, 7% for Part 2).

Due: Fri., 13/09/2019.

Part 1: create an Android mobile application

This lab exercise helps you learn Android application development and build your first Android application. We will particularly develop an Android application to retrieve and display daily movie images from the RSS feed: *http://movieweb.com/rss/all-news/*

Objectives

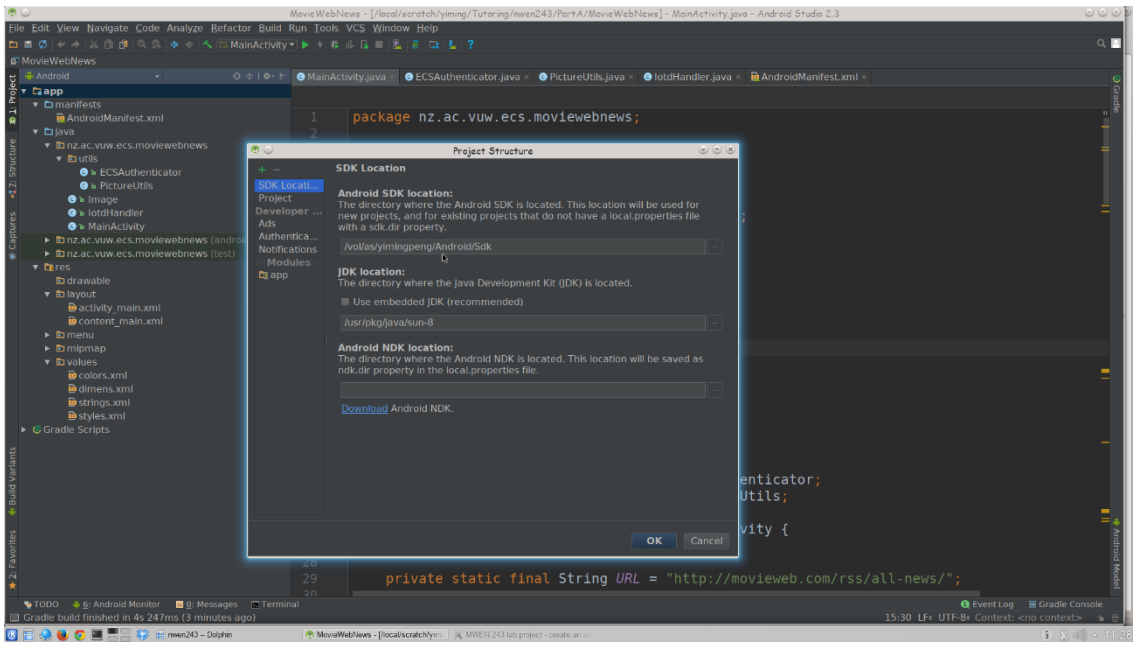
- Experience of using the Android development environment
- Experience of building simple Android applications

Requirements

- You are required to work on the given application development template and complete the code in missing parts as indicated in the template.
- You are recommended to demonstrate your work to your lab tutor.

Things to know:

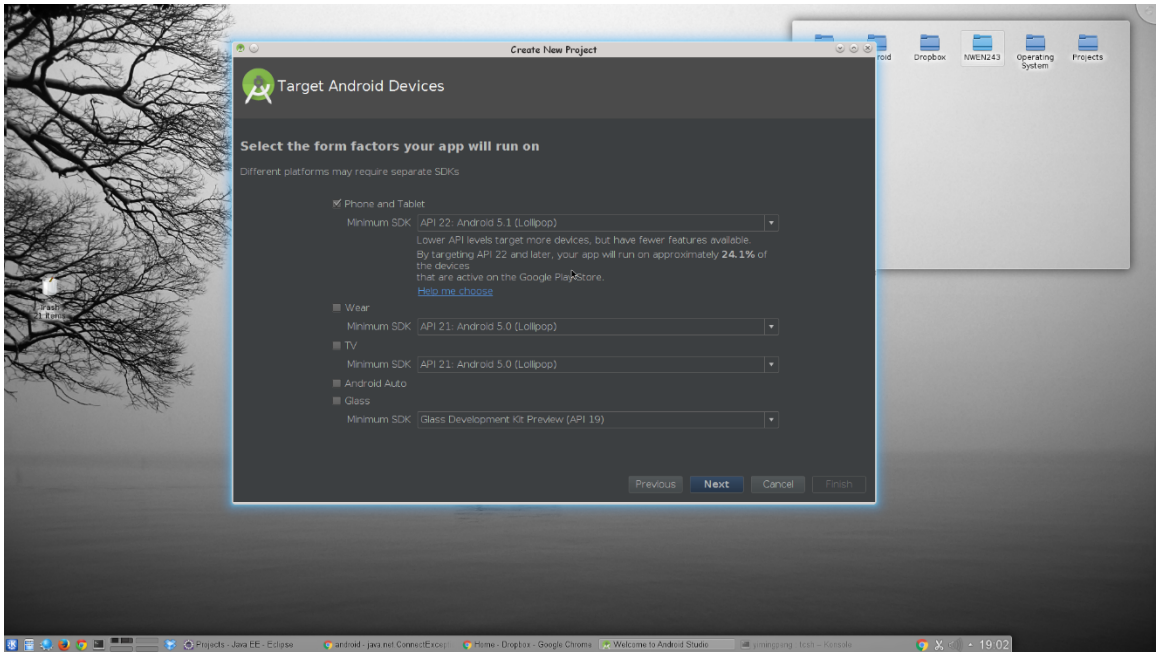
- Please set **Android SDK location (File – Project Structure – SDK Location)** to "/vol/as/**your_ecs_username**/Android/Sdk" as shown below.



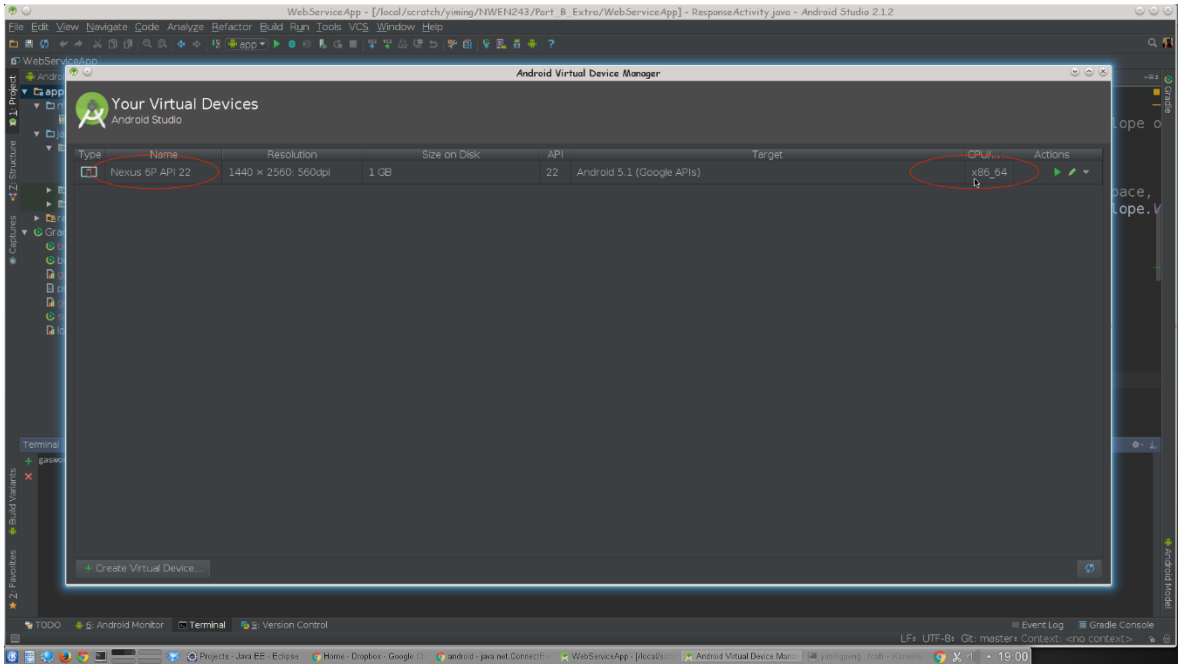
- The emulator by default picks up Android studio's proxy settings so just set it up in Android studio. You may follow the steps below: Eg File - Settings - Appearance & Behavior - System Settings - HTTP Proxy. Click on Auto-detect proxy settings. Tick Automatic proxy configuration URL: add http://proxy.ecs.vuw.ac.nz/proxy.pac

Important Settings

- Android SDK Version: The minimum SDK for the given project is "API22: Android 5.1 (Lollipop)".



