

Developing a JAX-WS EJB Stateless Session Bean Web Service

{scrollbar}

This tutorial will take you through the steps required in developing, deploying and testing a EJB Stateless Session Bean Web Service in Apache Geronimo. After completing this tutorial you should be able to understand how to develop simple JAX-WS compliant EJB web services in Apache Geronimo using Eclipse development environment.

It is highly recommended that you go through the [Developing a JAX-WS POJO Web Service](#) tutorial before jumping into this. We will be using the same sample from the tutorial to develop a EJB Web Service.

Client Development

Client development is excluded from this tutorial because there is no difference in creating a client for EJB Stateless Session Bean Web Service and a POJO Web Service.

You can refer to [Developing a JAX-WS POJO Web Service](#) tutorial for further knowledge about how to develop a client for Web Services.

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

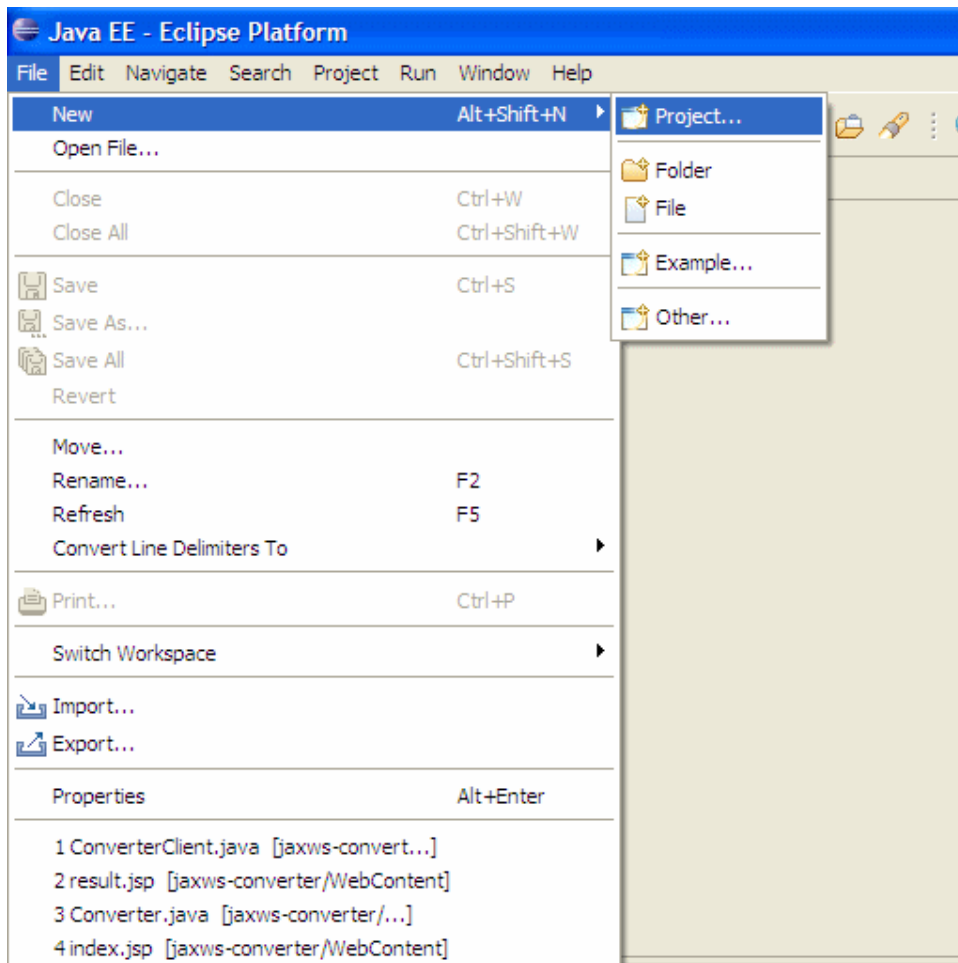
- Sun JDK 5.0+ (J2SE 1.5)
- Eclipse 3.3.1.1 (Eclipse Classic package of Europa distribution), which is platform specific
- Web Tools Platform (WTP) 2.0.1
- Data Tools Platform (DTP) 1.5.1
- Eclipse Modeling Framework (EMF) 2.3.1
- Graphical Editing Framework (GEF) 3.3.1

Details on installing eclipse are provided in the [Development environment](#) section. This tutorial will take you through the following steps:

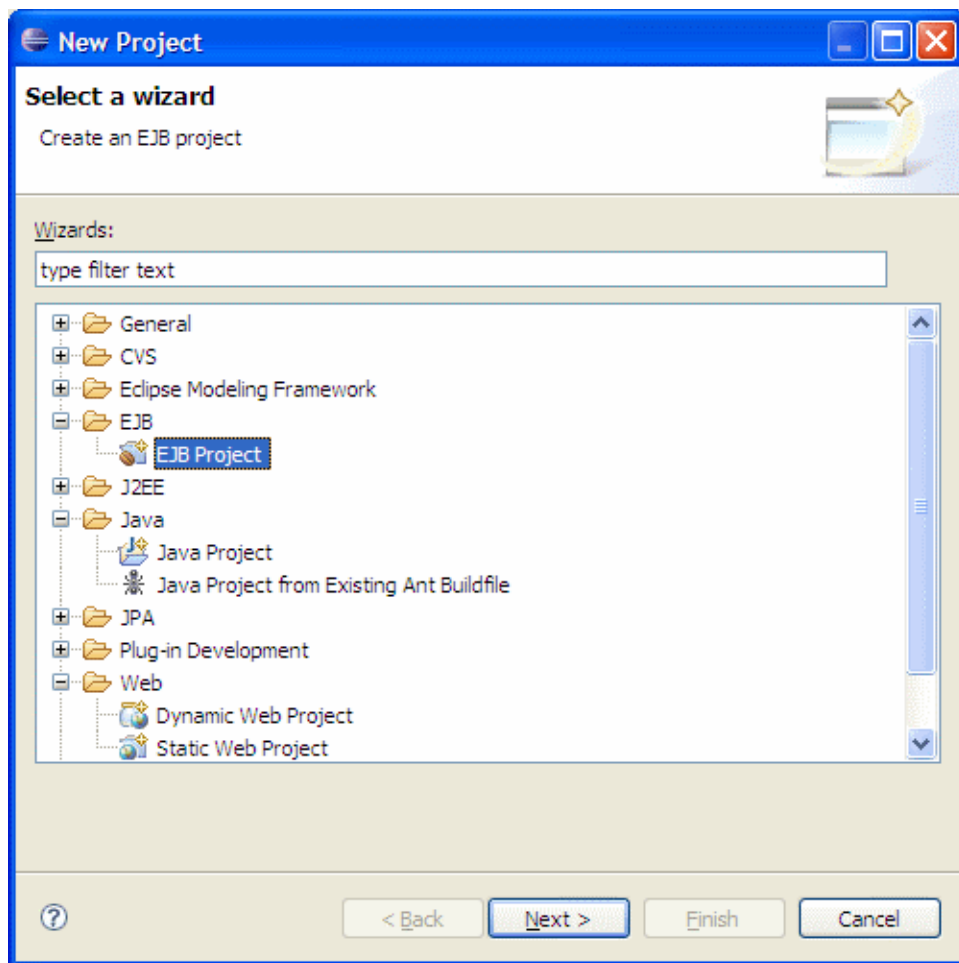
2listpipe

Setting Up Eclipse for Application Development

1. Create a Stateless Session EJB Project
 - Select **File->New->Project**



- In the popup window, select **EJB->EJB Project** and then click **Next**



- Type **jaxws-converterejb** as the **Project Name** and click **Next**..

New EJB Project

Create an EJB Project and add it to a new or existing Enterprise Application.

Project name:

Project contents:

☒ Use default

Directory:

Target Runtime

Configurations

A good starting point for working with Apache Geronimo v2.1 runtime. Additional facets can later be installed to add new functionality to the project.

EAR Membership

☐ Add project to an EAR

EAR Project Name:

- The default option should work for Geronimo, so click **Next**

New EJB Project

Select the facets that should be enabled for this project.

Configurations:

Project Facet	Version
<input checked="" type="checkbox"/> EJB Module	3.0
<input type="checkbox"/> EJBDoclet (XDoclet)	1.2.3
<input checked="" type="checkbox"/> Geronimo Deployment	1.2
<input checked="" type="checkbox"/> Java	5.0
<input type="checkbox"/> Java Persistence	1.0

- Here also default options will work, but the service and wsdl file will be located at the URL specified by the annotations

New EJB Project

EJB Module
Configure ejb module settings.

Source Folder:
ejbModule

☐ Create an EJB Client JAR module to hold the client interfaces and classes.

Name:
jaxws-converterejbClient

Client JAR URI:
jaxws-converterejbClient.jar

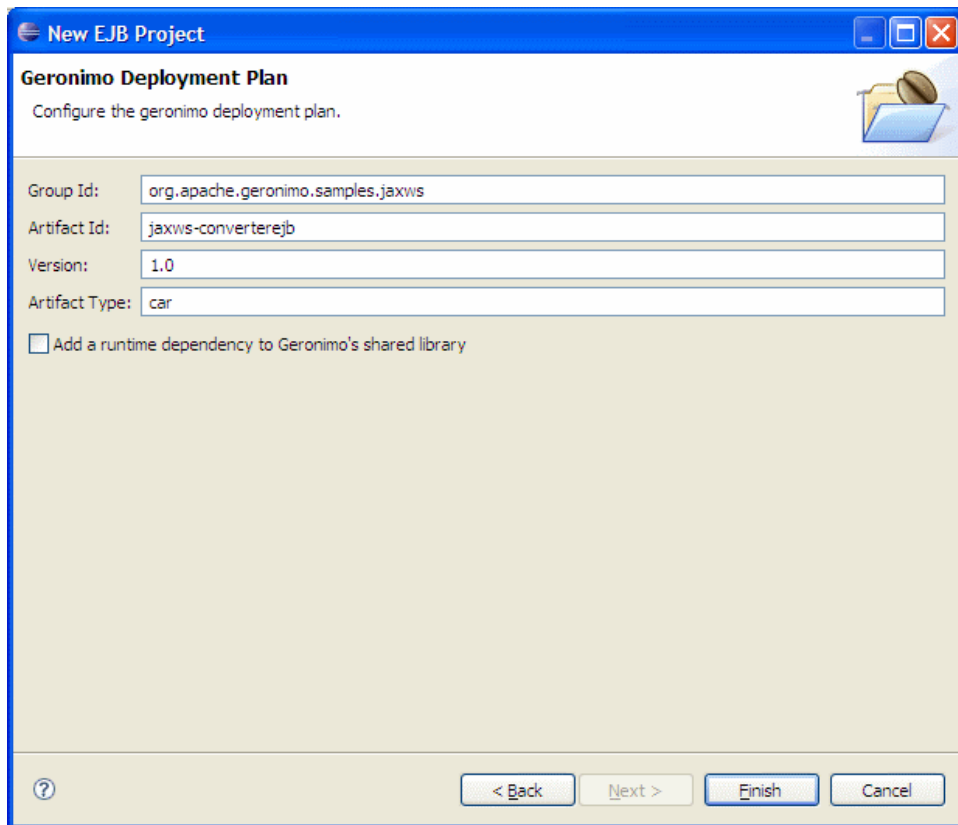
☐ Generate Deployment Descriptor

? < Back Next > Finish Cancel

Changing Service Location

If you want to change the service location to any custom URL you want, make sure that the check box **Generate Deployment Descriptor** is selected.

