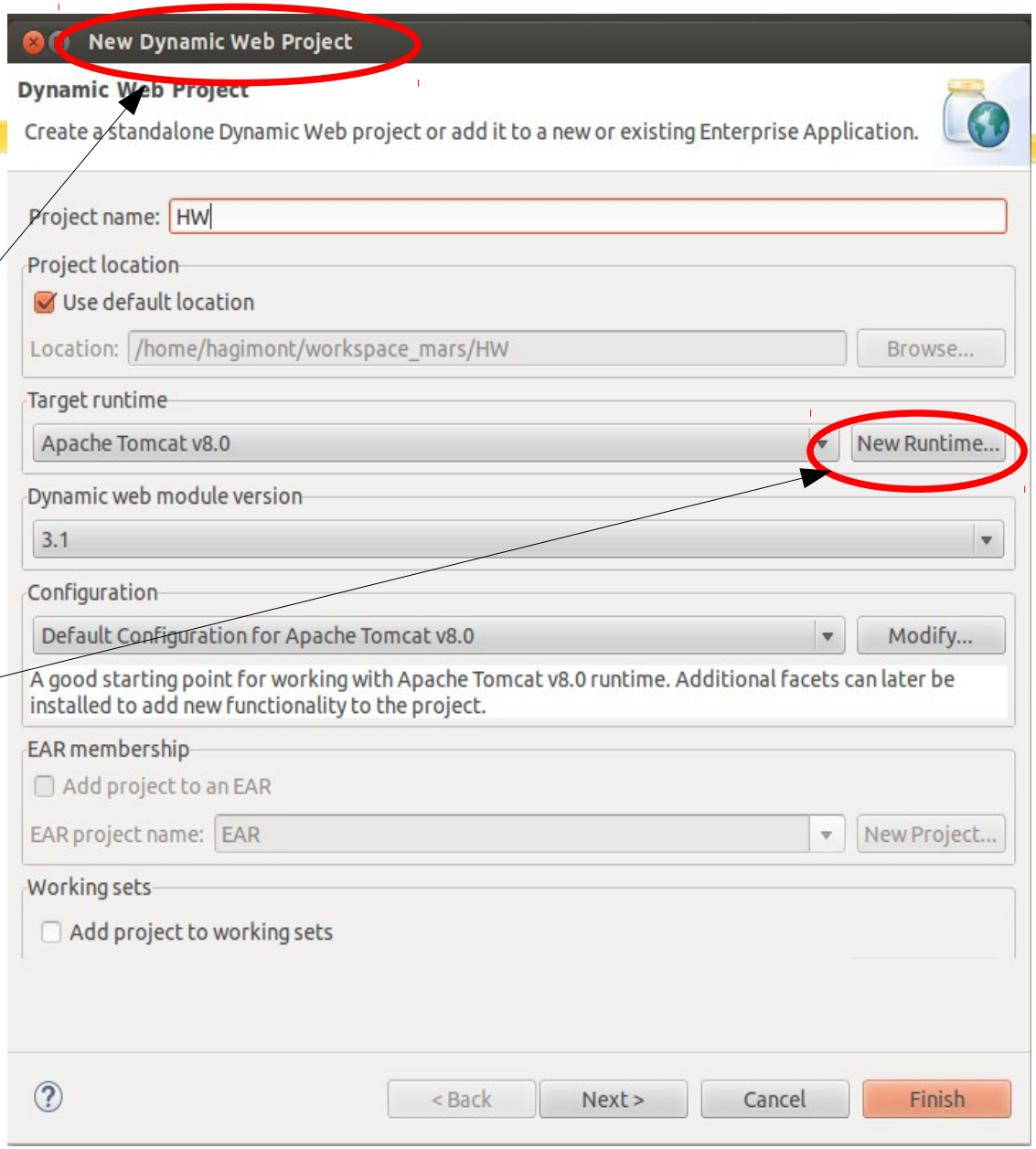
Creating a SOAP WS

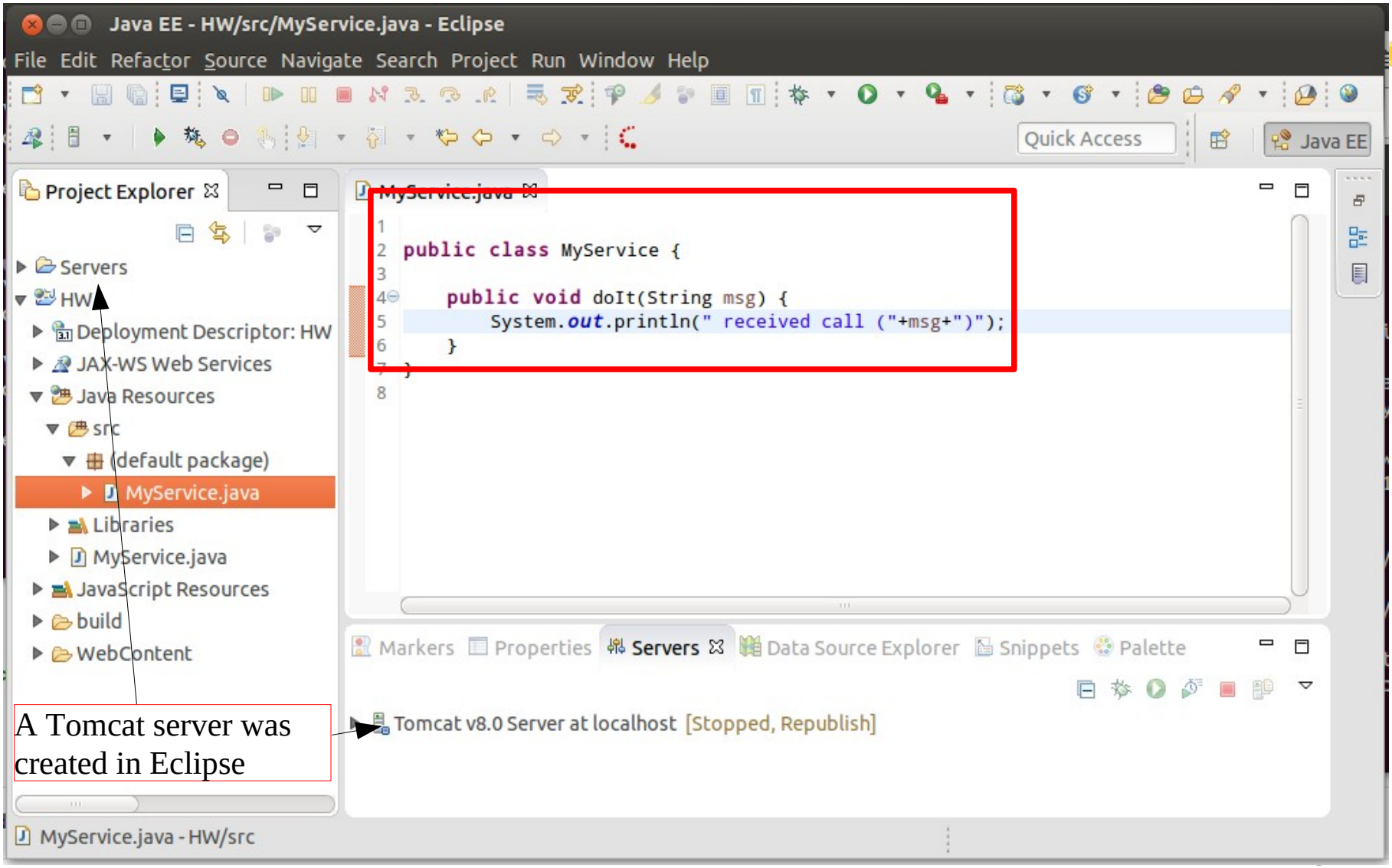
- Eclipse JEE
- Apache Axis
- Creation of a Web Service
  - From a Java class
  - In the Tomcat runtime
  - Generation of the WSDL file
- Creation of a client application
  - Generation of stubs from a WSDL file
  - Programming of the client