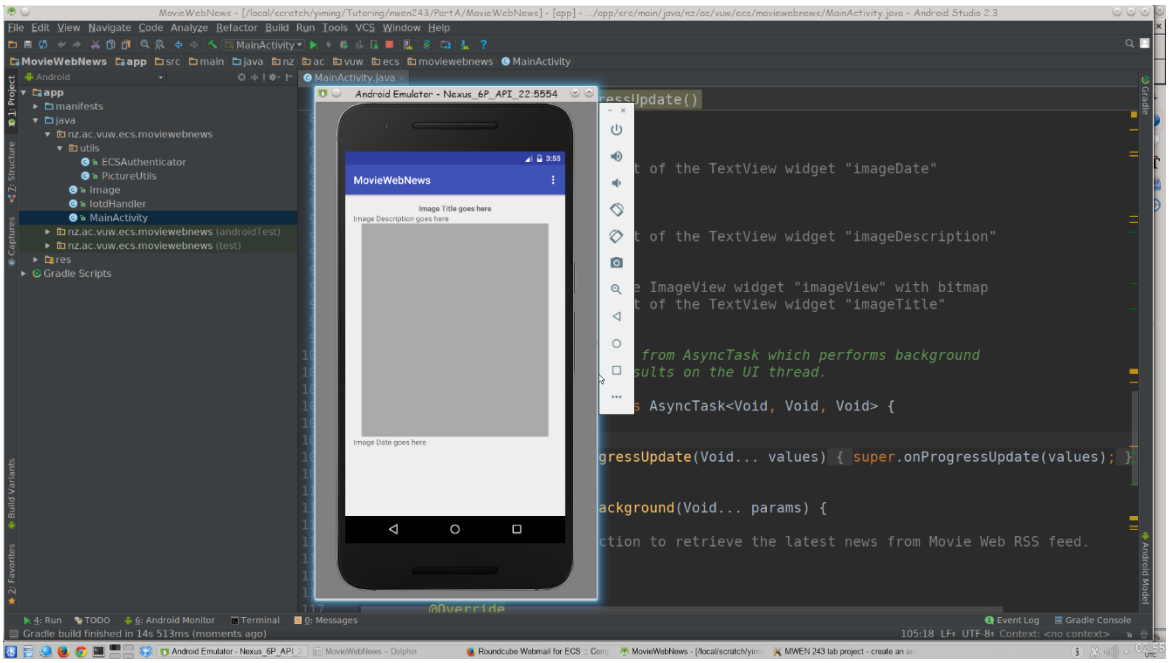
- Recommended Android Virtual Device: *Nexus 6P API 22*. Please refer to the settings shown below for more details.



Step-by-step description

Step. 1

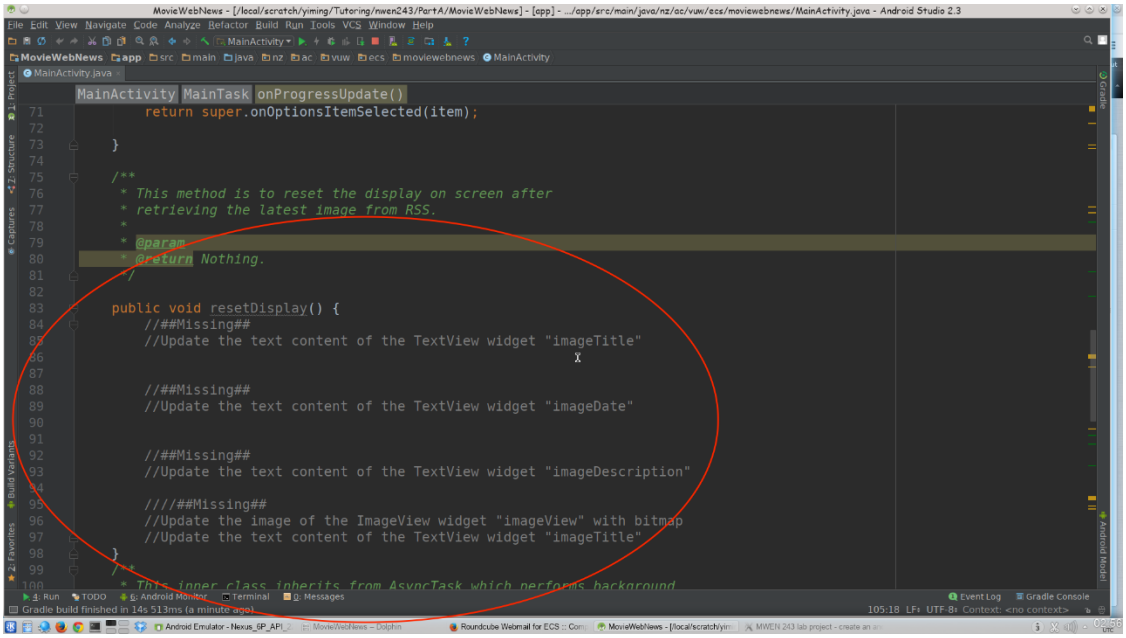
The project template provided can be compiled and executed easily. Please make the program compile and run on the emulator (you need to configure an Android virtual device through the device manager).



Step. 2

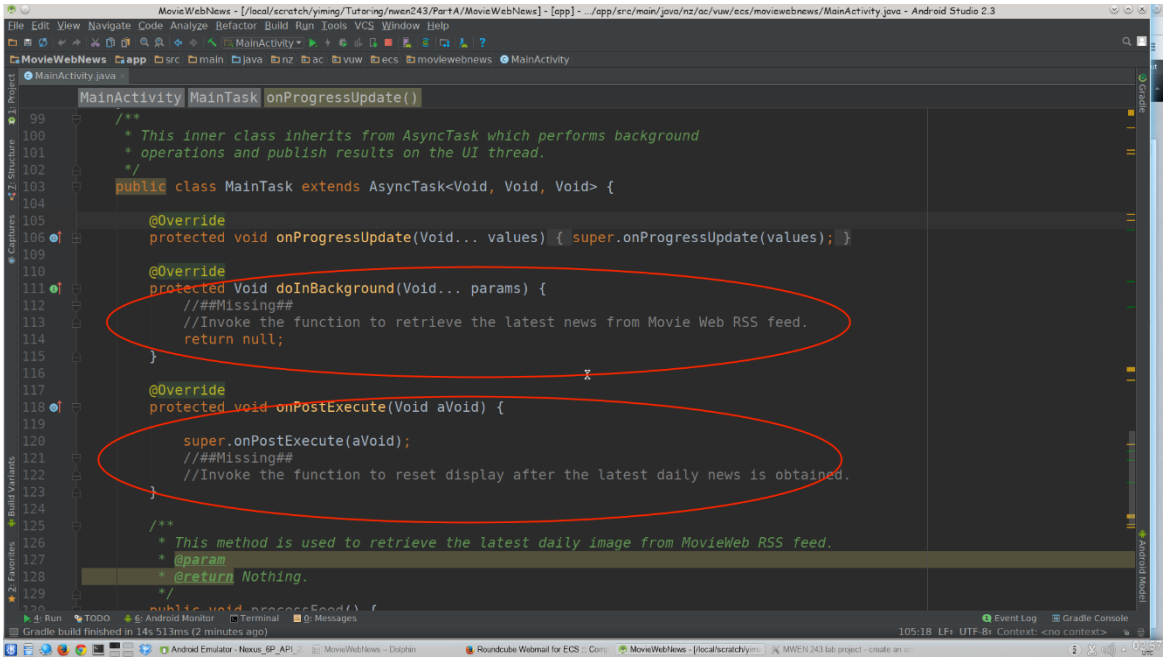
Please find and complete all missing parts in the source code, which are all labelled by "##Missing##". The line number of missing code are given below:

a. Line 84 – 97



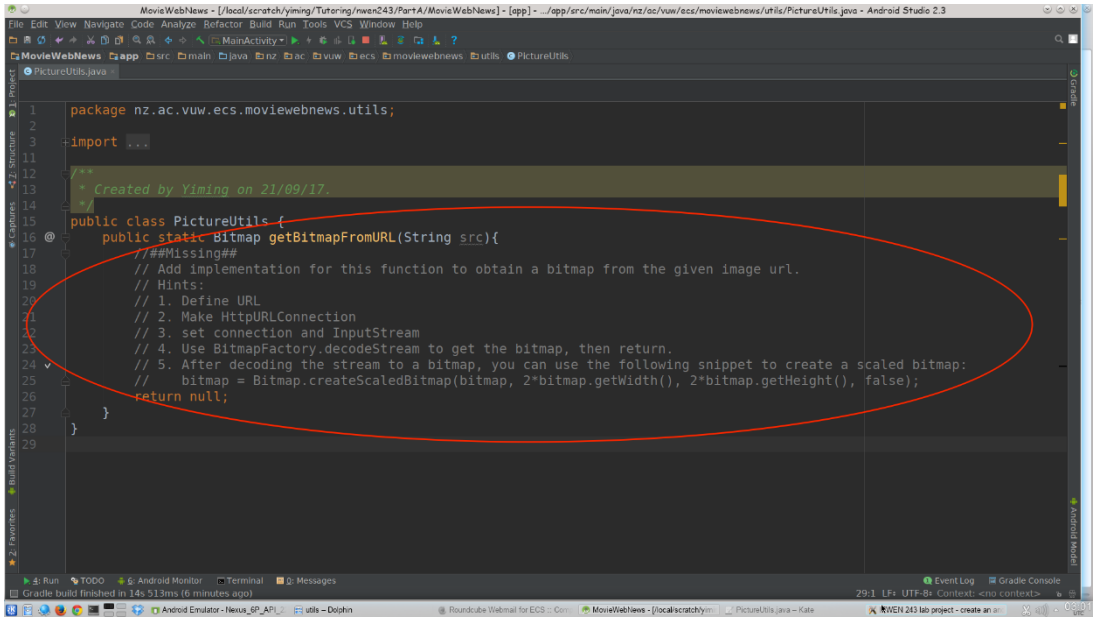
b. Line 112-113

c. Line 121-122



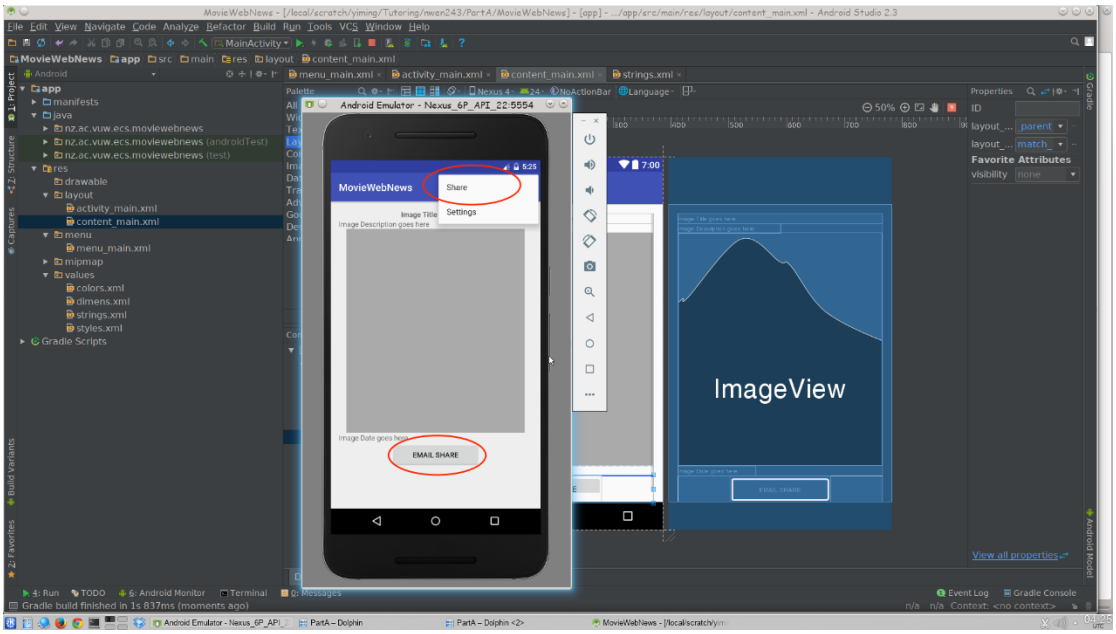
Step 3

Please complete the function "getBitmapFromURL(String src)" in "PictureUtils.java" for obtaining a bitmap from the given image URL.

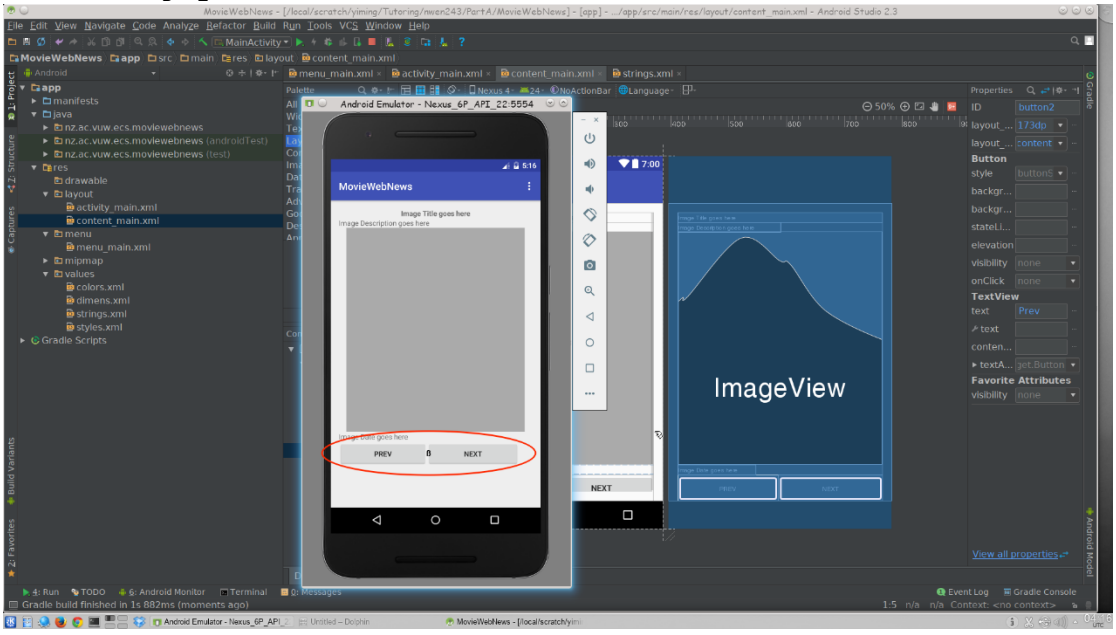


Challenging tasks:

1. Add a function to share the daily movie image with your friends. Either you can create a button on your main activity to share the image with its relative information via email, or you can add share actions to your action bar.



2. Add a function to switch across multiple images. For example, you may add two buttons, "Prev" and "Next" (see an example below). By clicking on those buttons, your Android app can display the previous movie image or the next movie image accordingly.



Hand in

Submit your complete Android app development project (in the form of a ZIP file) through the online submission system.

No technical report is required.

Grading

C grade – step 1 completed successfully.

B grade – step 2 completed successfully.

A grade – step 3 completed successfully.

Note: You can get **1 extra mark** for your total mark in this course if you can successfully complete the two challenging tasks above. Lab tutors however will not provide any support for the challenging tasks.

Part 2: Develop a simple Web Service Requester and Provider

In a networked environment, multiple software systems may often need to exchange information with each other. Web Service stands for an important technology to facilitate such information exchange especially over the Internet. In general terms, the software system that requests data is called a *service requester*, whereas the software system that processes the request and provides the data is called a *service provider*. In this lab project, you will build a simple service requester and service provider using Web Service and Java. Detailed steps are described below.

Objectives

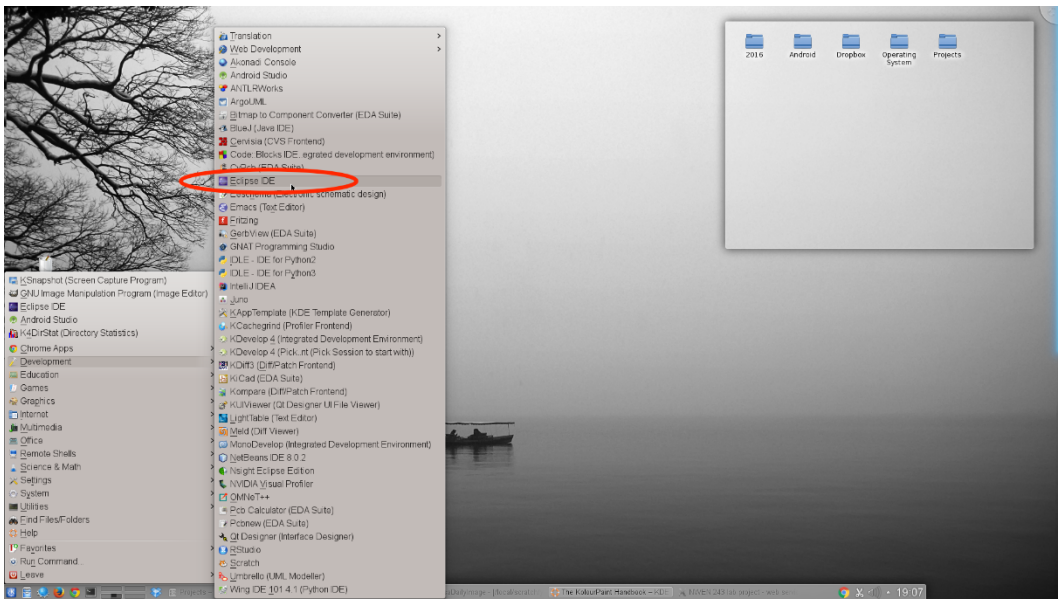
- Experience the Web Service concept and technology
- Develop simple Web Service provider and requester

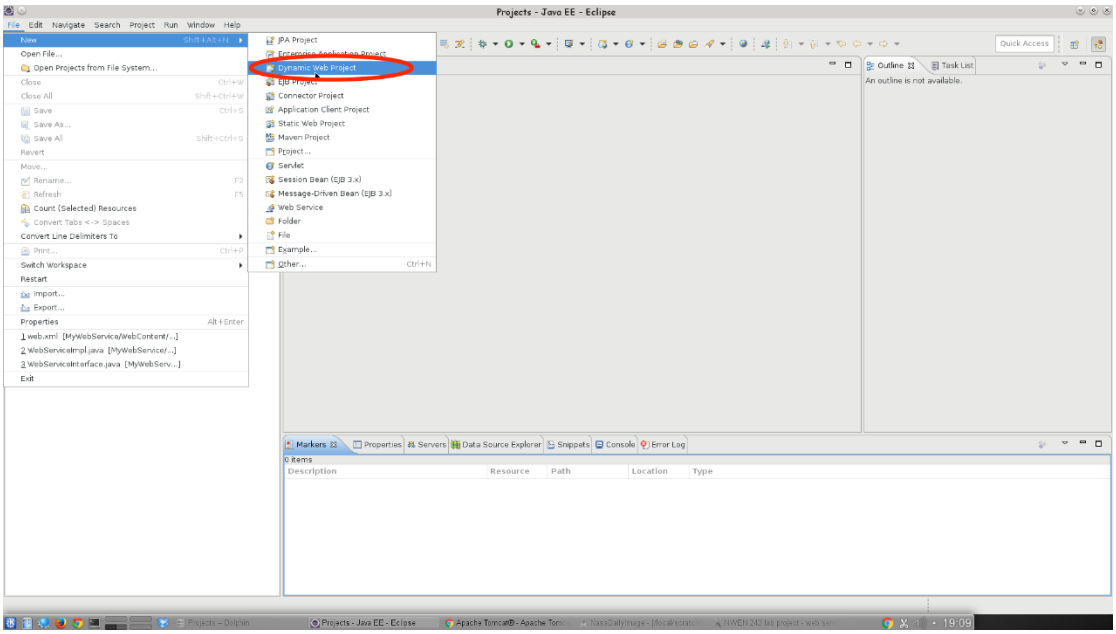
Requirements

- You will need to use Eclipse IDE for Java EE Developers to develop your Web Service project.
- You will need to use Apache Tomcat 6.0 to host your Web Service.
- You are recommended to demonstrate your work to your lab tutor.