- Modify the **Group Id** to **org.apache.geronimo.samples.jaxws** and the **artifact id** to **jaxws-converterejb**.



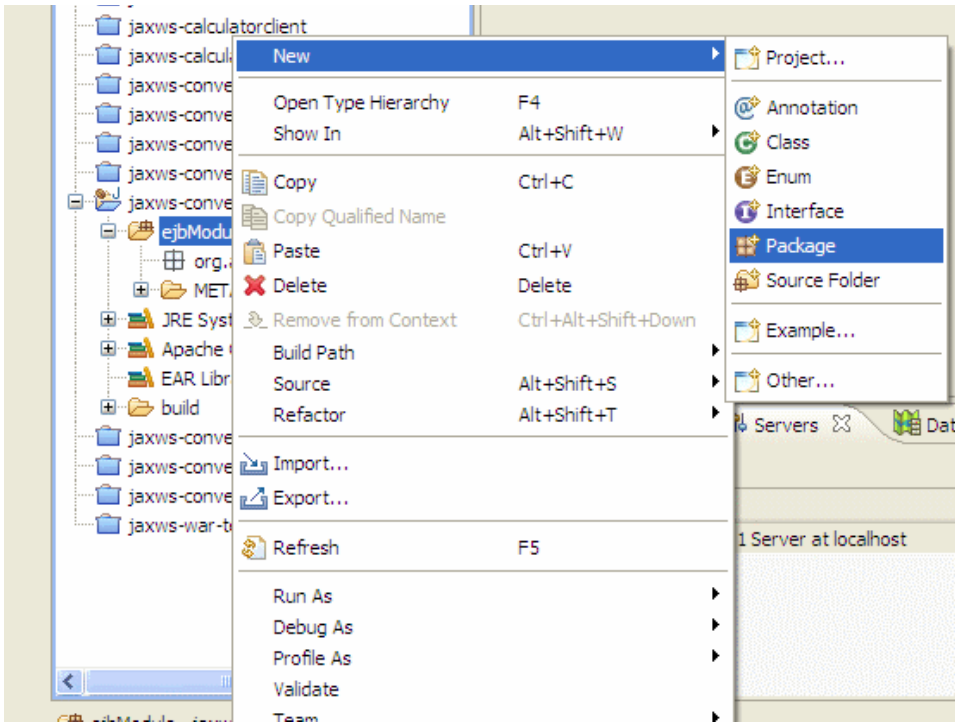
- Click **Finish**

This completes the configuration of Eclipse for application development.

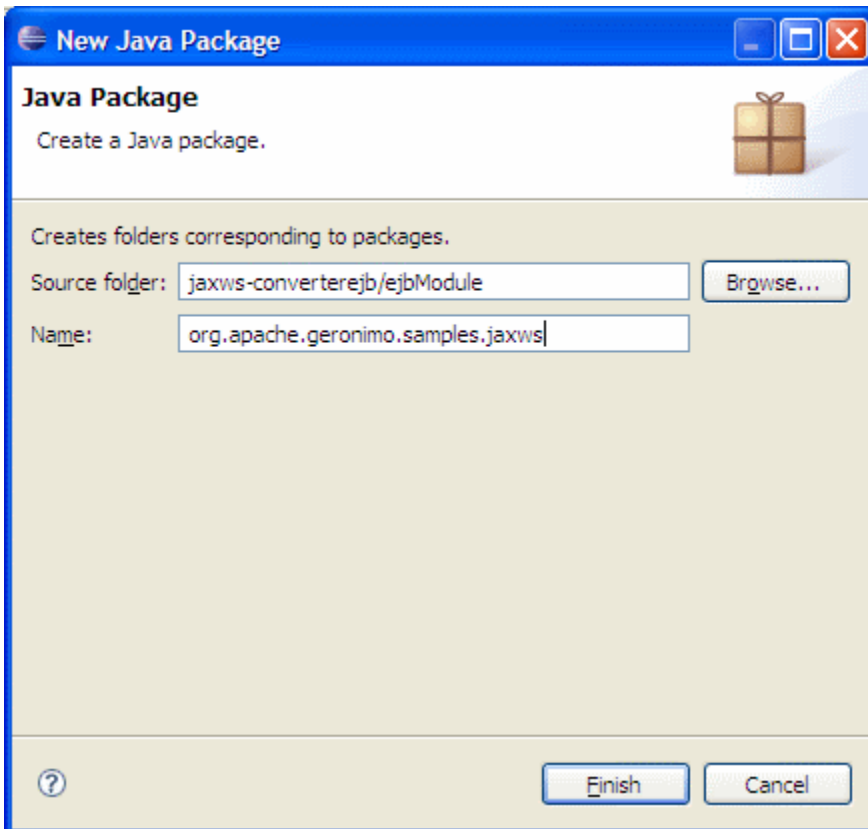
Creating the Web Services Implementation code

