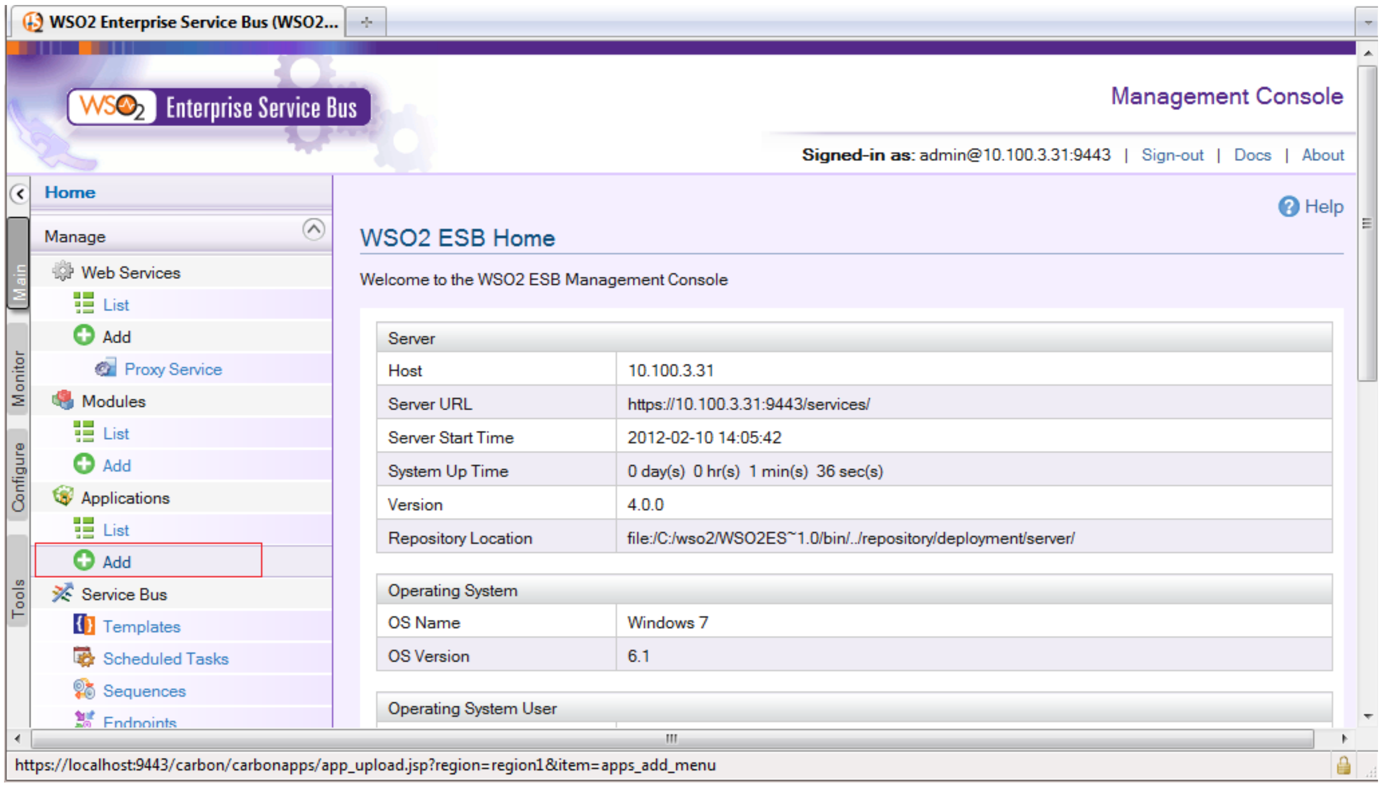
Deploying and Debugging

After creating your C-Apps and deployable archives, you are ready to deploy them to one or more Carbon servers (that is, servers for products based on WSO2 Carbon). You can then test them and debug your applications and artifacts as needed.

If you are running products in a cluster, you can use Deployment Synchronizer to keep the configurations on all nodes in the cluster in synch.

Deploying a C-App using the management console

To deploy the CAR file for the C-App to a Carbon server, start the server, log in to the management console for that product, go to **Application -> Add,** and browse for the CAR file that you have created.



WSO2 ESB Home

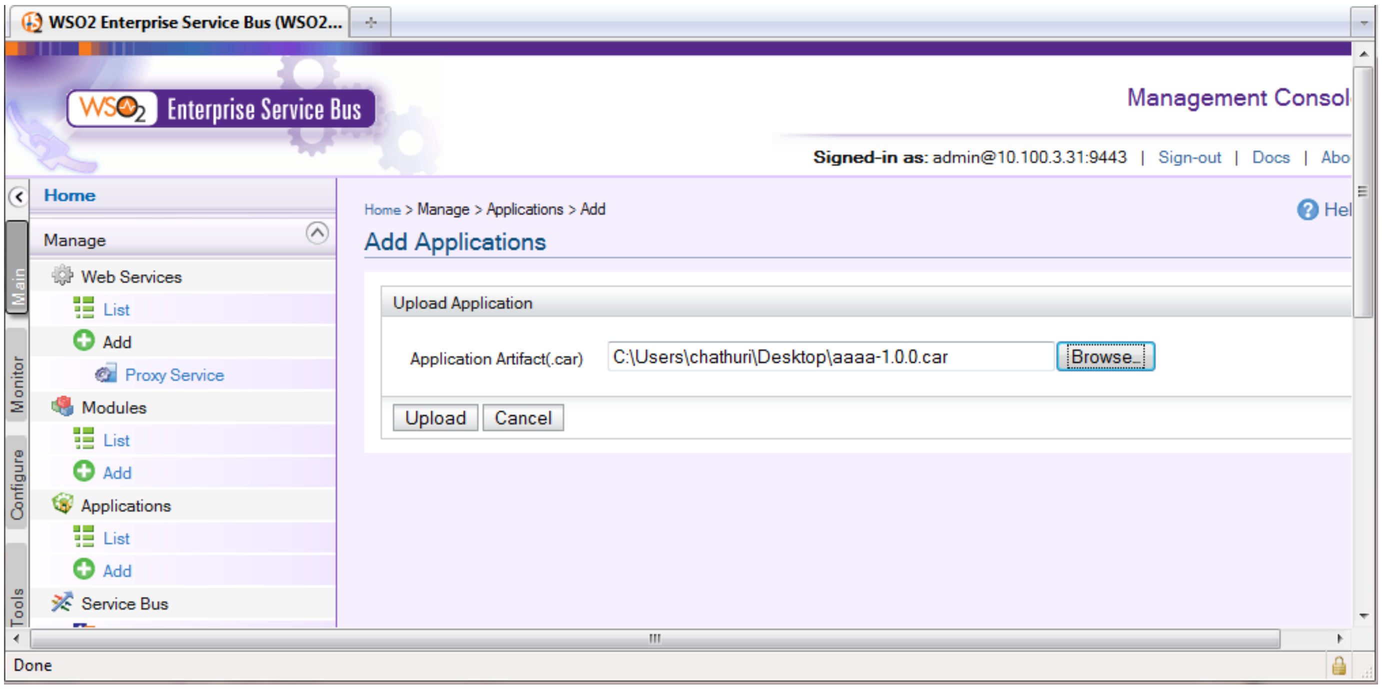
Welcome to the WSO2 ESB Management Console

Server	
Host	10.100.3.31
Server URL	https://10.100.3.31:9443/services/
Server Start Time	2012-02-10 14:05:42
System Up Time	0 day(s) 0 hr(s) 1 min(s) 36 sec(s)
Version	4.0.0
Repository Location	file:/C:/wso2/WSO2ES~1.0/bin/../repository/deployment/server/

Operating System	
OS Name	Windows 7
OS Version	6.1

Operating System User

https://localhost:9443/carbon/carbonapps/app_upload.jsp?region=region1&item=apps_add_menu



Add Applications

Upload Application

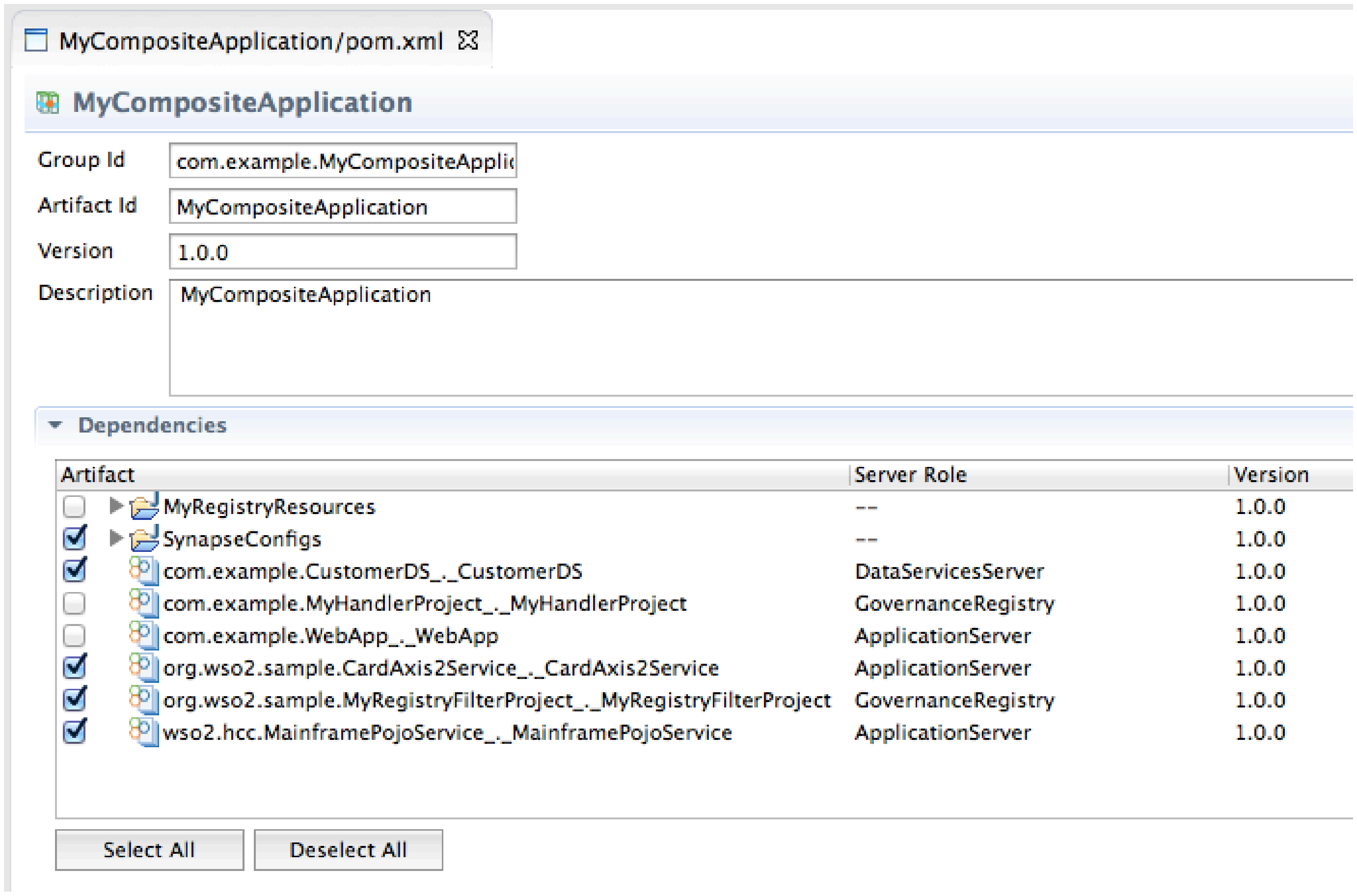
Application Artifact(.car) C:\Users\chathuri\Desktop\aaaa-1.0.0.car Browse...

Upload Cancel

Once you click **Upload**, you will be asked to **Refresh** the browser. Once you refresh, you will see that the **CAR** file has been deployed.

Deploying a C-App to a running server inside Eclipse

Before you deploy the created C-App project, you have to choose the artifacts you need to deploy. For that, double click on the **pom.xml** and select artifacts you need to deploy and save the changes.



