Apache Axis2 - Hello World! using Eclipse

javahelps.com/2016/04/apache-axis2-hello-world-using-eclipse.html

The previous article Apache Axis2 - Hello World! provides step by step guide to develop a very basic Axis2 Hello World application without using any IDEs. This article helps you to create a simple application in Axis2 using Eclipse IDE.

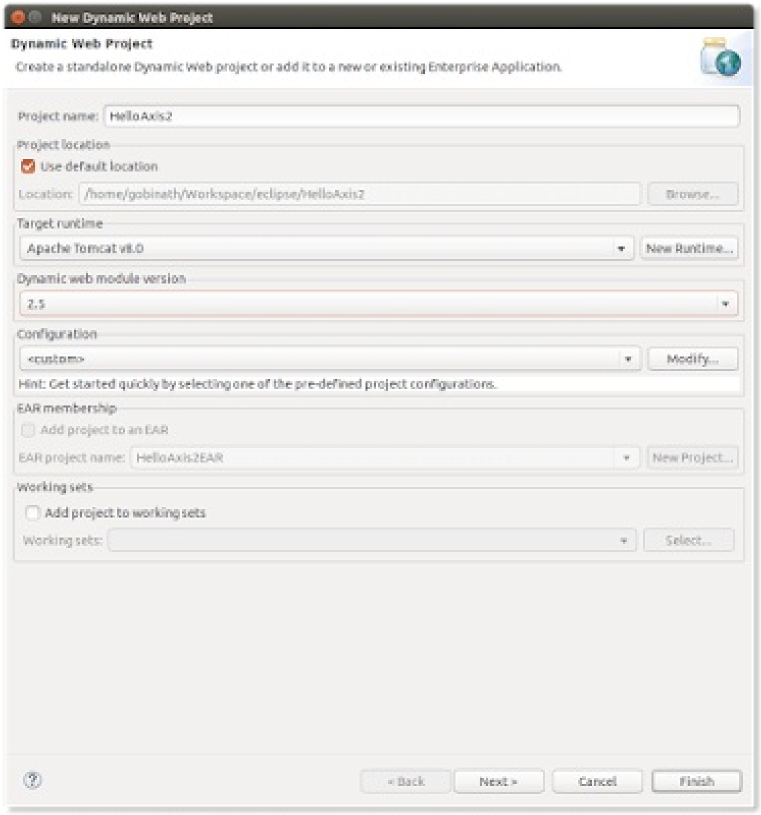
Prerequisite:

- Eclipse IDE for Java EE Developers (Follow this link to install Eclipse)
- Apache Tomcat (Follow this link to install and integrate with Eclipse)
- Apache Axis2 (Follow this link to install and integrate with Eclipse)

Update (01/05/2017): With the Axis2 library 1.7.4 there are some JSTL related errors in the JSP file. If you get such errors in your project, add the jstl-1.2.jar into the WebContent/lib directories.

Step 1:

Create a new Dynamic Web Project named *HelloAxis2* and change the *Dynamic web module version* to 2.5 because the current version of Axis2 core does not support versions higher than 2.5.


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: HelloAxis2

Project location

[✓] Use default location

Location: /home/gobinath/Workspace/eclipse/HelloAxis2

Target runtime: Apache Tomcat v8.0

Dynamic web module version: 2.5

Configuration: <custom>

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

EAR membership

[ ] Add project to an EAR

EAR project name: HelloAxis2EAR

Working sets

[ ] Add project to working sets

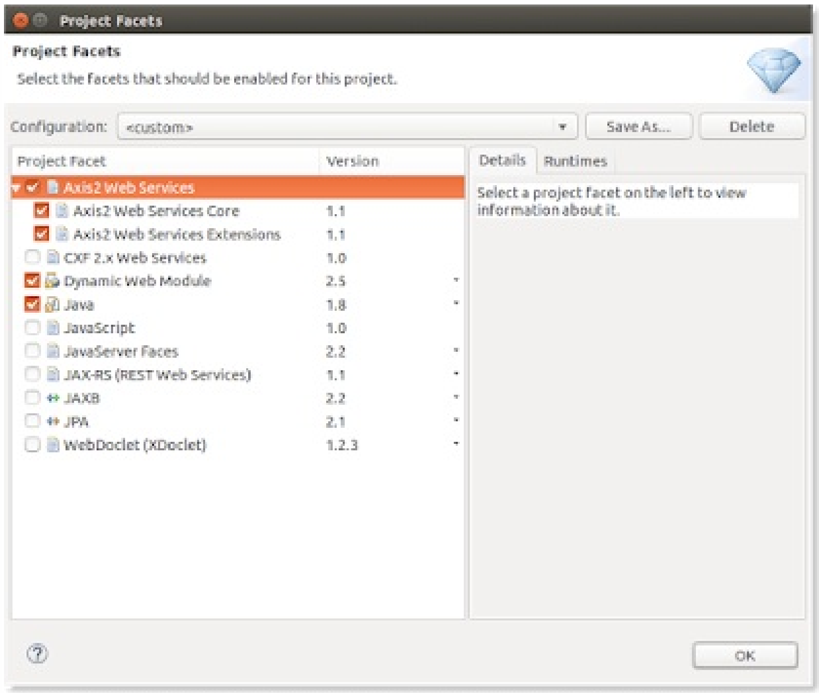
Working sets:

Step 2:

Click on the Modify button in Configuration sub-region.

Step 3:

Check *Axis2 Web Services*, click OK and click the Finish button.



Project Facets

Select the facets that should be enabled for this project.

Configuration: <custom>

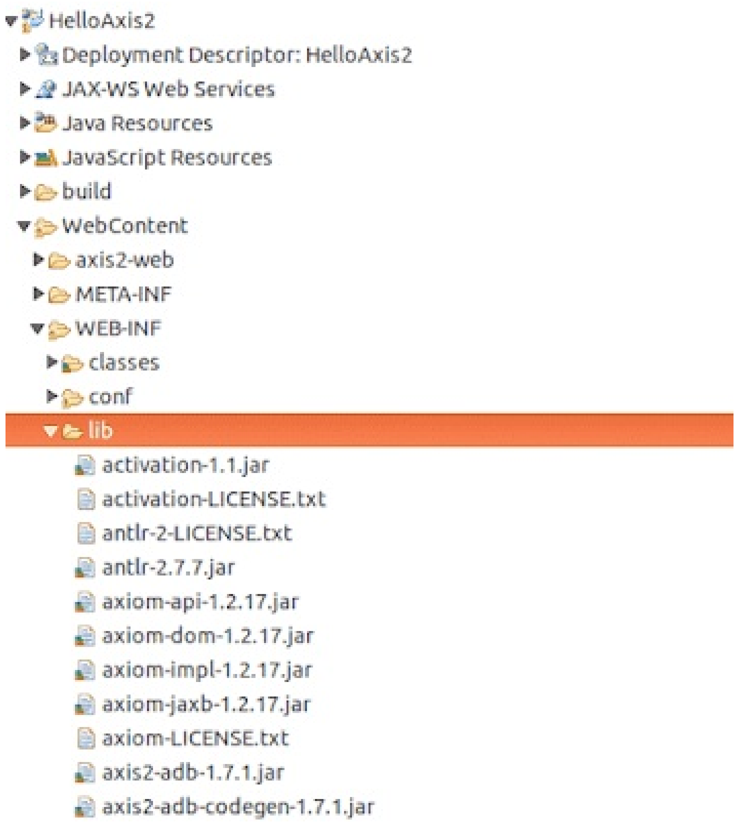
	Project Facet	Version
✓	Axis2 Web Services	
✓	Axis2 Web Services Core	1.1
✓	Axis2 Web Services Extensions	1.1
	CXF 2.x Web Services	1.0
✓	Dynamic Web Module	2.5
✓	Java	1.8
	JavaScript	1.0
	JavaServer Faces	2.2
	JAX-RS (REST Web Services)	1.1
	JAXB	2.2
	JPA	2.1
	WebDoclet (XDoclet)	1.2.3

Select a project facet on the left to view information about it.

Step 4:

From Axis2 1.7 onwards, Eclipse fails to copy *xmlschema-core* library to the project. It will throw NoClassDefFoundError: org/apache/ws/commons/schema/resolver/URIResolver when you run the project. To avoid this exception, copy the *xmlschema-core-x.x.x.jar* file from *$AXIS2_HOME/lib* folder to the Eclipse project directory *WebContent/WEB-INF/lib*.