Exercise

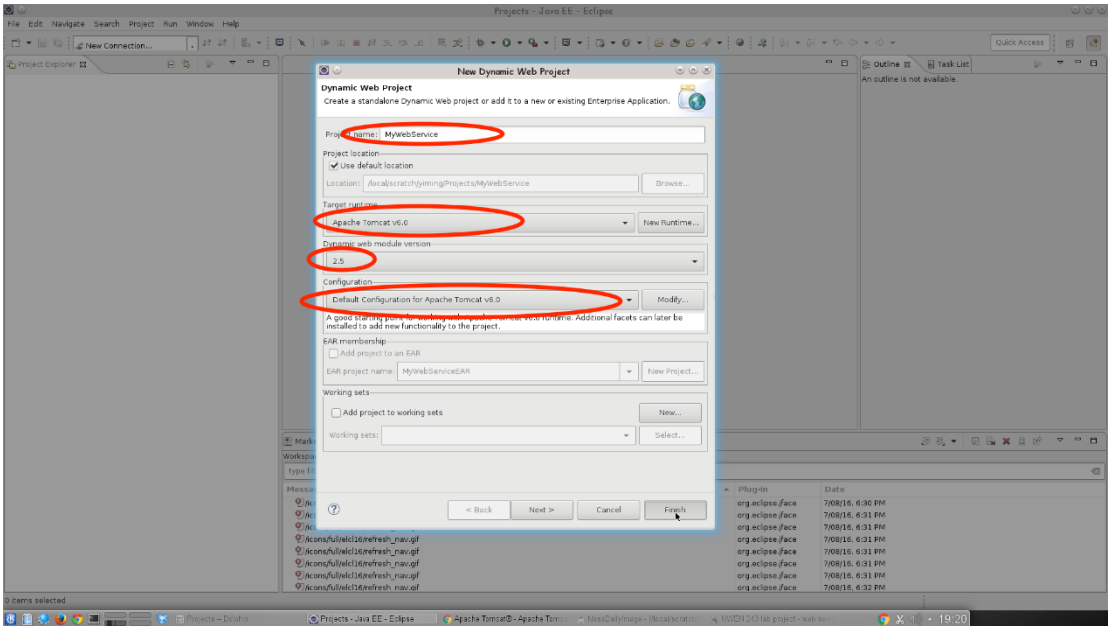
Step 1: Create a dynamic Web project using Eclipse IDE for Java EE Developers.





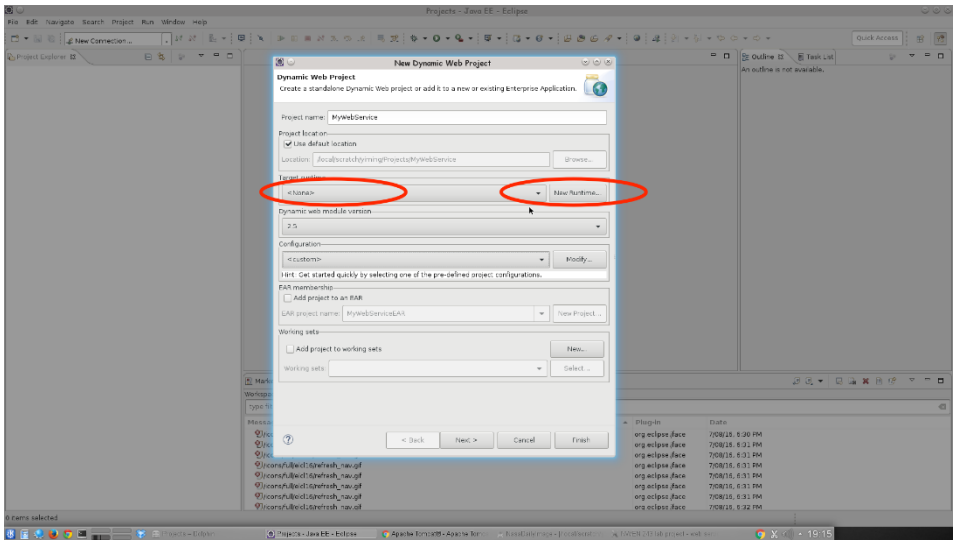
Step 2: Set up the project with the following settings:

- Define a project name (e.g., **MyWebService**)
- Choose target runtime to be "**Apache Tomcat v6.0**".
- Choose the dynamic web module version to be "**2.5**".
- Choose the configuration to be the "**default configuration for apache tomcat v6.0**".

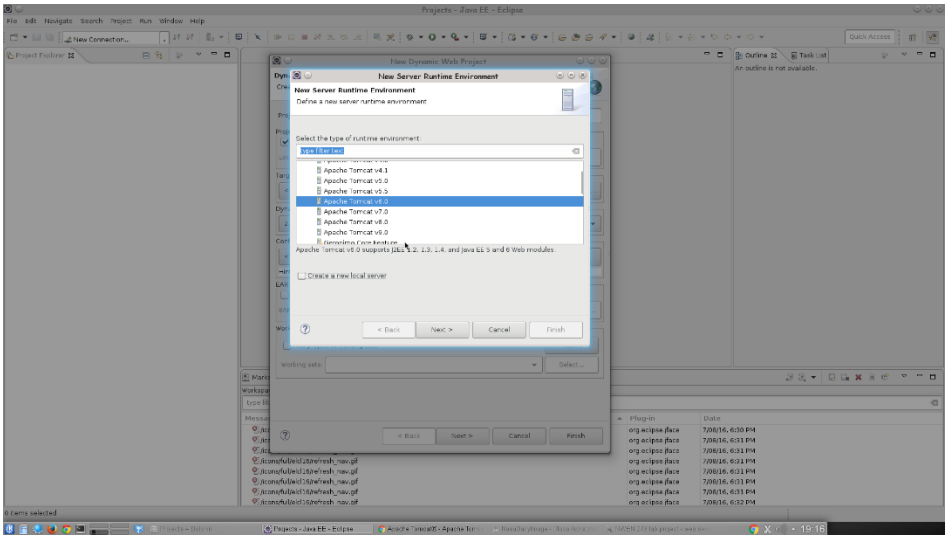


NOTE:

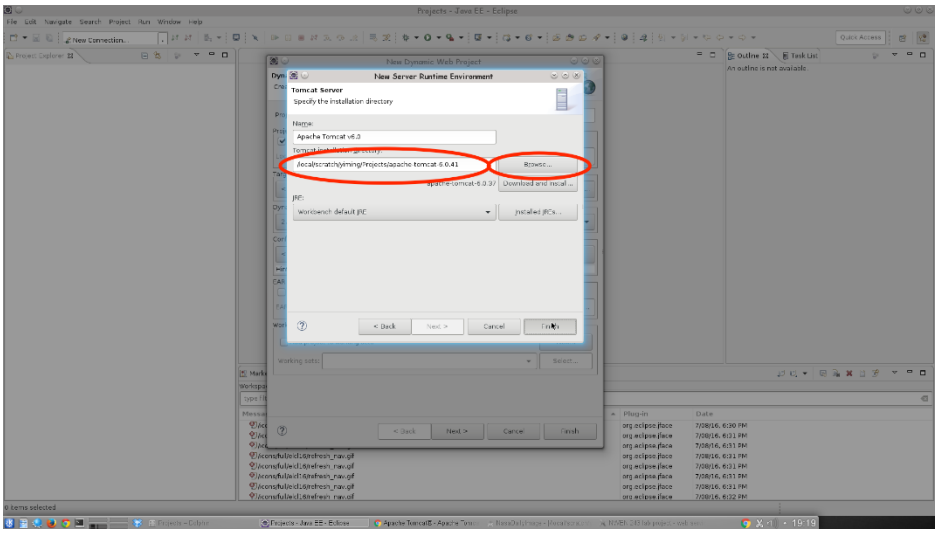
- If the target runtime shows "NONE" and has no options of "Apache Tomcat 6.0", you may choose to create a new runtime.



