Change the text size Reset A A
Discuss this article on **Stack Overflow**

Creating a Web Service Client in 3 Steps Using Eclipse

By **Saminda Wijeratne**

2 Oct, 2009 Level: Introductory Reads: 110063

When invoking a Web service which has complex requirements, you to make sure your Web service client can handle it. Saminda helps you to do this in just 3 simple steps. Saminda is a Senior Software Engineer at WSO2.

Saminda Wijeratne
Senior Software Engineer
WSO2 Inc

How to Create a Web Service Client in 3 Steps

With the increasing complexity of using Web services, the method of invoking them has also become increasingly complex. Users are required to learn a lot more than how to pass parameters to a specific url to get results. Learning the required domain is a waste of time if it is only a one time scenario or you don't want to know about Web service specifications at all.

Introduction

Writing a client manually is a time consuming process when having a very complex invocation procedure. From passing parameters to transportation methods and to applying security, the client has to be prepared before making the house call. Although these necessities are described in the WSDL file, understanding a long and complex WSDL file and converting the requirements into code can take hours specially if you lack the relevant knowledge. This procedure can be automated using the Apache Axis2 code generation libraries. This functionality now has been included as an Eclipse tool (one of the most popular open source IDEs in the world) so that within **few seconds** you can generate the client stubs according to the WSDL.

The following are the 3 steps,

1. Setting up the environment
2. Creating Client Stubs
3. Use the Generated Stubs to Invoke the Web service

The only tool that
lets you Develop Test Deploy
for on-premise and in the Cloud
composite applications
WSO2 Developer Studio
Eclipse-based development tool for SOA applications

Prerequisites

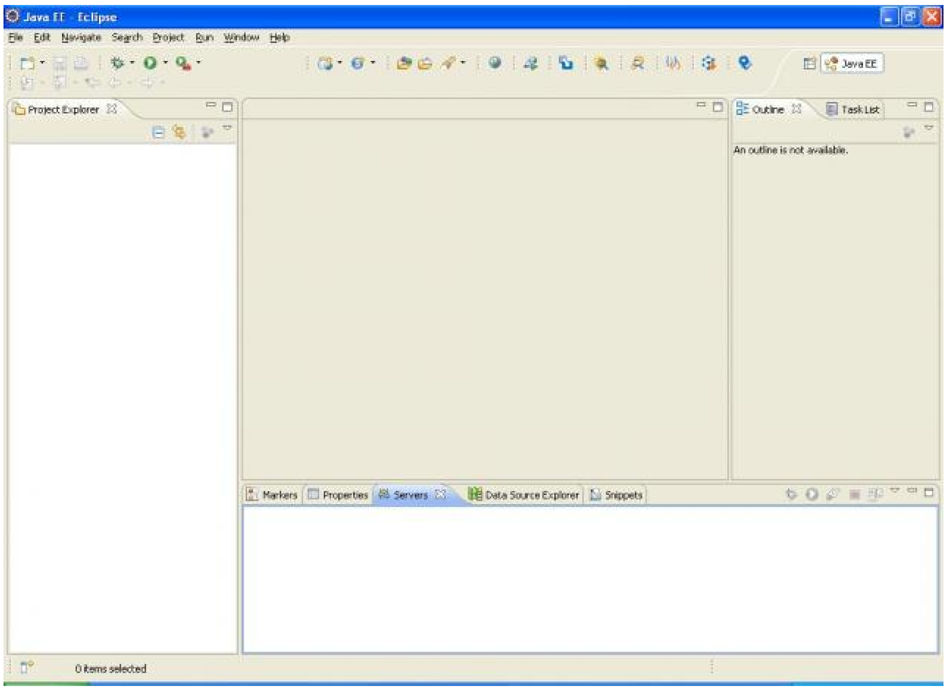
eclipse	3.4 or higher
Java	1.5 or higher
WSO2 WSAS	3.1.x
eclipse features	Web Service Authoring & Testing Tools - v1.1.0 WSO2 WSAS Server v3.1 - v1.0.0

Note: Visit the resources section of this tutorial to find out from where to download the prerequisites or click the above links.

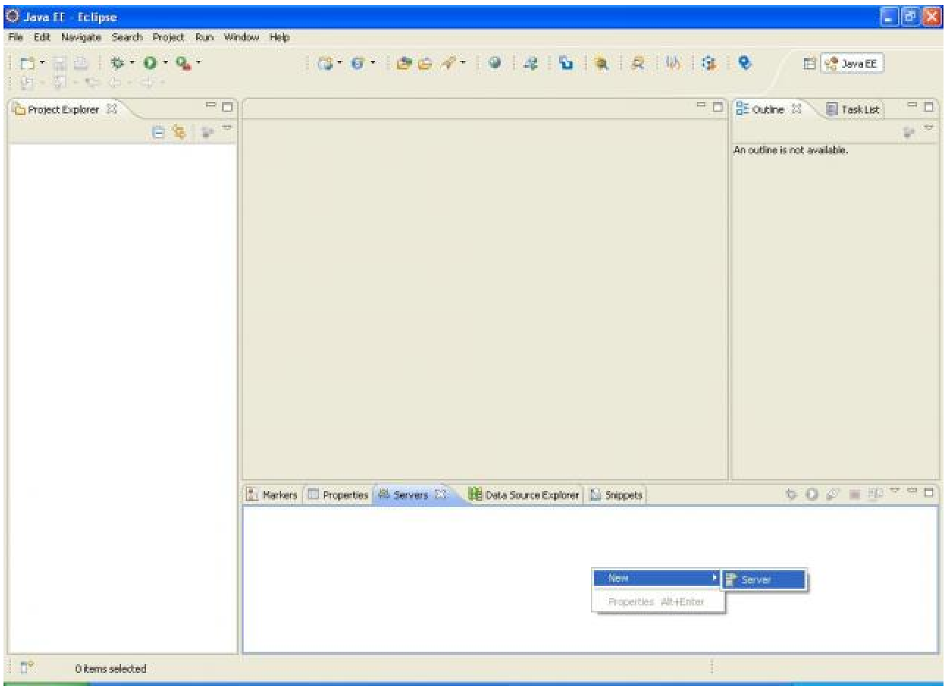
Step 1 - Setting up the environment - Adding a WSAS Server to eclipse