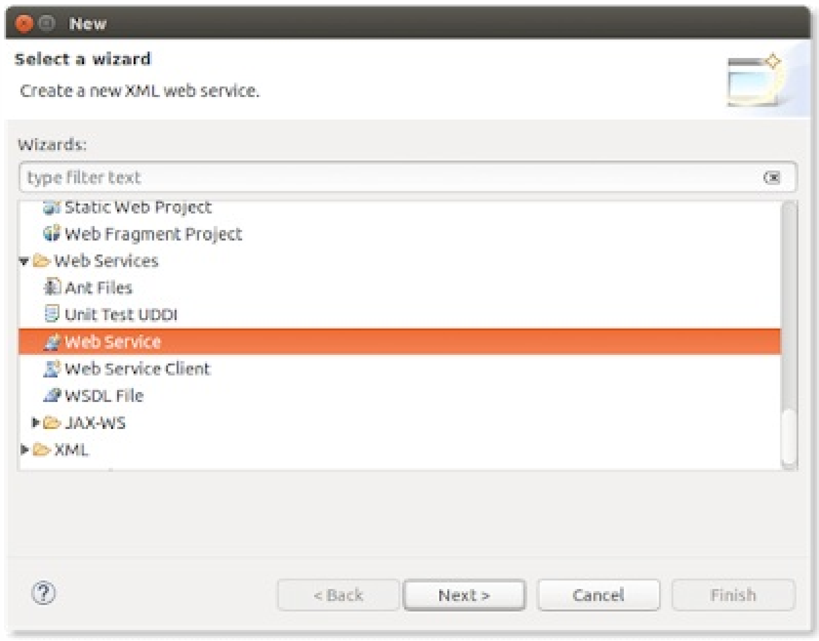
Step 5:
Create a new package *com.javahelps.helloaxis* in the *src* in the src folder and create a new class *HelloService* inside that package.



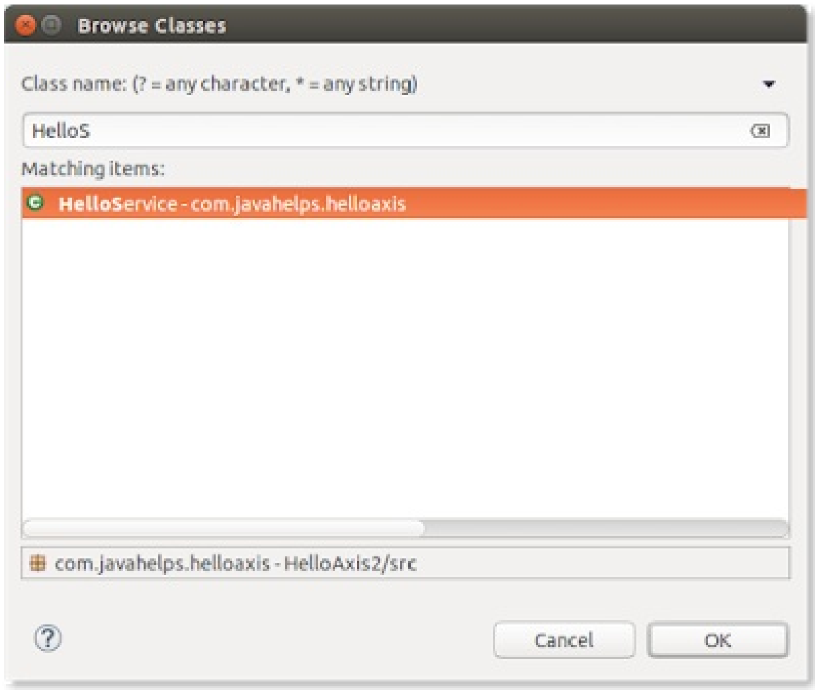
Step 6:
Add a method sayHello in the HelloService class as shown below. This is the business logic of our web service.

```
package com.javahelps.helloaxis;

public class HelloService {

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

}
```

Step 7: Right-click on the project and select New → Other → Web Service and click 'Next'



Step 8:
In the appeared dialog, click the Browse button and select the *HelloService* class as shown below.


Browse Classes

Class name: (? = any character, * = any string)