- Select “Apache Tomcat v6.0”

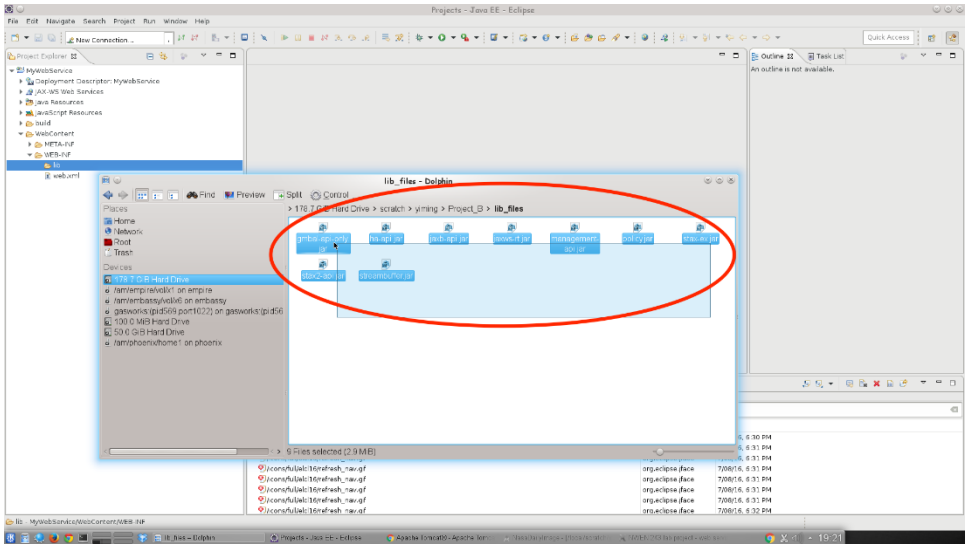


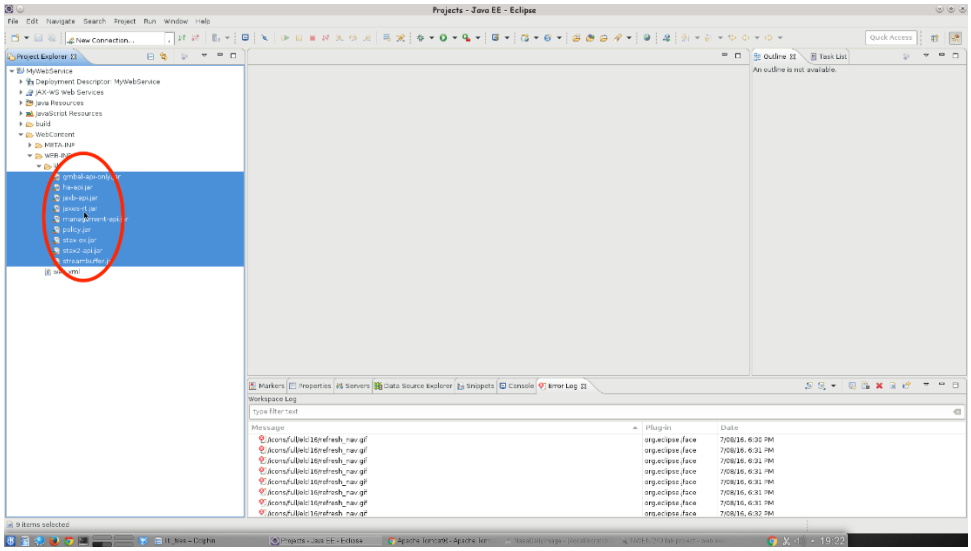
- Setup the path to the tomcat directory “apache-tomcat-6.0.41” provided through this project on the course website.



Step 3: Copy all important jar files (provided through this project on the course website) to “YOUR_PROJECT/WebContent/lib/”.

- jaxb-api.jar, jaxb-impl.jar, jaxws-rt.jar, stax-ex.jar, streambuffer.jar, gmbal-api-only.jar, ha-api.jar, jaxb-api.jar, management-api.jar, policy.jar, stax2-api.jar





Step 4: Create a package with your preferred name (e.g., example) under the directory "Java Resources/src/" in your project (e.g., MyWebService).

Step 5: Define a Java Interface (e.g., WebServiceInterface) by following the sample code given below.

```
package example;

import javax.jws.WebService;

@WebService
public interface WebServiceInterface {
	//Your own function signature goes here
	public String greet(String clientName);
}
```

Step 6: Define a Java class (e.g., WebServiceImpl) that implements the interface defined in step 5 by following the sample code below.

```
package example;

import javax.jws.WebService;

@WebService(
                endpointInterface = "example.WebServiceInterface",
                portName = "webservicePort",
                serviceName = "WebService")
public class WebServiceImpl implements WebServiceInterface {

        @Override
        public String greet(String clientName) {
                // TODO Auto-generated method stub
                // your own function implementation goes here

                return "Hi," + clientName + ", this is WebService!";
        }
}
```

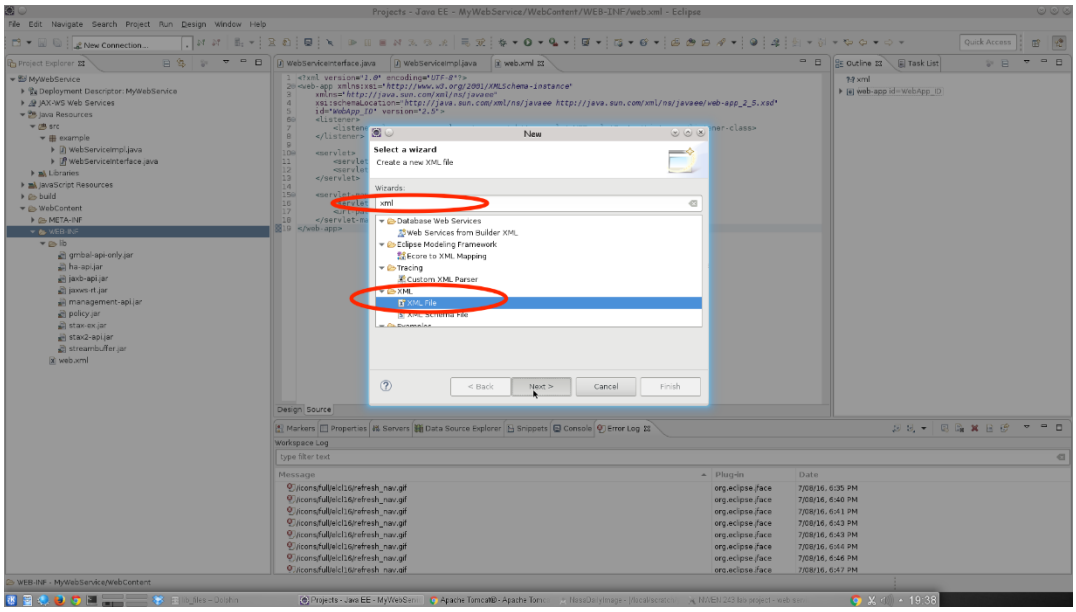
Step 7: Modify web.xml

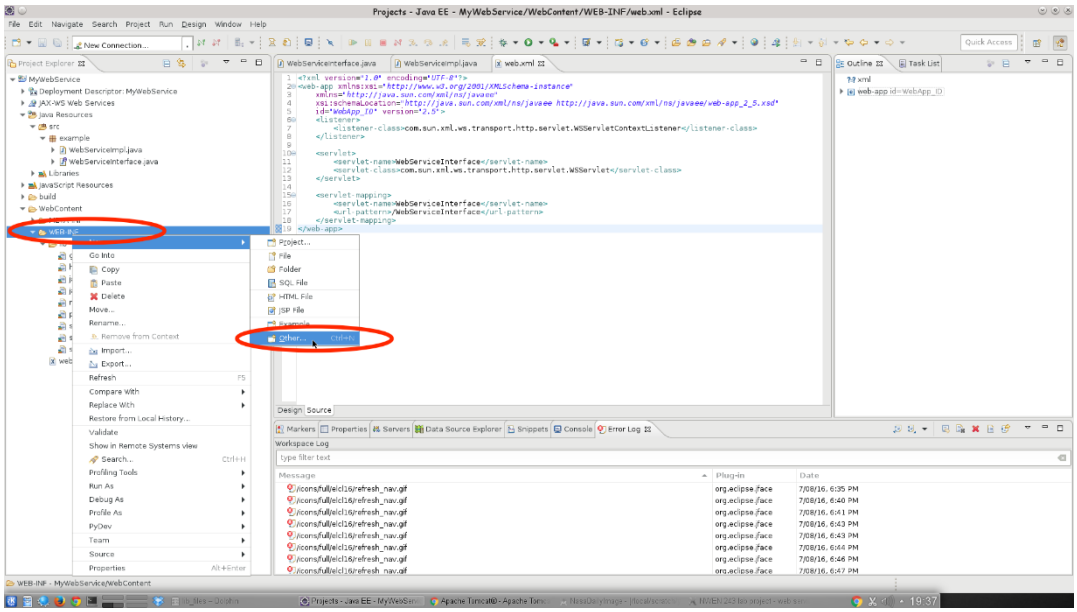
The file web.xml can be found in "PROJECT/WebContent/WEB-INF". Modify this file according to the template given below.

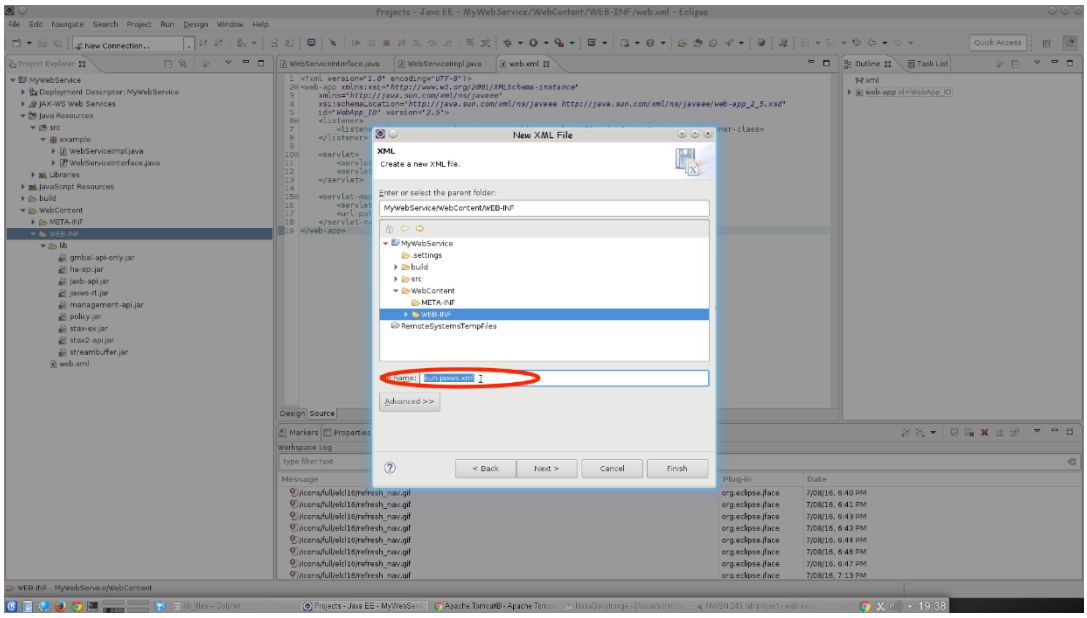
```
<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xmlns="http://java.sun.com/xml/ns/javaee"
        xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
http://java.sun.com/xml/ns/javaee/web-app_2_5.xsd"
        id="WebApp_ID" version="2.5">
        <listener>
                <listener-
class>com.sun.xml.ws.transport.http.servlet.WSServletContextListener</listener-
class>
        </listener>

        <servlet>
                <servlet-name>WebServiceInterface</servlet-name>
                <servlet-
class>com.sun.xml.ws.transport.http.servlet.WSServlet</servlet-class>
        </servlet>

        <servlet-mapping>
                <servlet-name>WebServiceInterface</servlet-name>
                <url-pattern>/WebServiceInterface</url-pattern>
        </servlet-mapping>
</web-app>
```