MyCompositeApplication/pom.xml

MyCompositeApplication

Group Id: com.example.MyCompositeApplic

Artifact Id: MyCompositeApplication

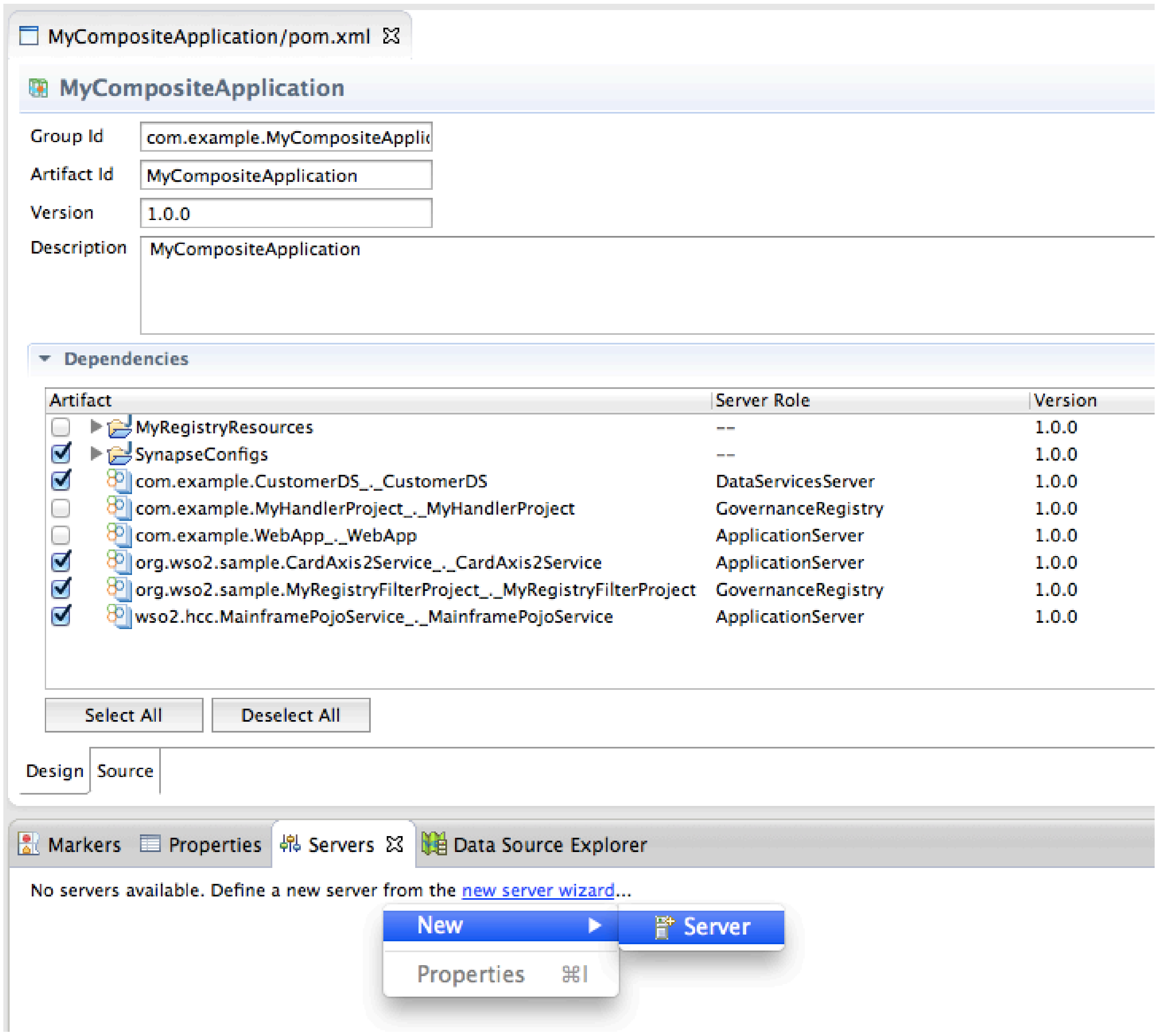
Version: 1.0.0

Description: MyCompositeApplication

Dependencies

	Artifact	Server Role	Version
☐	MyRegistryResources	--	1.0.0
✓	SynapseConfigs	--	1.0.0
✓	com.example.CustomerDS_._CustomerDS	DataServicesServer	1.0.0
☐	com.example.MyHandlerProject_._MyHandlerProject	GovernanceRegistry	1.0.0
☐	com.example.WebApp_._WebApp	ApplicationServer	1.0.0
✓	org.wso2.sample.CardAxis2Service_._CardAxis2Service	ApplicationServer	1.0.0
✓	org.wso2.sample.MyRegistryFilterProject_._MyRegistryFilterProject	GovernanceRegistry	1.0.0
✓	wso2.hcc.MainframePojoService_._MainframePojoService	ApplicationServer	1.0.0

Go to **Servers** view, right click and select **New -> Server**.



MyCompositeApplication/pom.xml

MyCompositeApplication

Group Id	com.example.MyCompositeApplic
Artifact Id	MyCompositeApplication
Version	1.0.0
Description	MyCompositeApplication

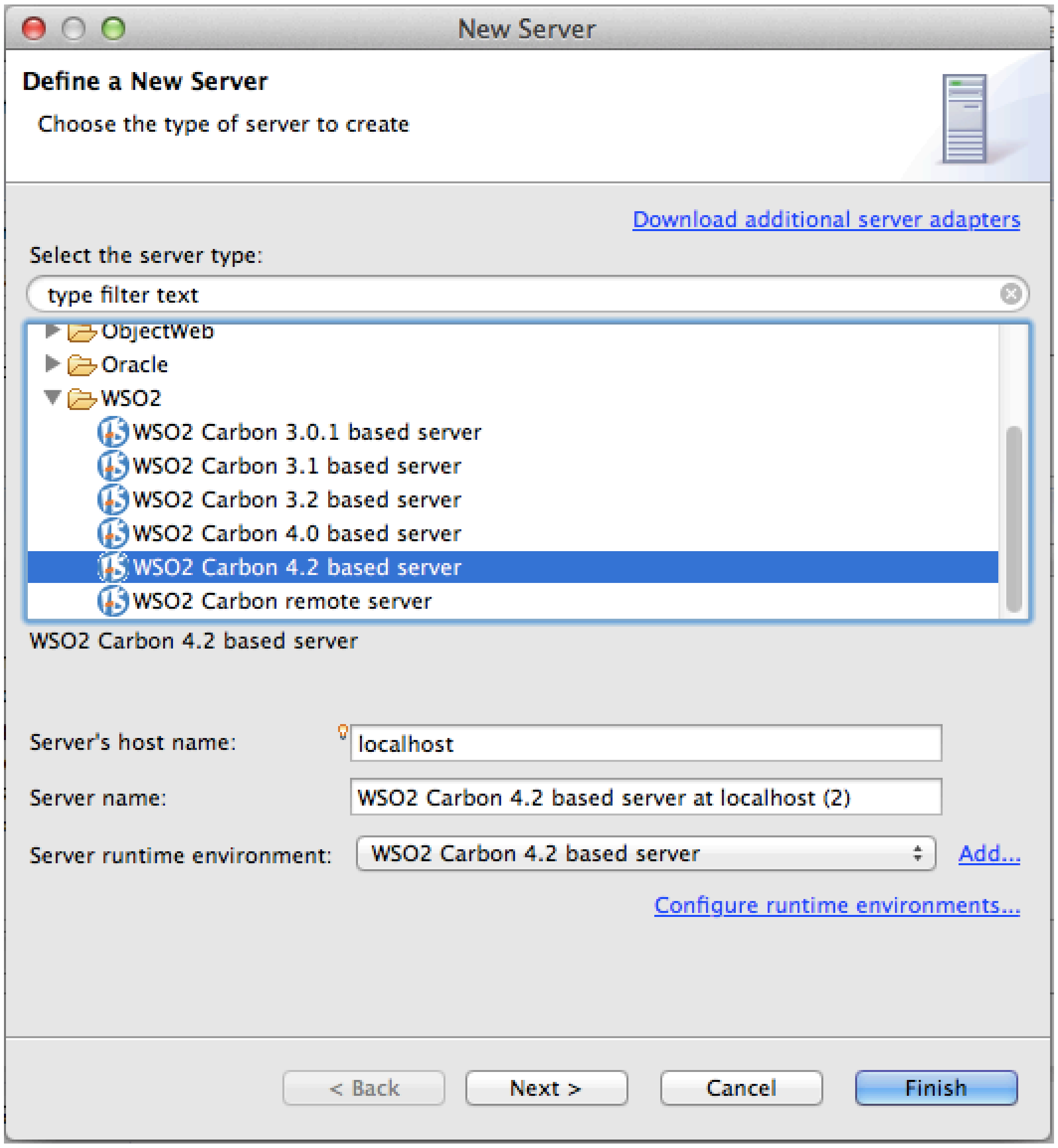
Dependencies

	Artifact	Server Role	Version
☐	MyRegistryResources	--	1.0.0
☑	SynapseConfigs	--	1.0.0
☑	com.example.CustomerDS_._CustomerDS	DataServicesServer	1.0.0
☐	com.example.MyHandlerProject_._MyHandlerProject	GovernanceRegistry	1.0.0
☐	com.example.WebApp_._WebApp	ApplicationServer	1.0.0
☑	org.wso2.sample.CardAxis2Service_._CardAxis2Service	ApplicationServer	1.0.0
☑	org.wso2.sample.MyRegistryFilterProject_._MyRegistryFilterProject	GovernanceRegistry	1.0.0
☑	wso2.hcc.MainframePojoService_._MainframePojoService	ApplicationServer	1.0.0

No servers available. Define a new server from the new server wizard...

Select the relevant version of the server under the **WSO2** category. For example, **WSO2 Carbon 4.2 based server**.

ⓘ To see the relevant version of the Carbon based server, refer to the Release Matrix .


New Server
Download additional server adapters
Add...
Configure runtime environments...
< Back
Next >
Cancel
Finish

Define a New Server

Choose the type of server to create

Select the server type:

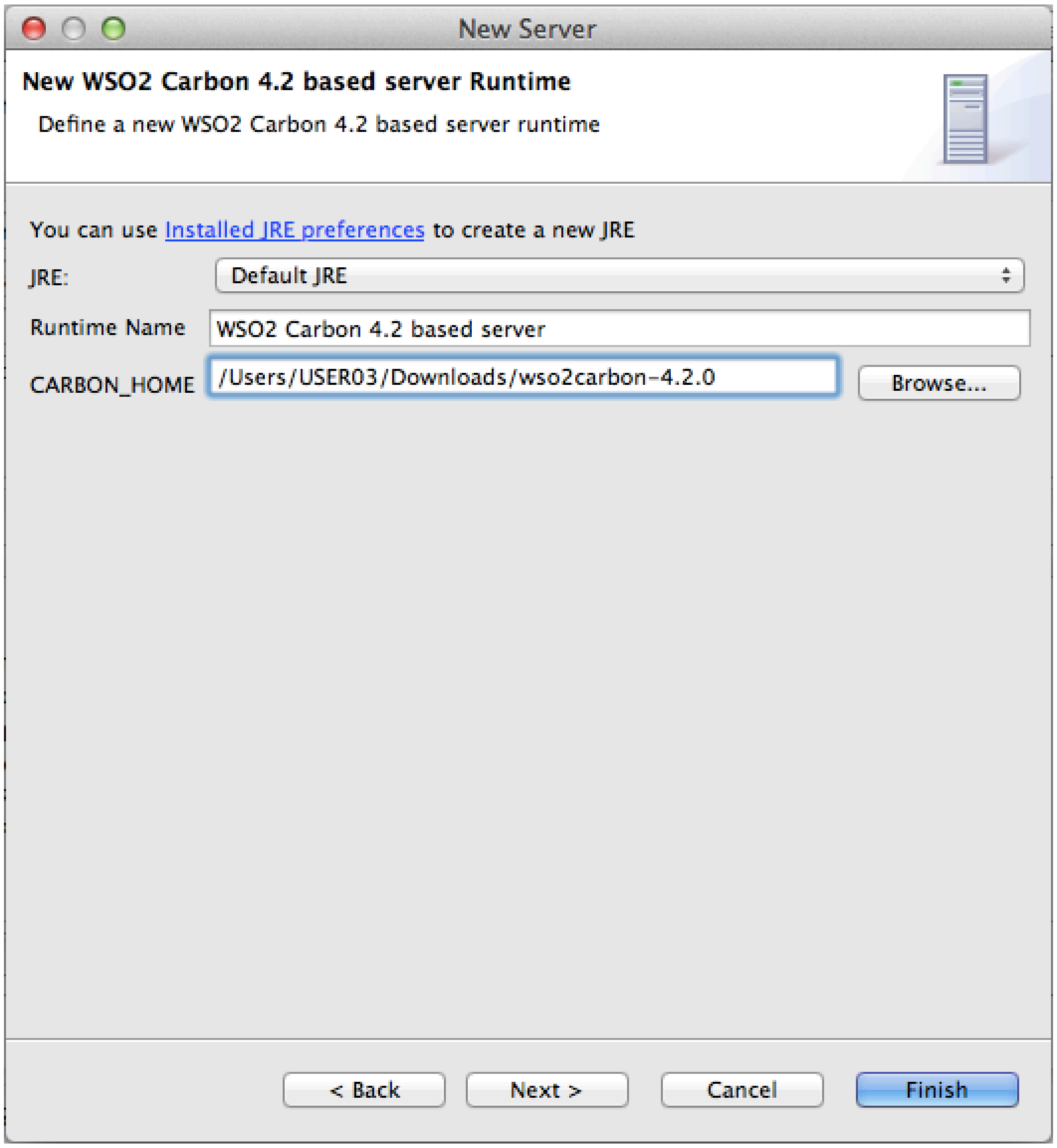
type filter text

- ObjectWeb
- Oracle
- WSO2
  - WSO2 Carbon 3.0.1 based server
  - WSO2 Carbon 3.1 based server
  - WSO2 Carbon 3.2 based server
  - WSO2 Carbon 4.0 based server
  - WSO2 Carbon 4.2 based server
  - WSO2 Carbon remote server

WSO2 Carbon 4.2 based server

Server's host name:	localhost
Server name:	WSO2 Carbon 4.2 based server at localhost (2)
Server runtime environment:	WSO2 Carbon 4.2 based server

If you have not added WSO2 Server instances before, you will need to specify the location of the downloaded WSO2 Server instance.