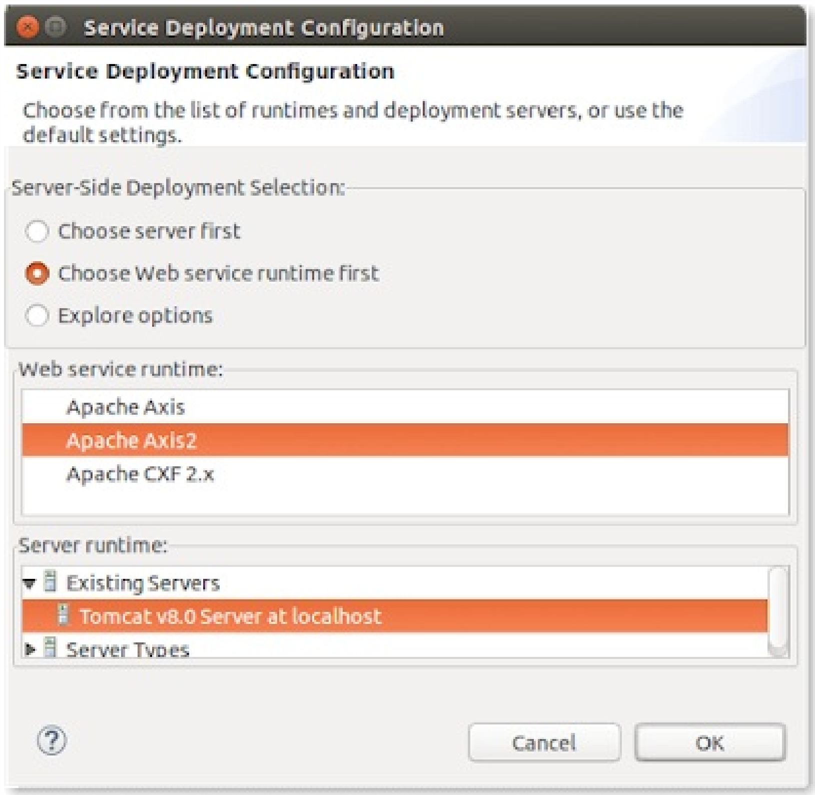
HelloS

Matching items:

HelloService - com.javahelps.helloaxis

com.javahelps.helloaxis - HelloAxis2/src

Step 9:
Click the "*Web service runtime: Apache Axis*" link under Configuration and select *Apache Axis 2*.


Service Deployment Configuration

Service Deployment Configuration

Choose from the list of runtimes and deployment servers, or use the default settings.

Server-Side Deployment Selection:

( ) Choose server first
(●) Choose Web service runtime first
( ) Explore options

Web service runtime:

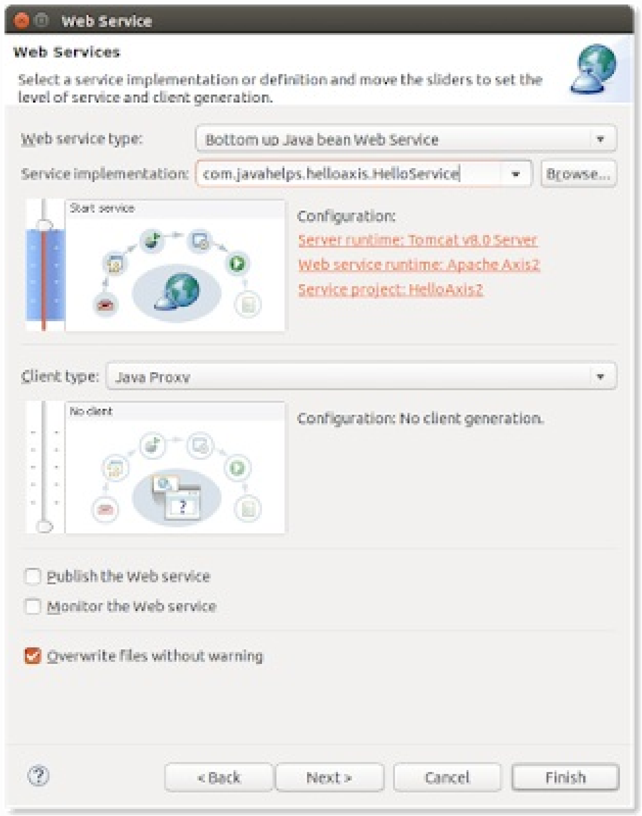
Apache Axis
Apache Axis2
Apache CXF 2.x

Server runtime:

Existing Servers
Tomcat v8.0 Server at localhost
Server Types

Step 10: Ensure that the "Service project" under Configuration is "HelloAxis2", which is your project name and click the 'Next'.

Step 11: Make sure that "Generate a default services.xml file" is selected and click "Finish".



Web Service

Web Services

Select a service implementation or definition and move the sliders to set the level of service and client generation.

