Developing a Hello World Blueprint application

Developing Aries applications

This application is a simple Hello World Blueprint application which shows basic principles of building an Aries application and definitions of bean, service and reference in Blueprint.

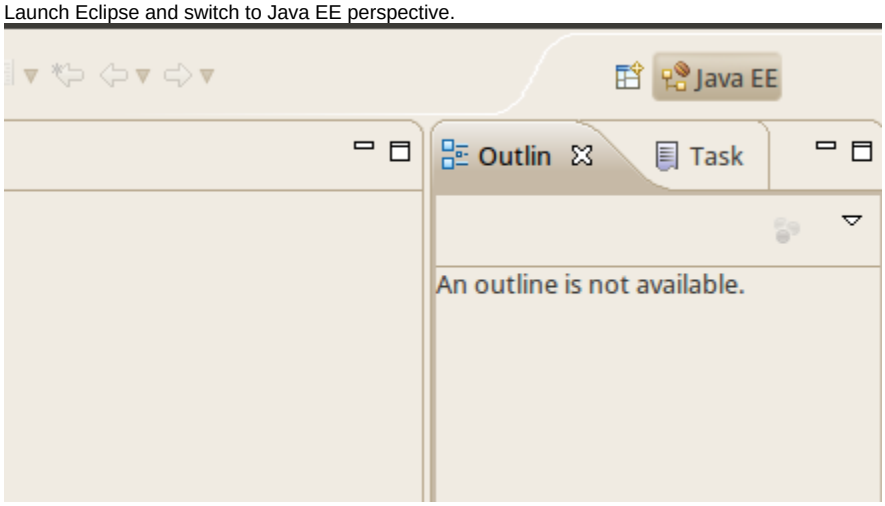
To run this tutorial, as a minimum you will be required to have installed the following prerequisite software:

1. SUN JDK 6.0+(J2SE 1.6)
2. Eclipse IDS for Java EE Developers, which is platform specific
3. Apache Geronimo Eclipse Plugin v3.0
4. OSGi Application Development Tools
5. Apache Geronimo Server v3.0

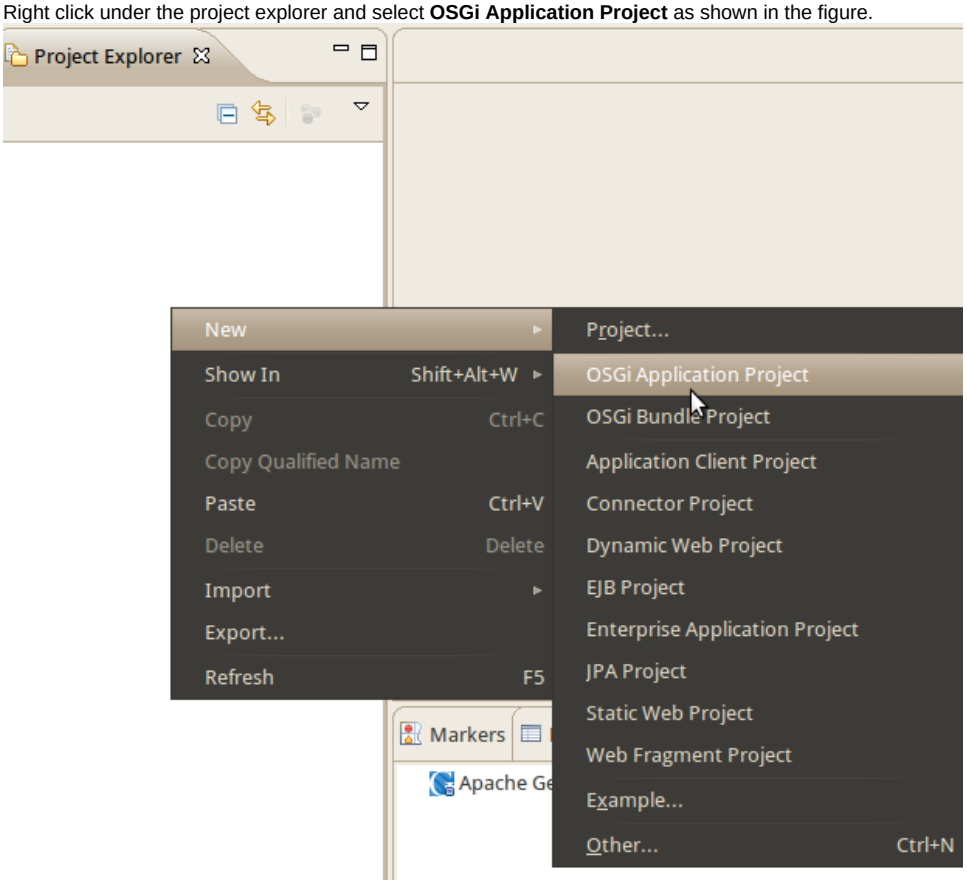
Details on installing Eclipse are provided in the Development environment section. This tutorial is organized in the following sections:

Creating an OSGi Enterprise Application using Eclipse

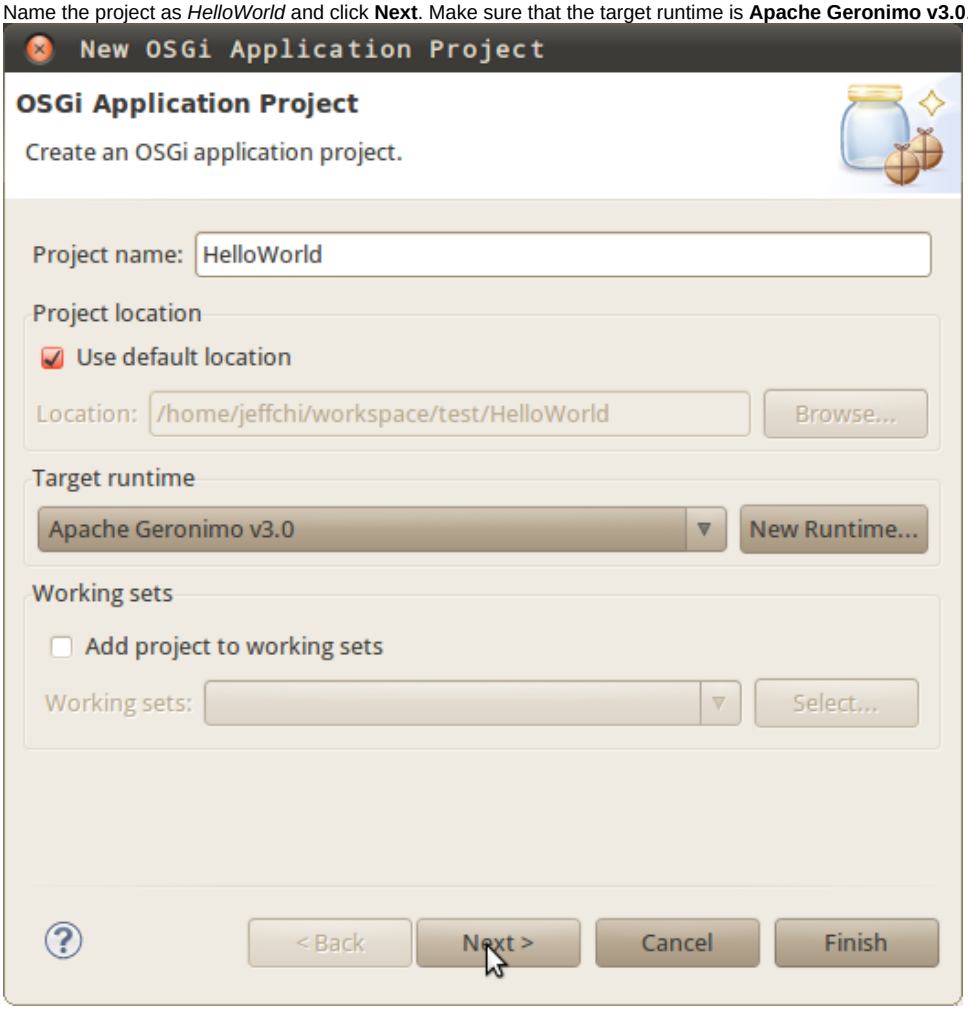
1. Launch Eclipse and switch to Java EE perspective.



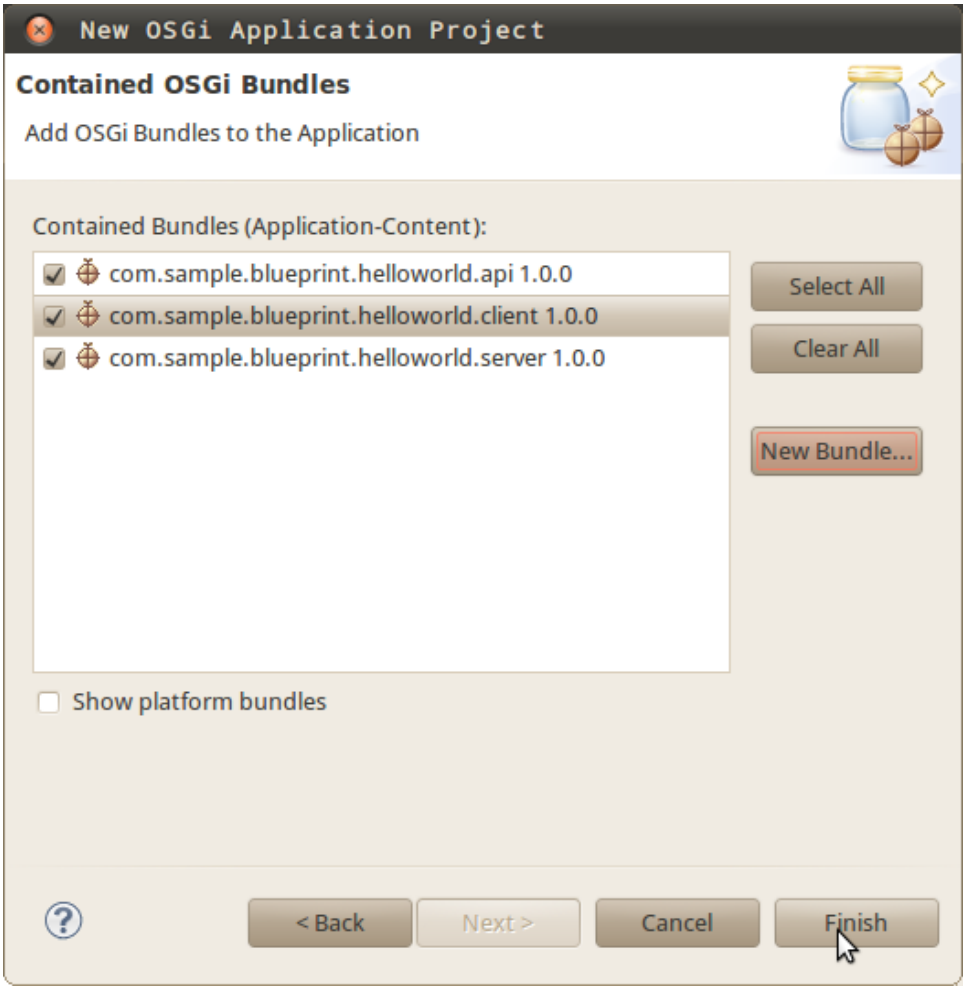
2. Right click under the project explorer and select **OSGi Application Project** as shown in the figure.