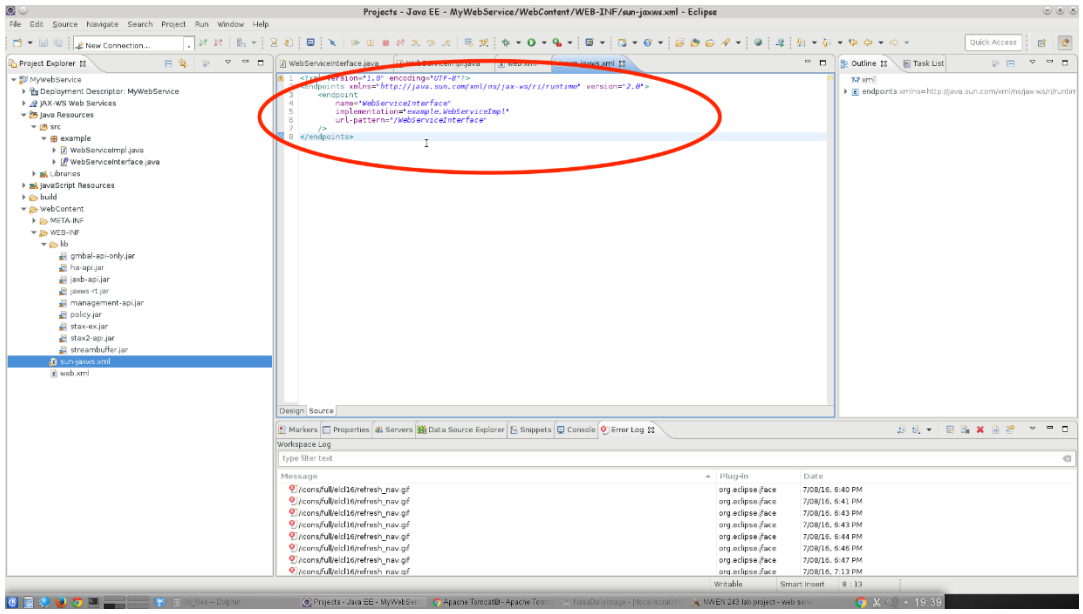
Step 8: Create a new XML file "sun-jaxws.xml" under the directory "/WebContent/WEB-INF/" of your project. Please prepare your file by following the example below.

```
<?xml version="1.0" encoding="UTF-8"?>
<endpoints xmlns="http://java.sun.com/xml/ns/jax-ws/ri/runtime"
version="2.0">
    <endpoint
        name="WebServiceInterface"
        implementation="example.WebServiceImpl"
        url-pattern="/WebServiceInterface"
    />
</endpoints>
```



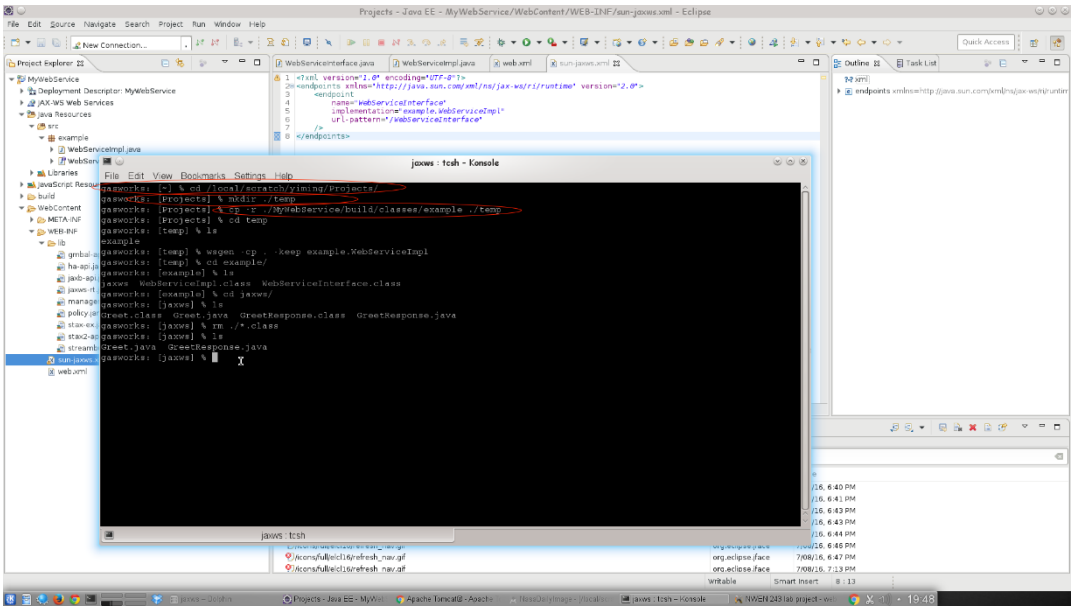




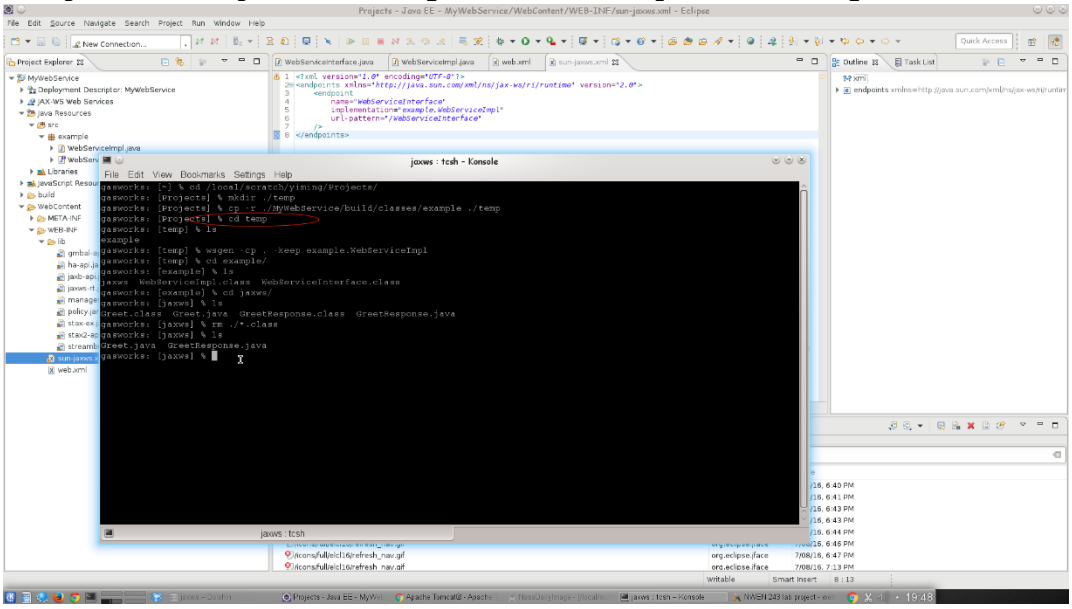


Step 9: Generate JAXB classes. JAXB stands for the Java Architecture for XML Binding. You will need to perform the following steps:

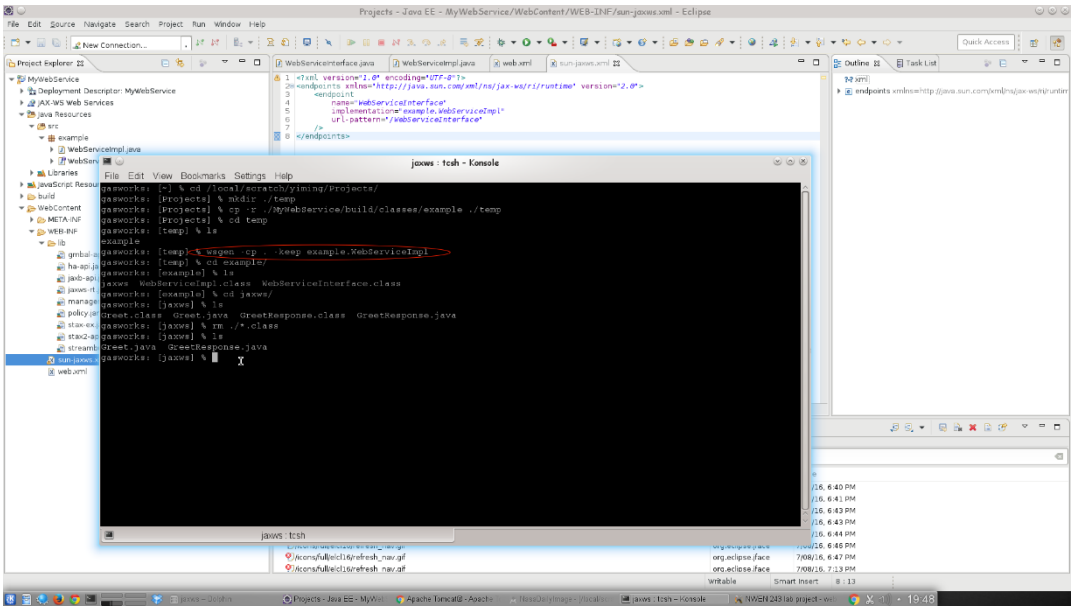
- Copy interface and implementation class files (e.g., WebServiceInterface.class and WebServiceImpl.class) in the directory "/YOUR_PROJECT/build/classes/" to a temp (and originally empty) directory (e.g., ~/temp/). If these class files are located in a package (e.g., example/), you need to copy the package as well.