Web service type: Bottom up Java bean Web Service

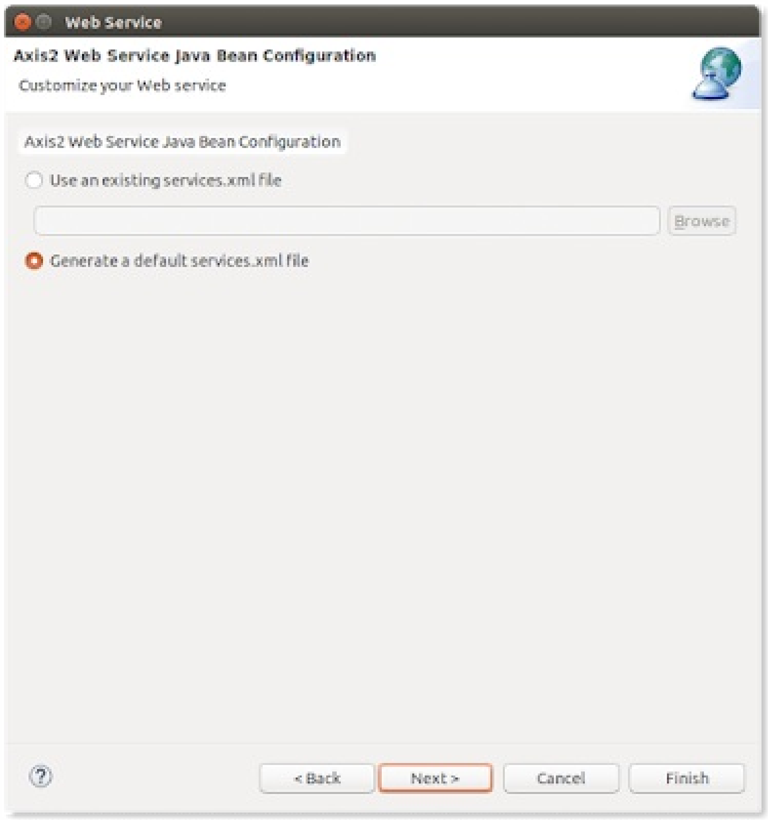
Service implementation: com.javahelps.helloaxis.HelloService

Configuration:
Server runtime: Tomcat v8.0 Server
Web service runtime: Apache Axis2
Service project: HelloAxis2

Client type: Java Proxy

Configuration: No client generation.

- [ ] Publish the Web service
- [ ] Monitor the Web service
- [x] Overwrite files without warning



Web Service

Axis2 Web Service Java Bean Configuration

Customize your Web service

Axis2 Web Service Java Bean Configuration

○ Use an existing services.xml file

● Generate a default services.xml file

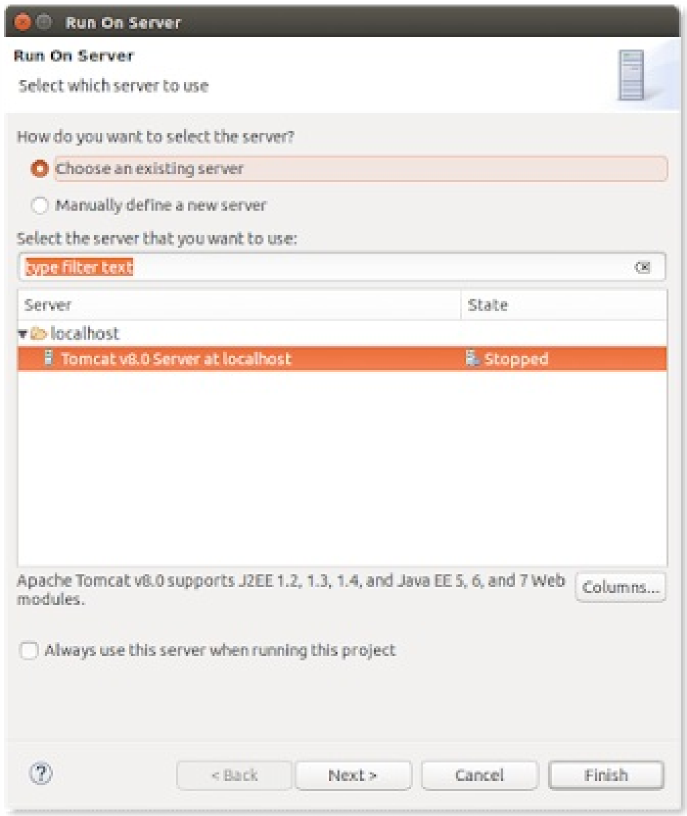
Step 12:
From Axis2 1.7 Message Exchange Pattern (MEP) URLS in the form http://www.w3.org/2004/08/wsdl/XXX and http://www.w3.org/2006/01/wsdl/XXX are no longer supported. Instead of them, we need to use http://www.w3.org/ns/wsdl/XXX. (Reference)

Therefore, open the *WebContent/WEB-INF/services/HelloService/META-INF/services.xml* file and change the messageReceivers as shown below.