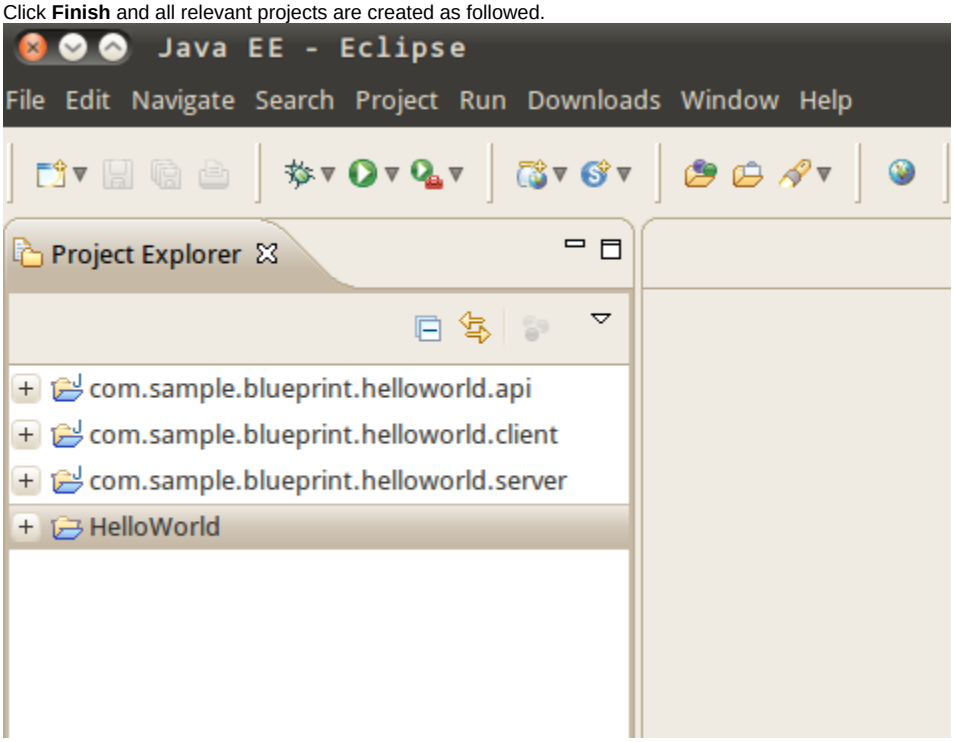
3. Name the project as *HelloWorld* and click **Next**. Make sure that the target runtime is **Apache Geronimo v3.0**.



4. Click **New Bundle...** to define api, client and server bundles to be included in this application. Note that interface and implementation classes should be kept in separate bundles so that the implementations could be replaced independently of their interfaces. Select **Change the active Target Platform** if necessary.