To deploy the Converter EJB Service, we are going to create a remote interface and a stateless session bean that implements the interface. The steps required are as follows:

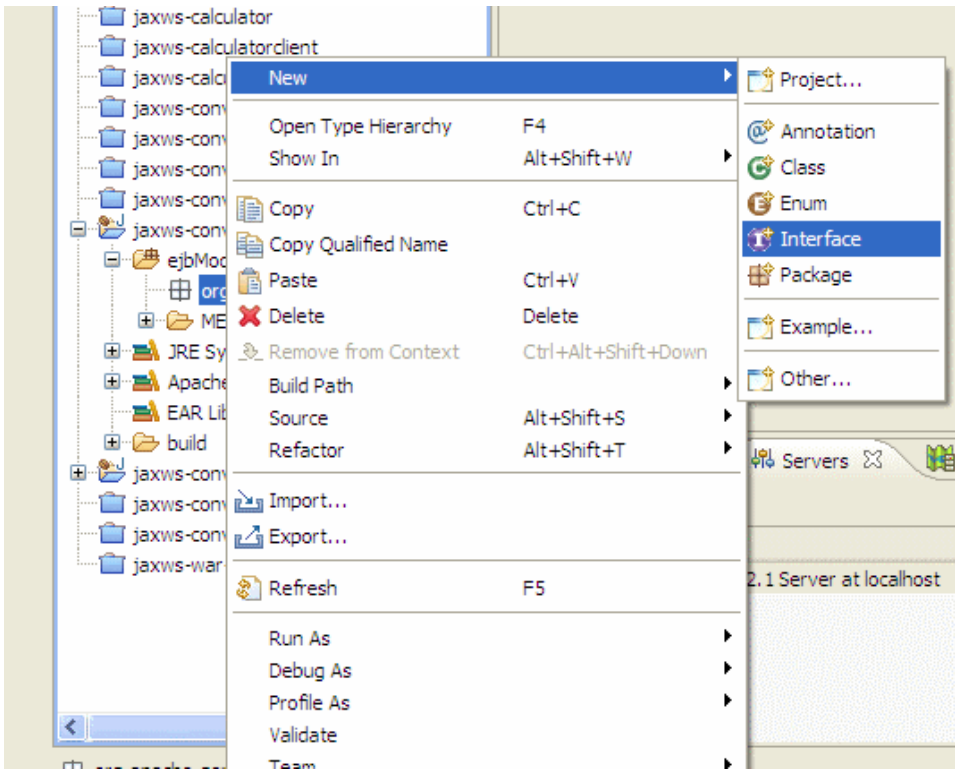
1. Right click on **ejbModule** and select **New->Package**



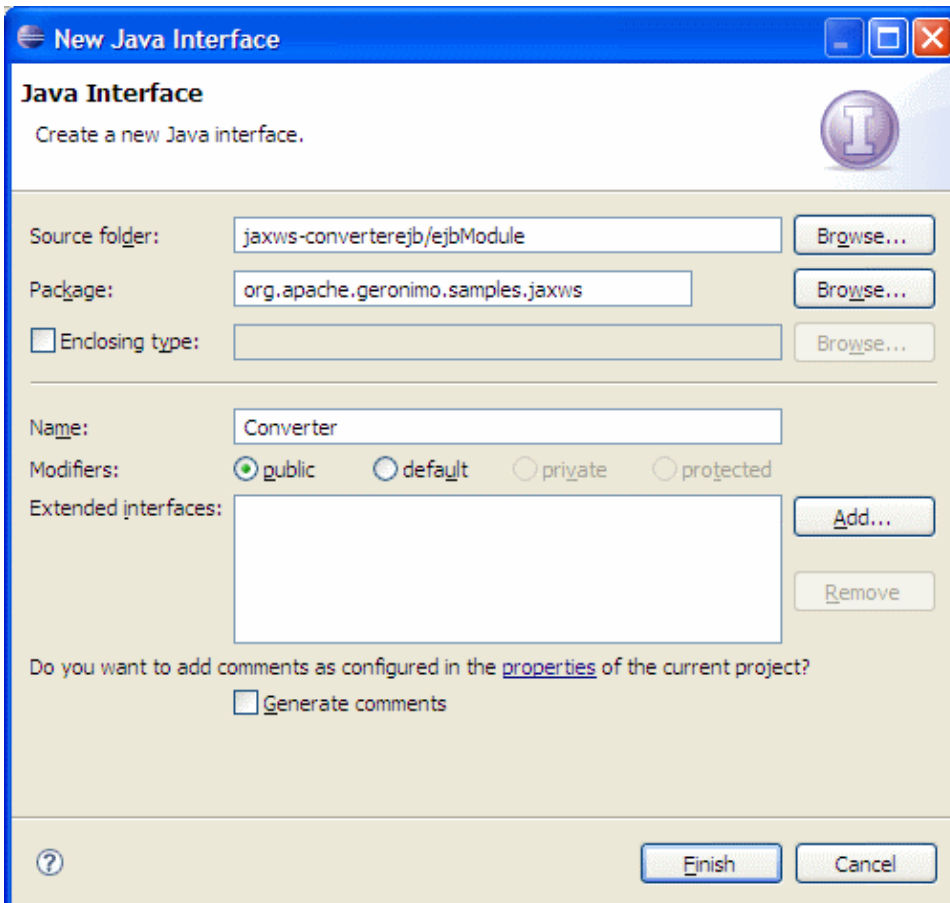
2. Name the package to **org.apache.geronimo.samples.jaxws** and click **Finish**