Create a Dynamic Web Project

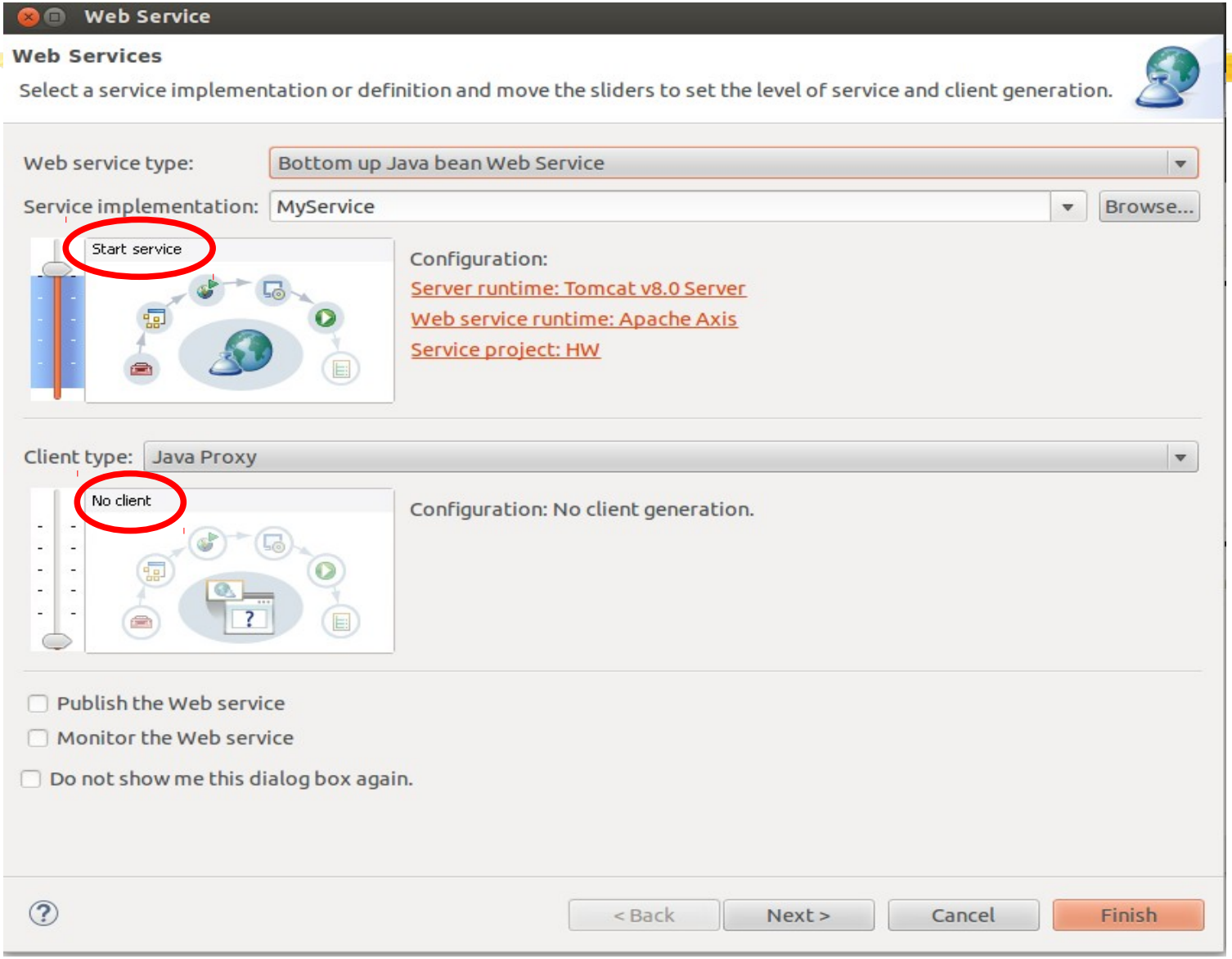
- Eclipse JEE
- Open JEE perspective
- Create a Dynamic Web Project
- Add your Tomcat runtime