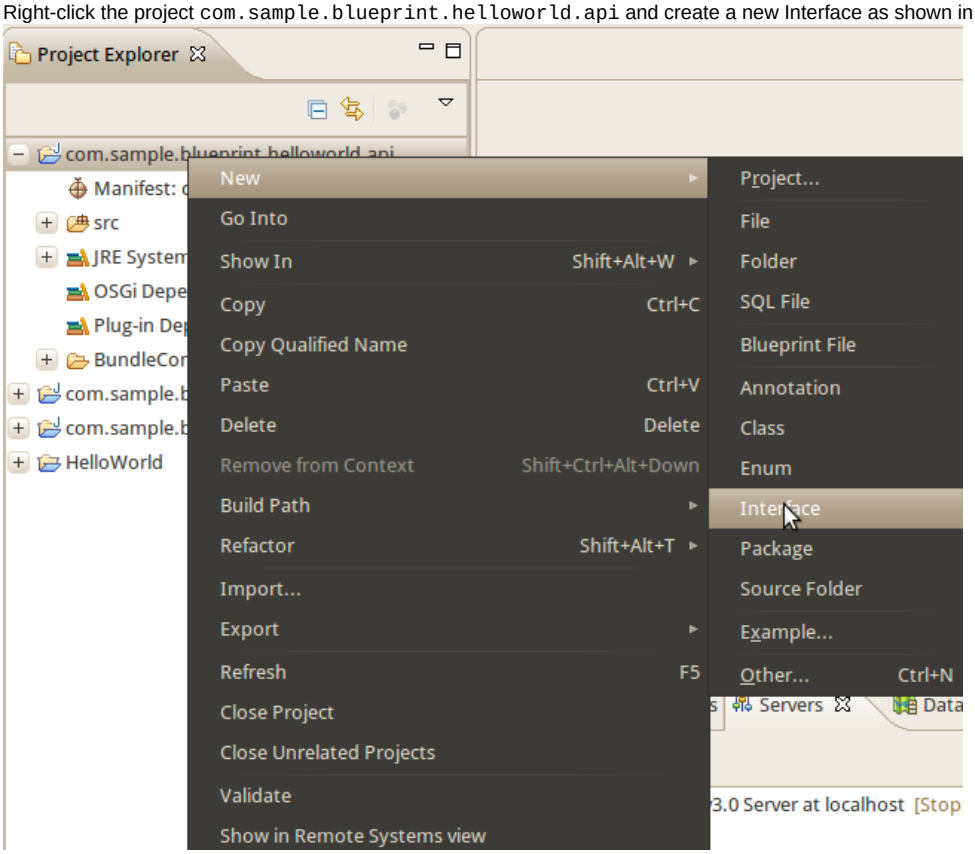


5. Click **Finish** and all relevant projects are created as followed.

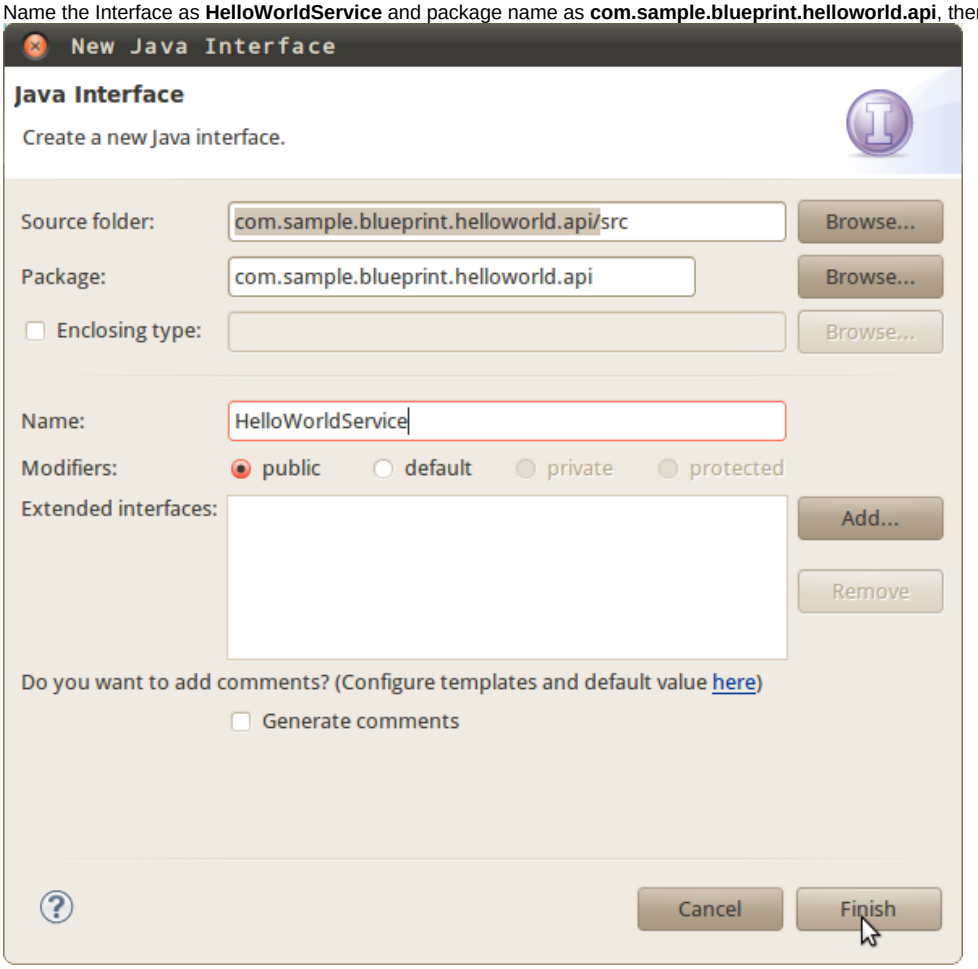


Adding a class to the api project

1. Right-click the project `com.sample.blueprint.helloworld.api` and create a new Interface as shown in the figure.



2. Name the Interface as **HelloWorldService** and package name as **com.sample.blueprint.helloworld.api**, then click **Finish**.