Open the Servers View in Eclipse by Window -> Show View -> Other... and select "Servers"

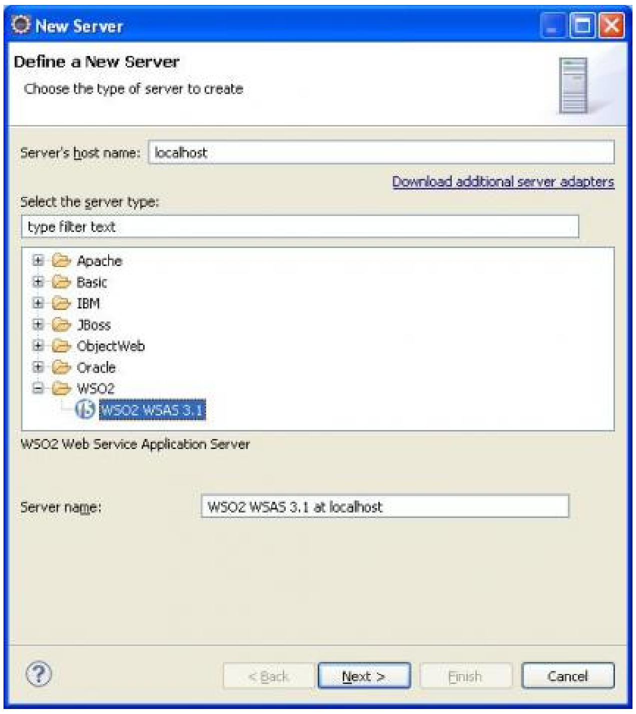
or just switch to "Java EE" perspective which will have "Server" view by default.



After opening the "Servers" view, if it does not contain a WSAS Server, right click on the view item content area and select New -> Server



Select "WSO2 WSAS 3.1" and click "Next"



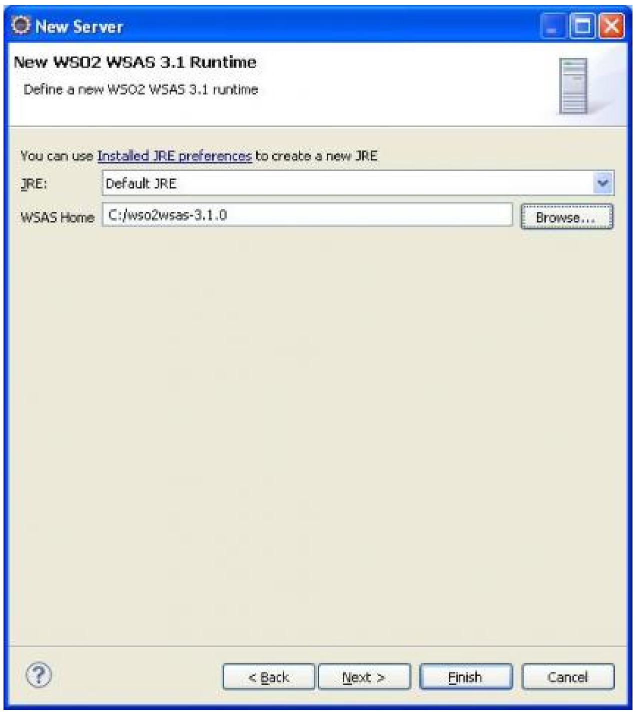
Server's host name: localhost

Select the server type: type filter text

WSO2 Web Service Application Server

Server name: WSO2 WSAS 3.1 at localhost

Specify the WSAS binary distribution location and click "Finish".