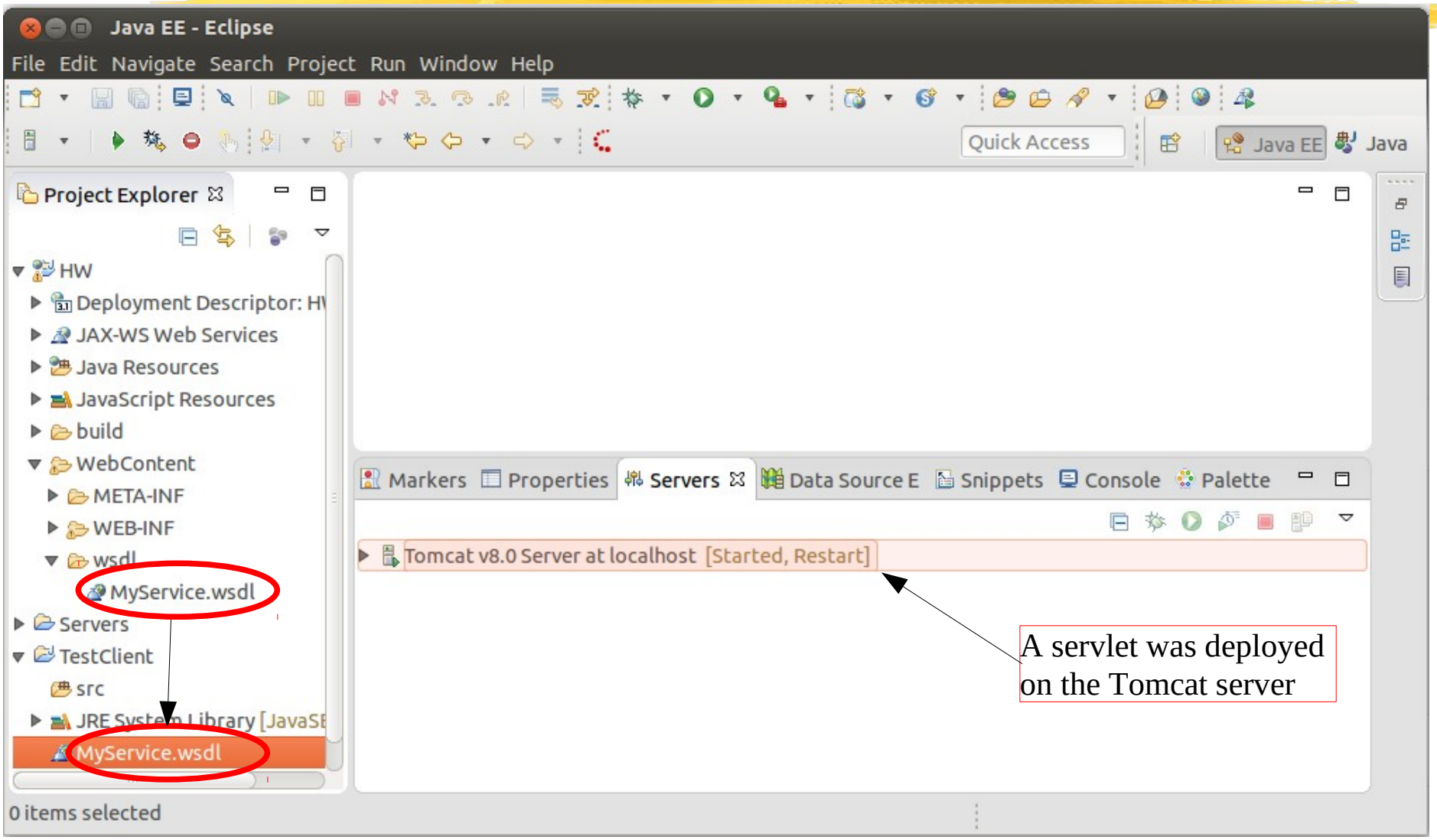
Create a Class