New WSO2 Carbon 4.2 based server Runtime

Define a new WSO2 Carbon 4.2 based server runtime

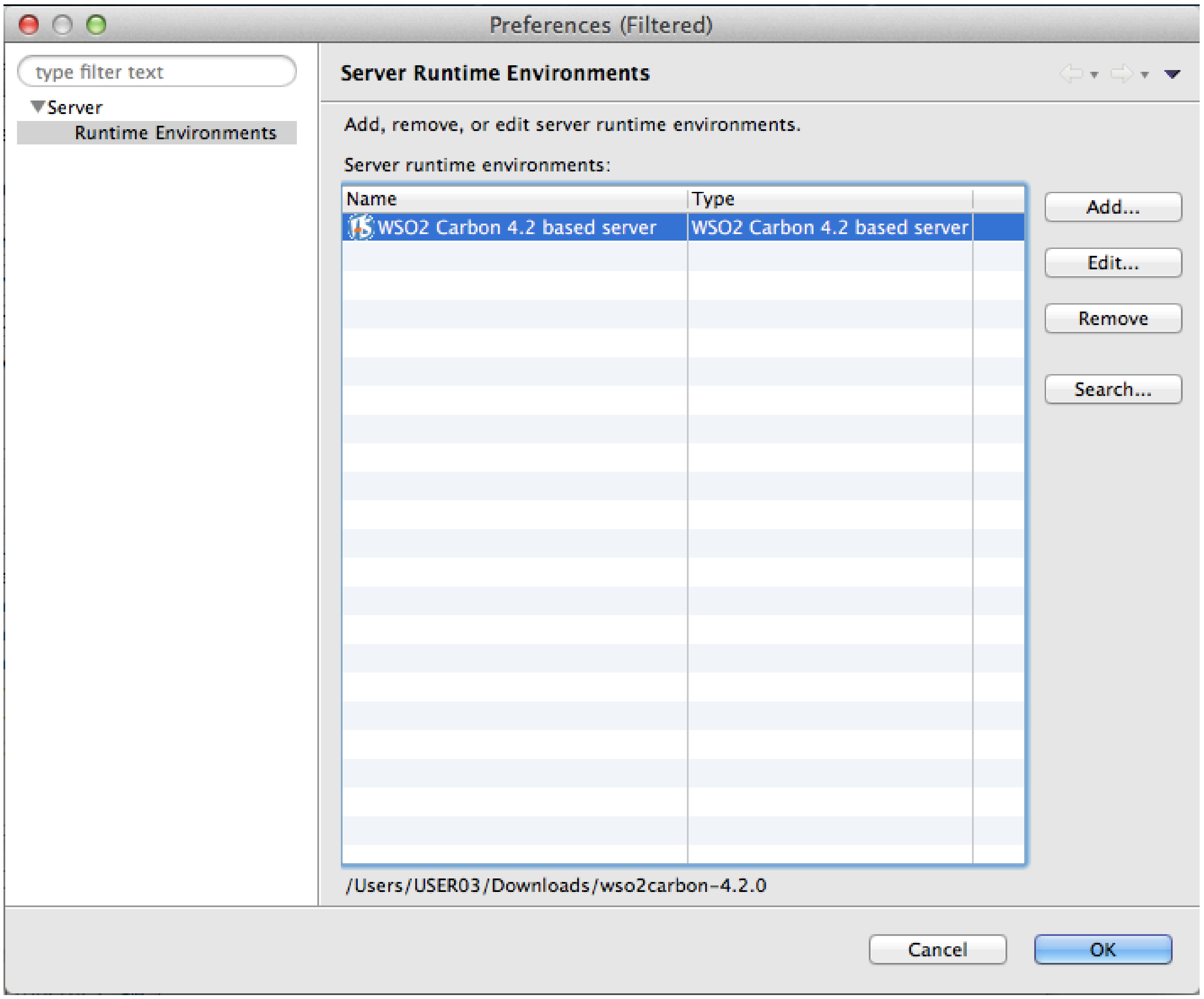
You can use Installed JRE preferences to create a new JRE

JRE: Default JRE

Runtime Name: WSO2 Carbon 4.2 based server

CARBON_HOME: /Users/USER03/Downloads/wso2carbon-4.2.0

Now you will see your version of the server (for example **WSO2 Carbon 4.2 based server**) added to the **Server Runtime Environments**. Click **OK**.



Server Runtime Environments

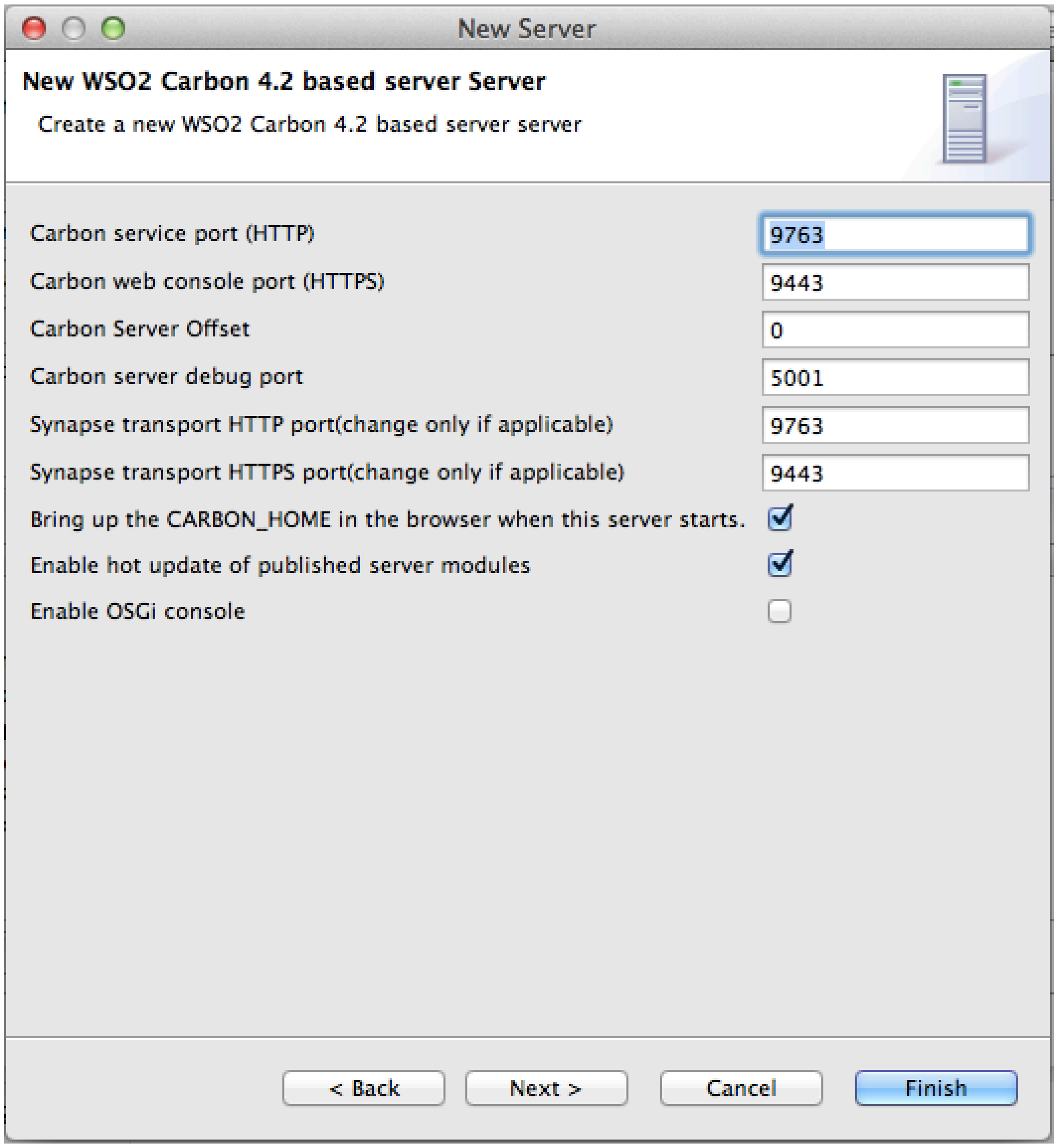
Add, remove, or edit server runtime environments.

Server runtime environments:

Name	Type
WSO2 Carbon 4.2 based server	WSO2 Carbon 4.2 based server

/Users/USER03/Downloads/wso2carbon-4.2.0

You can change ports if you want. If the default ports are not used by any other application, you can keep them as they are. After setting the ports click **Next**.

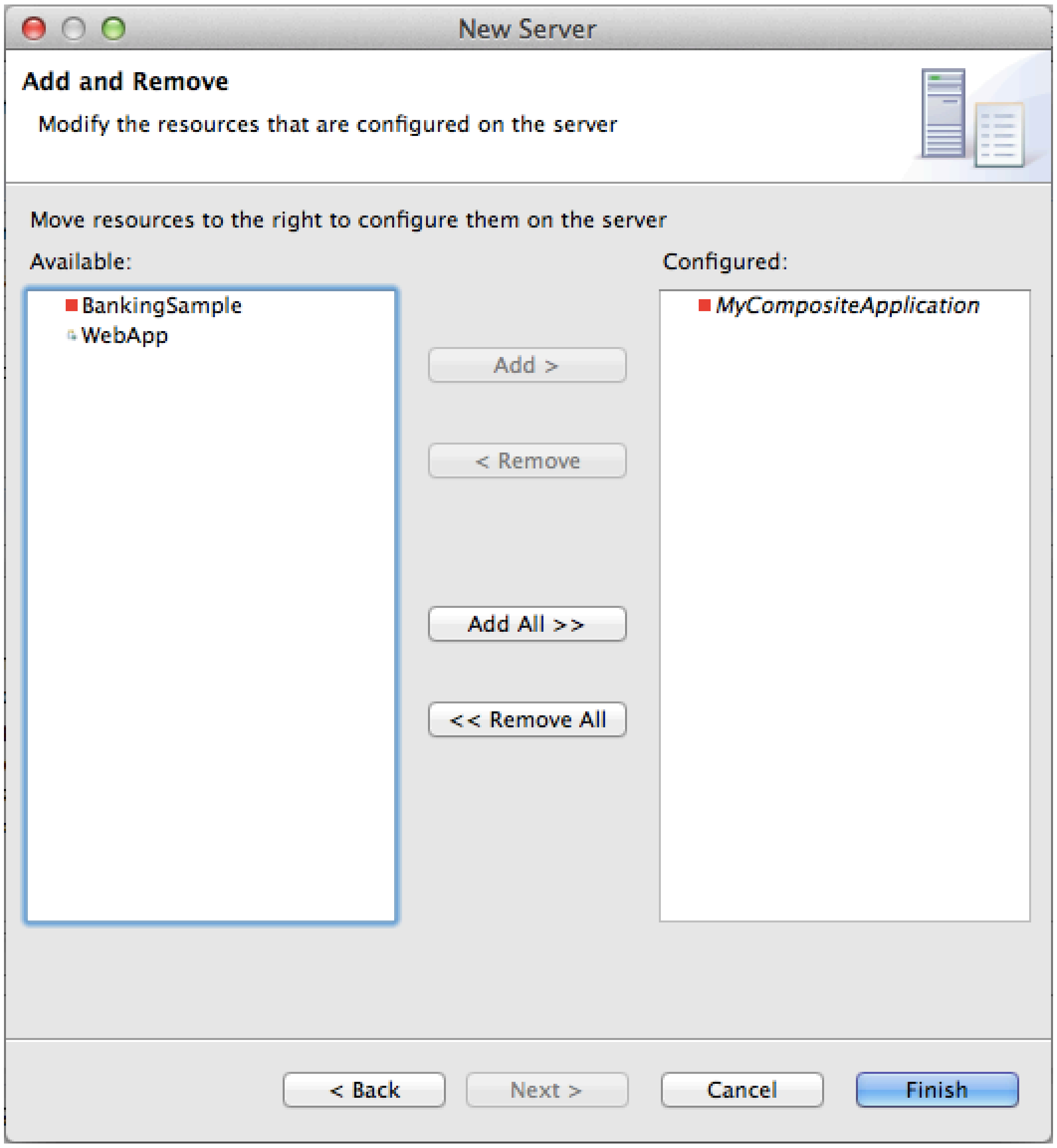


New WSO2 Carbon 4.2 based server Server

Create a new WSO2 Carbon 4.2 based server server

Carbon service port (HTTP)	9763
Carbon web console port (HTTPS)	9443
Carbon Server Offset	0
Carbon server debug port	5001
Synapse transport HTTP port(change only if applicable)	9763
Synapse transport HTTPS port(change only if applicable)	9443
Bring up the CARBON_HOME in the browser when this server starts.	✓
Enable hot update of published server modules	✓
Enable OSGi console	☐

Add the C-App project and click **Finish**.