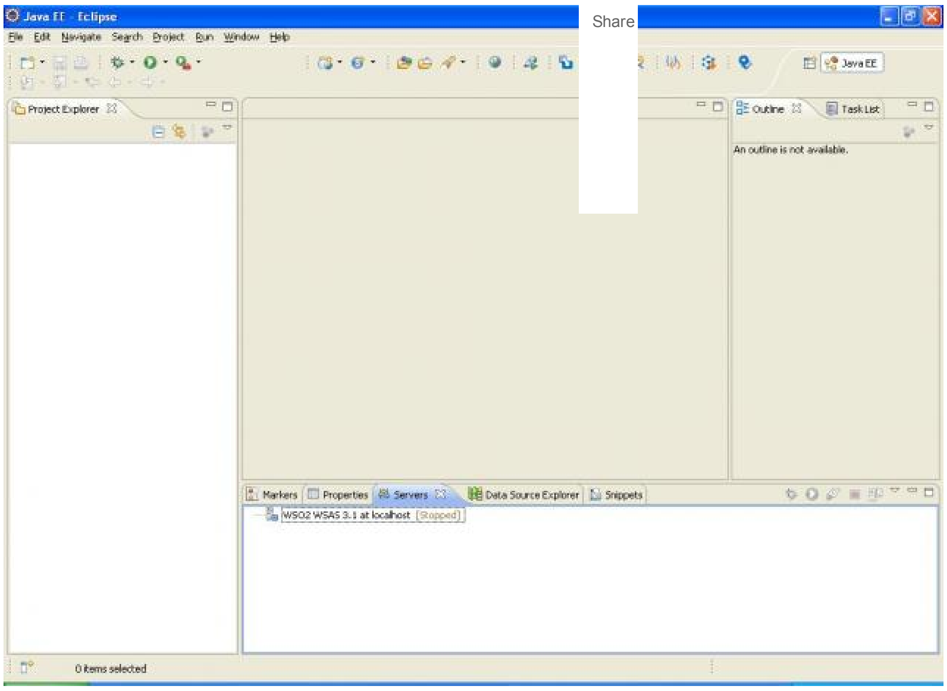


You can use Installed JRE preferences to create a new JRE

JRE: Default JRE

WSAS Home C:/wso2wsas-3.1.0

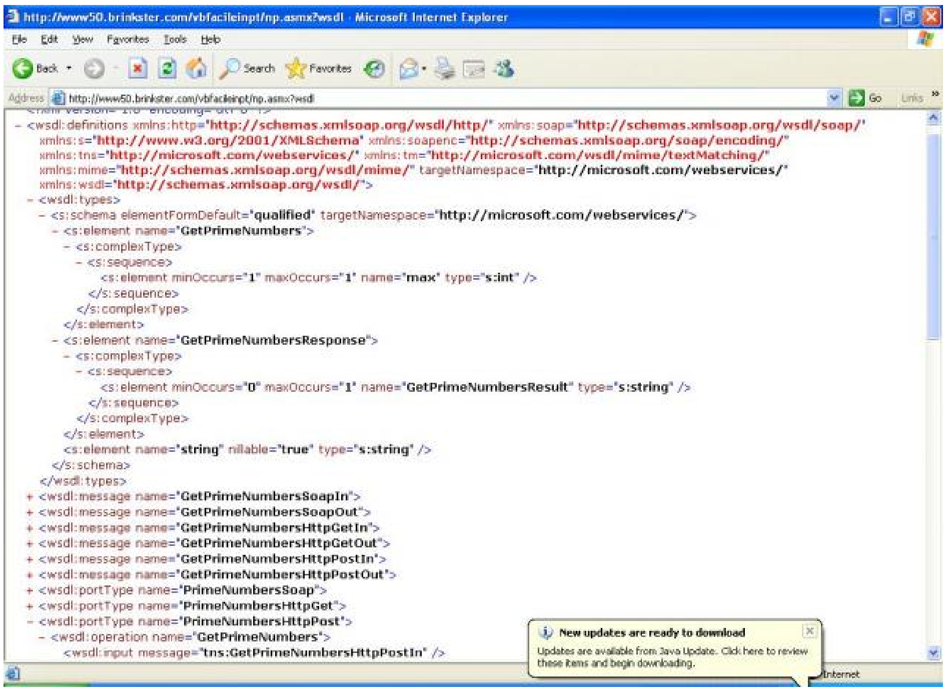
You will see a WSAS server being added. You don't need to start the server to do this tutorial. If you want to start the server click on the WSAS server in the list in Servers view and click the Start button of the view in upper right hand side.



Share

Step 2 - Create Client Stubs

As a first step, you need to get your hands on to the WSDL of the Web service. If it not hosted online, import the wsdl file into the eclipse workspace. In this tutorial, we will be using the WSDL at http://www50.brinkster.com/vbfacileinpt/np.asmx?WSDL. You may use any other valid WSDL.