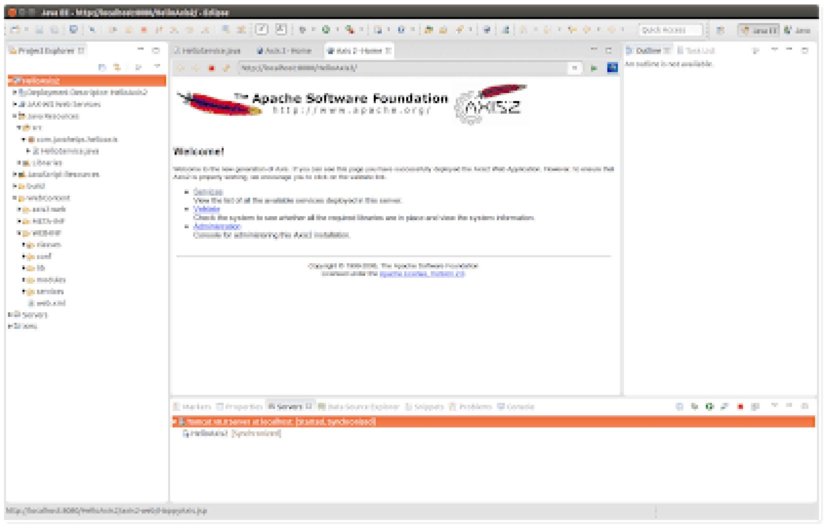
```
<service name="HelloService">
 <Description>
  Please Type your service description here
 </Description>

 <messageReceivers>
  <messageReceiver mep="http://www.w3.org/ns/wsdl/in-only"
   class="org.apache.axis2.rpc.receivers.RPCInOnlyMessageReceiver" />

  <messageReceiver mep="http://www.w3.org/ns/wsdl/in-out"
   class="org.apache.axis2.rpc.receivers.RPCMessageReceiver" />
 </messageReceivers>

 <parameter name="ServiceClass"
locked="false">com.javahelps.helloaxis.HelloService</parameter>
</service>
```

Step 13:
Right-click on the project and select Run As → Run on Server, select the Tomcat server and click 'Finish'.

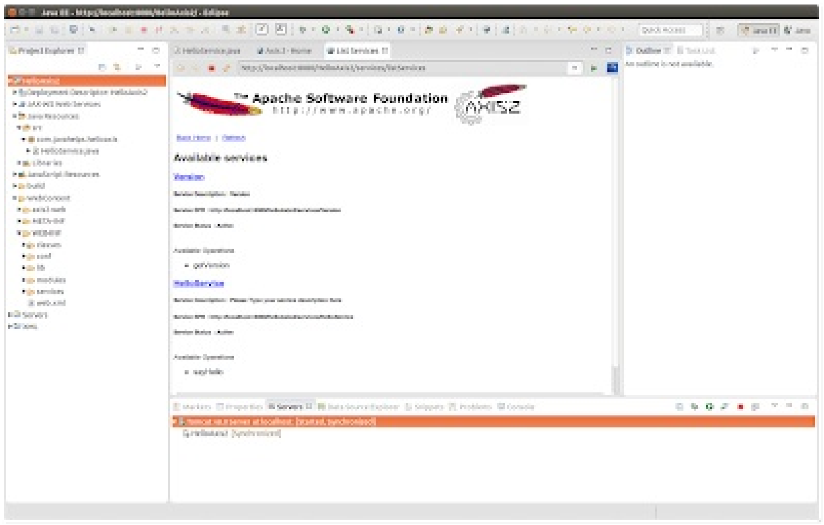
Step 14:
Click on the '*Services*' link in the appeared web page. It should list the available services including *HelloService*. Now we have successfully created our first web service using Axis2.