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

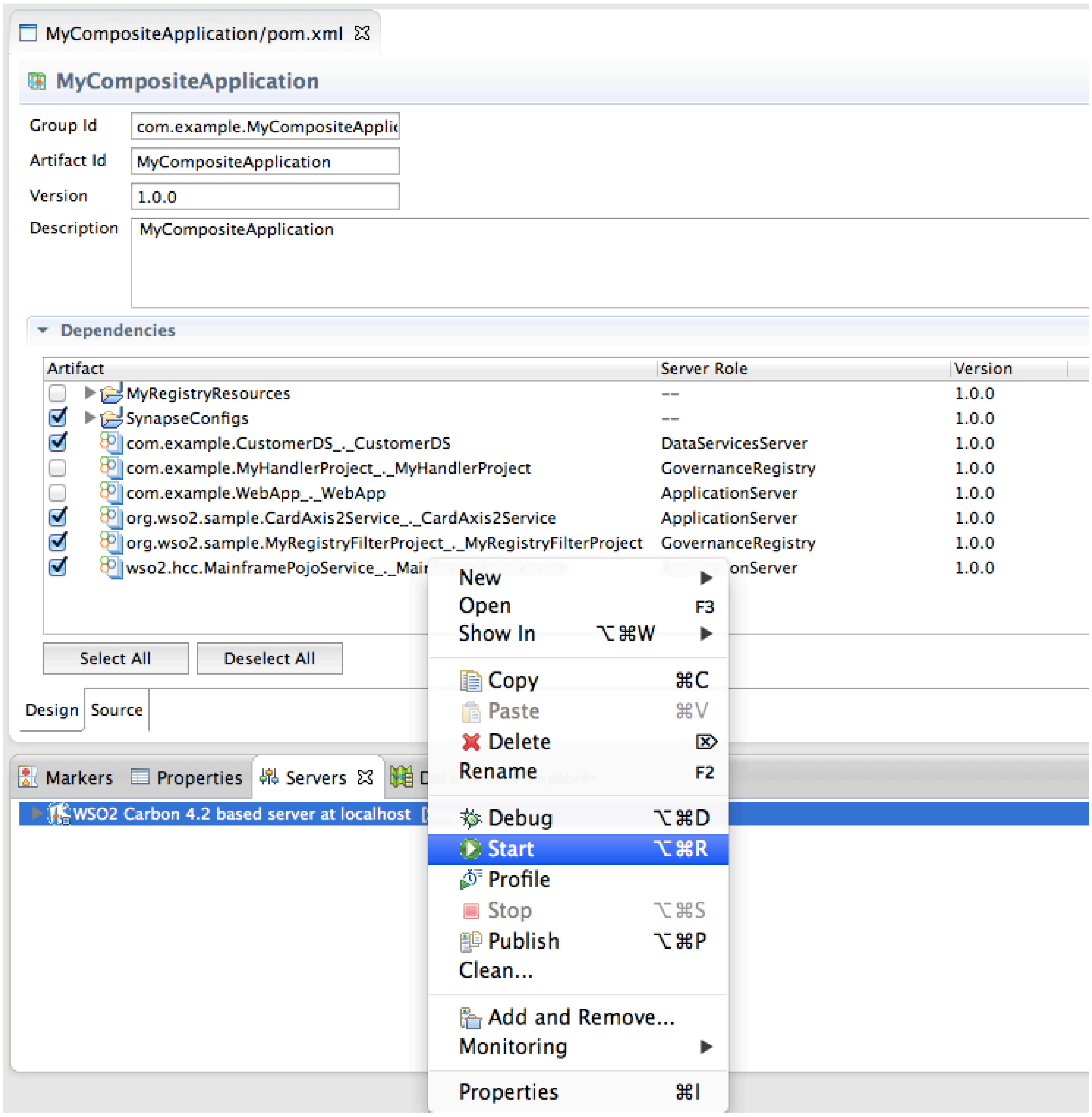
Available:

- BankingSample
- WebApp

Configured:

- *MyCompositeApplication*

Now you will see the newly added server listed in the **Servers** view. Right click on the **WSO2 Carbon x.x.x based server at localhost** and select **Start** from the menu.



Group Id	com.example.MyCompositeApplic
Artifact Id	MyCompositeApplication
Version	1.0.0
Description	MyCompositeApplication

Dependencies

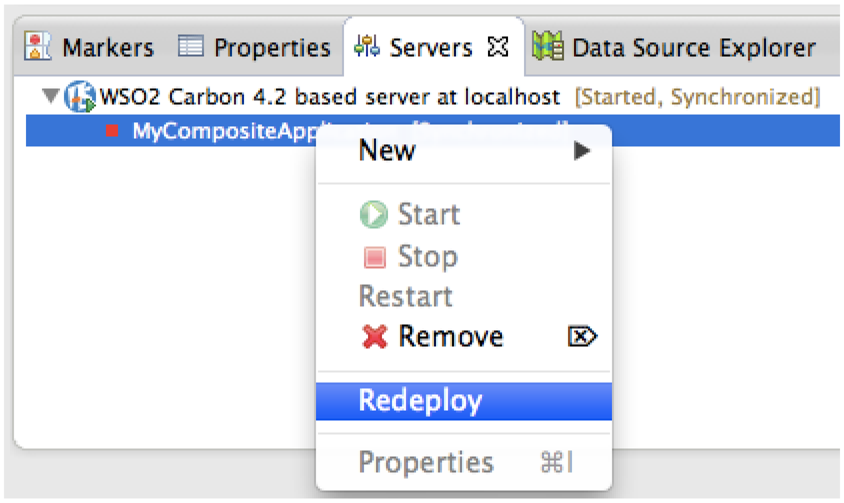
	Artifact	Server Role	Version
☐	MyRegistryResources	--	1.0.0
✓	SynapseConfigs	--	1.0.0
✓	com.example.CustomerDS_._CustomerDS	DataServicesServer	1.0.0
☐	com.example.MyHandlerProject_._MyHandlerProject	GovernanceRegistry	1.0.0
☐	com.example.WebApp_._WebApp	ApplicationServer	1.0.0
✓	org.wso2.sample.CardAxis2Service_._CardAxis2Service	ApplicationServer	1.0.0
✓	org.wso2.sample.MyRegistryFilterProject_._MyRegistryFilterProject	GovernanceRegistry	1.0.0
✓	wso2.hcc.MainframePojoService_._Mair	onServer	1.0.0

Redeploying a C-App to a running server inside Eclipse

After you have deployed your C-App to a running server inside Eclipse, you may want to change the content of your C-App. In that case you do not need to remove the deployed C-App from the server and deploy again to the server. Instead you can re-deploy an already deployed C-App. Deploy the C-App project according to the instructions given above and then do the necessary changes to your C-App project.

Open the **Servers** view in Eclipse using **Window => Show View => Servers**.

Click on the expandable icon of the Server you are running inside Eclipse. (This expandable icons available only if you have deployed a project in that server). Right-click on the project and click **Re-deploy**.



Then you will see that your Eclipse console is updated and you can notice that your C-App project is deployed again.

Deploying a C-App to multiple servers using the Maven plug-in

The WSO2 CAR Deploy plug-in for Maven allows you to generate the CAR file for a C-App and then deploy it (or undeploy it) on multiple local and remote Carbon servers, including StratosLive. To use this plug-in, you add it to the `pom.xml` file for your C-App and then use the `mvn clean deploy` command to generate and deploy the CAR.

Adding the plug-in to the POM file

1. Under the C-App you want to deploy, double-click the `pom.xml` file to open it in the editor.
2. Click the Source tab and navigate to the `maven-car-deploy-plugin` under `<build>` -> `<plugins>`.
3. Modify the `maven-car-deploy-plugin` properties as follows:
    a. Under `<carbonServers>`, create one `<CarbonServer>` for each Carbon server where you want to deploy the CAR file.
    b. Specify the following properties for each Carbon server:
        - If you are using a custom certificate with your Carbon server, import the public key of the custom certificate to your trust store, and then set the `<trustStorePath>`, `<trustStorePassword>`, and `<trustStoreType>` properties to the location, password, and type of your trust store.
        - Set the `<serverUrl>`, `<userName>`, and `<password>` properties to the URL (such as `https://localhost:9443`), user name, and password for the Carbon server.
        - In the `<operation>` element, specify whether to deploy or undeploy the CAR file for this Carbon server.
4. In the `<properties>` section at the end of the POM file, do the following:
    a. Add the following line to ensure that the CAR will be deployed:
    `<maven.car.deploy.skip>false</maven.car.deploy.skip>`
    b. If you want to deploy the CAR file to a remote Maven repository in addition to the Carbon servers, add the following line:
    `<maven.deploy.skip>false</maven.deploy.skip>`
    Note: You will be able to override both of these settings when you run the Maven command.
5. Save the file.

You are now ready to generate and deploy the CAR file for this C-App.

Generating and deploying the CAR file

You now run the `mvn clean deploy` command to generate the CAR file and deploy it to the Carbon servers in a single step. You can add parameters to the command to override the configuration. For example, to deploy the CAR file to the Carbon servers but not to the remote Maven repository, add the `-Dmaven.deploy.skip=true` paramet er, as follows:

```
mvn clean deploy -Dmaven.deploy.skip=true
```

If you want to deploy to the remote Maven repository but not the Carbon servers, add the `-Dmaven.`**`car`**`.deploy.skip=true` parameter, as follows:

```
mvn clean deploy -Dmaven.car.deploy.skip=true
```

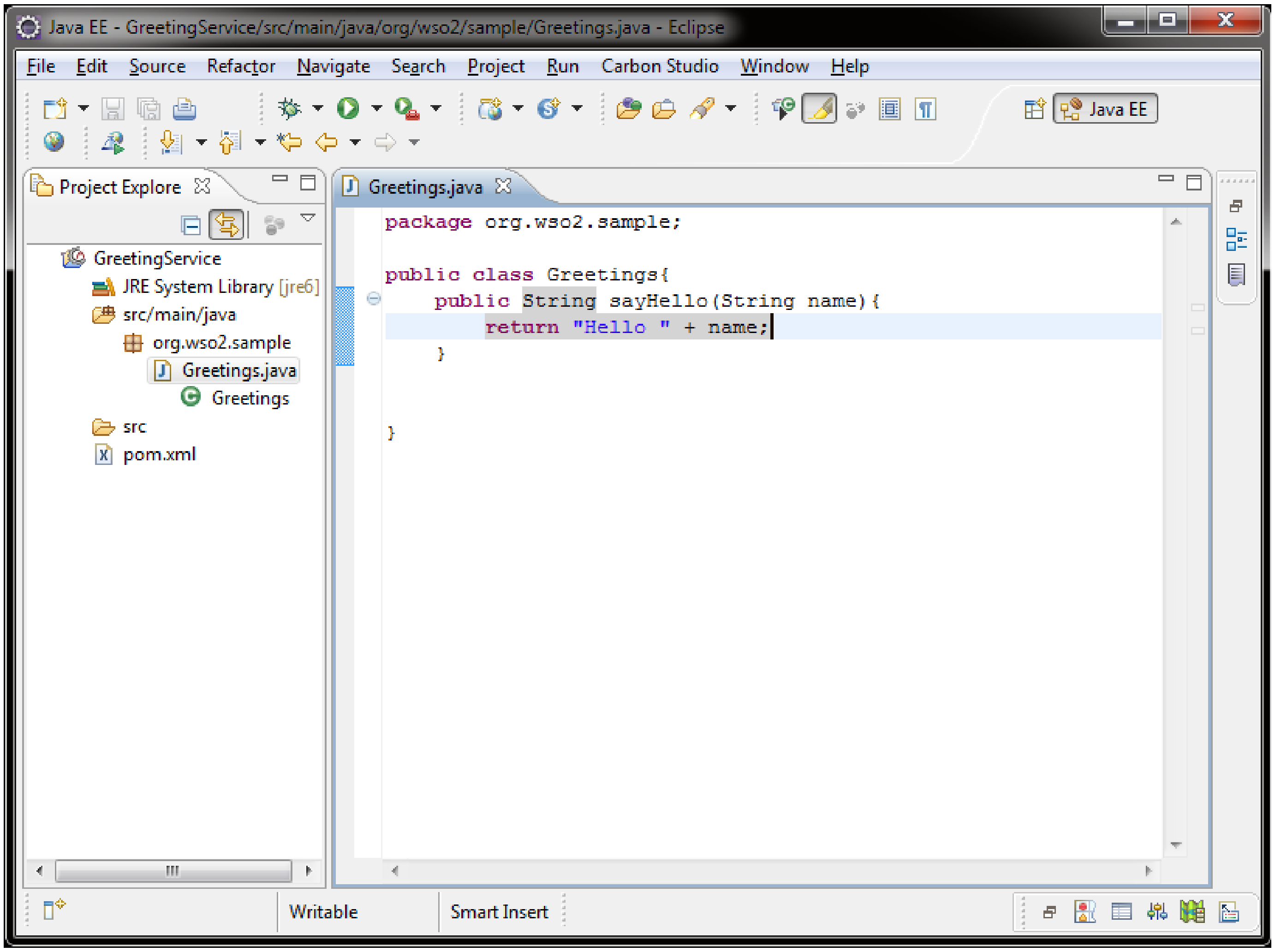
You can also skip deployment to both the remote repository AND the Carbon servers:

```
mvn clean deploy -Dmaven.deploy.skip=true -Dmaven.car.deploy.skip=true
```

Debugging your application

Suppose you want to develop a simple Apache Axis2 Service and want to debug your service when you invoke it. Let's see how to achieve this using WSO2 Developer Studio.

