NWEN 243 Lab Project 8: 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

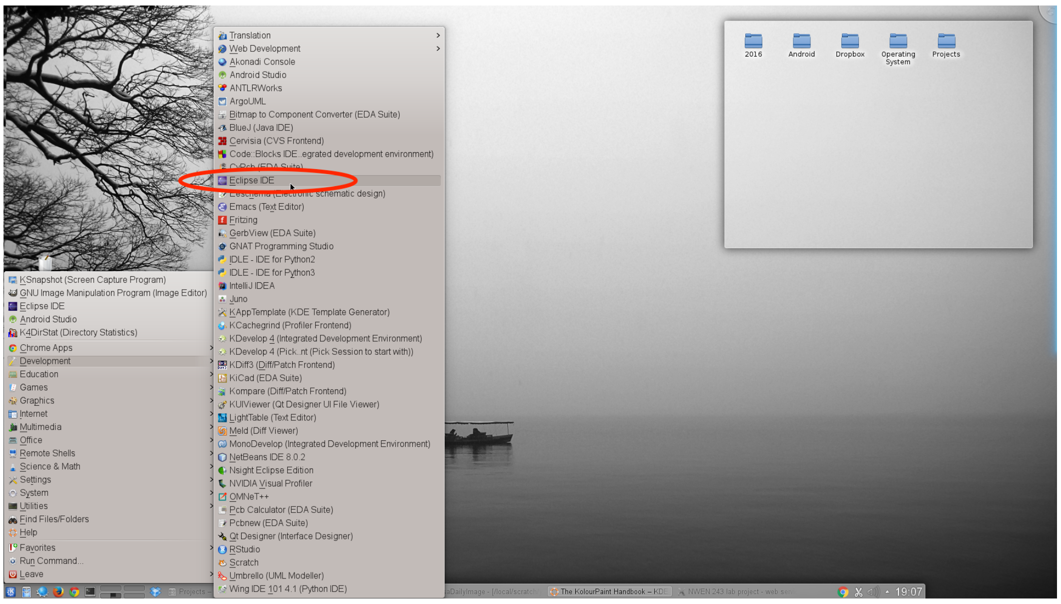
- Experience the Web Service concept and technology
- Develop simple Web Service provider and requester

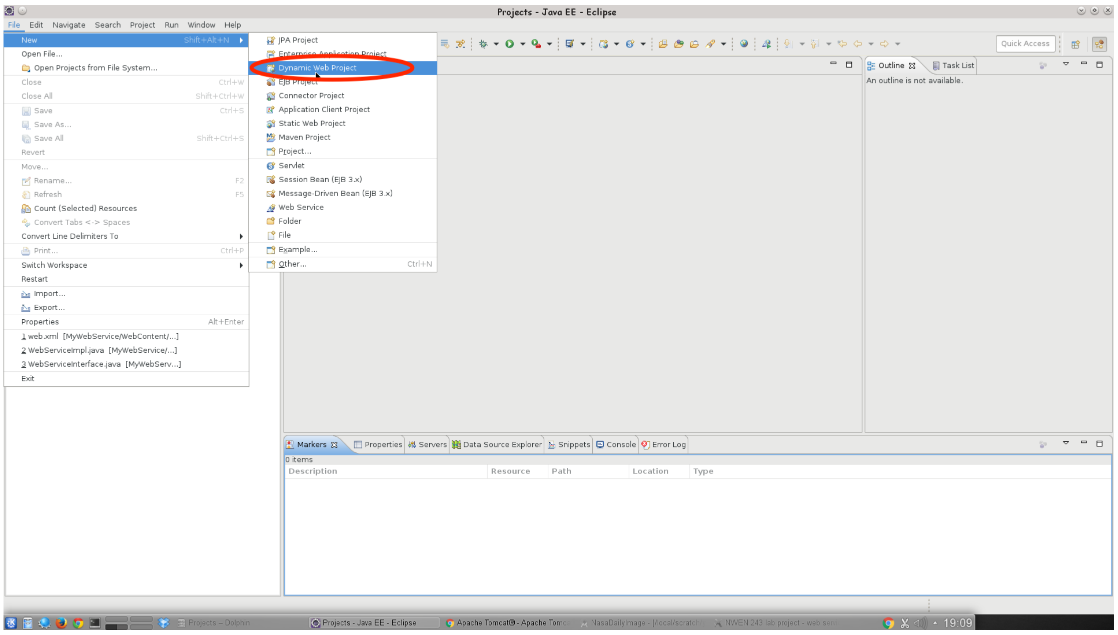
Requirements

- 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

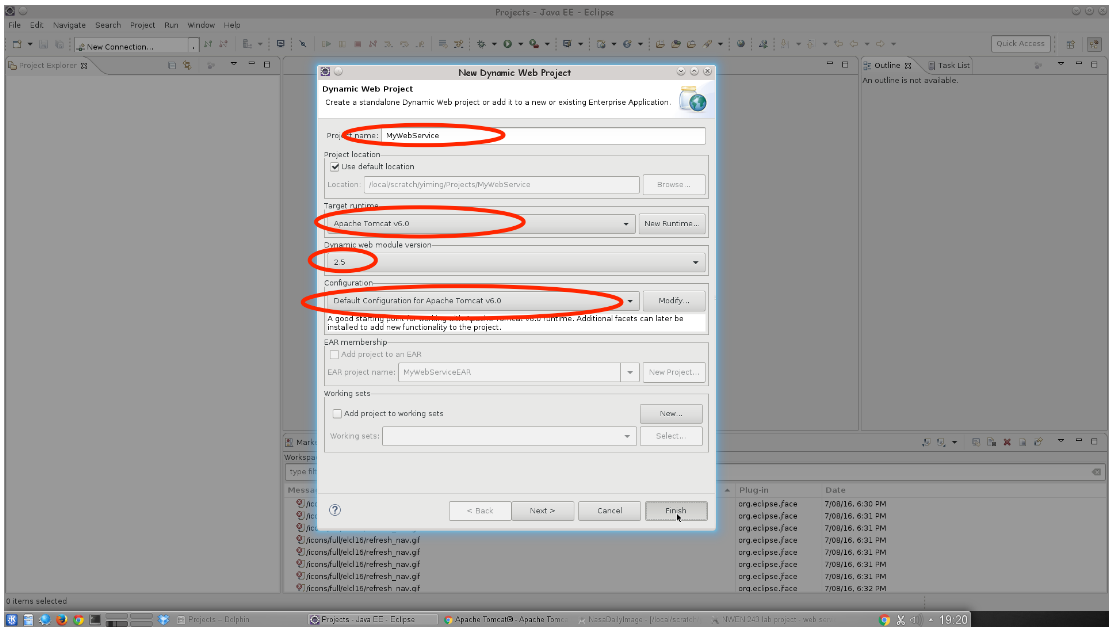
Step 1: Create a dynamic Web project using Eclipse IDE for Java EE Developers.





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**".



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: MyWebService

Project location

Use default location

Location: /local/scratch/yiming/Projects/MyWebService

Target runtime: Apache Tomcat v6.0

Dynamic web module version: 2.5

Configuration: Default Configuration for Apache Tomcat v6.0

A good starting point for working with Apache Tomcat v6.0 runtime. Additional facets can later be installed to add new functionality to the project.