3. Right click on the new package and select **New->Interface**

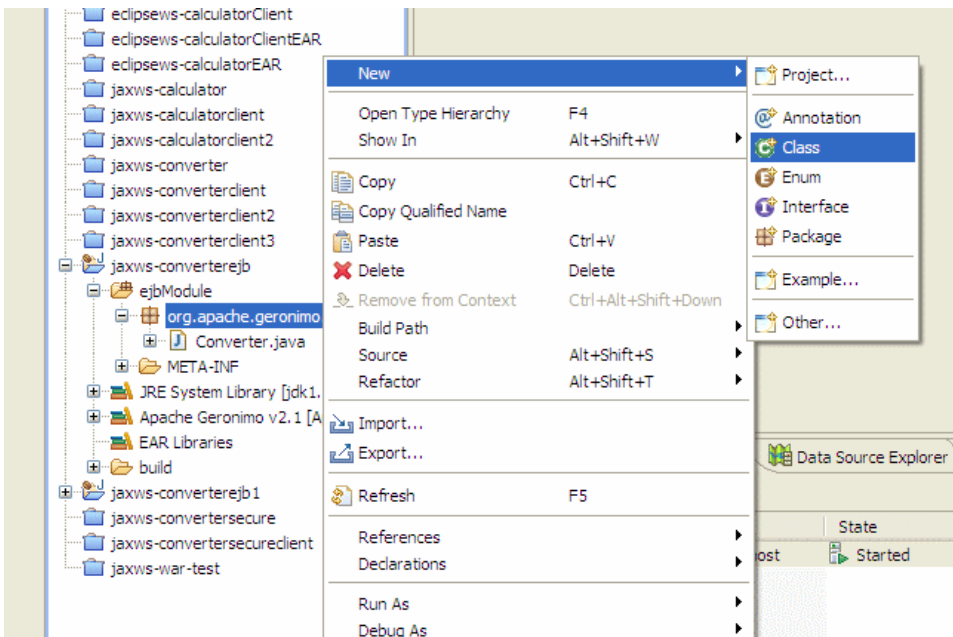


4. Name the interface as **Converter** and click **Finish**

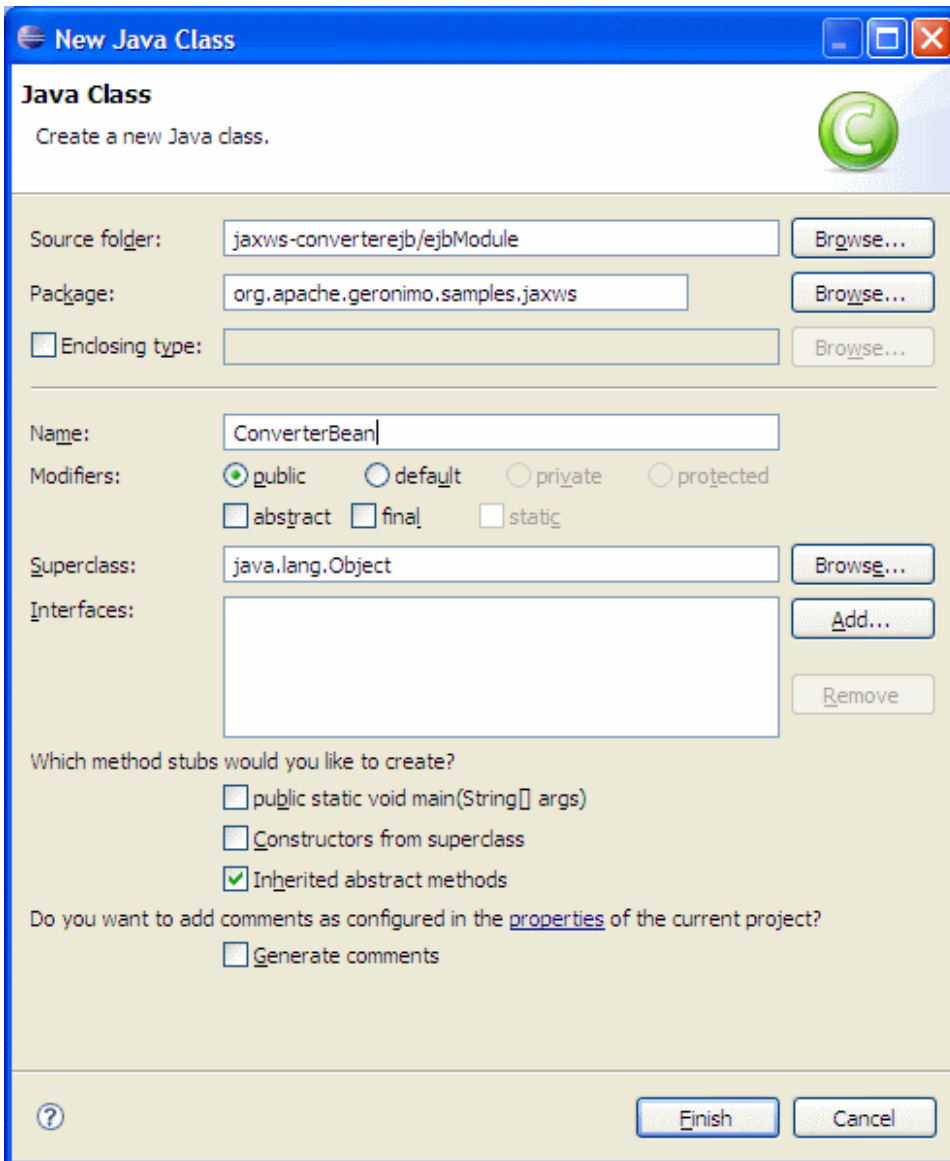


5. Add the following code to the Converter class Converter.javasolid package org.apache.geronimo.samples.jaxws; import java.math.BigDecimal; import javax.ejb.Remote; import javax.jws.WebService; @Remote @WebService(name = "ConverterPortType", targetNamespace = "http://jaxws.samples.geronimo.apache.org") public interface Converter { public BigDecimal dollarToRupees(BigDecimal dollars); public BigDecimal rupeesToEuro(BigDecimal rupees); }