First, create a simple GreetingService.

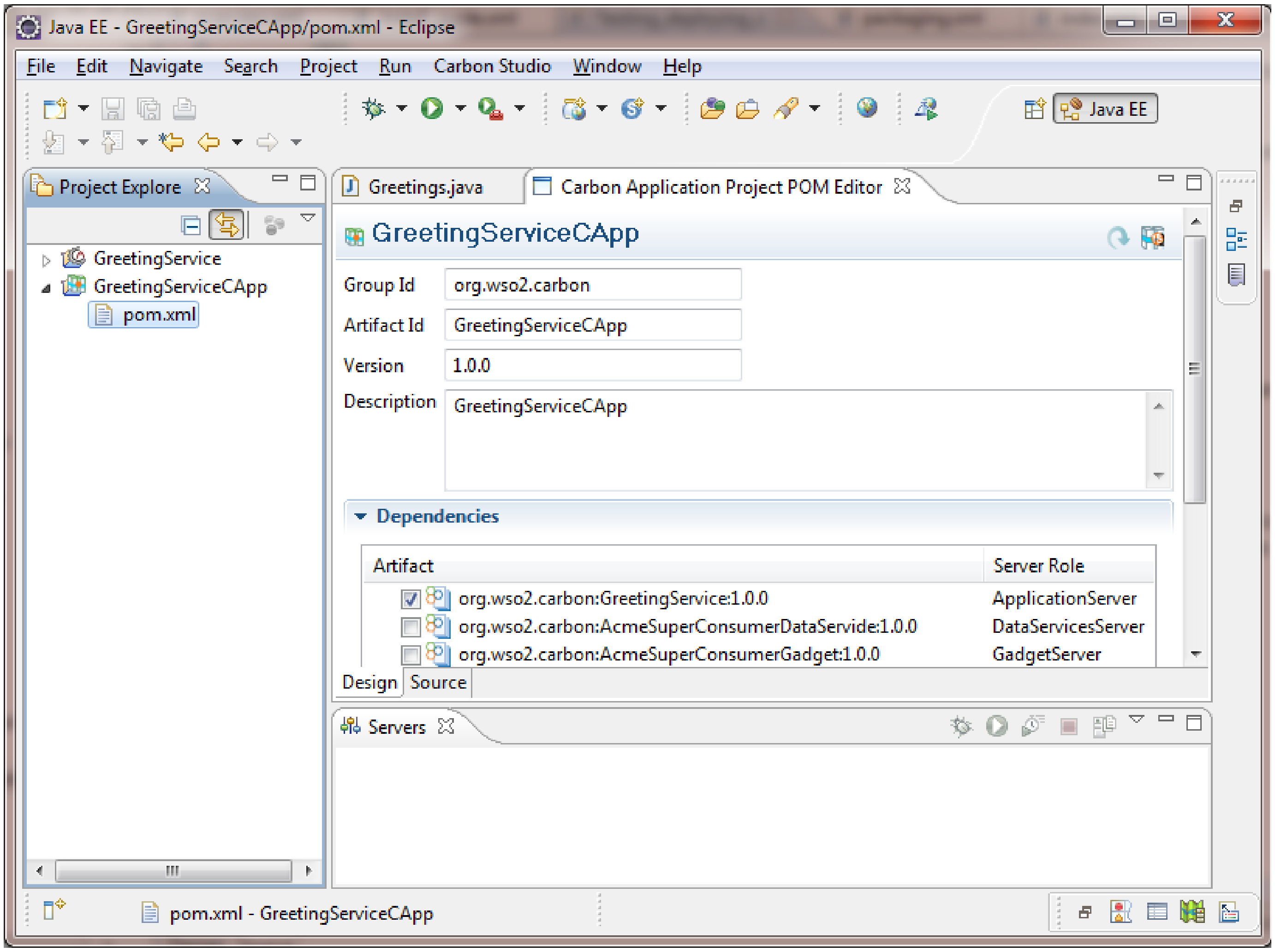

Java EE - GreetingService/src/main/java/org/wso2/sample/Greetings.java - Eclipse
File Edit Source Refactor Navigate Search Project Run Carbon Studio Window Help
Java EE
Project Explore
GreetingService
JRE System Library [jre6]
src/main/java
org.wso2.sample
Greetings.java
Greetings
src
pom.xml
Greetings.java
Writable
Smart Insert

```
package org.wso2.sample;

public class Greetings{
    public String sayHello(String name){
        return "Hello " + name;
    }


}
```

Then create a Composite Application Project to group it.


Java EE - GreetingServiceCApp/pom.xml - Eclipse
File Edit Navigate Search Project Run Carbon Studio Window Help
Java EE
Project Explore
GreetingService
GreetingServiceCApp
pom.xml
Greetings.java
Carbon Application Project POM Editor
Design
Source
Servers
pom.xml - GreetingServiceCApp

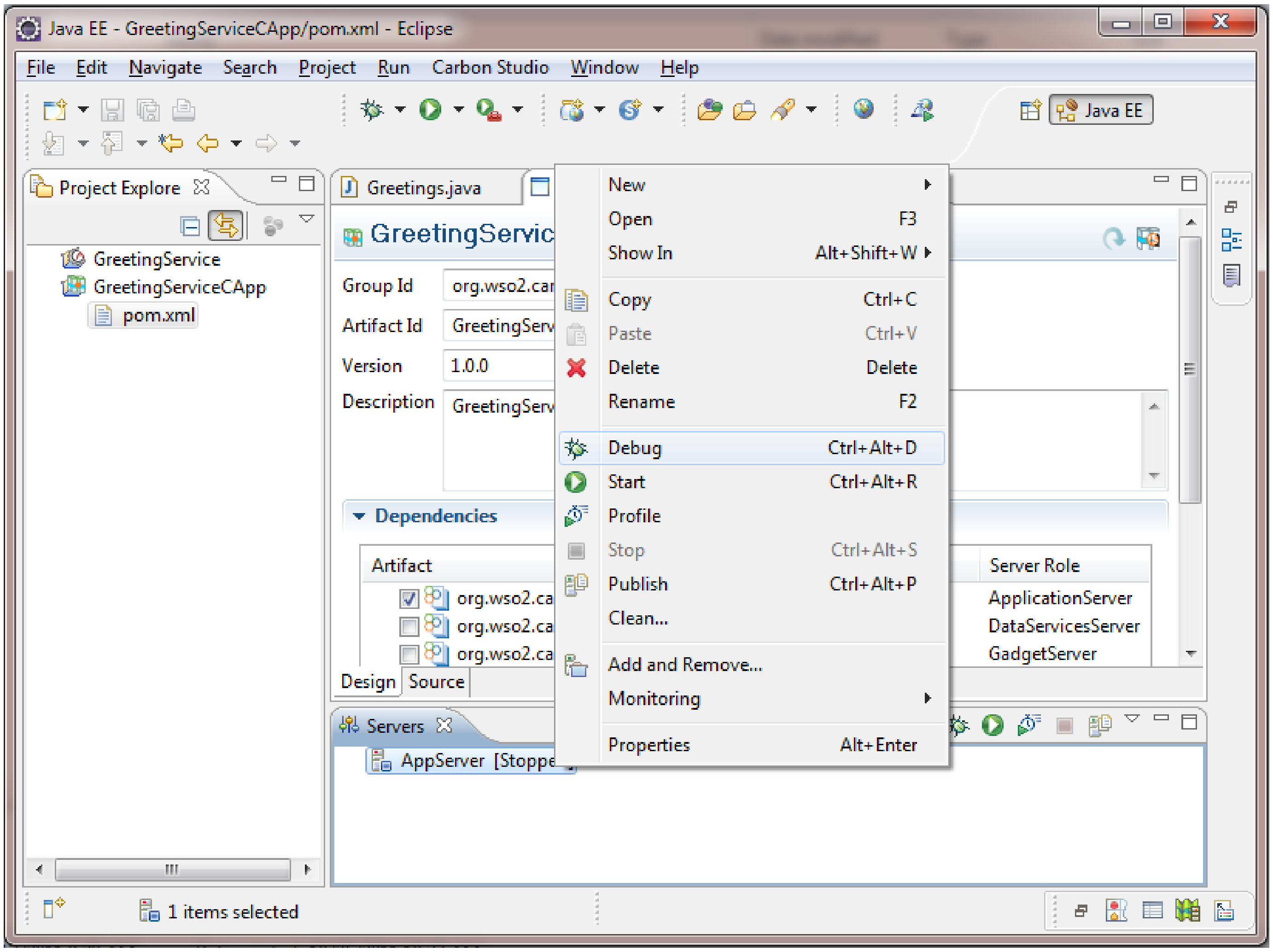
GreetingServiceCApp

Group Id: org.wso2.carbon
Artifact Id: GreetingServiceCApp
Version: 1.0.0
Description: GreetingServiceCApp

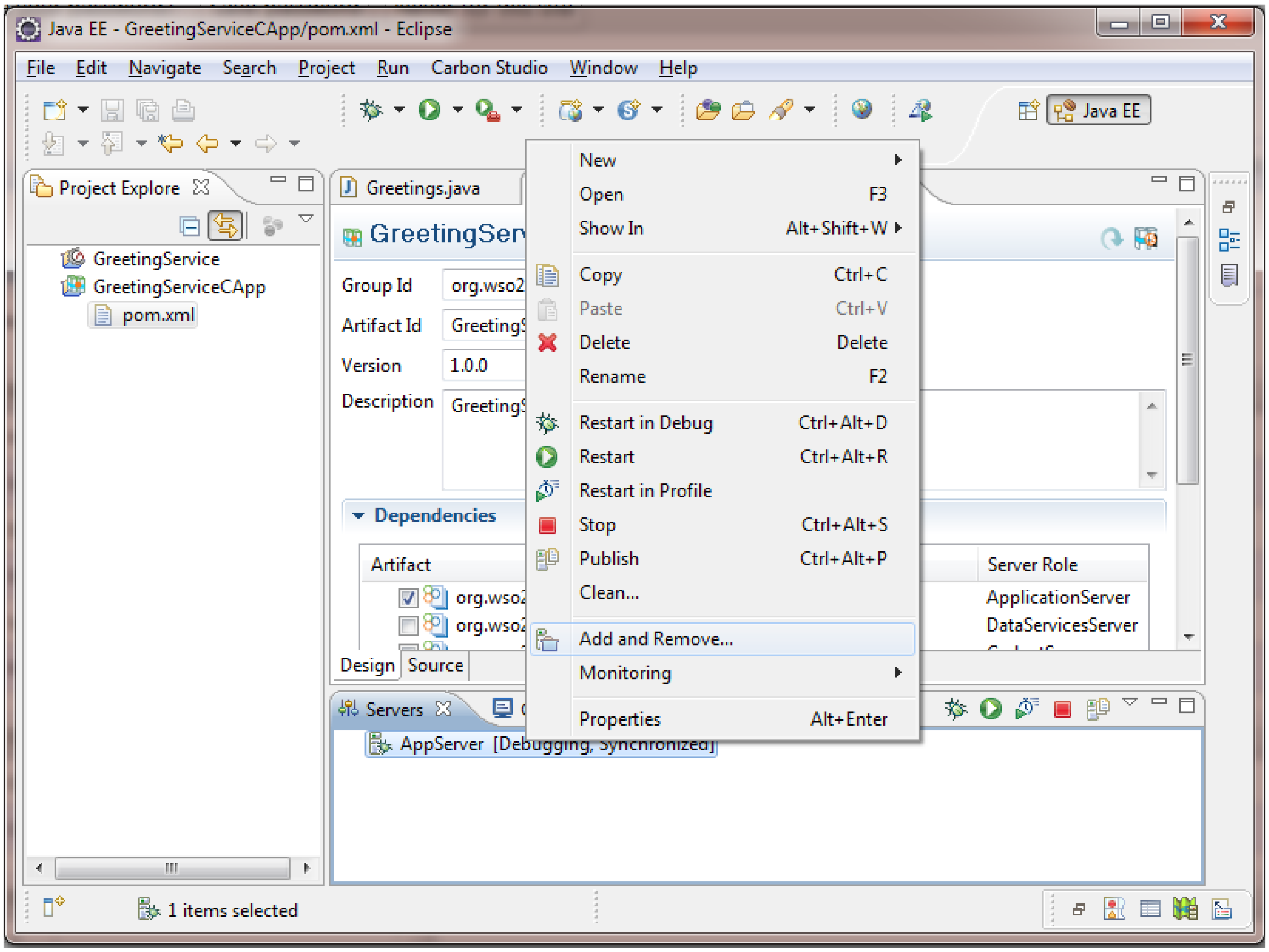
Dependencies

	Artifact	Server Role
✓	org.wso2.carbon:GreetingService:1.0.0	ApplicationServer
	org.wso2.carbon:AcmeSuperConsumerDataServide:1.0.0	DataServicesServer
	org.wso2.carbon:AcmeSuperConsumerGadget:1.0.0	GadgetServer