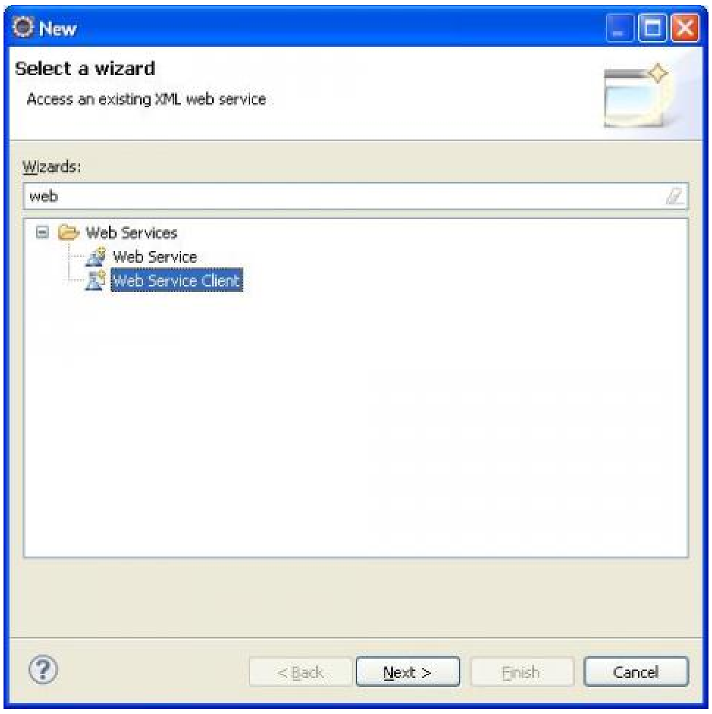

http://www50.brinkster.com/vbfacileinpt/np.asmx?wsdl - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back
Search
Favorites
Address http://www50.brinkster.com/vbfacileinpt/np.asmx?wsdl
Go
Links
Internet

```
- <wsdl:definitions xmlns:http="http://schemas.xmlsoap.org/wsdl/http/" xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
    xmlns:s="http://www.w3.org/2001/XMLSchema" xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
    xmlns:tns="http://microsoft.com/webservices/" xmlns:tm="http://microsoft.com/wsdl/mime/textMatching/"
    xmlns:mime="http://schemas.xmlsoap.org/wsdl/mime/" targetNamespace="http://microsoft.com/webservices/"
    xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/">
  - <wsdl:types>
    - <s:schema elementFormDefault="qualified" targetNamespace="http://microsoft.com/webservices/">
      - <s:element name="GetPrimeNumbers">
        - <s:complexType>
          - <s:sequence>
              <s:element minOccurs="1" maxOccurs="1" name="max" type="s:int" />
            </s:sequence>
          </s:complexType>
        </s:element>
      - <s:element name="GetPrimeNumbersResponse">
        - <s:complexType>
          - <s:sequence>
              <s:element minOccurs="0" maxOccurs="1" name="GetPrimeNumbersResult" type="s:string" />
            </s:sequence>
          </s:complexType>
        </s:element>
        <s:element name="string" nillable="true" type="s:string" />
      </s:schema>
    </wsdl:types>
  + <wsdl:message name="GetPrimeNumbersSoapIn">
  + <wsdl:message name="GetPrimeNumbersSoapOut">
  + <wsdl:message name="GetPrimeNumbersHttpGetIn">
  + <wsdl:message name="GetPrimeNumbersHttpGetOut">
  + <wsdl:message name="GetPrimeNumbersHttpPostIn">
  + <wsdl:message name="GetPrimeNumbersHttpPostOut">
  + <wsdl:portType name="PrimeNumbersSoap">
  + <wsdl:portType name="PrimeNumbersHttpGet">
  - <wsdl:portType name="PrimeNumbersHttpPost">
    - <wsdl:operation name="GetPrimeNumbers">
        <wsdl:input message="tns:GetPrimeNumbersHttpPostIn" />
```

New updates are ready to download
Updates are available from Java Update. Click here to review these items and begin downloading.

(At this point if you have imported the WSDL file to the workspace, select the WSDL file from the Project Explorer)

Go to File -> New -> Other... (or Ctrl + N)


New

Select a wizard

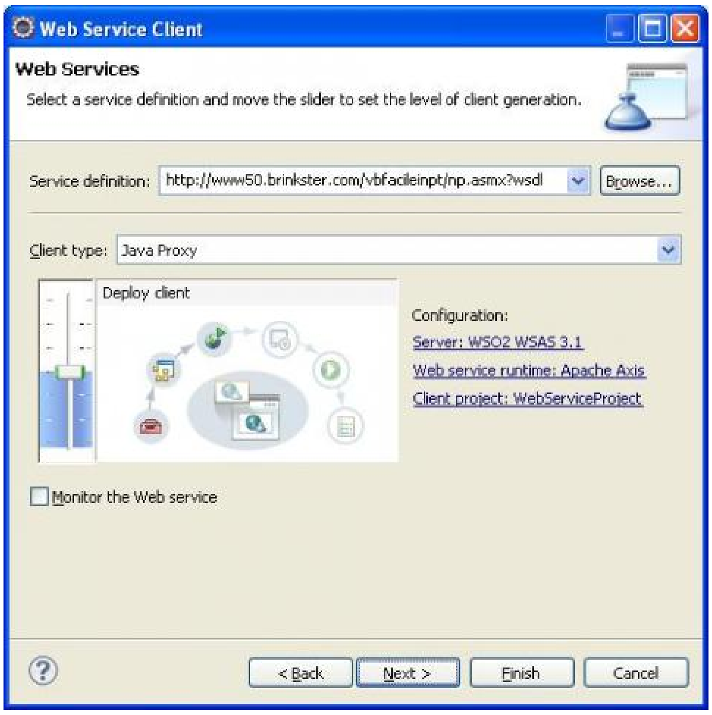
Access an existing XML web service

Wizards:

web

- Web Services
  - Web Service
  - Web Service Client

Select "Web Service Client" and click "Next"


Web Service Client

Web Services

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

Service definition: http://www50.brinkster.com/vbfacileinpt/np.asmx?wsdl

Client type: Java Proxy

Deploy client

Configuration:

Server: WSO2 WSAS 3.1

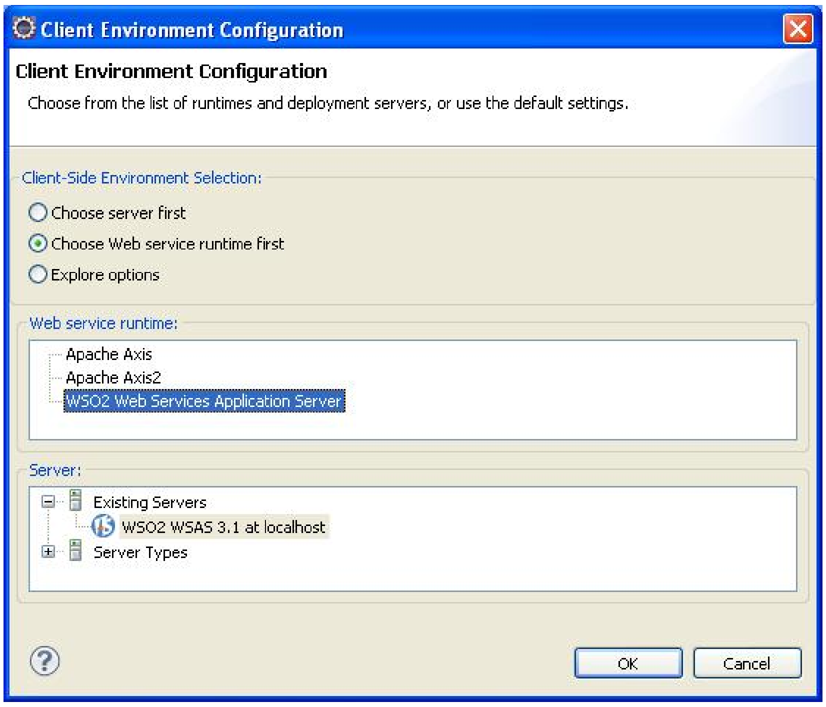
Web service runtime: Apache Axis

Client project: WebServiceProject

Monitor the Web service

Specify the WSDL file in the "Service Definition" text box if it is not already there.

Click the "Server" link and select "WSO2 WSAS Server 3.1" among existing servers or server types.


Client Environment Configuration

Client Environment Configuration

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

Client-Side Environment Selection:

- Choose server first
- [✓ Choose Web service runtime first]
- Explore options

Web service runtime:

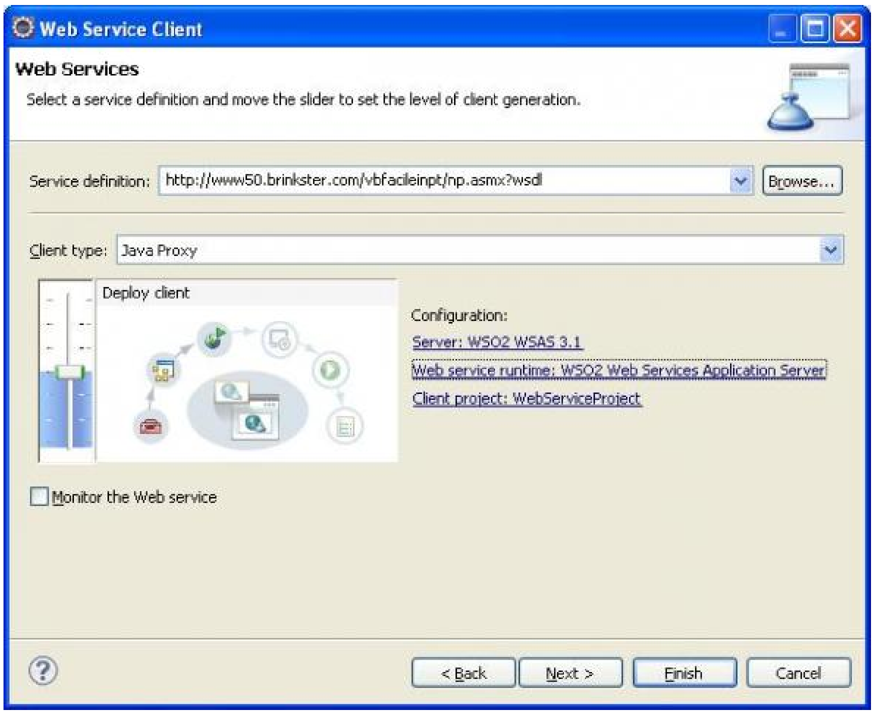
- Apache Axis
- Apache Axis2
- WSO2 Web Services Application Server

Server:

- Existing Servers
  - WSO2 WSAS 3.1 at localhost
- Server Types

Select "WSO2 Web Services Application Server".