Run On Server
Run On Server
Select which server to use
How do you want to select the server?
Choose an existing server
Manually define a new server
Select the server that you want to use:
type filter text
Server
State
localhost
Tomcat v8.0 Server at localhost
Stopped
Apache Tomcat v8.0 supports J2EE 1.2, 1.3, 1.4, and Java EE 5, 6, and 7 Web modules.
Columns...
Always use this server when running this project
< Back
Next >
Cancel
Finish

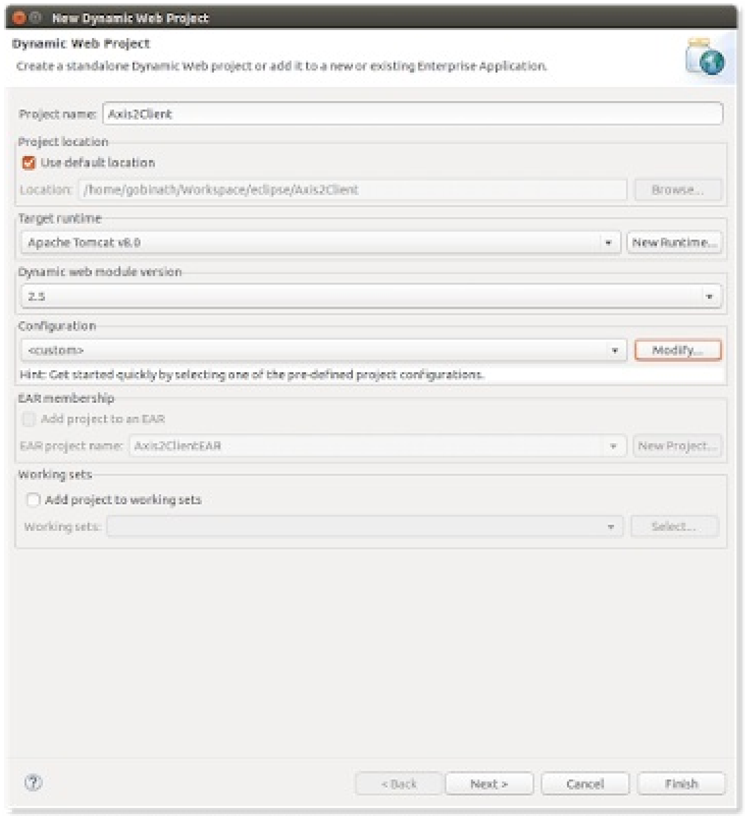


Step 15:
Click on ‘*HelloService*’ link and note the URL for future purpose.
http://localhost:8080/HelloAxis2/services/HelloService?wsdl



Step 16:
From this step onwards we are going to create a client for our web service. Create a new Dynamic Web Project named “Axis2Client” with same configurations as we did in *Step 1, 2 and 3*.

Step 17:
Copy and paste the *xmlschema-core-x.x.x.jar* file from *$AXIS2_HOME/lib* folder to the Eclipse project directory *WebContent/WEB-INF/lib* as we did in *Step 4*. If this library is not added to the client project, you may get NoClassDefFoundError:


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: Axis2Client

Project location

☑ Use default location

Location: /home/gobinath/Workspace/eclipse/Axis2Client

Target runtime: Apache Tomcat v8.0

Dynamic web module version: 2.5

Configuration: <custom>

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

EAR membership

☐ Add project to an EAR

EAR project name: Axis2ClientEAR

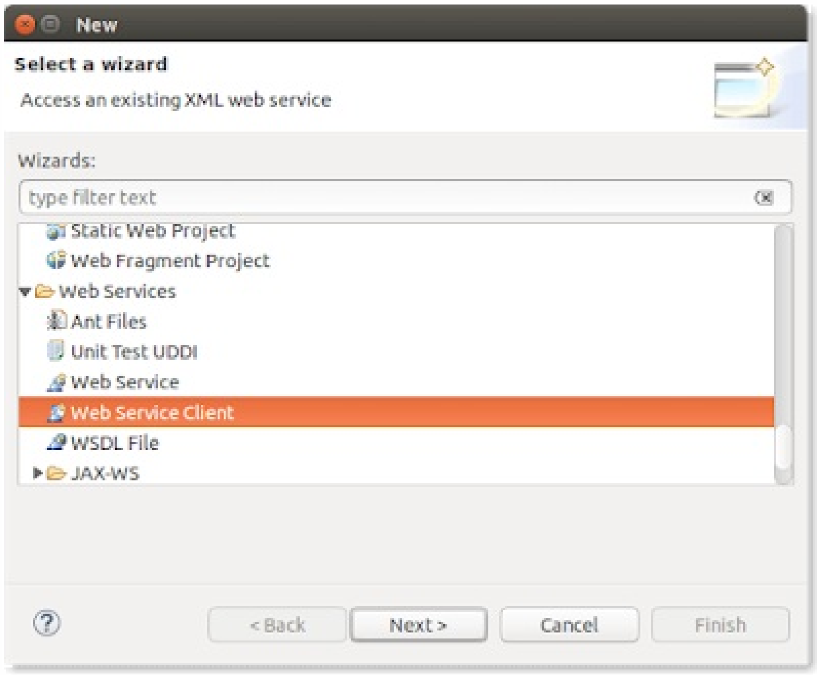
Working sets

☐ Add project to working sets

Working sets:

org/apache/ws/commons/schema/utils/NamespacePrefixList error when you run the client.