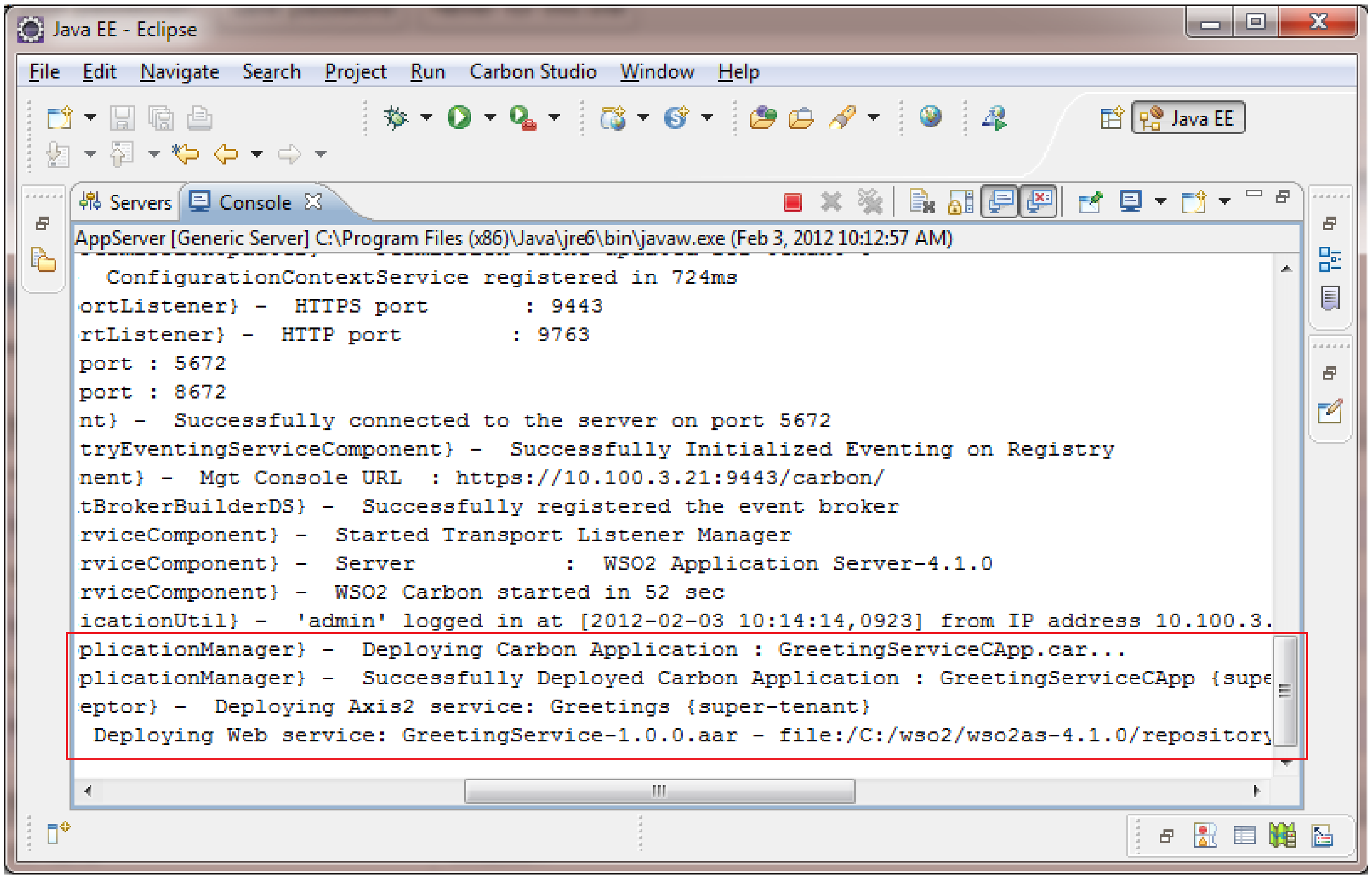
Now add a WSO2 Application Server instance to your Eclipse workspace. Steps to add a Carbon Server is described in the Deploy a C-App to a running Server inside Eclipse section. After adding the server, start the server in the **debug** mode.



After the server is successfully started, add the **GreetingServiceCApp** project.

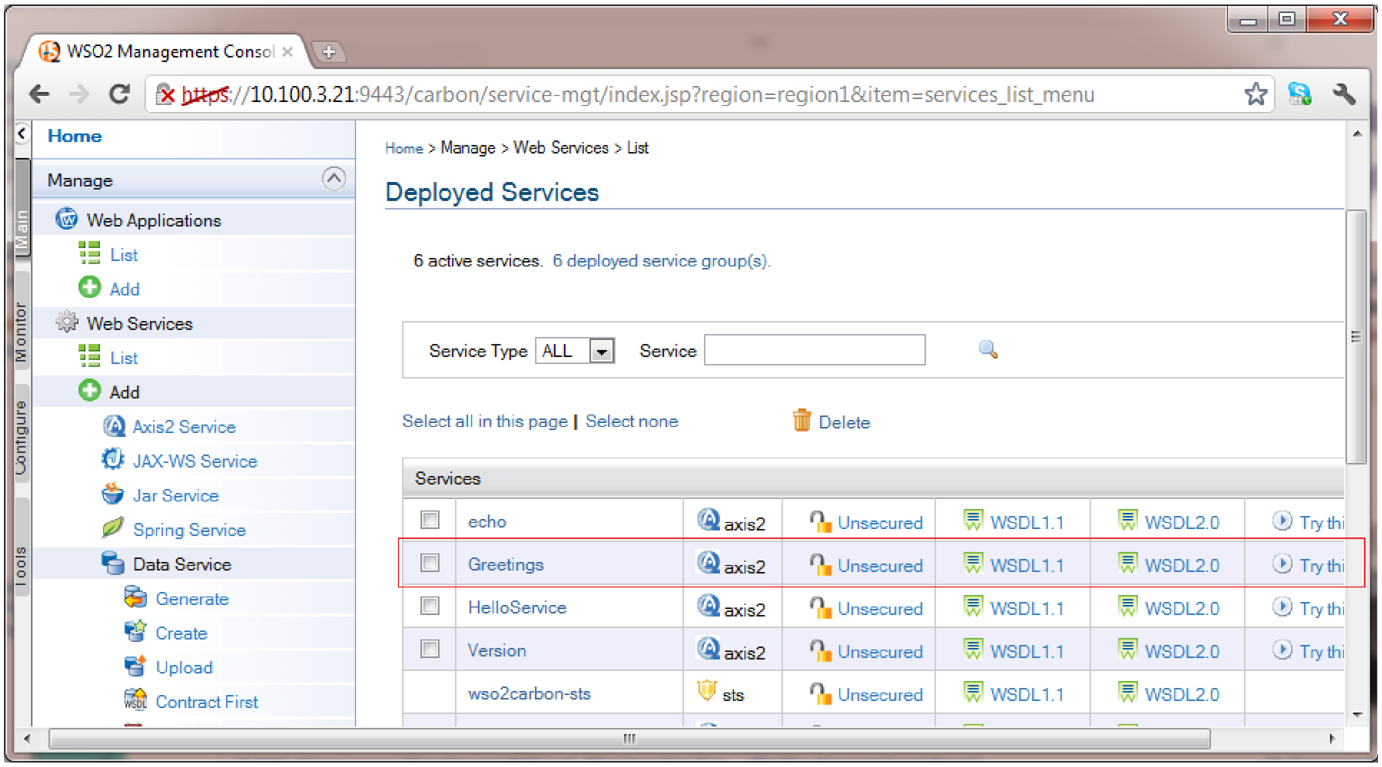


When your application has deployed successfully, you will see the following messages in your console.


Java EE - Eclipse
File Edit Navigate Search Project Run Carbon Studio Window Help
Servers Console
Java EE

```
AppServer [Generic Server] C:\Program Files (x86)\Java\jre6\bin\javaw.exe (Feb 3, 2012 10:12:57 AM)
  ConfigurationContextService registered in 724ms
ortListener} -  HTTPS port       : 9443
rtListener} -  HTTP port        : 9763
port : 5672
port : 8672
nt} -  Successfully connected to the server on port 5672
tryEventingServiceComponent} -  Successfully Initialized Eventing on Registry
nent} -  Mgt Console URL  : https://10.100.3.21:9443/carbon/
tBrokerBuilderDS} -  Successfully registered the event broker
rviceComponent} -  Started Transport Listener Manager
rviceComponent} -  Server           :  WSO2 Application Server-4.1.0
rviceComponent} -  WSO2 Carbon started in 52 sec
icationUtil} -  'admin' logged in at [2012-02-03 10:14:14,0923] from IP address 10.100.3.
plicationManager} -  Deploying Carbon Application : GreetingServiceCApp.car...
plicationManager} -  Successfully Deployed Carbon Application : GreetingServiceCApp {supe
eptor} -  Deploying Axis2 service: Greetings {super-tenant}
 Deploying Web service: GreetingService-1.0.0.aar - file:/C:/wso2/wso2as-4.1.0/repository
```

Now go to your web browser and check whether your application is being deployed under the **Web Services Listing** page.

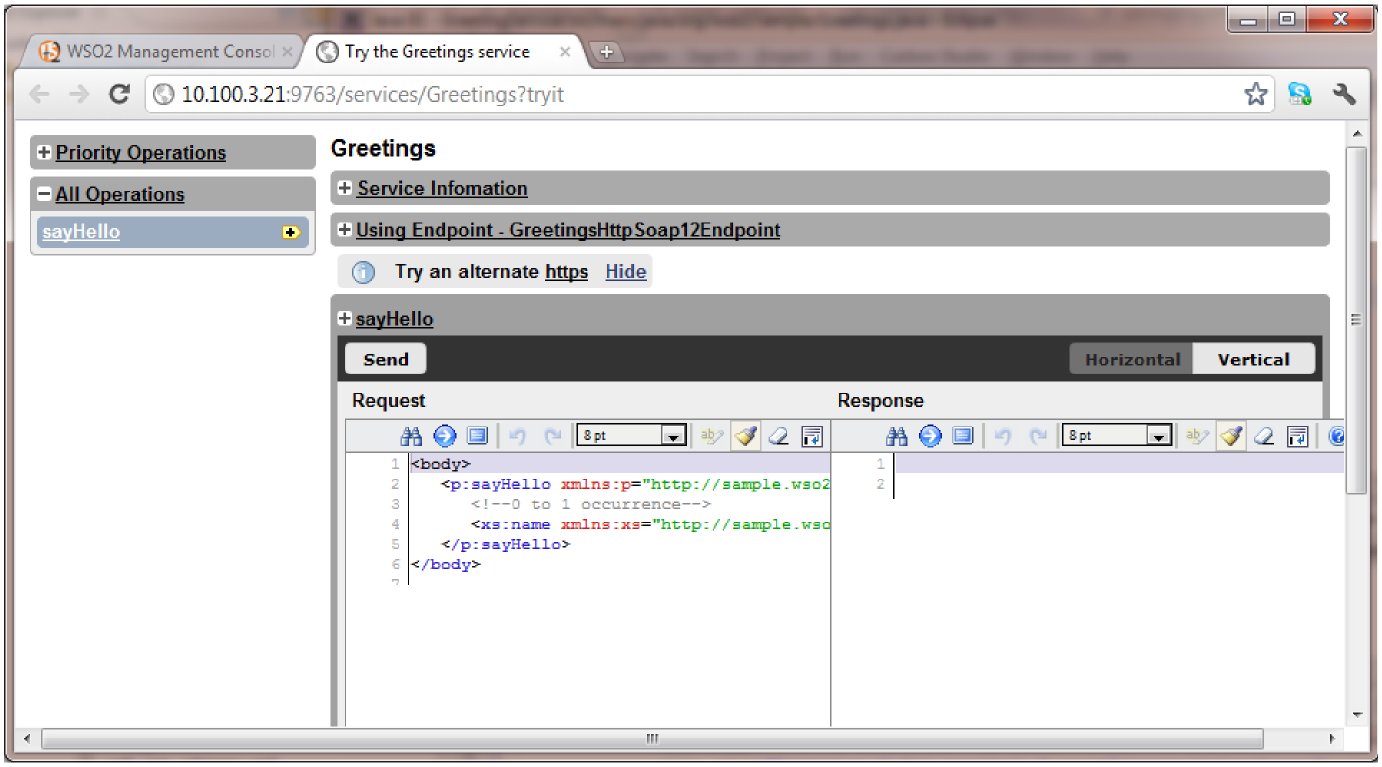


Deployed Services

6 active services. 6 deployed service group(s).