1. Right click on the new package and select **New->Class**



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



3. Add the following code to the ConverterBean class ConverterBean.javasolid package org.apache.geronimo.samples.jaxws; import java.math.BigDecimal; import javax.ejb.*; import javax.jws.WebService; @Stateless @WebService(serviceName = "Converter", portName = "ConverterPort", endpointInterface = "org.apache.geronimo.samples.jaxws.Converter", targetNamespace = "http://jaxws.samples.geronimo.apache.org") public class ConverterBean implements Converter { private BigDecimal rupeeRate = new BigDecimal("40.58"); private BigDecimal euroRate = new BigDecimal("0.018368"); public BigDecimal dollarToRupees(BigDecimal dollars) { BigDecimal result = dollars.multiply(rupeeRate); return result.setScale(2, BigDecimal.ROUND_UP); } public BigDecimal rupeesToEuro(BigDecimal rupees) { BigDecimal result = rupees.multiply(euroRate); return result.setScale(2, BigDecimal.ROUND_UP); } }

A little observation

If you have already gone through the [Developing a JAX-WS POJO Web Service](#) tutorial, you might have observed that there's not much difference between an EJB Web Service and POJO Web Service except for the EJB specific annotations.

This completes the development of the Web Service Implementation code.

Setting Up the Deployment Descriptor and Deployment Plan

Geronimo default location

If you are comfortable with the location that Geronimo deploys the EJB Web Service, you can skip this section and go to Deploy and Test Section.

1. Expand the **META-INF** directory present under **ejbModule** and add the following code to **ejb-jar.xml**

ejb-jar.xml

If ejb-jar.xml is not present, create a XML file and name it as ejb-jar.xml

1. Add the following code to ejb-jar.xml ejb-jar.xmlsolid <?xml version="1.0" encoding="UTF-8"?> <ejb-jar version="3.0" xmlns="http://java.sun.com/xml/ns/javaee" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="http://java.sun.com/xml/ns/javaee http://java.sun.com

```
com/xml/ns/javaee/ejb-jar_3_0.xsd"> <display-name>jaxws-converterejb</display-name> <enterprise-beans> <session> <ejb-name>jaxws-converterejb</ejb-name> <service-endpoint>org.apache.geronimo.samples.jaxws.Converter</service-endpoint> <ejb-class>org.apache.geronimo.samples.jaxws.ConverterBean</ejb-class> <session-type>Stateless</session-type> <transaction-type>Container</transaction-type> </session> </enterprise-beans> </ejb-jar>
```

