TUTORIAL

How to Call Web Service in Android Using SOAP

Posted by Chintan Rathod in Tutorials | Android Programming on May 29, 2012

Tags: call Web Service in Android, SOAP, Web Service Definition Language, Web Service in Android, Web Service in SOAP

In this tutorial, we will learn how to call a Web Service using SOAP (Simple Object Access Protocol).

Download Files: WebServiceDemo.zip

Introduction

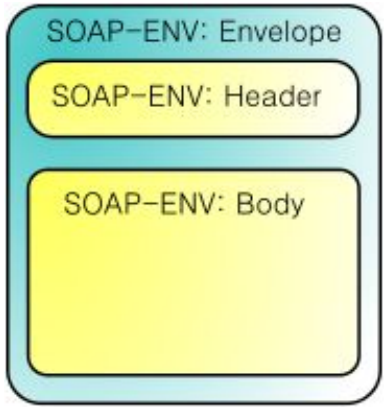
In this tutorial, we will learn how to call a Web Service using SOAP (Simple Object Access Protocol).

Prerequisites

Web Service, SOAP envelope, WSDL (Web Service Definition Language)

What is SOAP?

SOAP is a protocol specification for exchanging structured information in the implementation of Web Services in computer networks. It relies on Extensible Markup Language (XML) for its message format, and usually relies on other Application Layer protocols, most notably Hypertext Transfer Protocol (HTTP) and Simple Mail Transfer Protocol (SMTP), for message negotiation and transmission. The following is the structure of SOAP Envelope:



Step 1:

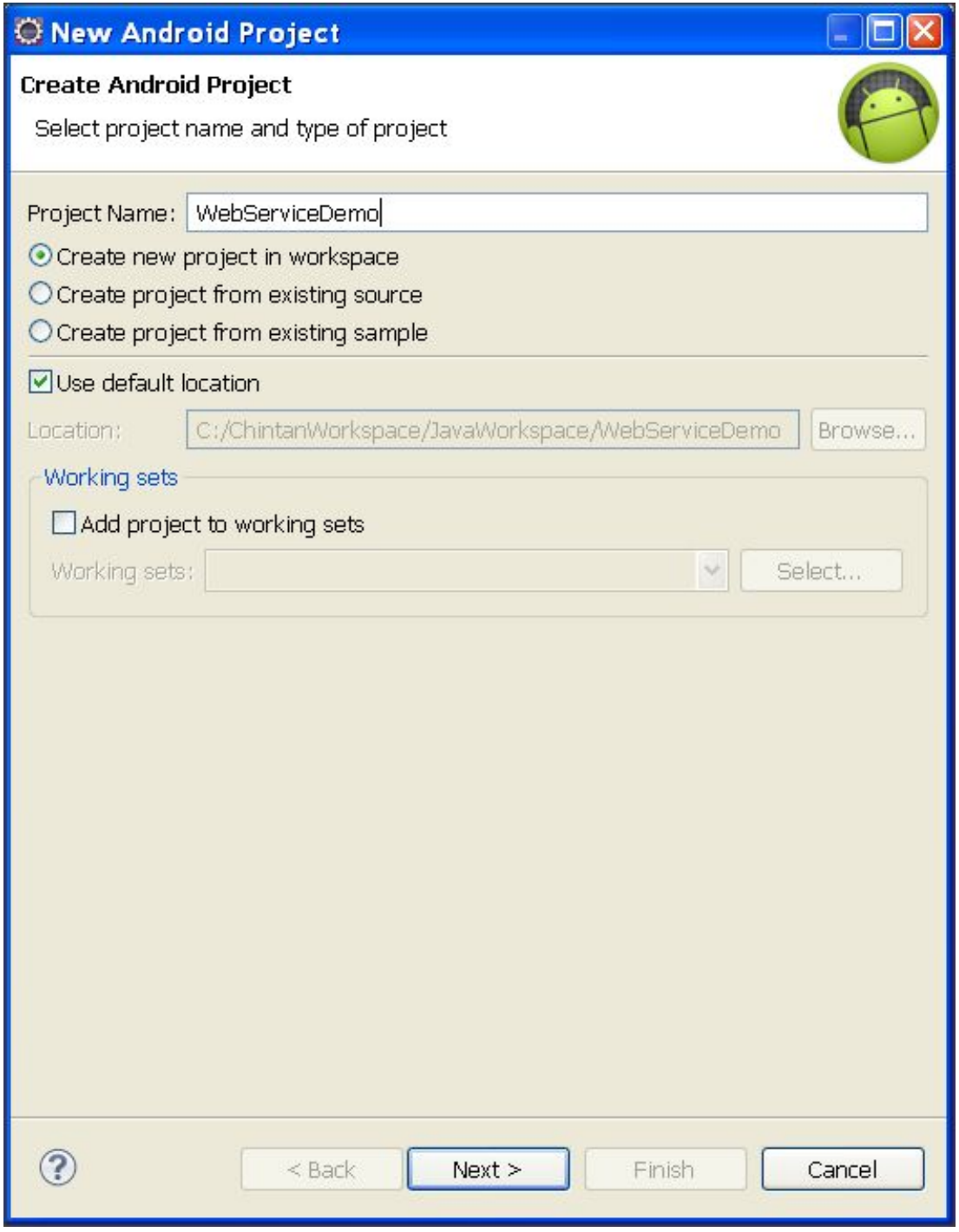
First create a "New Android Project". Name it "WebServiceDemo" like below.