EAR membership

Add project to an EAR

EAR project name: MyWebServiceEAR

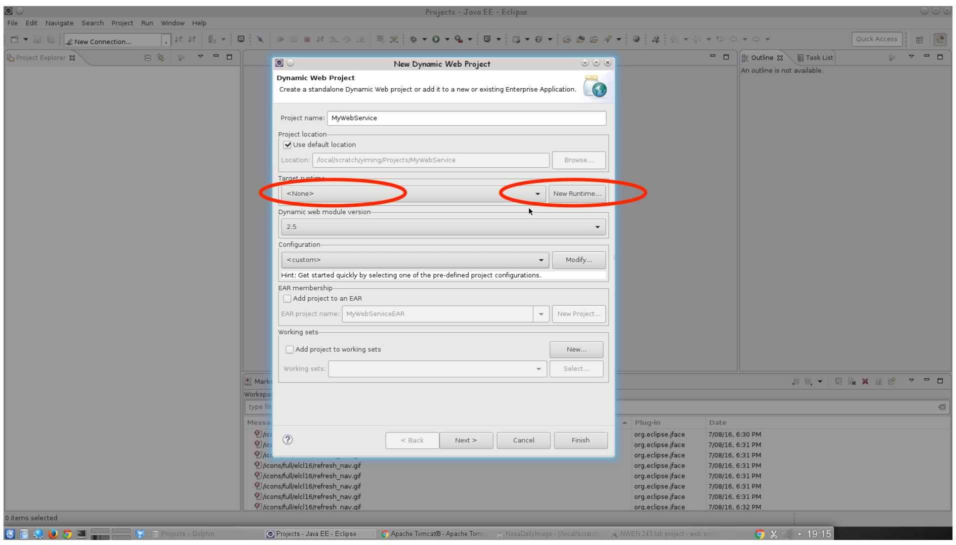
Working sets

Add project to working sets

Working sets:

NOTE:

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



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: MyWebService

Project location

☑ Use default location

Location: /local/scratch/yiming/Projects/MyWebService

Target runtime: <None> New Runtime...

Dynamic web module version: 2.5

Configuration: <custom> Modify...

Hint: Get started quickly by selecting one of the pre-defined project configurations.

EAR membership

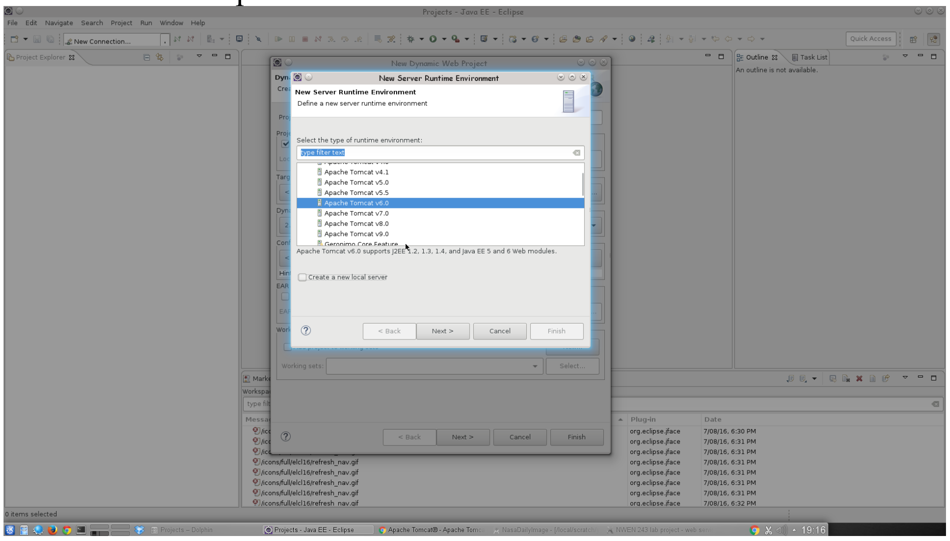
☐ Add project to an EAR

EAR project name: MyWebServiceEAR

Working sets

☐ Add project to working sets

- Select “Apache Tomcat v6.0”



New Server Runtime Environment

New Server Runtime Environment

Define a new server runtime environment

Select the type of runtime environment:

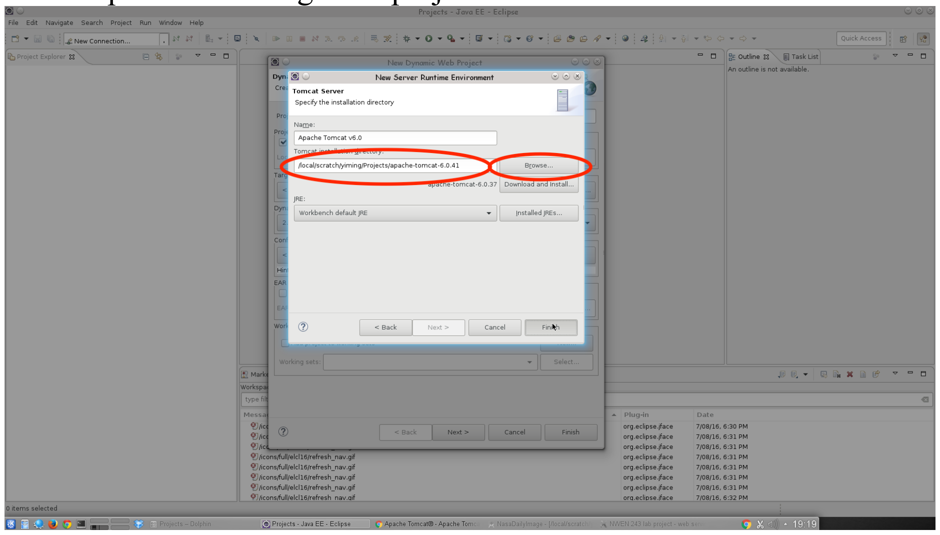
type filter text

- Apache Tomcat v4.1
- Apache Tomcat v5.0
- Apache Tomcat v5.5
- Apache Tomcat v6.0
- Apache Tomcat v7.0
- Apache Tomcat v8.0
- Apache Tomcat v9.0
- Geronimo Core Feature

Apache Tomcat v6.0 supports J2EE 1.2, 1.3, 1.4, and Java EE 5 and 6 Web modules.

☐ Create a new local server

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



New Server Runtime Environment

Tomcat Server

Specify the installation directory

Name: Apache Tomcat v6.0

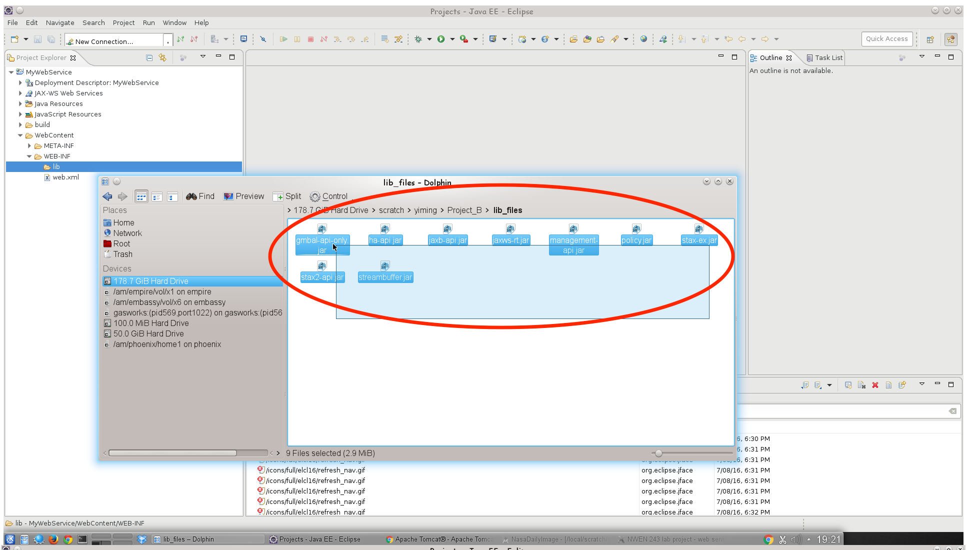
Tomcat installation directory: /local/scratch/yiming/Projects/apache-tomcat-6.0.41