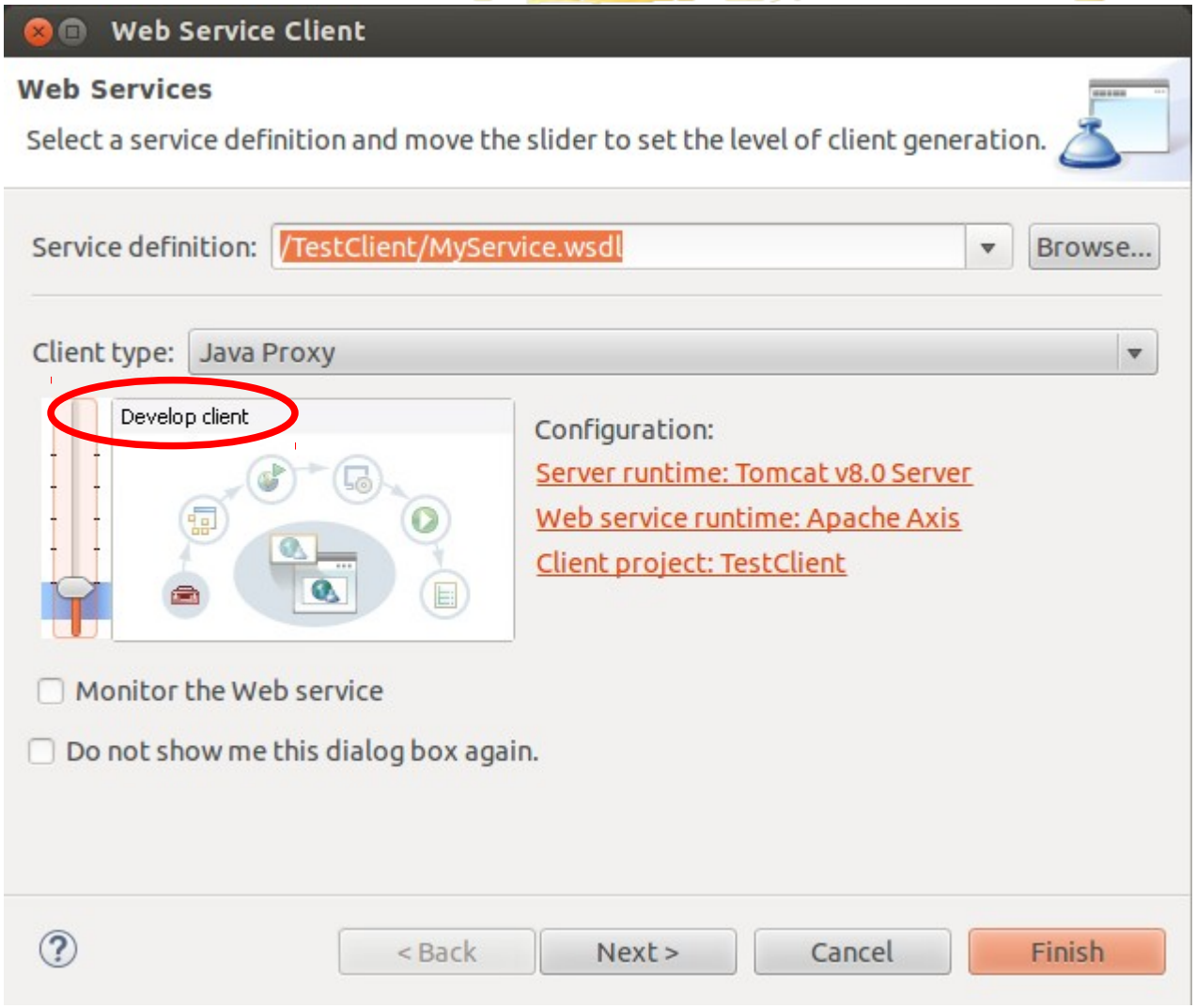
From source file : Web Service → create Web Service