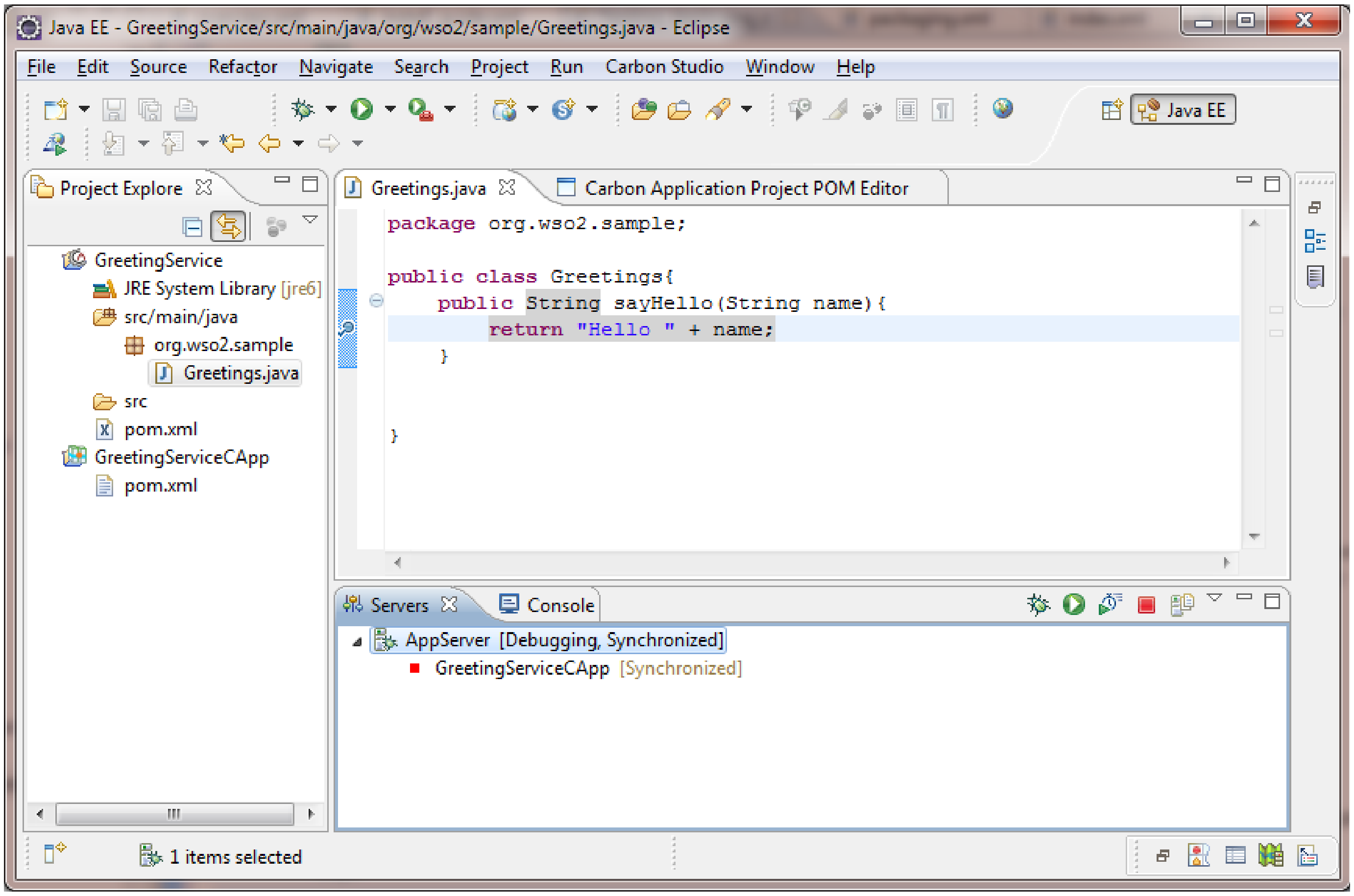
Services						
☐	echo	axis2	Unsecured	WSDL1.1	WSDL2.0	Try thi
☐	Greetings	axis2	Unsecured	WSDL1.1	WSDL2.0	Try thi
☐	HelloService	axis2	Unsecured	WSDL1.1	WSDL2.0	Try thi
☐	Version	axis2	Unsecured	WSDL1.1	WSDL2.0	Try thi
	wso2carbon-sts	sts	Unsecured	WSDL1.1	WSDL2.0	

Now click on the **Try This Service** option at the right hand corner. Once you click it, it will go to a separate page where it lists all the operations related to the web service you choose.



```
<body>
   <p:sayHello xmlns:p="http://sample.wso2
      <!--0 to 1 occurrence-->
      <xs:name xmlns:xs="http://sample.wso
   </p:sayHello>
</body>
```

Before you invoke the service, go back to your **Eclipse Workspace** and put a debug point inside your method.

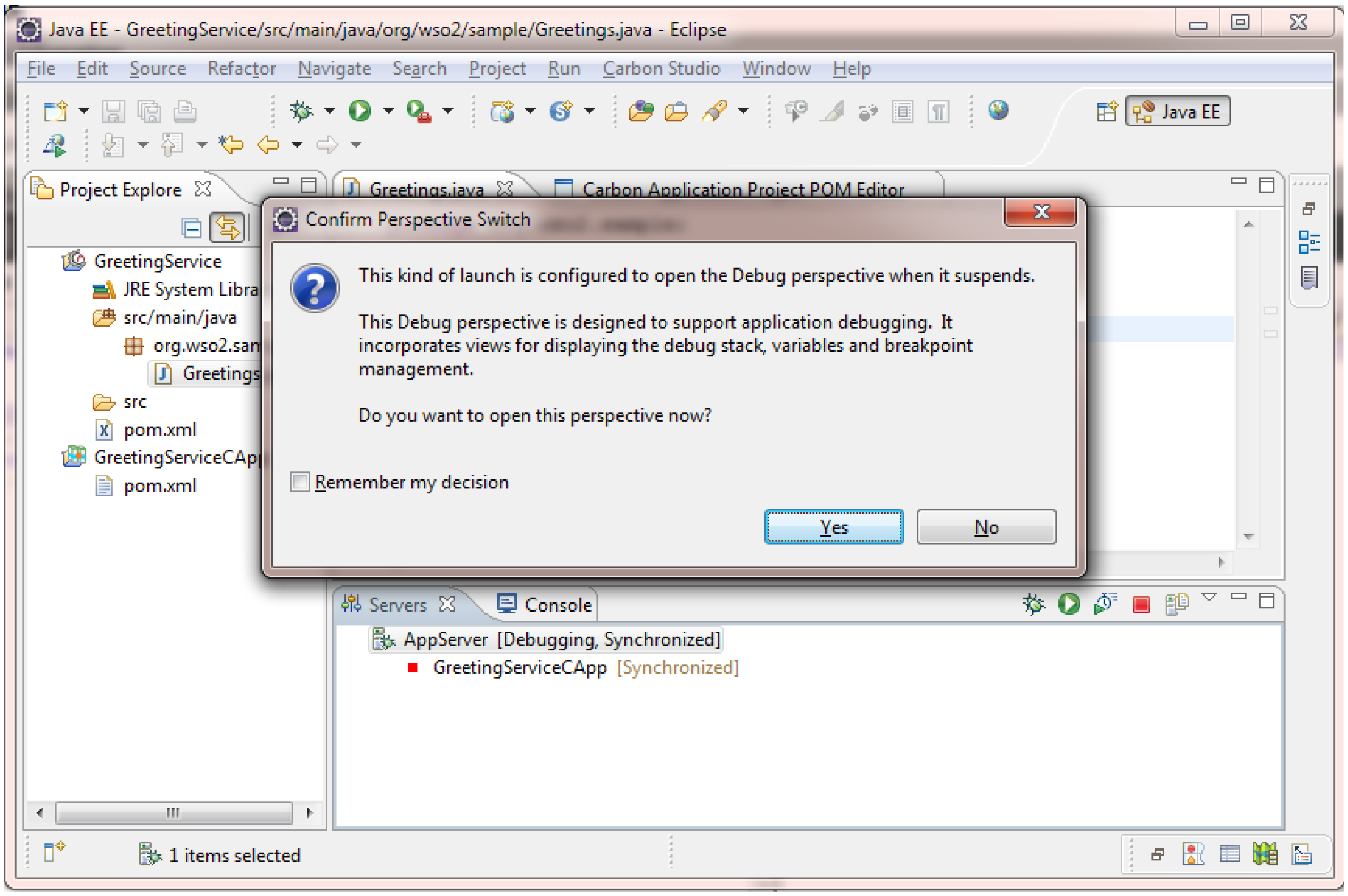

Java EE - GreetingService/src/main/java/org/wso2/sample/Greetings.java - Eclipse
File Edit Source Refactor Navigate Search Project Run Carbon Studio Window Help
Java EE
Project Explore
GreetingService
JRE System Library [jre6]
src/main/java
org.wso2.sample
Greetings.java
src
pom.xml
GreetingServiceCApp
pom.xml
Greetings.java
Carbon Application Project POM Editor
Servers
Console
AppServer [Debugging, Synchronized]
GreetingServiceCApp [Synchronized]
1 items selected

```
package org.wso2.sample;

public class Greetings{
    public String sayHello(String name){
        return "Hello " + name;
    }


}
```

Now go back to your web console and invoke the service. Once you invoke the service, it will ask whether to change in to the **Debug Perspective** in Eclipse.


Java EE - GreetingService/src/main/java/org/wso2/sample/Greetings.java - Eclipse
File Edit Source Refactor Navigate Search Project Run Carbon Studio Window Help
Java EE
Project Explore
Greetings.java
Carbon Application Project POM Editor
GreetingService
JRE System Libra
src/main/java
org.wso2.san
Greetings
src
pom.xml
GreetingServiceCAp
pom.xml
Servers
Console
AppServer [Debugging, Synchronized]
GreetingServiceCApp [Synchronized]
1 items selected

Confirm Perspective Switch

This kind of launch is configured to open the Debug perspective when it suspends.

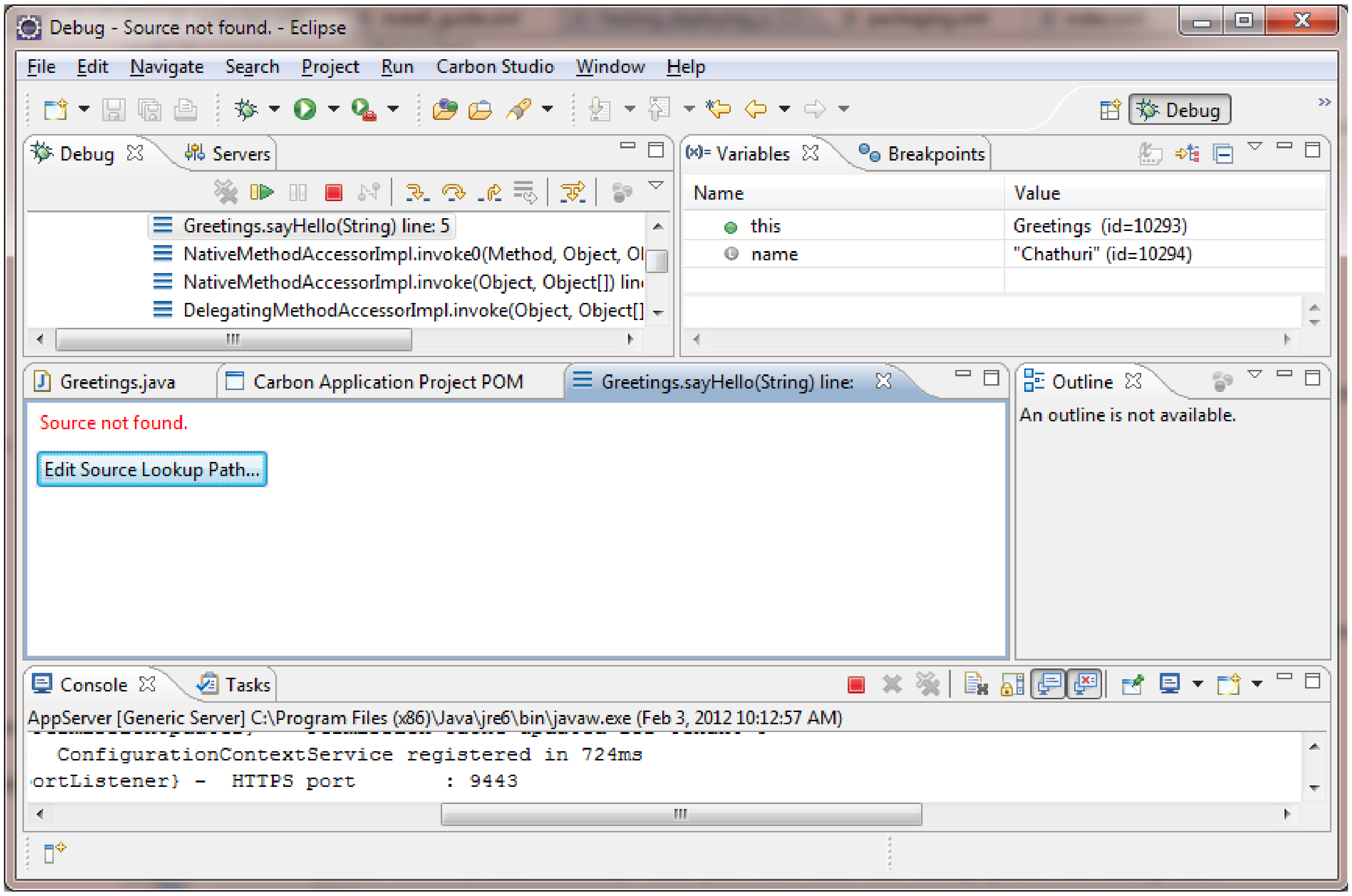
This Debug perspective is designed to support application debugging. It incorporates views for displaying the debug stack, variables and breakpoint management.

Do you want to open this perspective now?

Remember my decision

Yes No

Click **Yes**. If your Eclipse instance could not find the relavent source for the class, it will show a blank page as below.