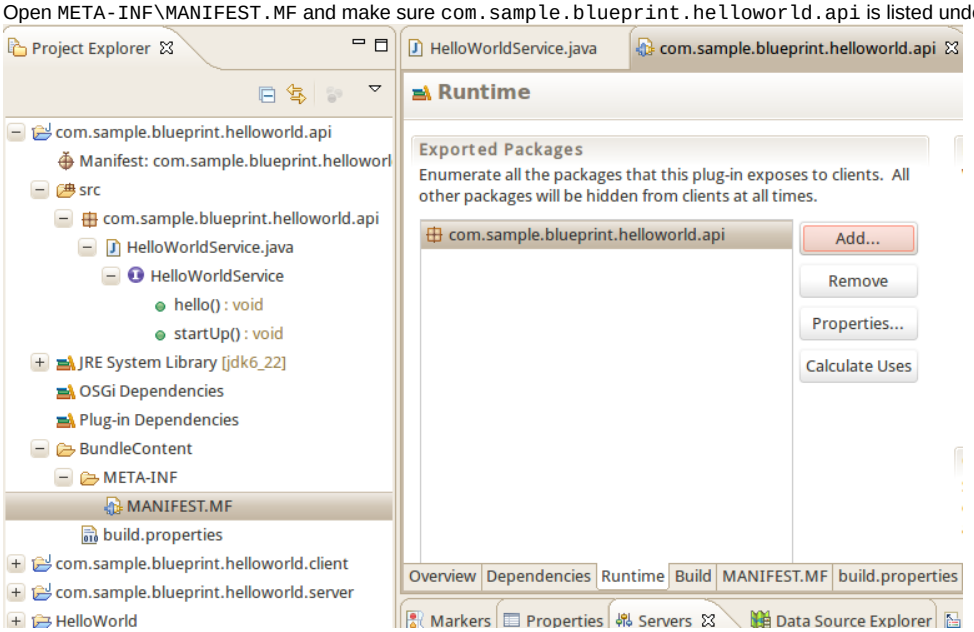


3. Modify the code of `HelloWorldService` as follows.

HelloWorldService.java

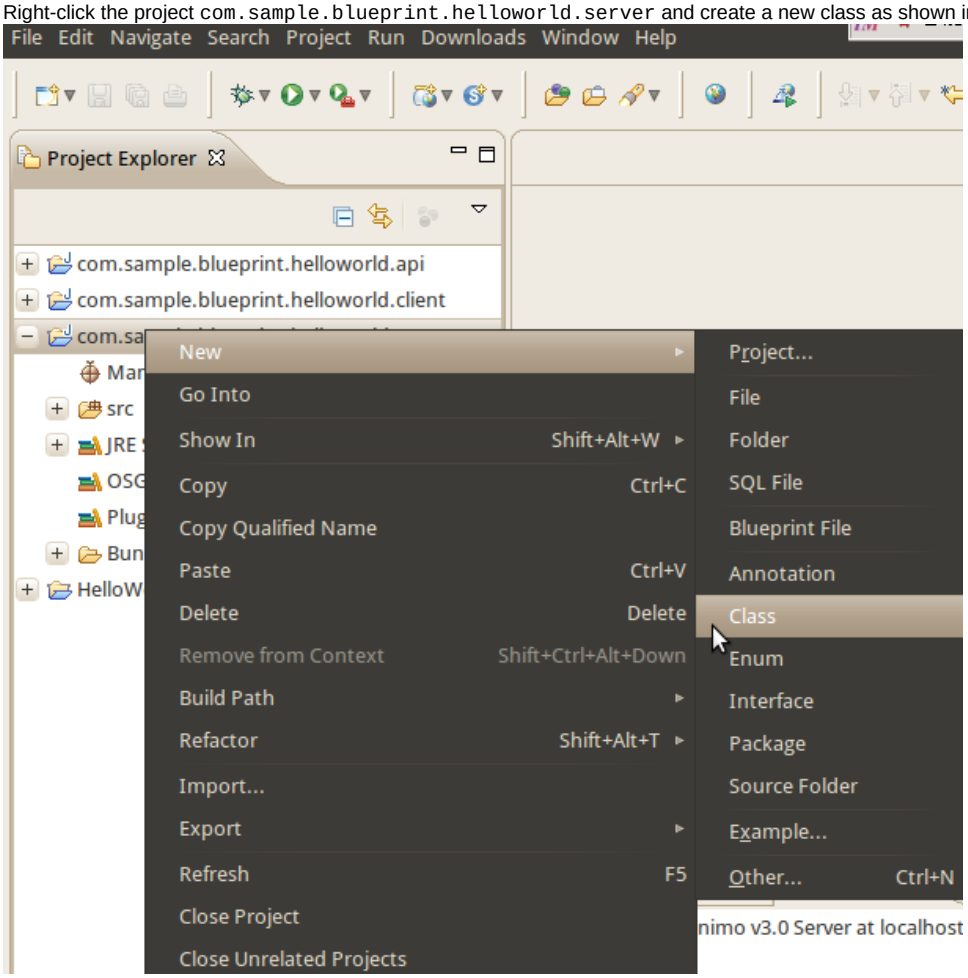
```
package com.sample.blueprint.helloworld.api;

public interface HelloWorldService {

        public void hello();

        public void startUp();

}
```

4. Open `META-INF\MANIFEST.MF` and make sure `com.sample.blueprint.helloworld.api` is listed under **Exported packages**.