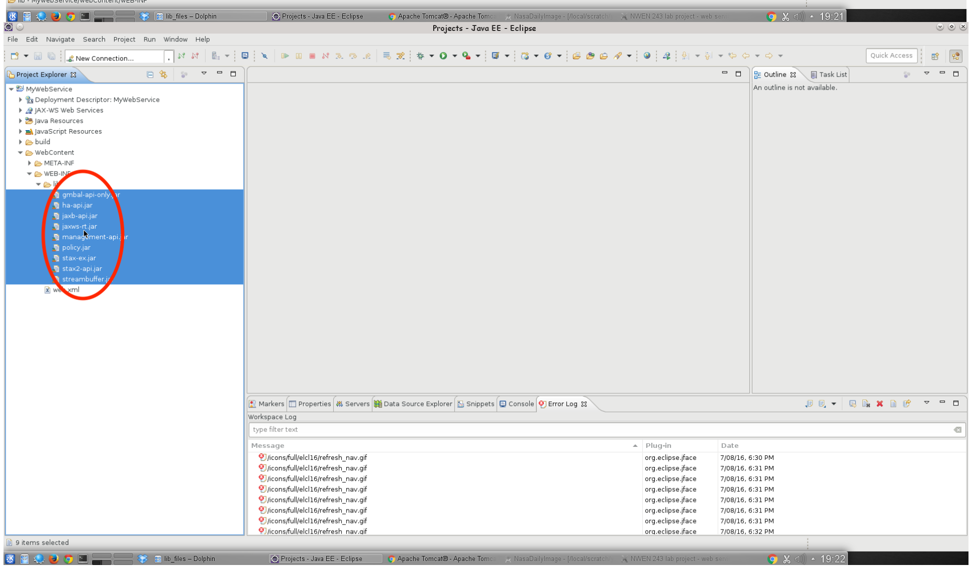
apache-tomcat-6.0.37

JRE: Workbench default JRE

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





```
Message                                  Plug-in            Date
/icons/full/elcl16/refresh_nav.gif       org.eclipse.jface  7/08/16, 6:30 PM
/icons/full/elcl16/refresh_nav.gif       org.eclipse.jface  7/08/16, 6:31 PM
/icons/full/elcl16/refresh_nav.gif       org.eclipse.jface  7/08/16, 6:31 PM
/icons/full/elcl16/refresh_nav.gif       org.eclipse.jface  7/08/16, 6:31 PM
/icons/full/elcl16/refresh_nav.gif       org.eclipse.jface  7/08/16, 6:31 PM
/icons/full/elcl16/refresh_nav.gif       org.eclipse.jface  7/08/16, 6:31 PM
/icons/full/elcl16/refresh_nav.gif       org.eclipse.jface  7/08/16, 6:31 PM
/icons/full/elcl16/refresh_nav.gif       org.eclipse.jface  7/08/16, 6:32 PM
```

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.

```java
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.

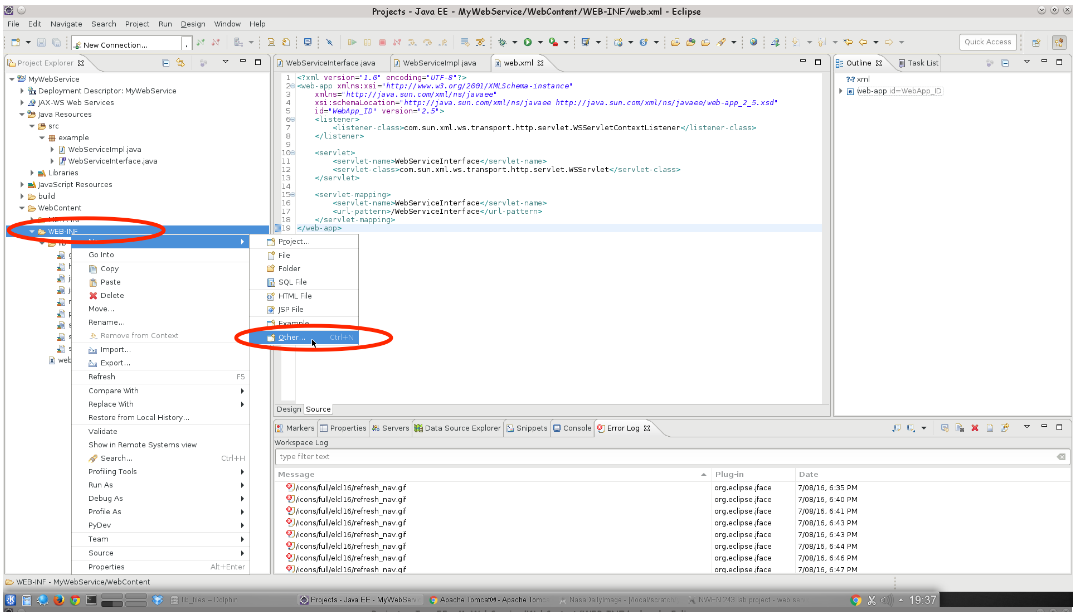
```xml
<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
       xmlns="http://java.sun.com/xml/ns/javaee"
       xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
http://java.sun.com/xml/ns/javaee/web-app_2_5.xsd"
       id="WebApp_ID" version="2.5">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