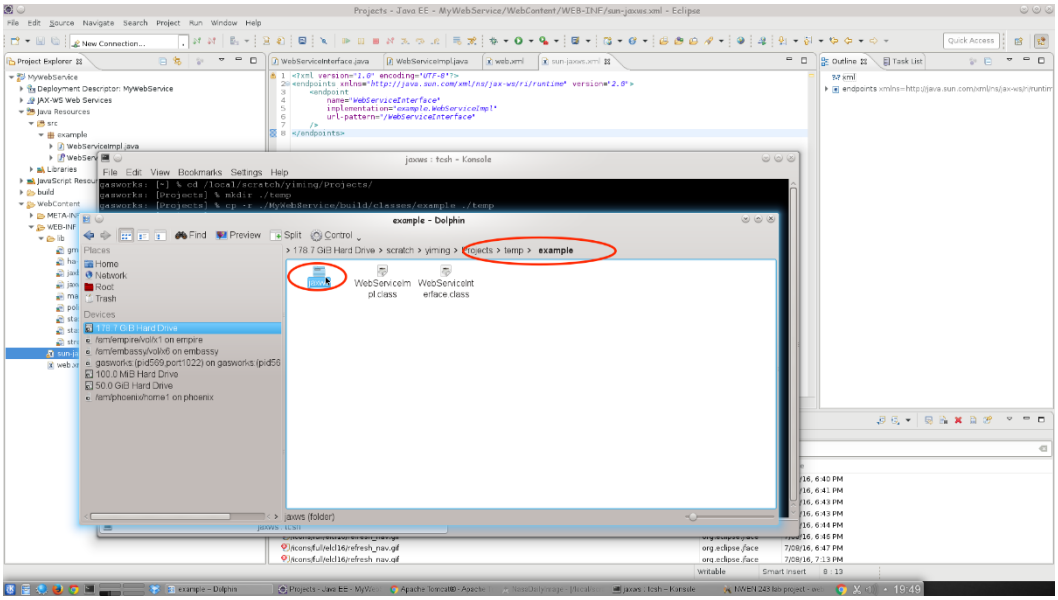
- Change working directory to the temp directory.

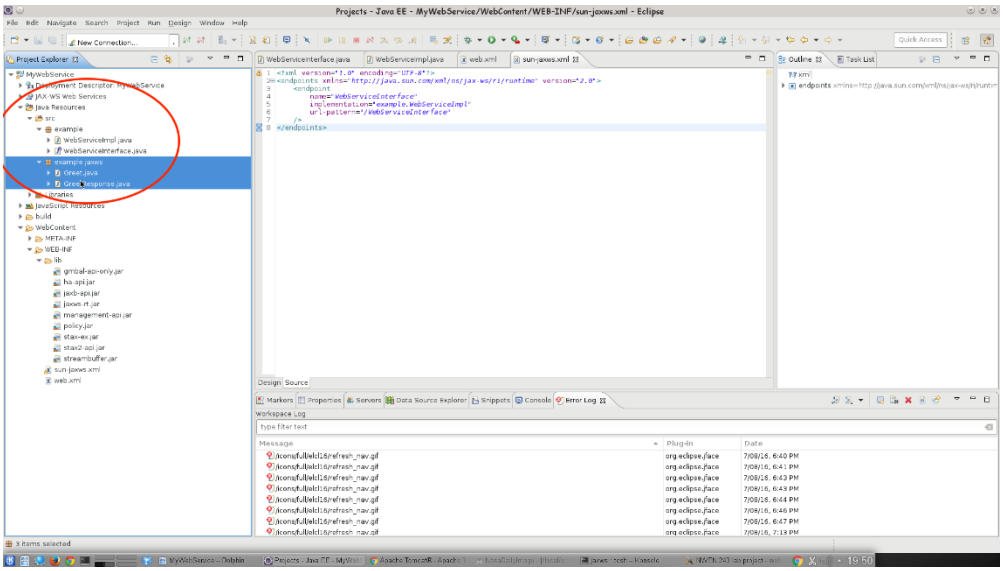


- Execute the command "**wsgen -cp . -keep WebServiceImpl**", where WebServiceImpl stands for the Java class that implements the service interface. The package name should also be included in the command if the implementation class is defined in a package (e.g., "**wsgen -cp . -keep example.WebServiceImpl**").



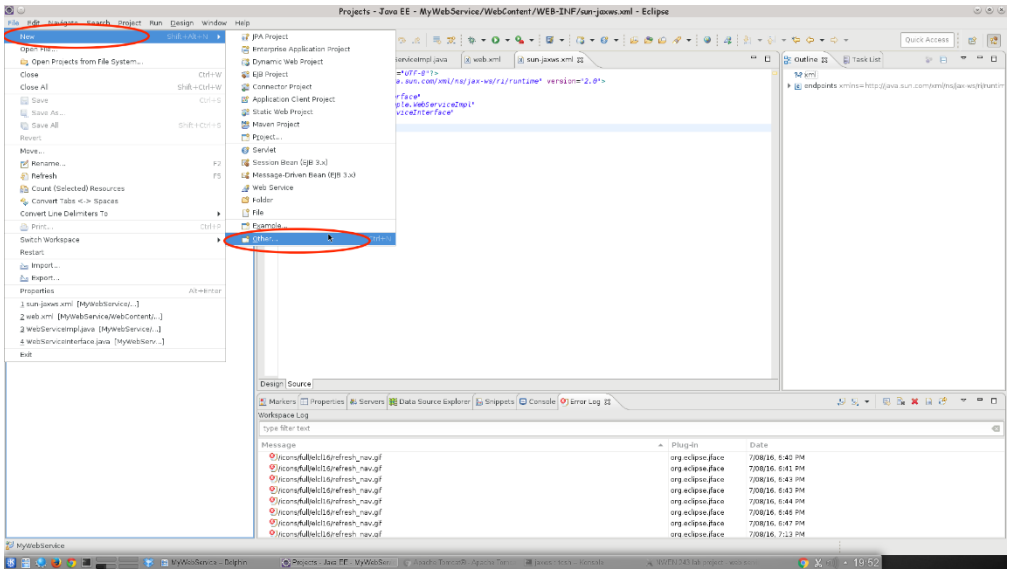
- At this stage, you should see four files (e.g., Greet.class, GreetResponse.class, Greet.java, GreetResponse.java) in the directory "jaxws". Delete the class files (e.g., Greet.class and GreetResponse.class). Then copy the entire package (e.g., jaxws) containing two java source files (e.g., Greet.java and GreetResponse.java) to your project directory - "/YOUR_PROJECT/Java Resources/src/example/".