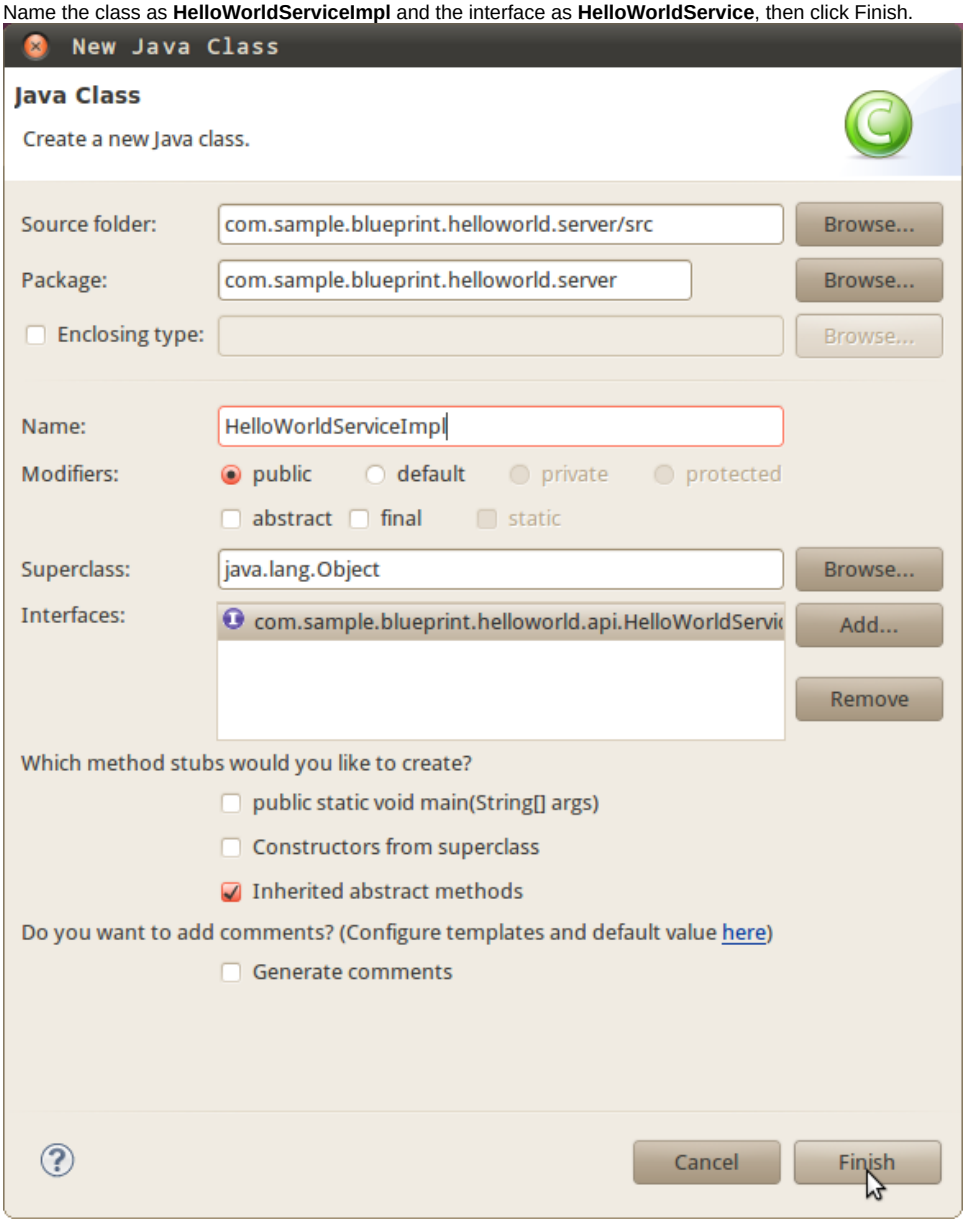


Implementing the HelloWorldService inteface

1. Right-click the project `com.sample.blueprint.helloworld.server` and create a new class as shown in the figure.



2. Name the class as **HelloWorldServiceImpl** and the interface as **HelloWorldService**, then click Finish.



3. Modify the code of `HelloWorldServiceImpl` as follows.

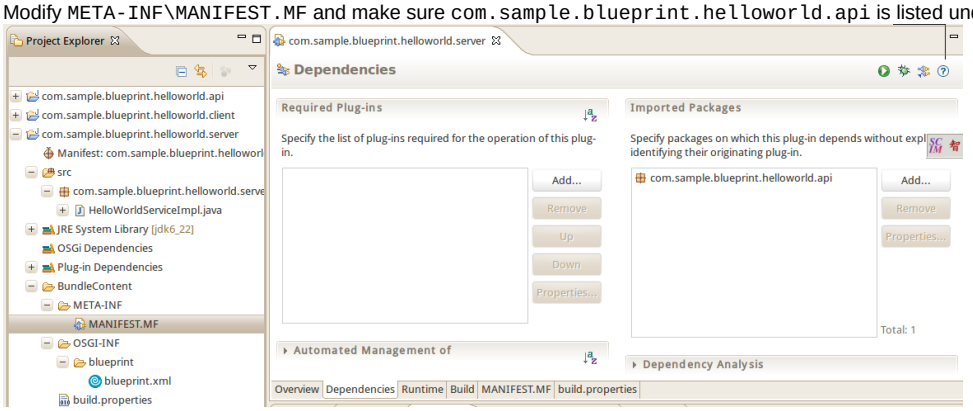
HelloWorldServiceImpl.java

```
package com.sample.blueprint.helloworld.server;
import com.sample.blueprint.helloworld.api.*;

public class HelloWorldServiceImpl implements HelloWorldService {

    public void hello() {
            System.out.println("======>>> A message from the server: Hello World!");
    }

    public void startUp() {
            System.out.println("======>>> Starting HelloWorld Server");
    }
}
```

4. Right-click the project name and create a Blueprint file, then modify the xml file as follows. The blueprint file is placed under `OSGI-INF\blueprint` directory automatically.

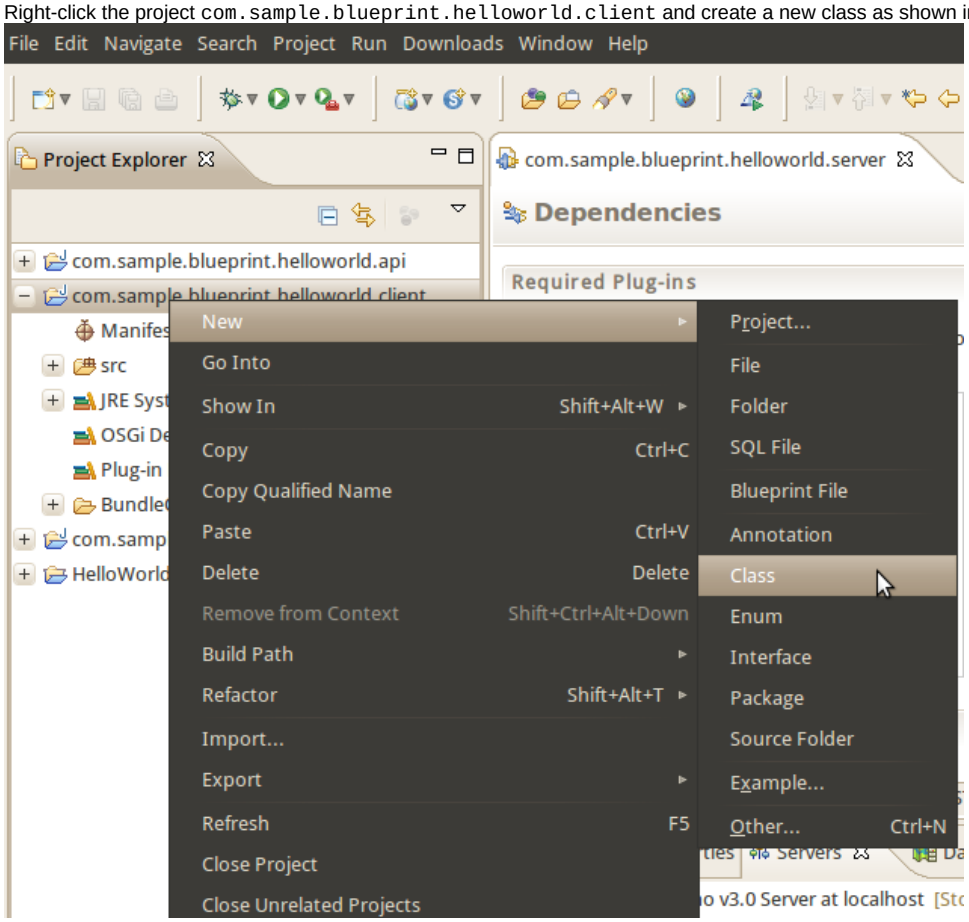
```
<?xml version="1.0" encoding="UTF-8"?>
<blueprint xmlns="http://www.osgi.org/xmlns/blueprint/v1.0.0">

      <bean id="helloservice"
            class="com.sample.blueprint.helloworld.server.HelloWorldServiceImpl"
            init-method="startUp">
      </bean>

      <service ref="helloservice"
          interface="com.sample.blueprint.helloworld.api.HelloWorldService"/>
</blueprint>
```