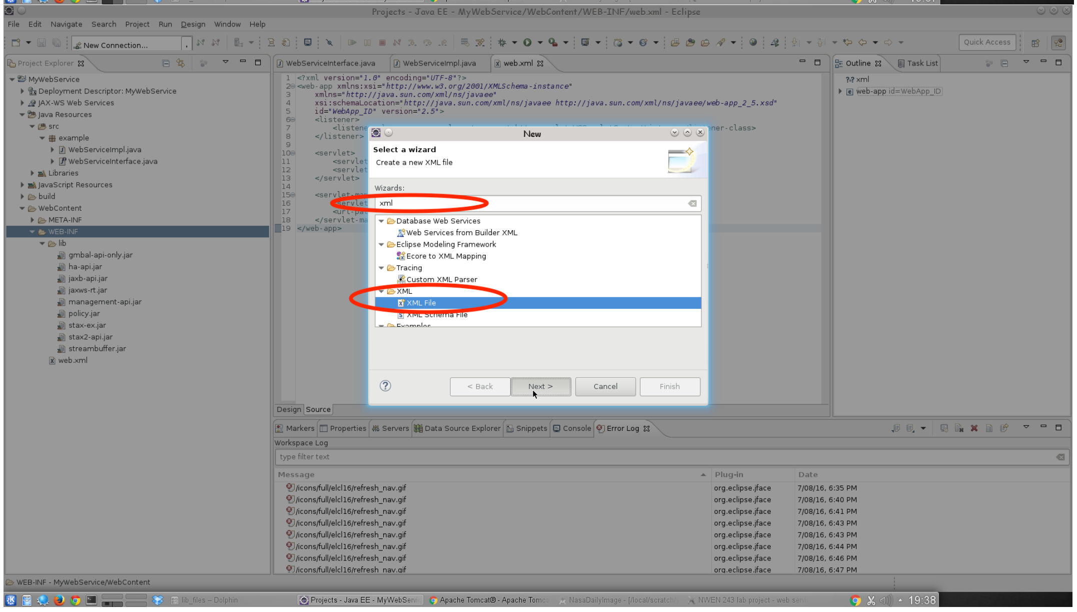


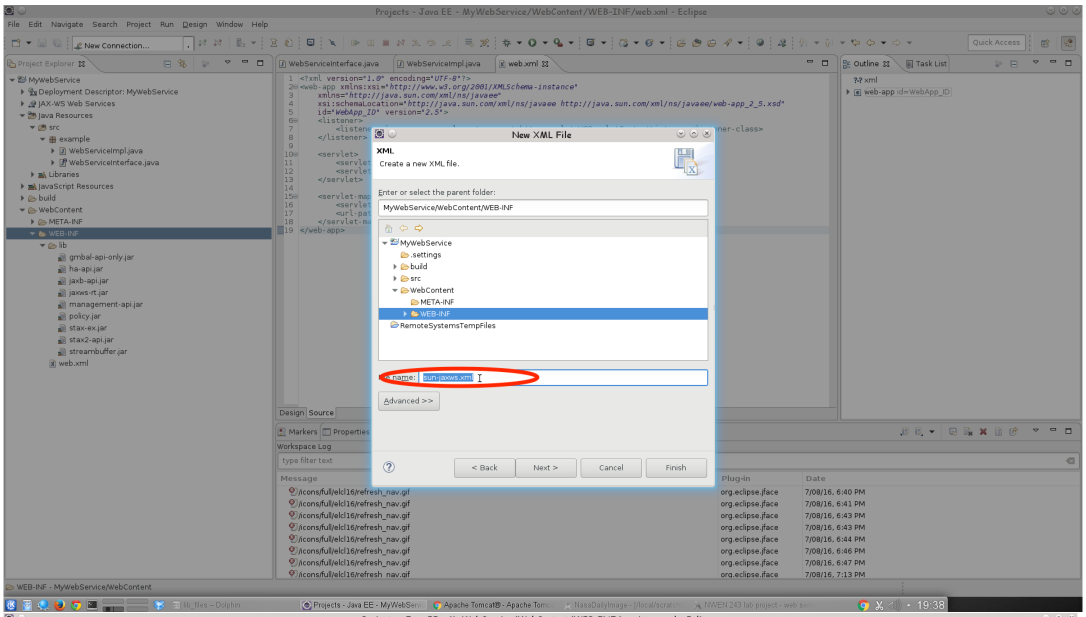
```xml
<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns="http://java.sun.com/xml/ns/javaee"
    xsi:schemaLocation="http://java.sun.com/xml/ns/javaee http://java.sun.com/xml/ns/javaee/web-app_2_5.xsd"
    id="WebApp_ID" version="2.5">
  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

```
Message                                         Plug-in             Date
/icons/full/elcl16/refresh_nav.gif              org.eclipse.jface   7/08/16, 6:35 PM
/icons/full/elcl16/refresh_nav.gif              org.eclipse.jface   7/08/16, 6:40 PM
/icons/full/elcl16/refresh_nav.gif              org.eclipse.jface   7/08/16, 6:41 PM
/icons/full/elcl16/refresh_nav.gif              org.eclipse.jface   7/08/16, 6:43 PM
/icons/full/elcl16/refresh_nav.gif              org.eclipse.jface   7/08/16, 6:43 PM
/icons/full/elcl16/refresh_nav.gif              org.eclipse.jface   7/08/16, 6:44 PM
/icons/full/elcl16/refresh_nav.gif              org.eclipse.jface   7/08/16, 6:46 PM
/icons/full/elcl16/refresh_nav.gif              org.eclipse.jface   7/08/16, 6:47 PM
```





Projects - Java EE - MyWebService/WebContent/WEB-INF/web.xml - Eclipse

```
<?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>
        <listene                                              ner-class>
    </listener>

    <servlet>
        <servlet
        <servlet
    </servlet>

    <servlet-map
        <servlet
        <url-pat
    </servlet-ma
</web-app>
```

New XML File

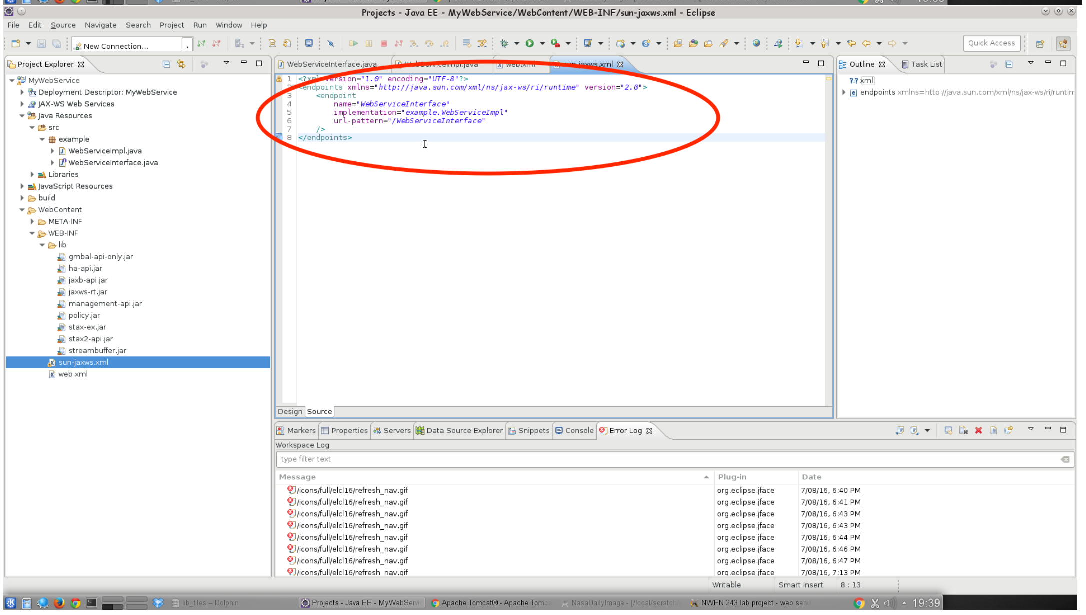
XML

Create a new XML file.

Enter or select the parent folder: MyWebService/WebContent/WEB-INF

File name: sun-jaxws.xml

```
Message                                  Plug-in              Date
/icons/full/elcl16/refresh_nav.gif       org.eclipse.jface    7/08/16, 6:40 PM
/icons/full/elcl16/refresh_nav.gif       org.eclipse.jface    7/08/16, 6:41 PM
/icons/full/elcl16/refresh_nav.gif       org.eclipse.jface    7/08/16, 6:43 PM
/icons/full/elcl16/refresh_nav.gif       org.eclipse.jface    7/08/16, 6:43 PM
/icons/full/elcl16/refresh_nav.gif       org.eclipse.jface    7/08/16, 6:44 PM
/icons/full/elcl16/refresh_nav.gif       org.eclipse.jface    7/08/16, 6:46 PM
/icons/full/elcl16/refresh_nav.gif       org.eclipse.jface    7/08/16, 6:47 PM
/icons/full/elcl16/refresh_nav.gif       org.eclipse.jface    7/08/16, 7:13 PM
```



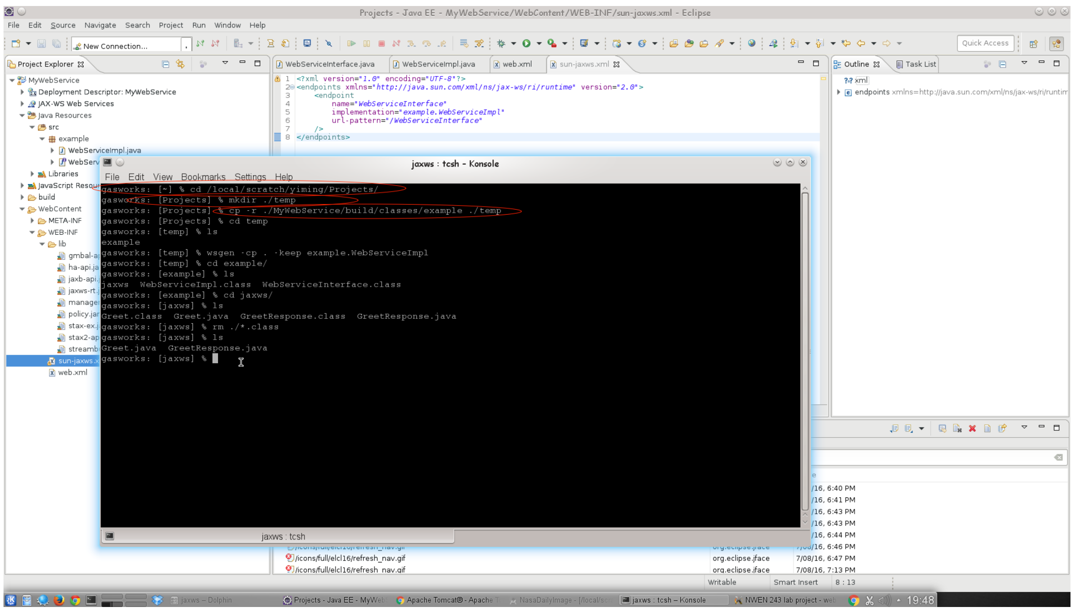
Projects - Java EE - MyWebService/WebContent/WEB-INF/sun-jaxws.xml - Eclipse