Step 18:
Right click on the Axis2Client project, select New → Other → Web Service Client and click Next.

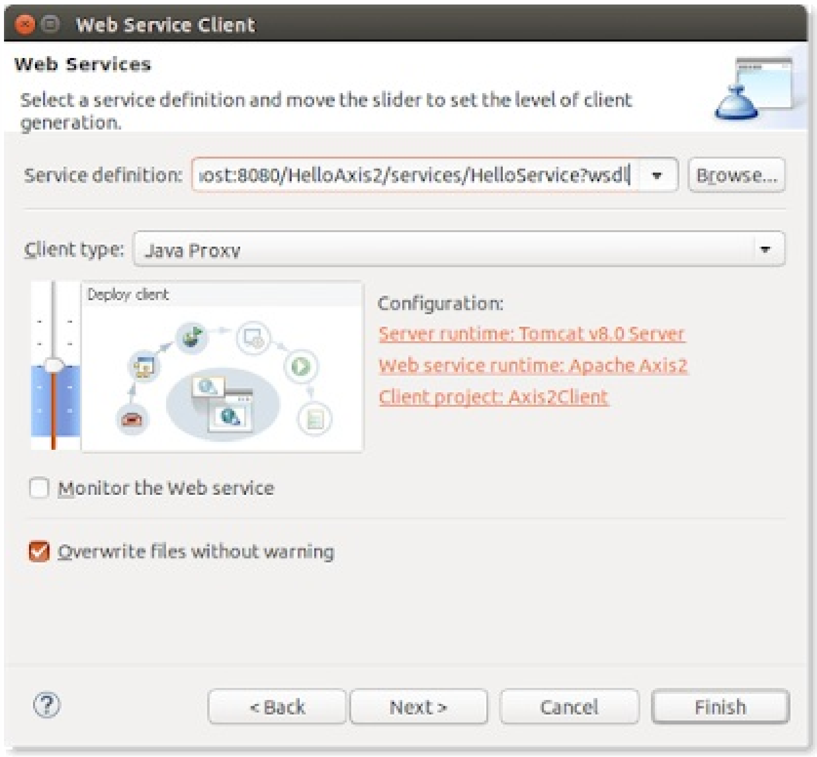

New

Select a wizard

Access an existing XML web service

Wizards:

Step 19:
Provide the URL noted in *Step 15* in place of Service definition and change the “Web service runtime” to Apache Axis2 as we did in *Step 9*.


Web Service Client

Web Services

Select a service definition and move the slider to set the level of client generation.

Service definition: ost:8080/HelloAxis2/services/HelloService?wsdl

Client type: Java Proxy

Configuration:
Server runtime: Tomcat v8.0 Server
Web service runtime: Apache Axis2
Client project: Axis2Client

Monitor the Web service

[✓ Overwrite files without warning]

It will create *HelloServiceStub* and *HelloServiceCallbackHandler* classes in the src folder with a package *com.javahelps.helloaxis*.

Step 20:
Create a new class *Client* in the same package *com.javahelps.helloaxis* and modify the code as shown here.

```
package com.javahelps.helloaxis;

public class Client {
    public static void main(String[] args) throws Exception {
        // Create the stub object
        HelloServiceStub stub = new HelloServiceStub();

        // Create the request
        HelloServiceStub.SayHello request = new HelloServiceStub.SayHello();

        // Set the parameters
        request.setName("Gobinath");

        // Invoke the service
        HelloServiceStub.SayHelloResponse response = stub.sayHello(request);
        String res = response.get_return(); // Hello Gobinath
        System.out.println("Response : " + res);
    }
}
```

Step 21:

Save all the changes and run the *Client* class as a Java application.

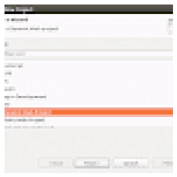
Find the project on Git Hub.

- Twitter
- Facebook
- Google
- Tumblr
- Pinterest

Related Posts

- Parse PCAP files in Java

- Jersey 2.x - Hello, world!



- Bridge Design Pattern

- Introduction to Interface (with Java 8 Features)

- Object Oriented Programming

- Microservices in a minute