1. Also add the following code to the `openejb-jar.xml` present at the same location. `openejb-jar.xml`

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<ns4:openejb-jar xmlns="http://geronimo.apache.org/xml/ns/j2ee/connector-1.2" xmlns:ns2="http://geronimo.apache.org/xml/ns/deployment-1.2" xmlns:ns3="http://geronimo.apache.org/xml/ns/naming-1.2" xmlns:ns4="http://openejb.apache.org/xml/ns/openejb-jar-2.2" xmlns:ns5="http://openejb.apache.org/xml/ns/pkgen-2.1" xmlns:ns6="http://geronimo.apache.org/xml/ns/j2ee/application-2.0" xmlns:ns7="http://geronimo.apache.org/xml/ns/security-2.0" xmlns:ns8="http://geronimo.apache.org/xml/ns/j2ee/web-2.0.1" xmlns:ns9="http://java.sun.com/xml/ns/persistence" xmlns:ns10="http://geronimo.apache.org/xml/ns/j2ee/application-client-2.0">
  <ns2:environment>
    <ns2:moduleId>
      <ns2:groupId>org.apache.geronimo.samples.jaxws</ns2:groupId>
      <ns2:artifactId>jaxws-converterejb</ns2:artifactId>
      <ns2:version>1.0</ns2:version>
      <ns2:type>car</ns2:type>
    </ns2:moduleId>
  </ns2:environment>
  <ns4:enterprise-beans>
    <ns4:session>
      <ns4:ejb-name>jaxws-converterejb</ns4:ejb-name>
      <ns4:web-service-address>ADD_CUSTOM_URL</ns4:web-service-address>
    </ns4:session>
  </ns4:enterprise-beans>
</ns4:openejb-jar>
```

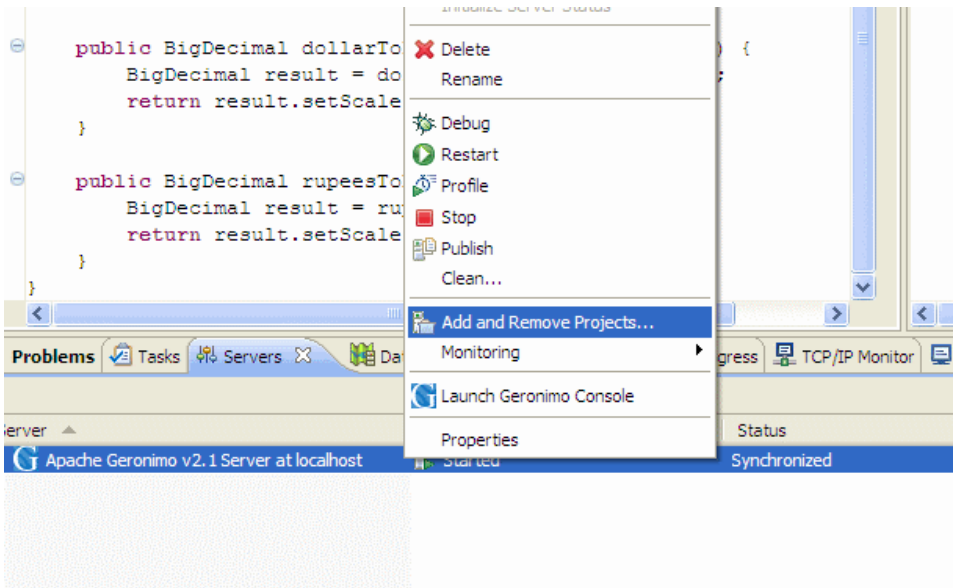
This completes the setting up of Deployment descriptor and Deployment Plan.

Deploy and Test the Web Service

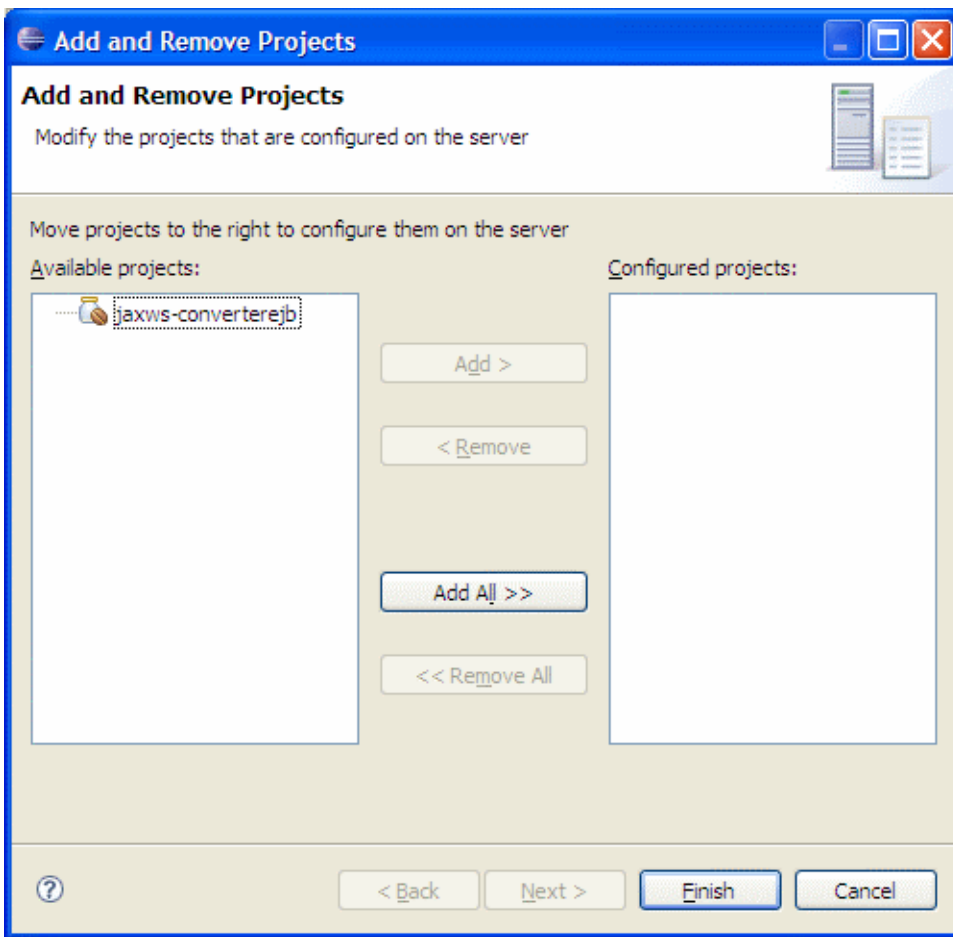