```
<?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>
```

```
Message                                  Plug-in              Date
/icons/full/elcl16/refresh_nav.gif       org.eclipse.jface    7/08/16, 6:40 PM
/icons/full/elcl16/refresh_nav.gif       org.eclipse.jface    7/08/16, 6:41 PM
/icons/full/elcl16/refresh_nav.gif       org.eclipse.jface    7/08/16, 6:43 PM
/icons/full/elcl16/refresh_nav.gif       org.eclipse.jface    7/08/16, 6:43 PM
/icons/full/elcl16/refresh_nav.gif       org.eclipse.jface    7/08/16, 6:44 PM
/icons/full/elcl16/refresh_nav.gif       org.eclipse.jface    7/08/16, 6:46 PM
/icons/full/elcl16/refresh_nav.gif       org.eclipse.jface    7/08/16, 6:47 PM
/icons/full/elcl16/refresh_nav.gif       org.eclipse.jface    7/08/16, 7:13 PM
```

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

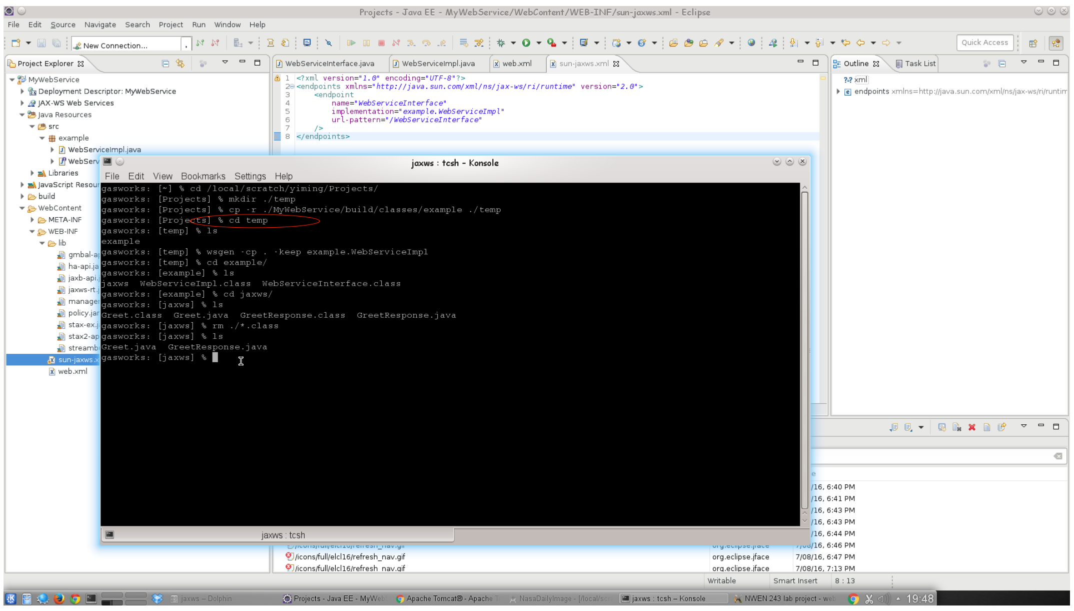
- Copy interface and implementation class files (e.g., WebServiceInterface.class and WebServiceImpl.class) in the directory "/YOUR_PROJECT/build/classes/" to a temp (and originally empty) directory (e.g., ~/temp/). If these class files are located in a package (e.g., example/), you need to copy the package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



```
<?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>
```

```
gasworks: [~] % cd /local/scratch/yiming/Projects/
gasworks: [Projects] % mkdir ./temp
gasworks: [Projects] % cp -r ./MyWebService/build/classes/example ./temp
gasworks: [Projects] % cd temp
gasworks: [temp] % ls
example
gasworks: [temp] % wsgen -cp . -keep example.WebServiceImpl
gasworks: [temp] % cd example/
gasworks: [example] % ls
jaxws  WebServiceImpl.class  WebServiceInterface.class
gasworks: [example] % cd jaxws/
gasworks: [jaxws] % ls
Greet.class  Greet.java  GreetResponse.class  GreetResponse.java
gasworks: [jaxws] % rm ./*.class
gasworks: [jaxws] % ls
Greet.java  GreetResponse.java
gasworks: [jaxws] %
```

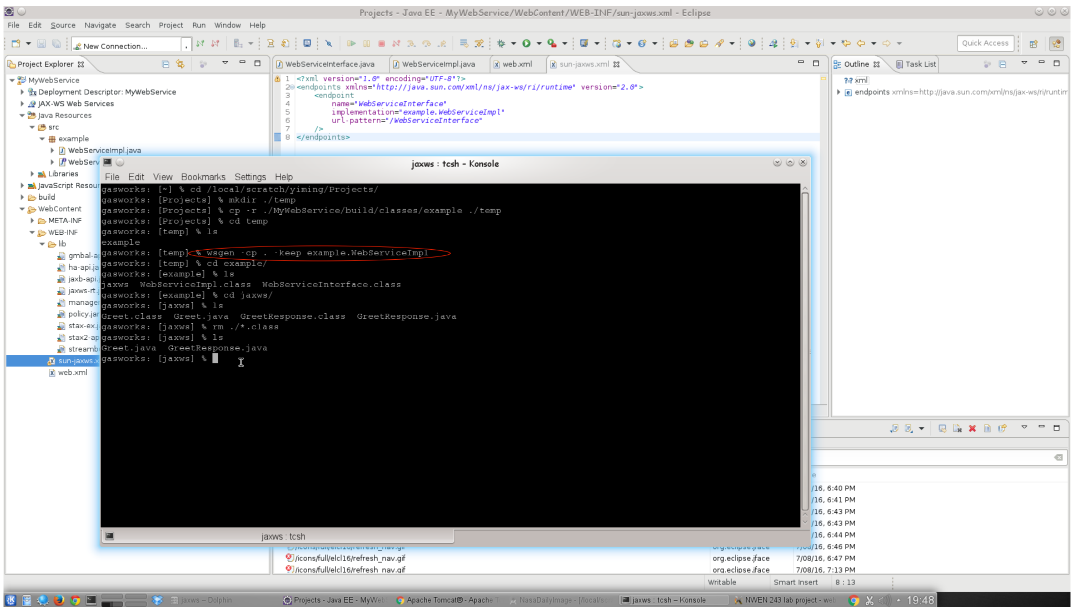
- Change working directory to the temp directory.