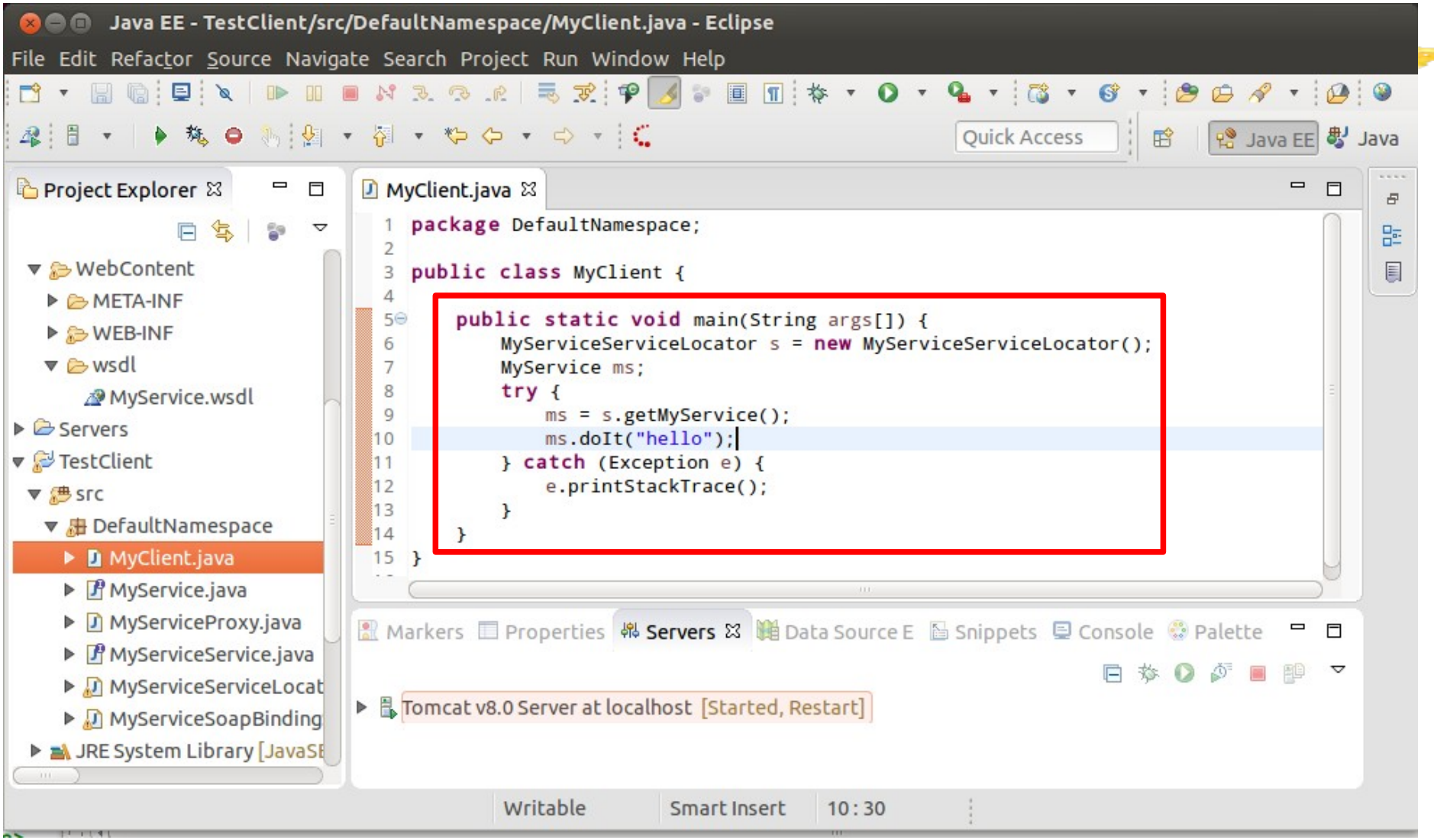
Copy the generated WSDL file in a new Java project