Click "OK"


Web Service Client

Web Services

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

Service definition: http://www50.brinkster.com/vbfacileinpt/np.asmx?wsdl

Client type: Java Proxy

Deploy client

Configuration:

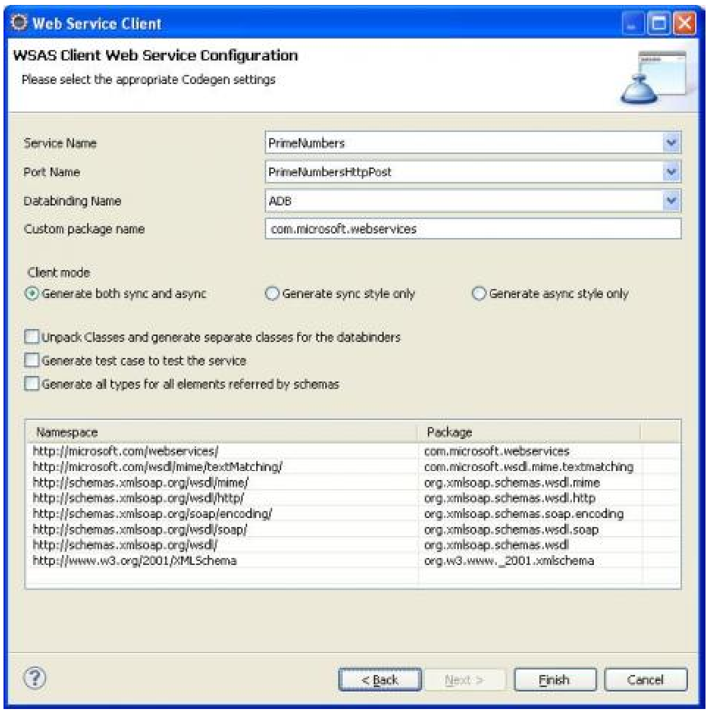
Server: WSO2 WSAS 3.1

Web service runtime: WSO2 Web Services Application Server

Client project: WebServiceProject

Monitor the Web service

Click "Next"



WSAS Client Web Service Configuration

Please select the appropriate Codegen settings

Service Name: PrimeNumbers

Port Name: PrimeNumbersHttpPost

Databinding Name: ADB

Custom package name: com.microsoft.webservices

Client mode: [✓ Generate both sync and async] Generate sync style only Generate async style only

☐ Unpack Classes and generate separate classes for the databinders

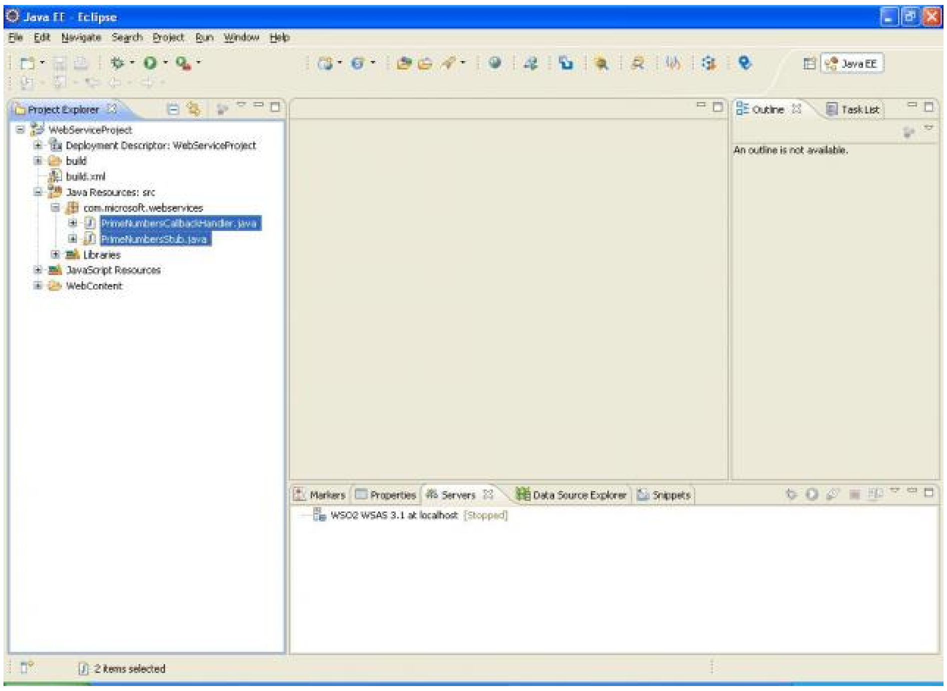
☐ Generate test case to test the service

☐ Generate all types for all elements referred by schemas

Namespace	Package
http://microsoft.com/webservices/	com.microsoft.webservices
http://microsoft.com/wsdl/mime/textMatching/	com.microsoft.wsdl.mime.textmatching
http://schemas.xmlsoap.org/wsdl/mime/	org.xmlsoap.schemas.wsdl.mime
http://schemas.xmlsoap.org/wsdl/http/	org.xmlsoap.schemas.wsdl.http
http://schemas.xmlsoap.org/soap/encoding/	org.xmlsoap.schemas.soap.encoding
http://schemas.xmlsoap.org/wsdl/soap/	org.xmlsoap.schemas.wsdl.soap
http://schemas.xmlsoap.org/wsdl/	org.xmlsoap.schemas.wsdl
http://www.w3.org/2001/XMLSchema	org.w3.www._2001.xmlschema

In this page, you can customize how the Client Stubs are generated. For now, lets leave the settings as it is. Click "Finish"

The related stubs will be generated in the given Eclipse project.