```
<?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>
```

```
gasworks: [~] % cd /local/scratch/yiming/Projects/
gasworks: [Projects] % mkdir ./temp
gasworks: [Projects] % cp -r ./MyWebService/build/classes/example ./temp
gasworks: [Projects] % cd temp
gasworks: [temp] % ls
example
gasworks: [temp] % wsgen -cp . -keep example.WebServiceImpl
gasworks: [temp] % cd example/
gasworks: [example] % ls
jaxws  WebServiceImpl.class  WebServiceInterface.class
gasworks: [example] % cd jaxws/
gasworks: [jaxws] % ls
Greet.class  Greet.java  GreetResponse.class  GreetResponse.java
gasworks: [jaxws] % rm ./*.class
gasworks: [jaxws] % ls
Greet.java  GreetResponse.java
gasworks: [jaxws] %
```

- 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**").

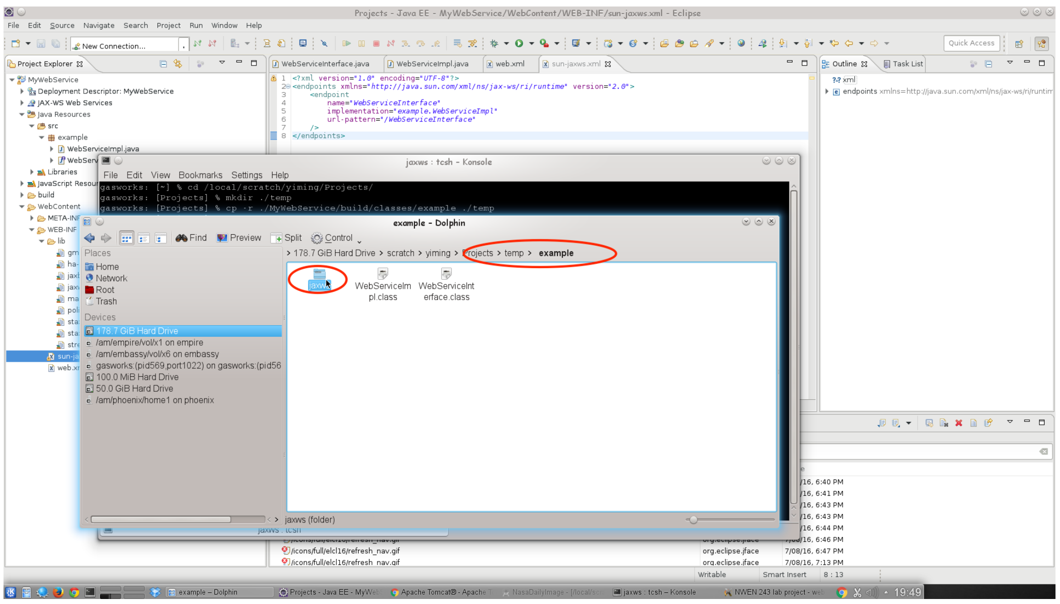


Projects - Java EE - MyWebService/WebContent/WEB-INF/sun-jaxws.xml - Eclipse
jaxws : tcsh - Konsole

```xml
<?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>
```

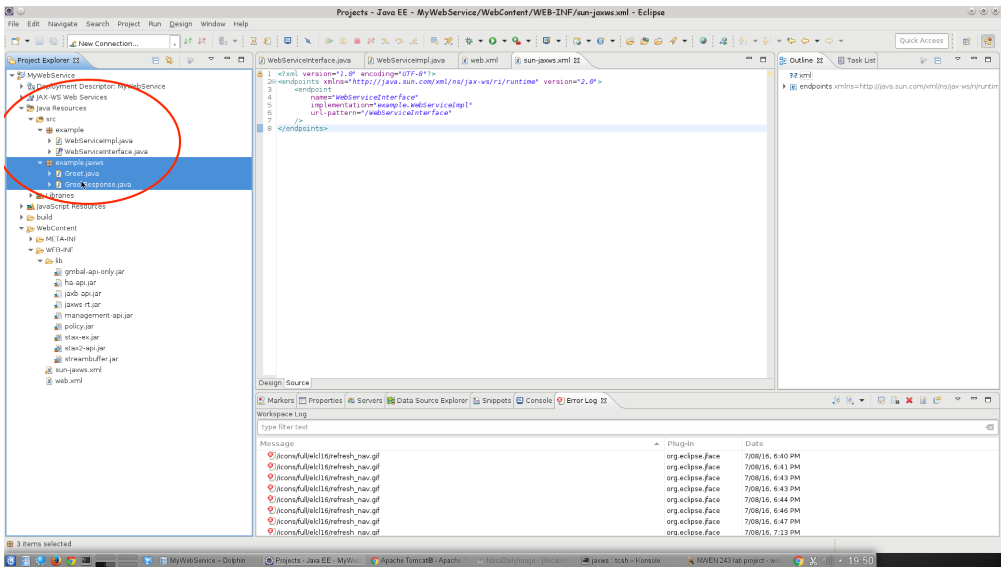
```
gasworks: [~] % cd /local/scratch/yiming/Projects/
gasworks: [Projects] % mkdir ./temp
gasworks: [Projects] % cp -r ./MyWebService/build/classes/example ./temp
gasworks: [Projects] % cd temp
gasworks: [temp] % ls
example
gasworks: [temp] % wsgen -cp . -keep example.WebServiceImpl
gasworks: [temp] % cd example/
gasworks: [example] % ls
jaxws  WebServiceImpl.class  WebServiceInterface.class
gasworks: [example] % cd jaxws/
gasworks: [jaxws] % ls
Greet.class  Greet.java  GreetResponse.class  GreetResponse.java
gasworks: [jaxws] % rm ./*.class
gasworks: [jaxws] % ls
Greet.java  GreetResponse.java
gasworks: [jaxws] %
```

- 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/”.



Projects - Java EE - MyWebService/WebContent/WEB-INF/sun-jaxws.xml - Eclipse
jaxws : tcsh - Konsole
example – Dolphin
Places
Home
Network
Root
Trash
Devices
178.7 GiB Hard Drive
> 178.7 GiB Hard Drive > scratch > yiming > Projects > temp > example
jaxws
WebServiceImpl.class
WebServiceInterface.class
jaxws (folder)

```
gasworks: [~] % cd /local/scratch/yiming/Projects/
gasworks: [Projects] % mkdir ./temp
gasworks: [Projects] % cp -r ./MyWebService/build/classes/example ./temp
```

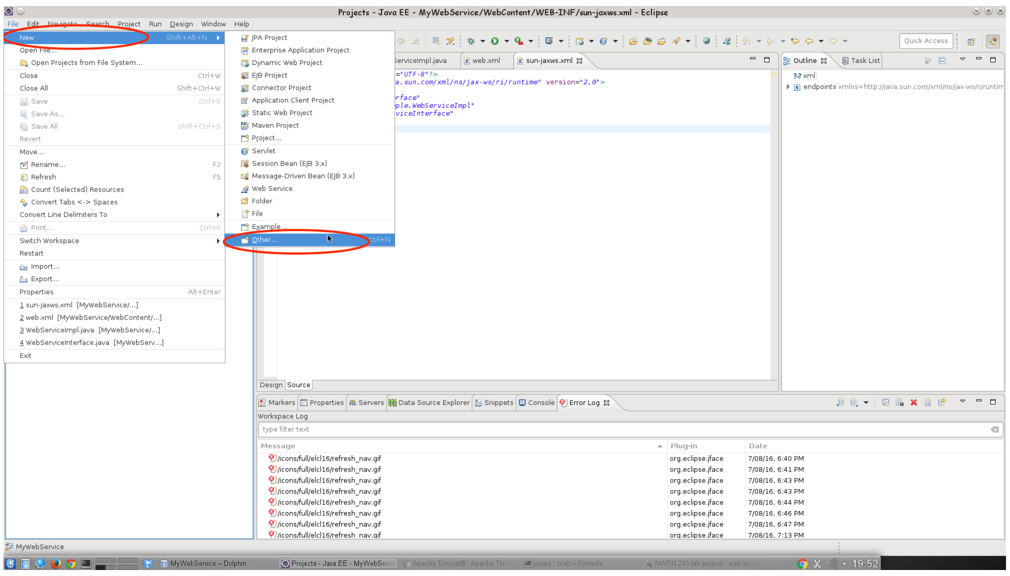