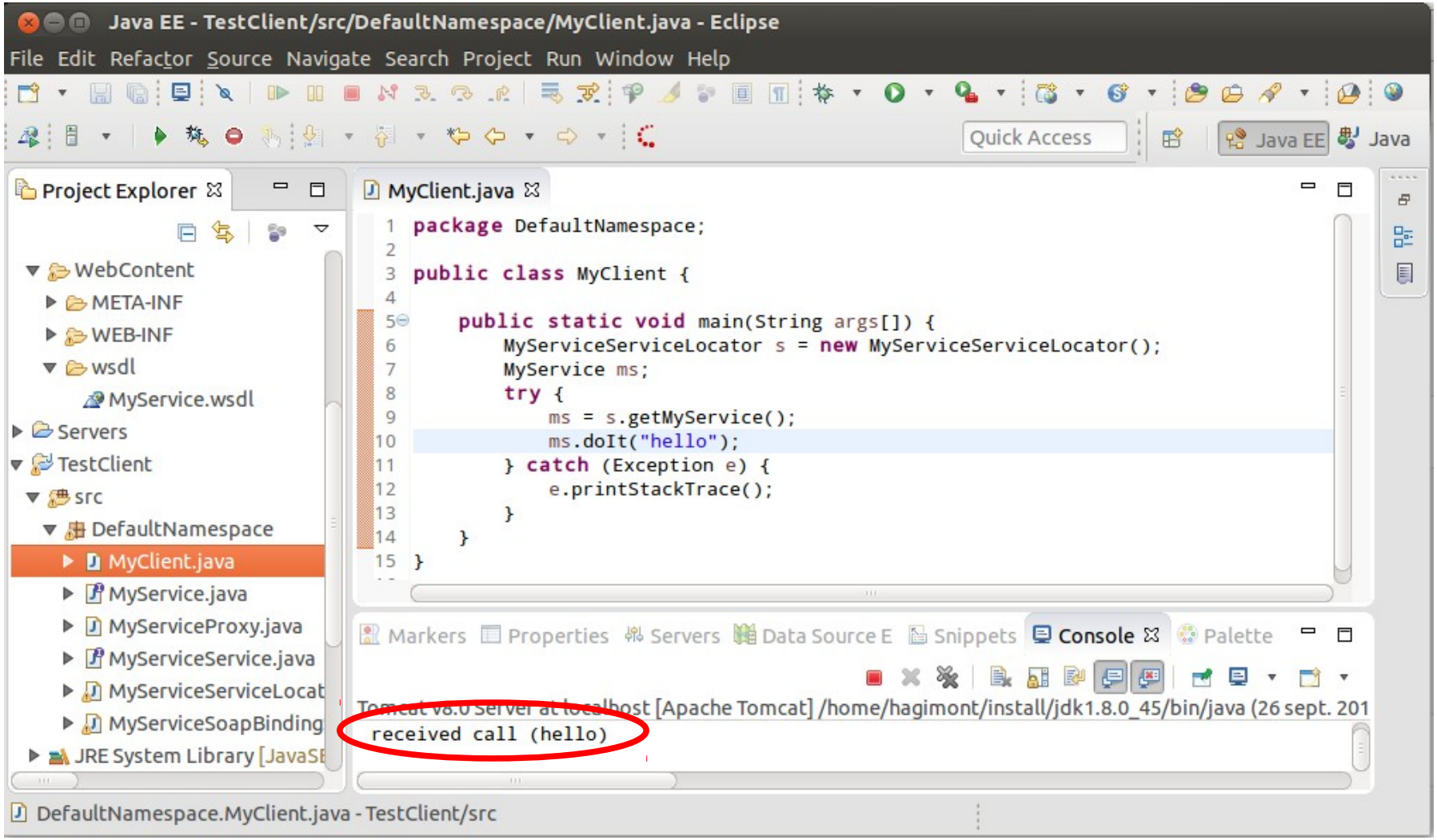
From the WSDL file
Web Service → Generate Client (Develop Client)



Program a client



Run


Java EE - TestClient/src/DefaultNamespace/MyClient.java - Eclipse
File Edit Refactor Source Navigate Search Project Run Window Help
Quick Access
Java EE
Java
Project Explorer
WebContent
META-INF
WEB-INF
wsdl
MyService.wsdl
Servers
TestClient
src
DefaultNamespace
MyClient.java
MyService.java
MyServiceProxy.java
MyServiceService.java
MyServiceServiceLocat
MyServiceSoapBinding
JRE System Library [JavaSE
MyClient.java
1 package DefaultNamespace;
2
3 public class MyClient {
4
5 public static void main(String args[]) {
6 MyServiceServiceLocator s = new MyServiceServiceLocator();
7 MyService ms;
8 try {
9 ms = s.getMyService();
10 ms.doIt("hello");
11 } catch (Exception e) {
12 e.printStackTrace();
13 }
14 }
15 }
Markers
Properties
Servers
Data Source E
Snippets
Console
Palette
Tomcat v8.0 Server at localhost [Apache Tomcat] /home/hagimont/install/jdk1.8.0_45/bin/java (26 sept. 201
received call (hello)
DefaultNamespace.MyClient.java - TestClient/src

Creating a REST WS with resteasy

- WS class

```
@Path("/rest")
public class Facade {

    static Hashtable<String, Person> ht = new Hashtable<String, Person>();

    @POST
    @Path("/addperson")
    @Consumes({ "application/json" })
    public Response addPerson(Person p) {
         ht.put(p.getId(), p);
         return Response.status(201).entity("person added").build();
    }

    @GET
    @Path("/getperson")
    @Produces({ "application/json" })
    public Person getPerson(@QueryParam("id") String id) {
         return ht.get(id);
    }

    @GET
    @Path("/listpersons")
    @Produces({ "application/json" })
    public Collection<Person> listPersons() {
         return ht.values();
    }

}
```