Now, we will look into the steps involved in deploying and testing our web service without any clients.

Deploy

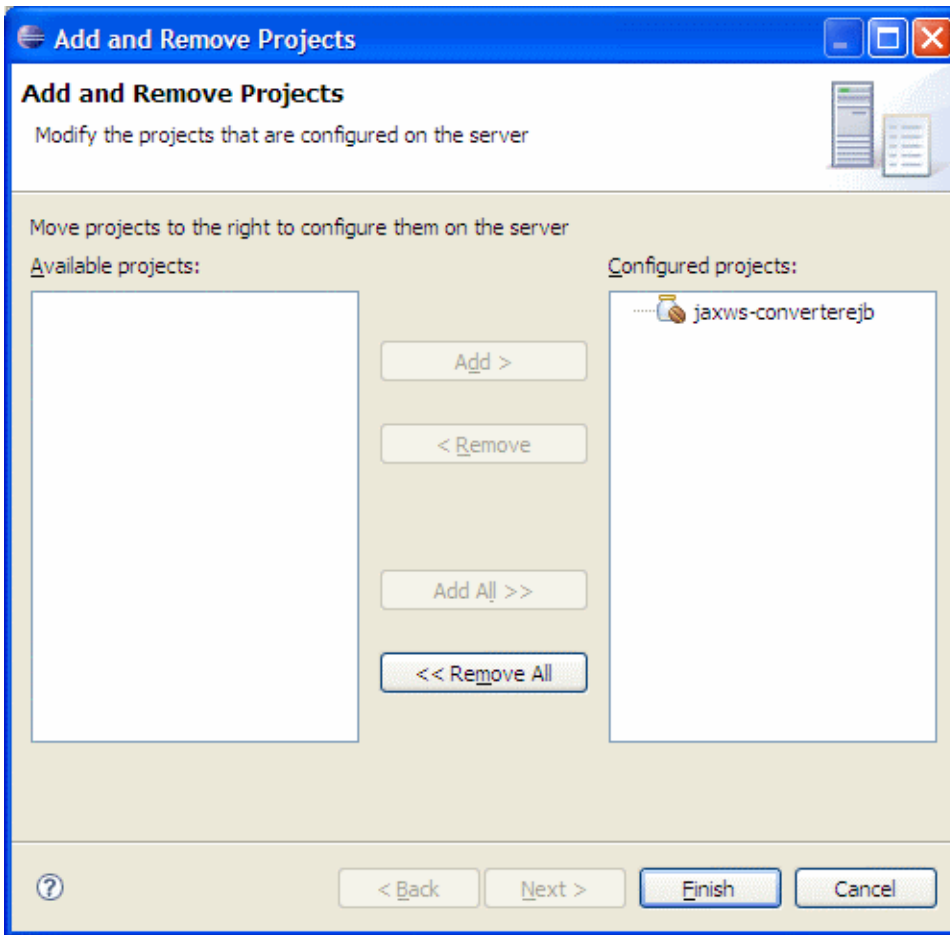
1. Right click on the **Apache Geronimo** Server Runtime present in the servers view and select **Add or Remove Projects**



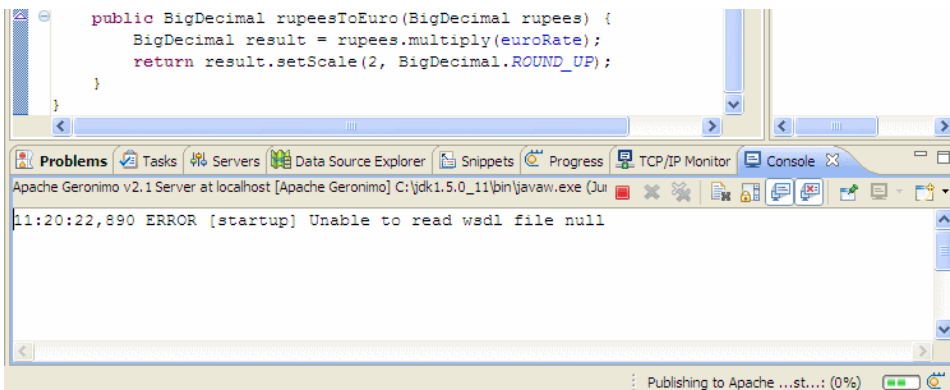
2. In the popup dialog, select the **jaxws-converterejb** project and click **Add**



3. Make sure that **jaxws-converter** is in the **configured projects** list and then click **Finish**



4. Wait for some time till the server status is changed to synchronized. Errors at Deploy time
If you see any errors at deploy time like **Unable to read WSDL file null**, simply ignore them.



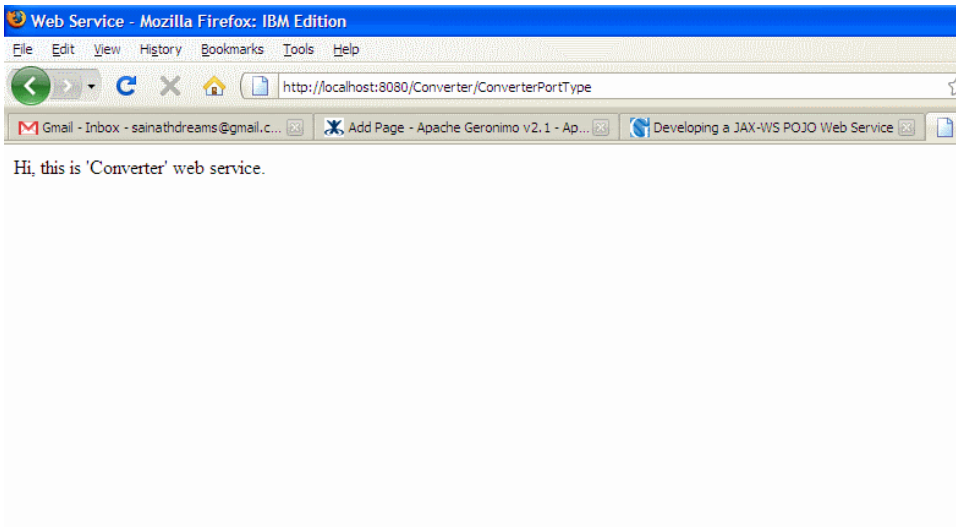
Testing

1. Once the application is deployed on to the server, Launch a browser and go to the following url.
<http://localhost:8080/Converter/ConverterPortType>

Custom location

If you have followed the steps to deploy the service onto custom location, go to the url http://localhost:8080/CUSTOM_URL

1. Now you should see the screen telling that service is located at the following URL



Using Web Service Explorer in Eclipse

You can also use Web Services Explorer present in Eclipse to rapidly test your web service without developing a client.

To know how to use Web Services Explorer in Eclipse, one can refer to the [Developing a JAX-WS POJO Web Service#Using Web Services Explorer in Eclipse](#)