5. Modify `META-INF\MANIFEST.MF` and make sure `com.sample.blueprint.helloworld.api` is listed under **Import-Pakcage**.

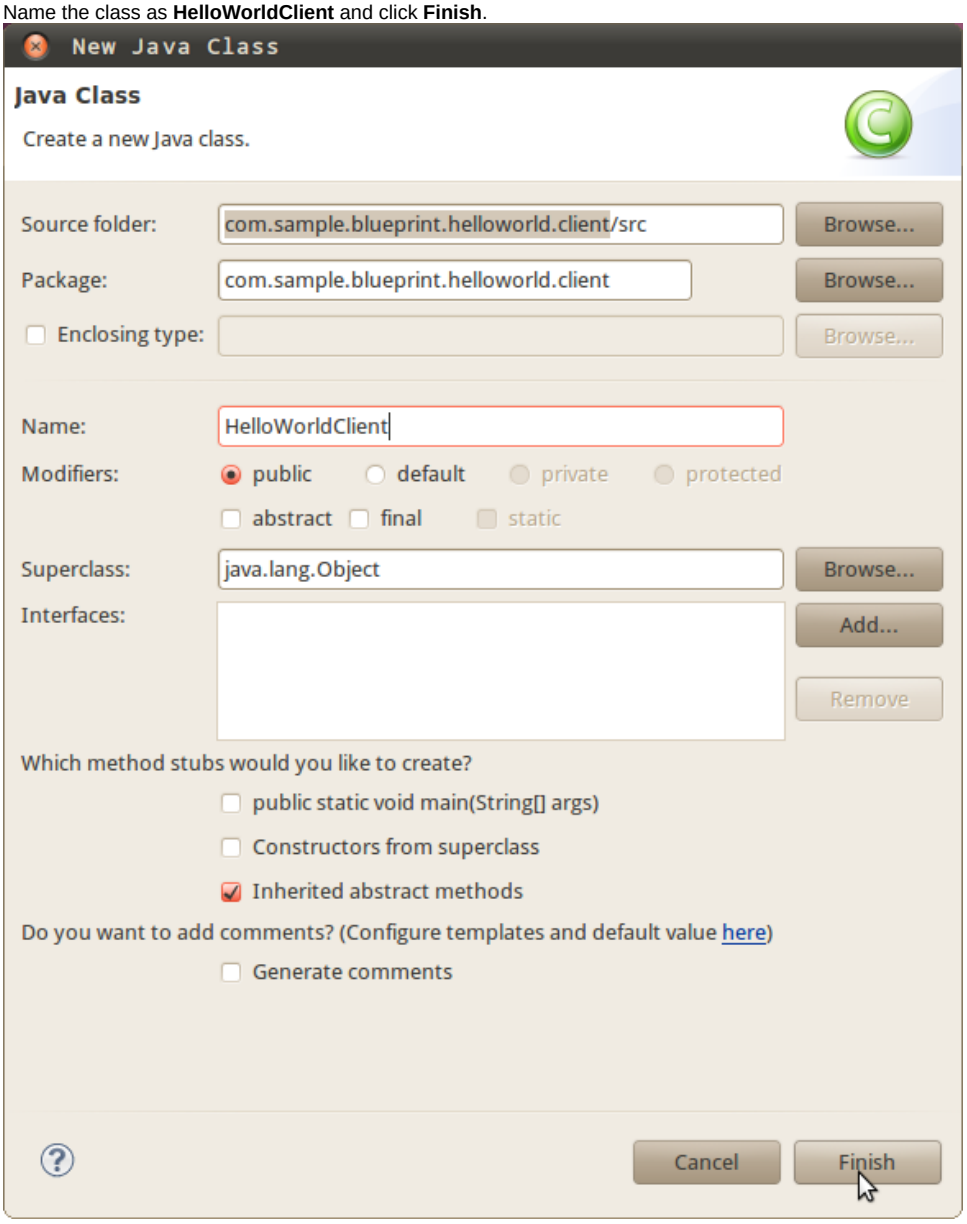


Creating the client to consume the services

1. Right-click the project `com.sample.blueprint.helloworld.client` and create a new class as shown in the figure.



2. Name the class as **HelloWorldClient** and click **Finish**.



3. Modify the code of `HelloWorldClient` as follows.

HelloWorldClient.java

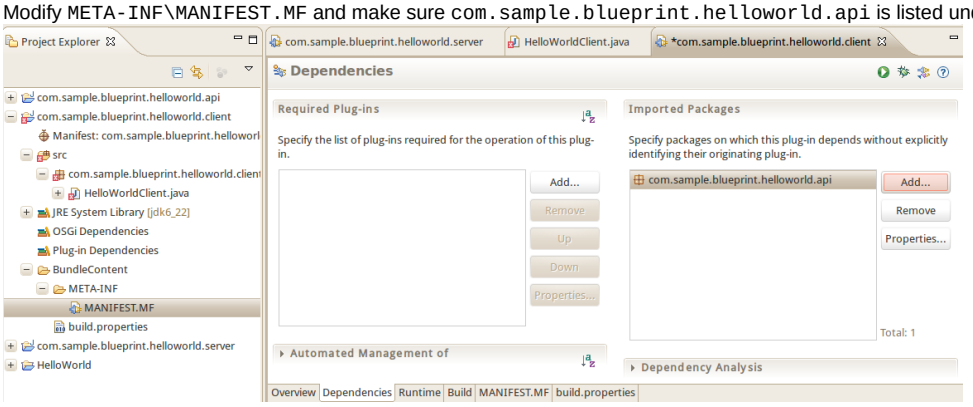
```
package com.sample.blueprint.helloworld.client;
import com.sample.blueprint.helloworld.api.HelloWorldService;

public class HelloWorldClient {

        HelloWorldService helloWorldService = null;

        public void startUp() {
                System.out
                                .println("========>>>>Client HelloWorld: About to execute a method from
the Hello World service");
                helloWorldService.hello();
                System.out
                                .println("========>>>>Client HelloWorld: ... if you didn't just see a
Hello World message something went wrong");
        }
```

```
        public HelloWorldService getHelloWorldService() {
                return helloWorldService;
        }

        public void setHelloWorldService(HelloWorldService helloWorldService) {