Receives a JSON
Deserialized into a Java object
void method

Returns an object
Serialized into a JSON
Receives an id HTTP parameter

Person is a simple POJO

Creating a REST WS with resteasy

- Add the RestEasy jars in Tomcat
- In eclipse
  - Create a Dynamic Web Project
  - Add RestEasy jars in the buildpath
  - Create a package
  - Implement the WS class (Facade + Person)
  - Add a class RestApp

```
public class RestApp extends Application {
          private Set<Object> singletons = new HashSet<Object>();
          public RestApp() {
                    singletons.add(new Facade());
          }
          public Set<Object> getSingletons() {
                    return singletons;
          }
}
```

Creating a REST WS with resteasy

- Add a web.xml descriptor in the WebContent/WEB-INF folder

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xmlns="http://xmlns.jcp.org/xml/ns/javaee"
xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/javaee
http://xmlns.jcp.org/xml/ns/javaee/web-app_3_1.xsd" version="3.1">
  <display-name>essai-server</display-name>
  <servlet>
    <servlet-name>resteasy-servlet</servlet-name>
    <servlet-class>
            org.jboss.resteasy.plugins.server.servlet.HttpServletDispatcher
        </servlet-class>
    <init-param>
      <param-name>javax.ws.rs.Application</param-name>
      <param-value>pack.RestApp</param-value>
    </init-param>
  </servlet>
  <servlet-mapping>
    <servlet-name>resteasy-servlet</servlet-name>
    <url-pattern>/rest/*</url-pattern>
  </servlet-mapping>
</web-app>
```

- Export the war in Tomcat

Creating a REST WS with resteasy

- Just write a documentation which says that
  - The WS is available at http://localhost:8080/rs-server-person/rest
  - Method addperson with POST receives a person JSON :

```
{
  "id":"00000",
   "firstname":"Alain",
   "lastname":"Tchana",
   "phone":"0102030405",
   "email":"alain.tchana@enseeiht.fr"
}
```

  - Method getperson with GET receives an HTTP parameter id and returns a person JSON
  - Method listperson returns a JSON including a set of persons
- A user may use any tool (not only RestEasy)

Invoking a REST WS with resteasy

- From the previous documentation, a client can write the interface

```
@Path("/rest")
public interface FacadeInterface {

    @POST
    @Path("/addperson")
    @Consumes({ "application/json" })
    public Response addPerson(Person p);

    @GET
    @Path("/getperson")
    @Produces({ "application/json" })
    public Person getPerson(@QueryParam("id") String id);

    @GET
    @Path("/listpersons")
    @Produces({ "application/json" })
    public Collection<Person> listPersons();

}
```

Invoking a REST WS with resteasy

- And write a class which invokes the WS

```
public class Client {

   public static void main(String args[]) {

       final String path = "http://localhost:8080/rs-server-person";

       ResteasyClient client = new ResteasyClientBuilder().build();
       ResteasyWebTarget target = client.target(UriBuilder.fromPath(path));
       FacadeInterface proxy = target.proxy(FacadeInterface.class);

       Response resp;
       resp = proxy.addPerson(new Person("007","James Bond"));
       System.out.println("HTTP code: " + resp.getStatus()
                                  +"  message: "+resp.readEntity(String.class));
       resp.close();
       resp = proxy.addPerson(new Person("006","Dan Hagi"));
       System.out.println("HTTP code: " + resp.getStatus()
                                  +"  message: "+resp.readEntity(String.class));
       resp.close();

       Collection<Person> l = proxy.listPersons();
       for (Person p : l) System.out.println("list Person: "+p.getId()+"/"+p.getName());

       Person p = proxy.getPerson("006");
       System.out.println("get Person: "+p.getId()+"/"+p.getName());
   }

}
```

Invoking a REST WS with resteasy

- In eclipse
  - Create a Java Project
  - Add RestEasy jars in the buildpath
  - Implement the Java bean that correspond to the JSON
    - Automatic generation with https://www.site24x7.com/tools/json-to-java.html
  - Implement the interface and the client class (FacadeInterface + Client)
  - Run