```xml
<?xml version="1.0" encoding="UTF-8"?>
<endpoints xmlns="http://java.sun.com/xml/ns/jax-ws/ri/runtime" version="2.0">
    <endpoint
        name="WebServiceInterface"
        implementation="example.WebServiceImpl"
        url-pattern="/WebServiceInterface"
    />
</endpoints>
```

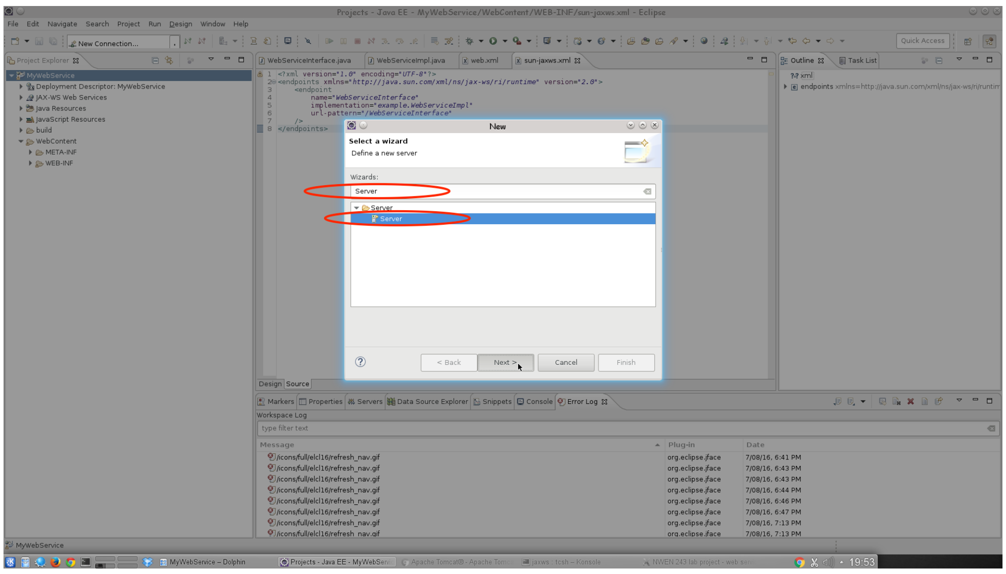
```
Message                                  Plug-in             Date
/icons/full/elcl16/refresh_nav.gif       org.eclipse.jface   7/08/16, 6:40 PM
/icons/full/elcl16/refresh_nav.gif       org.eclipse.jface   7/08/16, 6:41 PM
/icons/full/elcl16/refresh_nav.gif       org.eclipse.jface   7/08/16, 6:43 PM
/icons/full/elcl16/refresh_nav.gif       org.eclipse.jface   7/08/16, 6:43 PM
/icons/full/elcl16/refresh_nav.gif       org.eclipse.jface   7/08/16, 6:44 PM
/icons/full/elcl16/refresh_nav.gif       org.eclipse.jface   7/08/16, 6:46 PM
/icons/full/elcl16/refresh_nav.gif       org.eclipse.jface   7/08/16, 6:47 PM
/icons/full/elcl16/refresh_nav.gif       org.eclipse.jface   7/08/16, 7:13 PM
```

Step 10: Define Apache Tomcat v6.0 Server.

- Create a new server



```
Message                                  Plug-in             Date
/icons/full/elcl16/refresh_nav.gif       org.eclipse.jface   7/08/16, 6:40 PM
/icons/full/elcl16/refresh_nav.gif       org.eclipse.jface   7/08/16, 6:41 PM
/icons/full/elcl16/refresh_nav.gif       org.eclipse.jface   7/08/16, 6:43 PM
/icons/full/elcl16/refresh_nav.gif       org.eclipse.jface   7/08/16, 6:43 PM
/icons/full/elcl16/refresh_nav.gif       org.eclipse.jface   7/08/16, 6:44 PM
/icons/full/elcl16/refresh_nav.gif       org.eclipse.jface   7/08/16, 6:46 PM
/icons/full/elcl16/refresh_nav.gif       org.eclipse.jface   7/08/16, 6:47 PM
/icons/full/elcl16/refresh_nav.gif       org.eclipse.jface   7/08/16, 7:13 PM
```


Projects - Java EE - MyWebService/WebContent/WEB-INF/sun-jaxws.xml - Eclipse