Debug - Source not found. - Eclipse
File Edit Navigate Search Project Run Carbon Studio Window Help
Debug
Debug Servers
Variables Breakpoints
Greetings.java
Carbon Application Project POM
Greetings.sayHello(String) line:
Outline
An outline is not available.
Edit Source Lookup Path...
Console Tasks

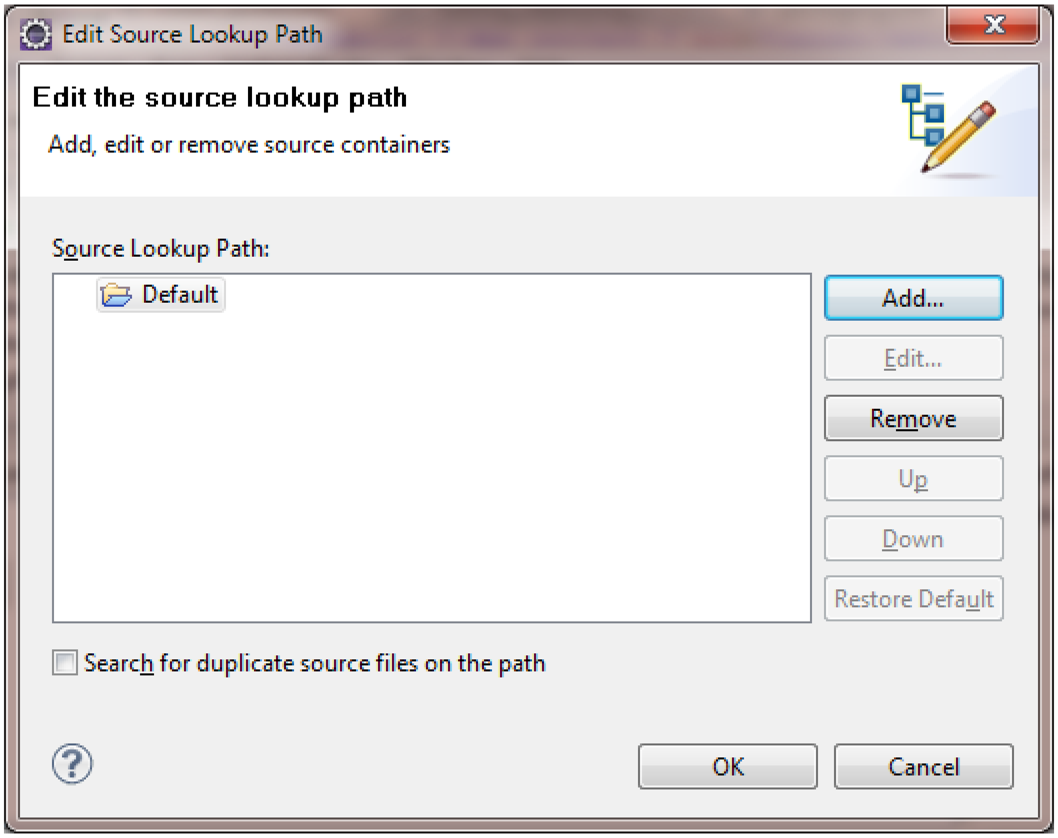
Greetings.sayHello(String) line: 5
NativeMethodAccessorImpl.invoke0(Method, Object, Ol
NativeMethodAccessorImpl.invoke(Object, Object[]) lin
DelegatingMethodAccessorImpl.invoke(Object, Object[]

Name	Value
this	Greetings (id=10293)
name	"Chathuri" (id=10294)

Source not found.

```
AppServer [Generic Server] C:\Program Files (x86)\Java\jre6\bin\javaw.exe (Feb 3, 2012 10:12:57 AM)
  ConfigurationContextService registered in 724ms
ortListener} -  HTTPS port        : 9443
```

Click **Edit Source Lookup Path**.



Edit the source lookup path

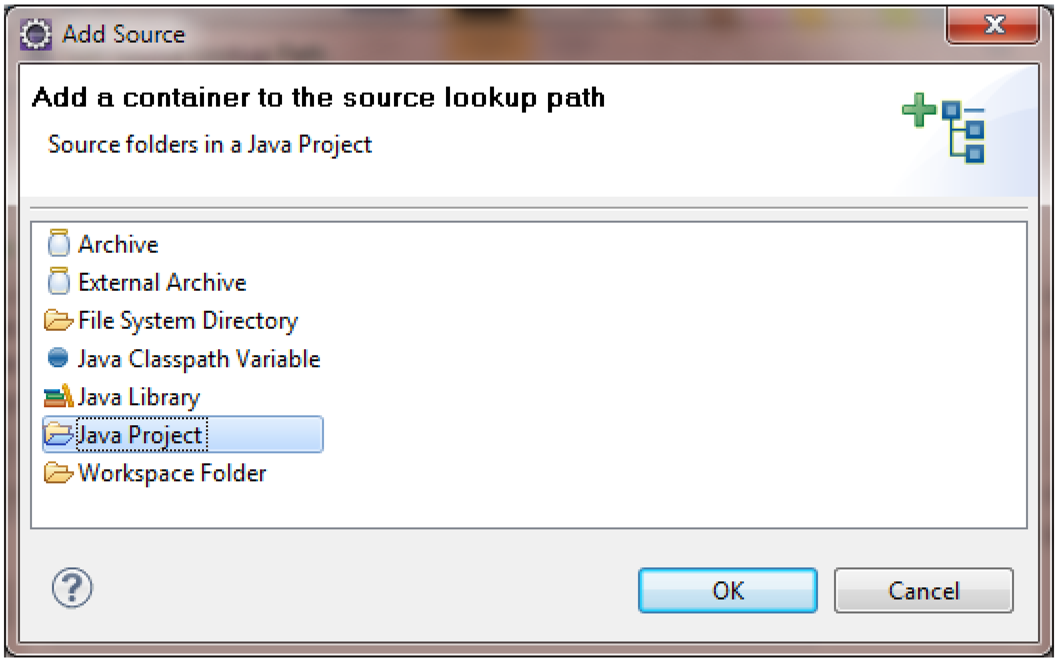
Add, edit or remove source containers

Source Lookup Path:

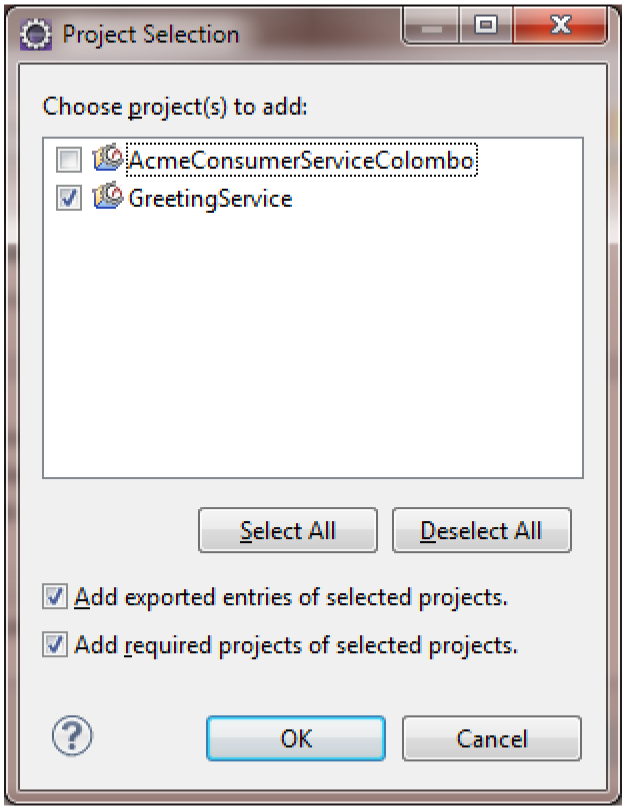
Default

☐ Search for duplicate source files on the path

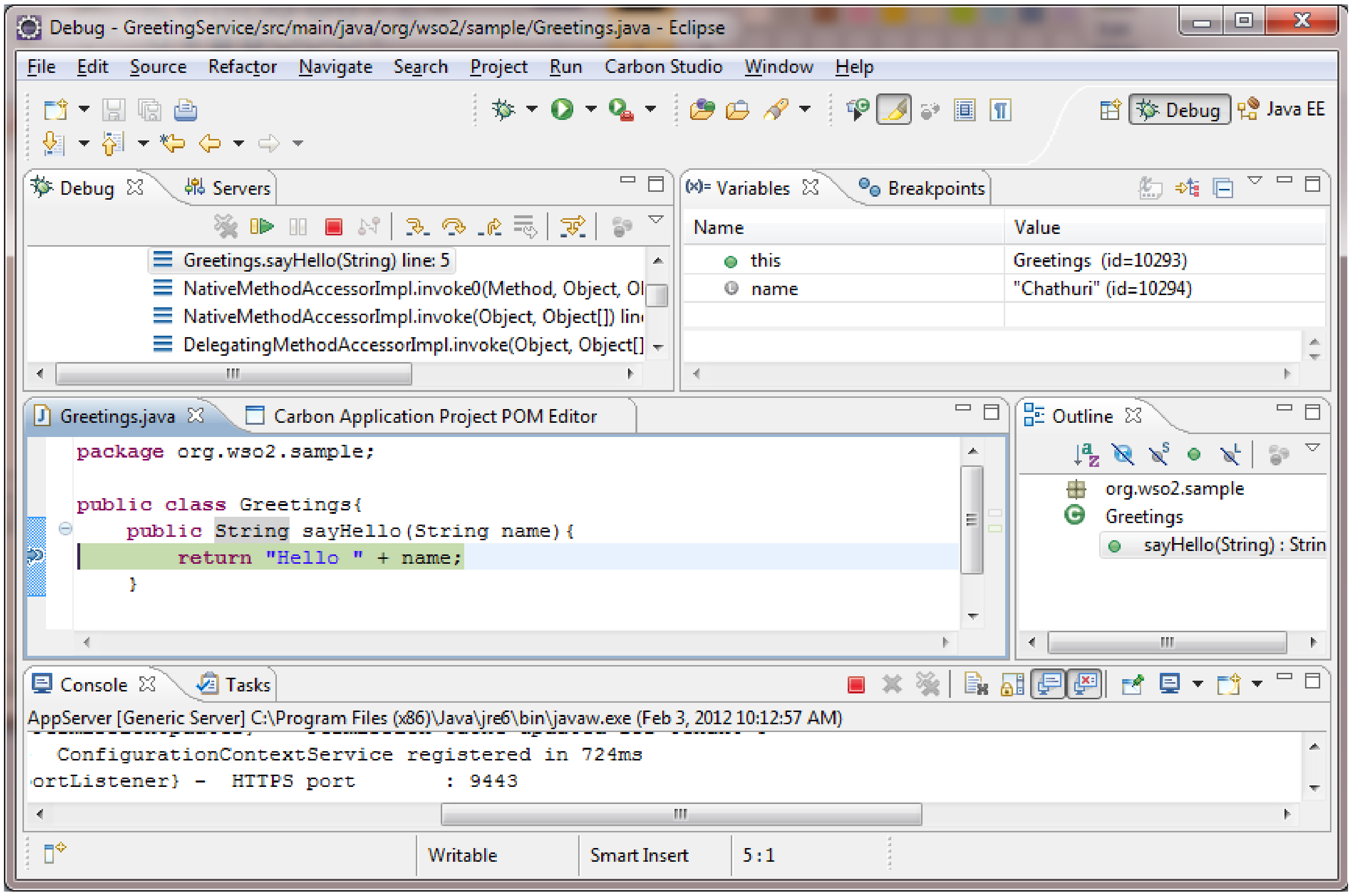
Click **Add** and select the **Java Project**.



Select the **GreetingService** project from the list and click **OK**.


Project Selection
Choose project(s) to add:
AcmeConsumerServiceColombo
GreetingService
Select All
Deselect All
Add exported entries of selected projects.
Add required projects of selected projects.
OK
Cancel

Now you will see the source being attached and you will see the debug point being hit.


Debug - GreetingService/src/main/java/org/wso2/sample/Greetings.java - Eclipse
File Edit Source Refactor Navigate Search Project Run Carbon Studio Window Help
Debug
Java EE
Debug
Servers
Variables
Breakpoints
Greetings.java
Carbon Application Project POM Editor
Outline
Console
Tasks
Writable
Smart Insert
5 : 1

```
Greetings.sayHello(String) line: 5
NativeMethodAccessorImpl.invoke0(Method, Object, Ol
NativeMethodAccessorImpl.invoke(Object, Object[]) lin
DelegatingMethodAccessorImpl.invoke(Object, Object[]
```

Name	Value
this	Greetings (id=10293)
name	"Chathuri" (id=10294)

```
package org.wso2.sample;

public class Greetings{
    public String sayHello(String name){
        return "Hello " + name;
    }
```

```
org.wso2.sample
Greetings
  sayHello(String) : Strin
```

```
AppServer [Generic Server] C:\Program Files (x86)\Java\jre6\bin\javaw.exe (Feb 3, 2012 10:12:57 AM)
 ConfigurationContextService registered in 724ms
ortListener} -  HTTPS port       : 9443
```

In a similar manner, you can debug any application which will be associated with **Java Sources**.