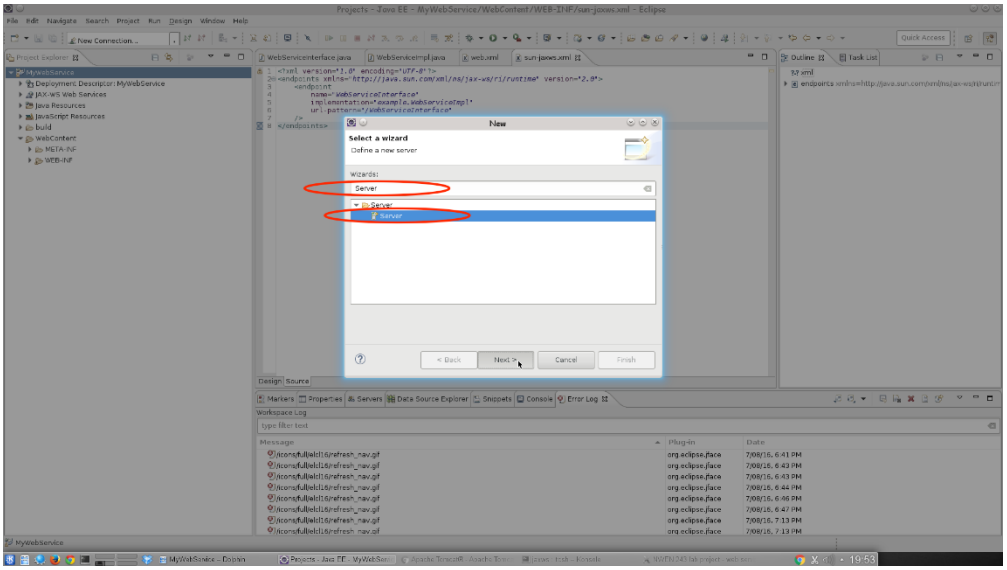




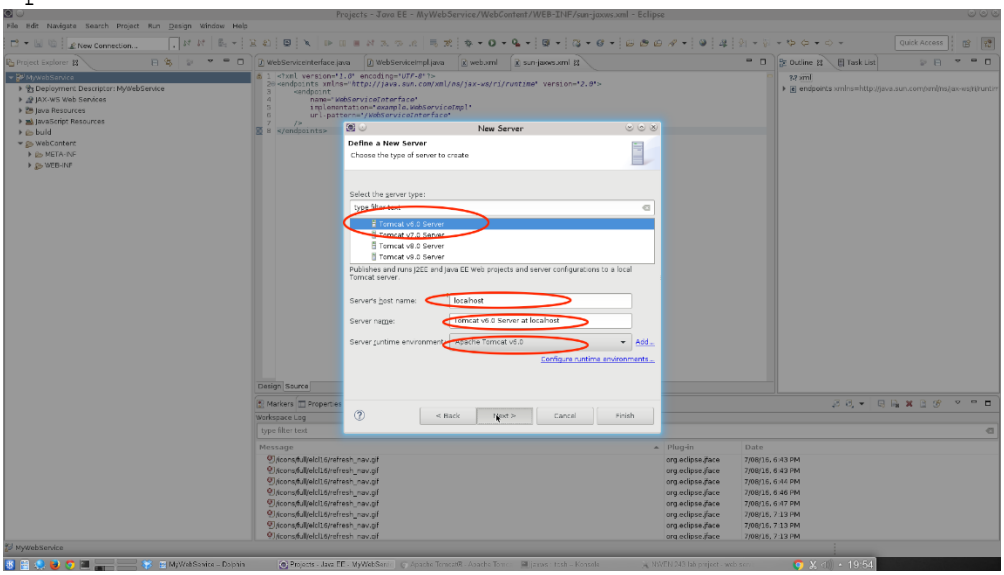
Step 10: Define Apache Tomcat v6.0 Server.

- Create a new server

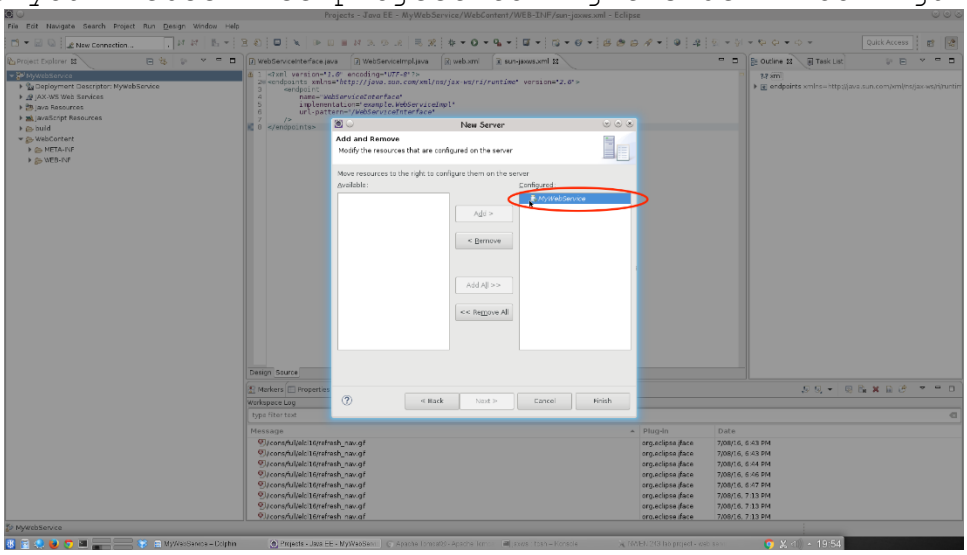




- Setup the server

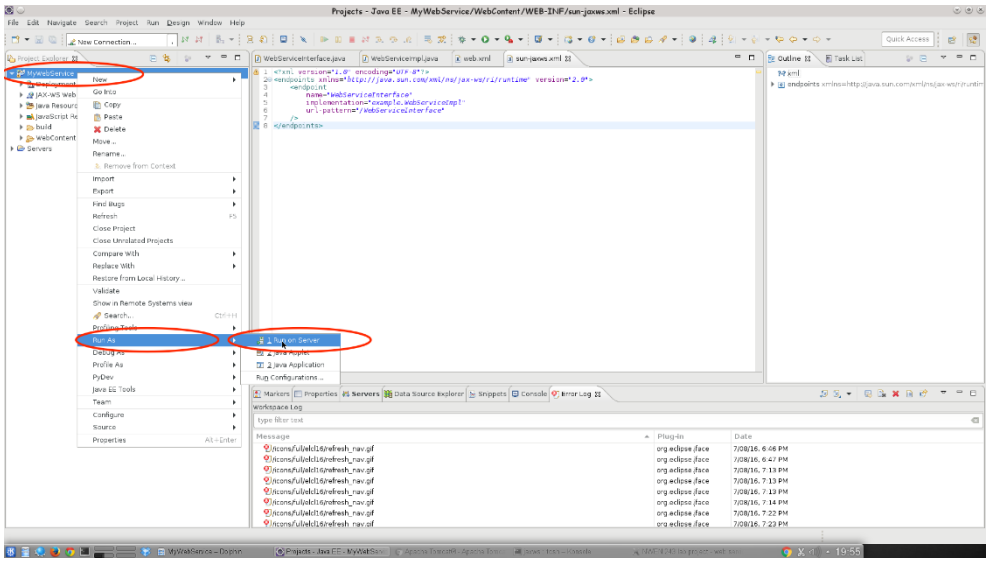


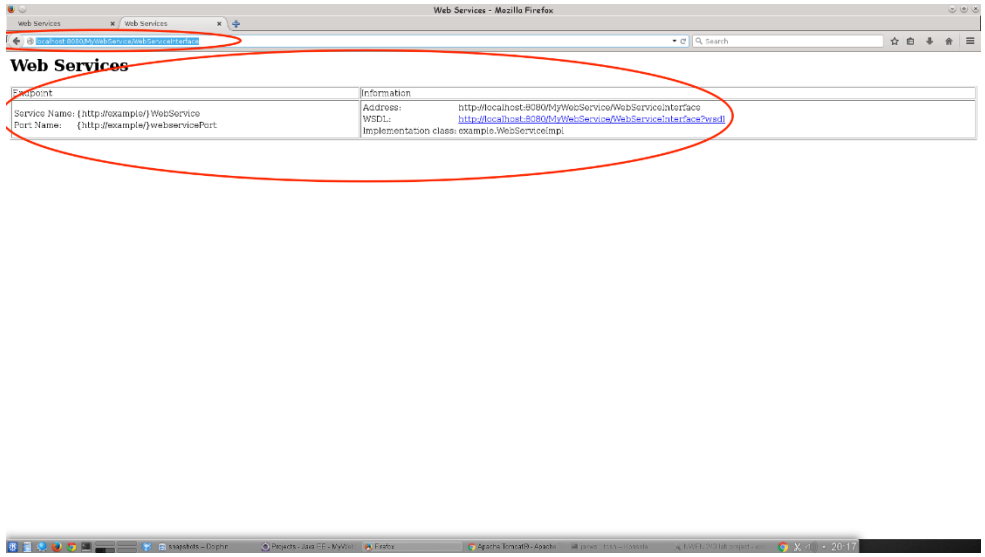
- Move your WebService project to right side - "Configured".



Note: Please do not start your Tomcat server at this stage.

Step 11: Deploy the Web Service by right clicking on your web project and choose to run on server. Now you can open a browser and enter a URL (e.g., http://localhost:8080/MyWebService/WebServiceInterface) to check the WSDL service description. If you can see the service description, it means that your Web Service is running properly!





Step 12: Create a web service requester

Up to this point, you have successfully developed and deployed a Web Service provider. Let us develop a web service requester program by following the sample code below. Afterwards, please run your requester program as a normal Java application and check whether it can receive correct response from your Web Service provider.

```java
package example;
import java.net.MalformedURLException;
import java.net.URL;

import javax.xml.namespace.QName;
import javax.xml.ws.Service;

public class WebServiceClient {

	public static void main(String[] args) throws MalformedURLException {
		// TODO Auto-generated method stub
		URL url = new URL("http://localhost:8080/MyWebService/WebServiceInterface?wsdl");
		QName qname = new QName("http://example/", "WebService");
		Service service = Service.create(url, qname);
		WebServiceInterface callWebService = service.getPort(WebServiceInterface.class);
		System.out.println("Service output: \n" + callWebService.greet("WebServiceClient"));
	}
}
```

Step 13: Now you may have an idea of how to develop and deploy a simple Web Service. Following this, please design and implement a Web Service that reports the current temperature of any given city in the world.

Hand in

Submit your Web Service project online in the form of a zip file. Please zip your entire eclipse project to make it easy for tutors to check your submitted code.

No technical report is required.

Grading

C grade – manage to define and implement a simple Web Service provider (e.g. the greeting web service provider described above). The provider can be deployed successfully on a Tomcat server.

B grade – manage to develop a simple Web Service requester successfully (e.g. the greeting web service requester described above). Upon running the requester program, desirable response can be received from the service provider.

A grade – manage to develop and deploy a weather report service provider successfully. The service provider accepts city and country input and can provide expected temperature output, as verified by the correspondingly service requester program. However, the temperature reported by the service provider is randomly generated locally without accessing any information from the Internet.

A+ grade – the weather report service provider will provide the temperature information based on data retrieved from the weather API provided by https://openweathermap.org/api

Note: Lab tutors will not provide support for the task for the A+ grade.