```
<?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>
```

New

Select a wizard
Define a new server

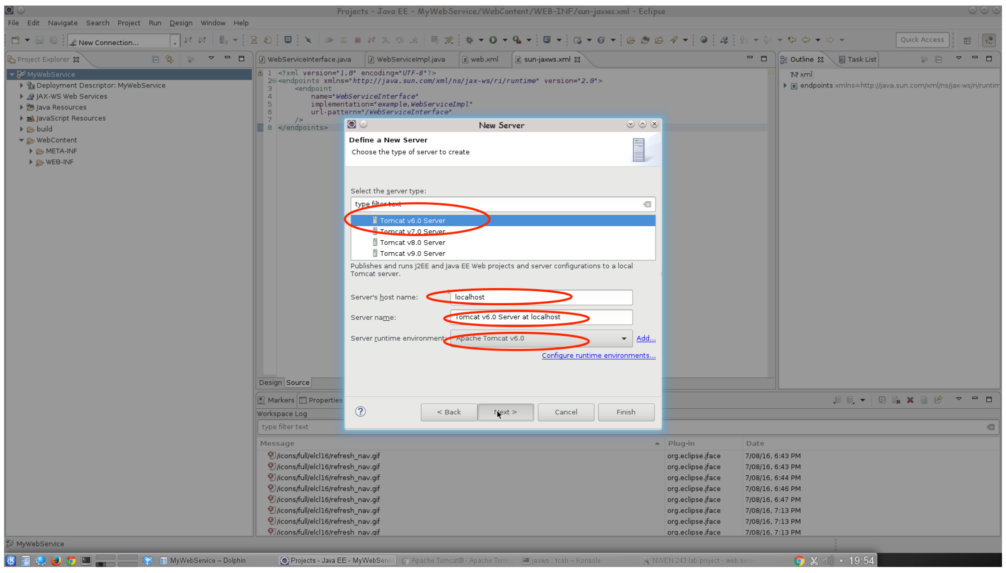
Wizards: Server

Server
    Server

< Back | Next > | Cancel | Finish

File Edit Navigate Search Project Run Design Window Help
Project Explorer
MyWebService
Deployment Descriptor: MyWebService
JAX-WS Web Services
Java Resources
JavaScript Resources
build
WebContent
META-INF
WEB-INF
WebServiceInterface.java
WebServiceImpl.java
web.xml
sun-jaxws.xml
Outline
Task List
endpoints xmlns=http://java.sun.com/xml/ns/jax-ws/ri/runtim
Markers Properties Servers Data Source Explorer Snippets Console Error Log
Workspace Log
Message
Plug-in
Date
/icons/full/elcl16/refresh_nav.gif
org.eclipse.jface
19:53

- Setup the server


New Server

Define a New Server
Choose the type of server to create

Select the server type: type filter text

Tomcat v6.0 Server
Tomcat v7.0 Server
Tomcat v8.0 Server
Tomcat v9.0 Server

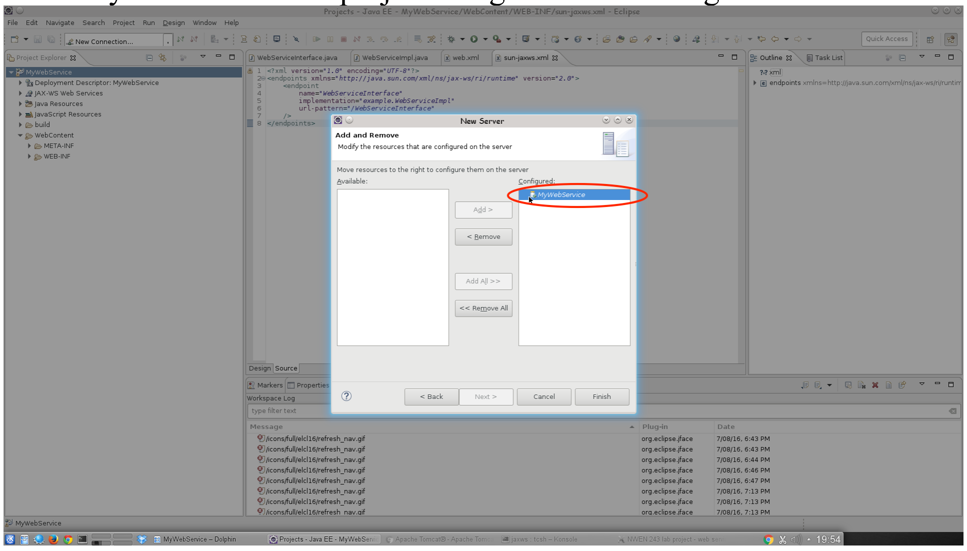
Publishes and runs J2EE and Java EE Web projects and server configurations to a local Tomcat server.

Server's host name: localhost
Server name: Tomcat v6.0 Server at localhost
Server runtime environment: Apache Tomcat v6.0 Add...
Configure runtime environments...

< Back | Next > | Cancel | Finish

Projects - Java EE - MyWebService/WebContent/WEB-INF/sun-jaxws.xml - Eclipse
19:54

- Move your WebService project to right side - “Configured”.


New Server

Add and Remove
Modify the resources that are configured on the server

Move resources to the right to configure them on the server
Available:
Configured:
MyWebService

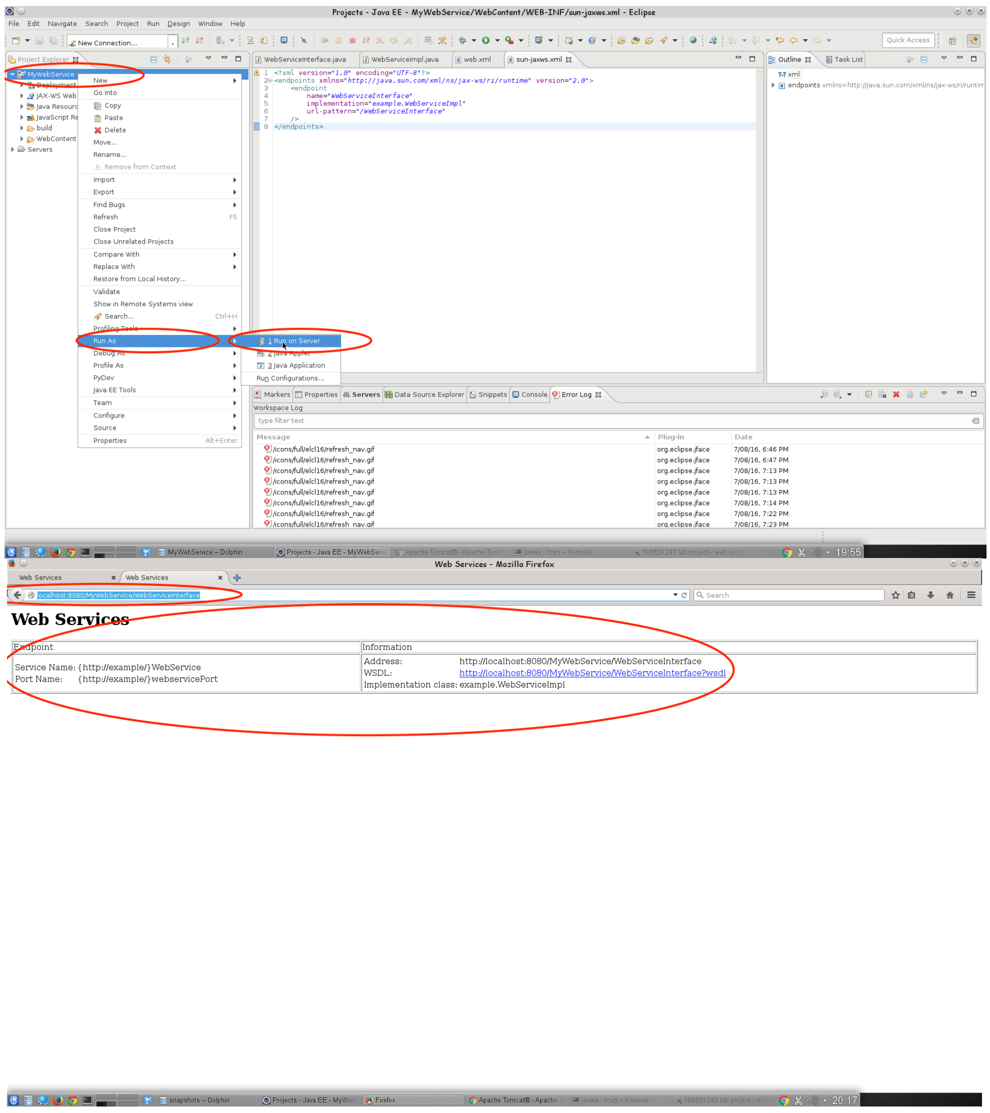
Add > | < Remove | Add All >> | << Remove All

< Back | Next > | Cancel | Finish

Projects - Java EE - MyWebService/WebContent/WEB-INF/sun-jaxws.xml - Eclipse
19:54

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!



```
<?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>
```

```
Message                                  Plug-in             Date
/icons/full/elcl16/refresh_nav.gif       org.eclipse.jface   7/08/16, 6:46 PM
/icons/full/elcl16/refresh_nav.gif       org.eclipse.jface   7/08/16, 6:47 PM
/icons/full/elcl16/refresh_nav.gif       org.eclipse.jface   7/08/16, 7:13 PM
/icons/full/elcl16/refresh_nav.gif       org.eclipse.jface   7/08/16, 7:13 PM
/icons/full/elcl16/refresh_nav.gif       org.eclipse.jface   7/08/16, 7:13 PM
/icons/full/elcl16/refresh_nav.gif       org.eclipse.jface   7/08/16, 7:14 PM
/icons/full/elcl16/refresh_nav.gif       org.eclipse.jface   7/08/16, 7:22 PM
/icons/full/elcl16/refresh_nav.gif       org.eclipse.jface   7/08/16, 7:23 PM
```

Web Services

Endpoint	Information
Service Name: {http://example/}WebService Port Name: {http://example/}webservicePort	Address: http://localhost:8080/MyWebService/WebServiceInterface WSDL: http://localhost:8080/MyWebService/WebServiceInterface?wsdl Implementation class: example.WebServiceImpl

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.

```
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
		URLurl=newURL("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("Serviceoutput:\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

It is recommended (but not compulsory) for you to demo your work to your lab tutor by the end of week 12 (the last teaching week). This involves running the Web Service provider successfully and running the Web Service requester that will display the desirable response received from the service provider.

You also need to 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. Please refer to the submission link for the online submission deadline.

No technical report is required for this lab. However you need to submit a PDF document that includes your answers to all topic questions (topic questions are provided through a separate PDF document retrievable from the lab project page of our course website).

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.