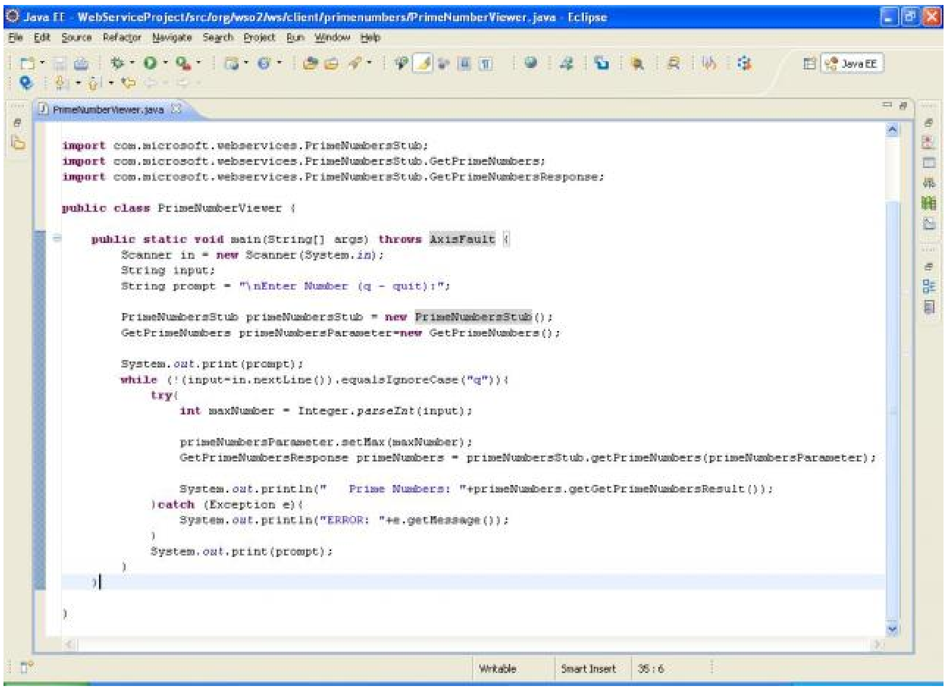
Step 3 - Use the Generate Stubs to Invoke the Web Service

Inside an existing class or in a new class we'll add the java code to invoke the Web service now. As long as the stub classes are accessible you can use any class in any project in the workspace. I'm creating a new class "PrimeNumberViewer" inside the stub generated project itself ("WebServiceProject") adding the following method.

```
public static void main(String[] args) throws AxisFault {
    Scanner in = new Scanner(System.in);
    String input;
    String prompt = "\nEnter Number (q - quit):";

    PrimeNumbersStub primeNumbersStub = new PrimeNumbersStub();
    GetPrimeNumbers primeNumbersParameter=new GetPrimeNumbers();
     System.out.print(prompt);
```

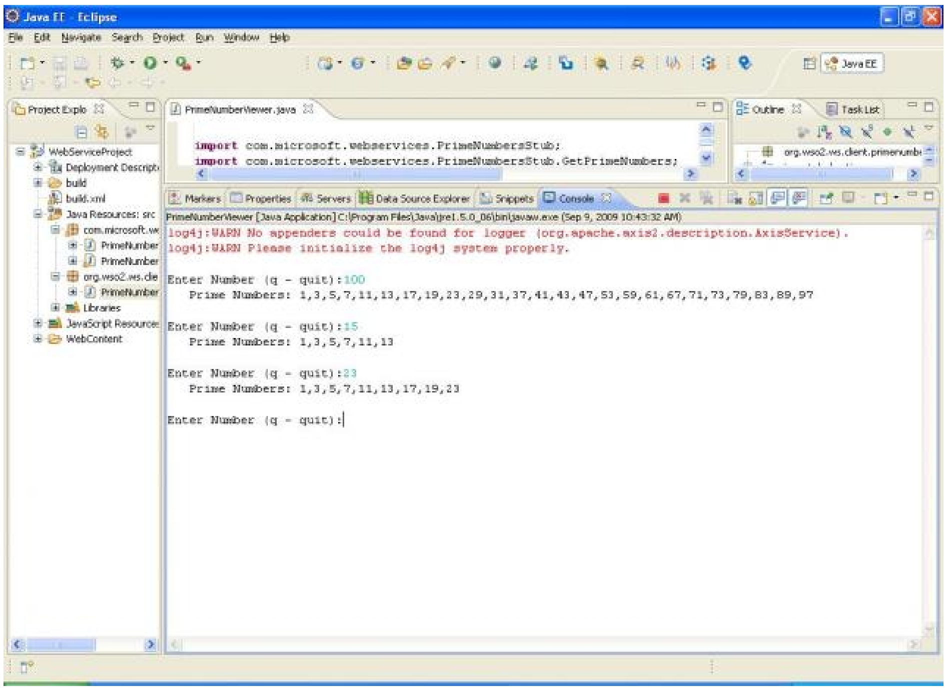
```
            while (!(input=in.nextLine()).equalsIgnoreCase("q")){
                try{
                    int maxNumber = Integer.parseInt(input);
                    primeNumbersParameter.setMax(maxNumber);
                    GetPrimeNumbersResponse primeNumbers = primeNumbersStub.getPrimeNumbers(primeNumbersParameter);
                    System.out.println("   Prime Numbers: "+primeNumbers.getGetPrimeNumbersResult());
                }catch (Exception e){
                    System.out.println("ERROR: "+e.getMessage());
                }
                System.out.print(prompt);
            }
        }
```