                this.helloWorldService = helloWorldService;

        }
}
```

4. Right-click the project name and create a Blueprint file, then modify the xml file as follows. The xml is placed under `OSGI-INF\blueprint` directory automatically.

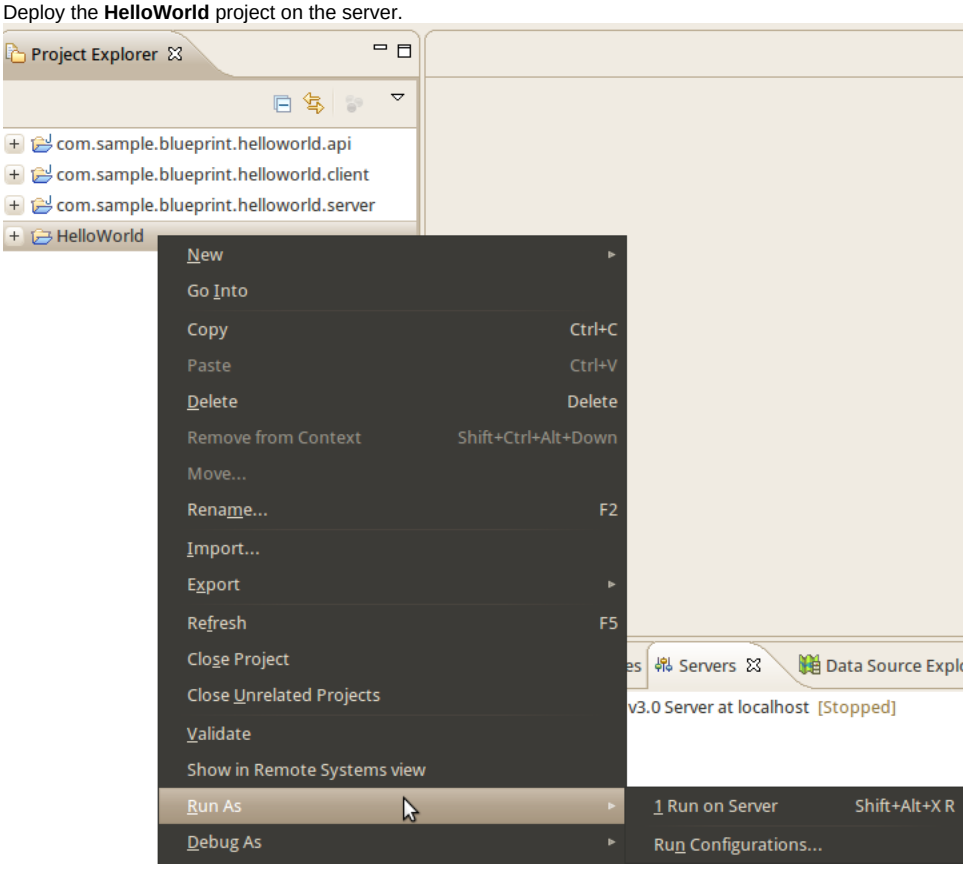
```
<?xml version="1.0" encoding="UTF-8"?>
<blueprint xmlns="http://www.osgi.org/xmlns/blueprint/v1.0.0">
        <reference id="helloservice"
                interface="com.sample.blueprint.helloworld.api.HelloWorldService" />

        <bean id="helloclient" class="com.sample.blueprint.helloworld.client.HelloWorldClient"
                init-method="startUp">
                <property name="helloWorldService" ref="helloservice" />
        </bean>
</blueprint>
```

5. Modify `META-INF\MANIFEST.MF` and make sure `com.sample.blueprint.helloworld.api` is listed under **Import-Pakcage**.



Run and deploy

1. Deploy the **HelloWorld** project on the server.



2. Check the console of the server, the messages displays as each bundles initialized sequentially.

```
Geronimo Application Server started
======>>> Starting HelloWorld Server
=========>>>>Client HelloWorld: About to execute a method from the Hello World service
======>>> A message from the server: Hello World!
========>>>>Client HelloWorld: ... if you didn't just see a Hello World message something went wrong
```