Only the bolded text/commands in the above script are related to the web service invocation. Rest are just there to create a fancy program.



```
import com.microsoft.webservices.PrimeNumbersStub;
import com.microsoft.webservices.PrimeNumbersStub.GetPrimeNumbers;
import com.microsoft.webservices.PrimeNumbersStub.GetPrimeNumbersResponse;

public class PrimeNumberViewer {

    public static void main(String[] args) throws AxisFault {
        Scanner in = new Scanner(System.in);
        String input;
        String prompt = "\nEnter Number (q - quit):";

        PrimeNumbersStub primeNumbersStub = new PrimeNumbersStub();
        GetPrimeNumbers primeNumbersParameter=new GetPrimeNumbers();

        System.out.print(prompt);
        while (!(input=in.nextLine()).equalsIgnoreCase("q")){
            try{
                int maxNumber = Integer.parseInt(input);

                primeNumbersParameter.setMax(maxNumber);
                GetPrimeNumbersResponse primeNumbers = primeNumbersStub.getPrimeNumbers(primeNumbersParameter);

                System.out.println("   Prime Numbers: "+primeNumbers.getGetPrimeNumbersResult());
            }catch (Exception e){
                System.out.println("ERROR: "+e.getMessage());
            }
            System.out.print(prompt);
        }
    }

}
```

Now just execute the above method. In my case I'll just execute it as a java class.



```
PrimeNumberViewer [Java Application] C:\Program Files\Java\jre1.5.0_06\bin\javaw.exe (Sep 9, 2009 10:43:32 AM)
log4j:WARN No appenders could be found for logger (org.apache.axis2.description.AxisService).
log4j:WARN Please initialize the log4j system properly.

Enter Number (q - quit):100
   Prime Numbers: 1,3,5,7,11,13,17,19,23,29,31,37,41,43,47,53,59,61,67,71,73,79,83,89,97

Enter Number (q - quit):15
   Prime Numbers: 1,3,5,7,11,13

Enter Number (q - quit):23
   Prime Numbers: 1,3,5,7,11,13,17,19,23

Enter Number (q - quit):
```

Explaining the use of client stubs

- To invoke the client service, first you need to create an object from "<ServiceName>Stub" class. In my case, this class is "**PrimeNumbersStub**" and the object created is **primeNumbersStub.**

- It is from invoking methods in this object (**primeNumbersStub**) you will invoke the Web service. (The Web service port names become the names methods in this object)

eg: **primeNumbersStub.getPrimeNumbers(....)**

- If the Web service requires some parameters to be passed to it, then you will need to pass a parameter to this method depicting the Web service port type.

eg: **primeNumbersStub.getPrimeNumbers(primeNumbersParameter);**
here the **primeNumbersParameter** is an object which keeps the required parameters. Normally it would contain getter/setter methods for the parameters (**primeNumbersParameter.setMax(***maxNumber***);**)

- The return object of the service invocation method would contain the response received from the server. In this example, the response object is **primeNumbers** of the type **GetPrimeNumbersResponse**. There will be a getter to retrieve the expected result/value of the Web service.

eg: **primeNumbers.getGetPrimeNumbersResult()**

Resources

1. The Java class which uses the generated client stubs to invoke the service can be found here.
2. To download WSO2 WSAS please visit https://wso2.org/projects/wsas/java
3. To download WSO2 WSAS Tools please visit http://wso2.org/projects/tools or use the update site URL http://tools.wso2.org/eclipse

Summary

It is easier to create a Web service rather than finding issues related to it. But these tools help you in debugging your Web service just like debugging your Java application in Eclipse.

Click here to find out how to create a web service

Author

Saminda Wijeratne, Senior Software Engineer, WSO2, samindaw@wso2.com

WSO2 Advantage
The Connected Business
API-centric SOA
Integrated Complete Platform
Open Source
Revolutionary Middleware
End to End Governance

Products
Middleware Platform
Overview
API Manager
Application Server
Business Activity Monitor
Business Process Server
Business Rules Server
Carbon
Cloud Gateway
Complex Event Processor
Data Services Server
Elastic Load Balancer
Enterprise Service Bus
Enterprise Store
Governance Registry
Identity Server

Cloud
Cloud Overview
Private Cloud
WSO2 Private PaaS
WSO2 App Factory
Managed Cloud
Managed Cloud Overview
Service Level Agreement
Public Cloud
App Cloud
API Cloud

Use Cases
Technology Challenges
IT Challenges
Business Challenges

Resources
Events Calendar
WSO2 Library
White Papers
Case Studies
Analyst Reports
Presentations
On-demand Webinars
Videos
Product Documentation
Research
Partners

Support
Enterprise Support
Production Support
Development Support
Professional Services
Getting Started
Evaluation Support
QuickStart
Training and Certification
Community Support
StackOverFlow
Product Documentation
Knowledge Base Library
Support System Login

Message Broker
Storage Server
Task Server
User Engagement Server

Development Tools

WSO2 Developer Studio

Mobile Platform

WSO2 Enterprise Mobility Manager

Subscribe to the newsletter Legal Privacy Report a problem with this page