- Live Currency Converter Using ASP.NET C#
- Uploading Images to Database Using ASP.NET C#
- Uploading and Downloading PDF Files From Database Using ASP.NET C#
- Uploading and Downloading Word Files From Database Using ASP.NET C#
- WCF Method Overloading
- Div Rotation Using HTML5 and CSS3
- Ribbon Control in WPF 4.5
- Simple ASP.NET Survey Application

View All

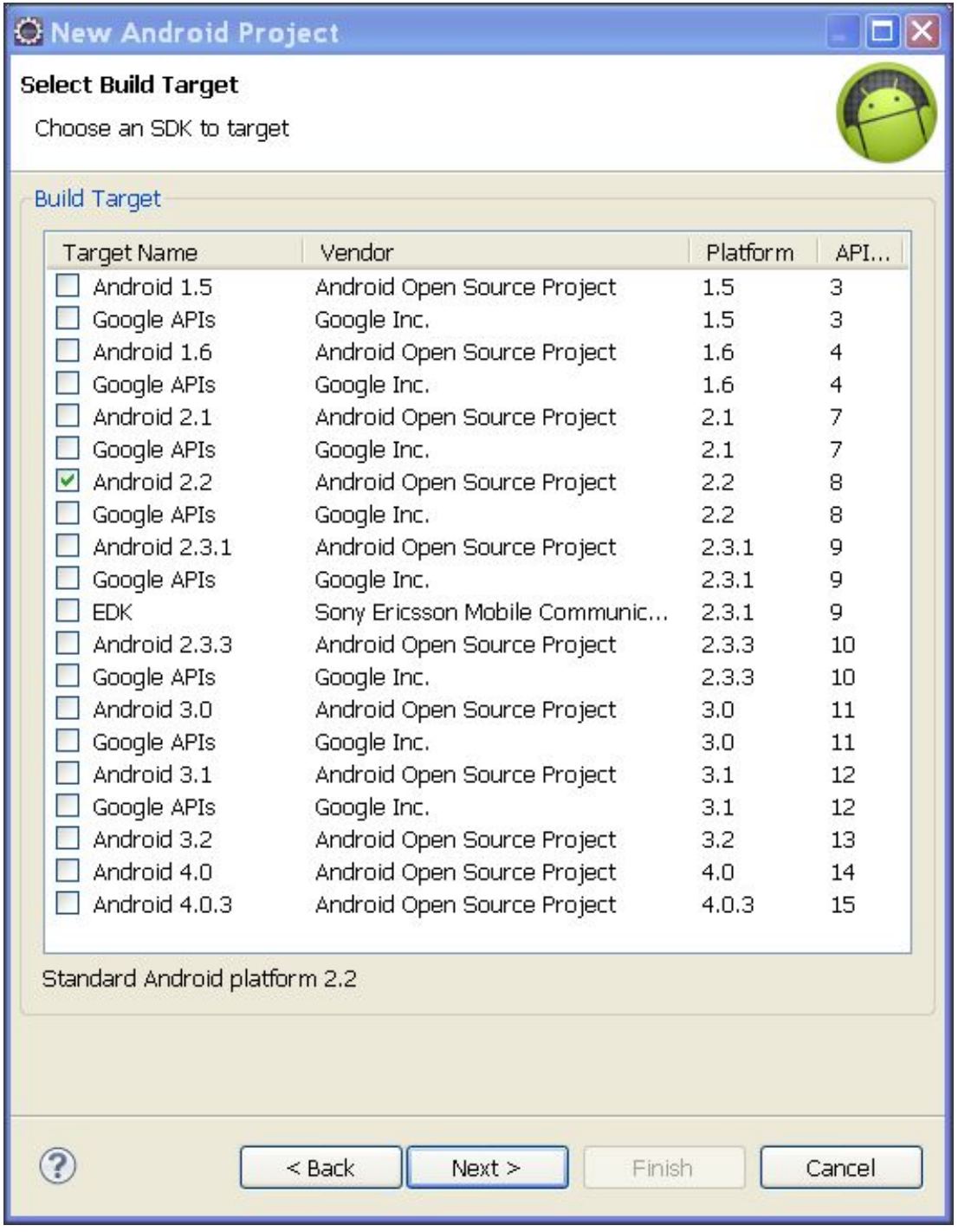
MOST LIKED ARTICLE

- WCF Introduction and Contracts - Day 1
- Create SSRS Report Using Report Wizard
- CREATE READ UPDATE and DELETE - CRUD Operation Using LINQ to SQL
- WCF - Data Contract - Day 4
- WCF Message Exchange Patterns: Day 3



HOT KEYWORDS

Android android application Android articles Android LinearLayout Android o.s android programming Android ScrollView Android ScrollView in VS2010 Android tutorials AutoCompleteTextView in Android DroidDraw Google Android GridView in Android install Android LinearLayout LinearLayout in Android Mobile Mobile Development Mono For Android MonoAndroid MultiCompleteTextView in Android ScrollView in Android VS 20101 what is Android Widgets

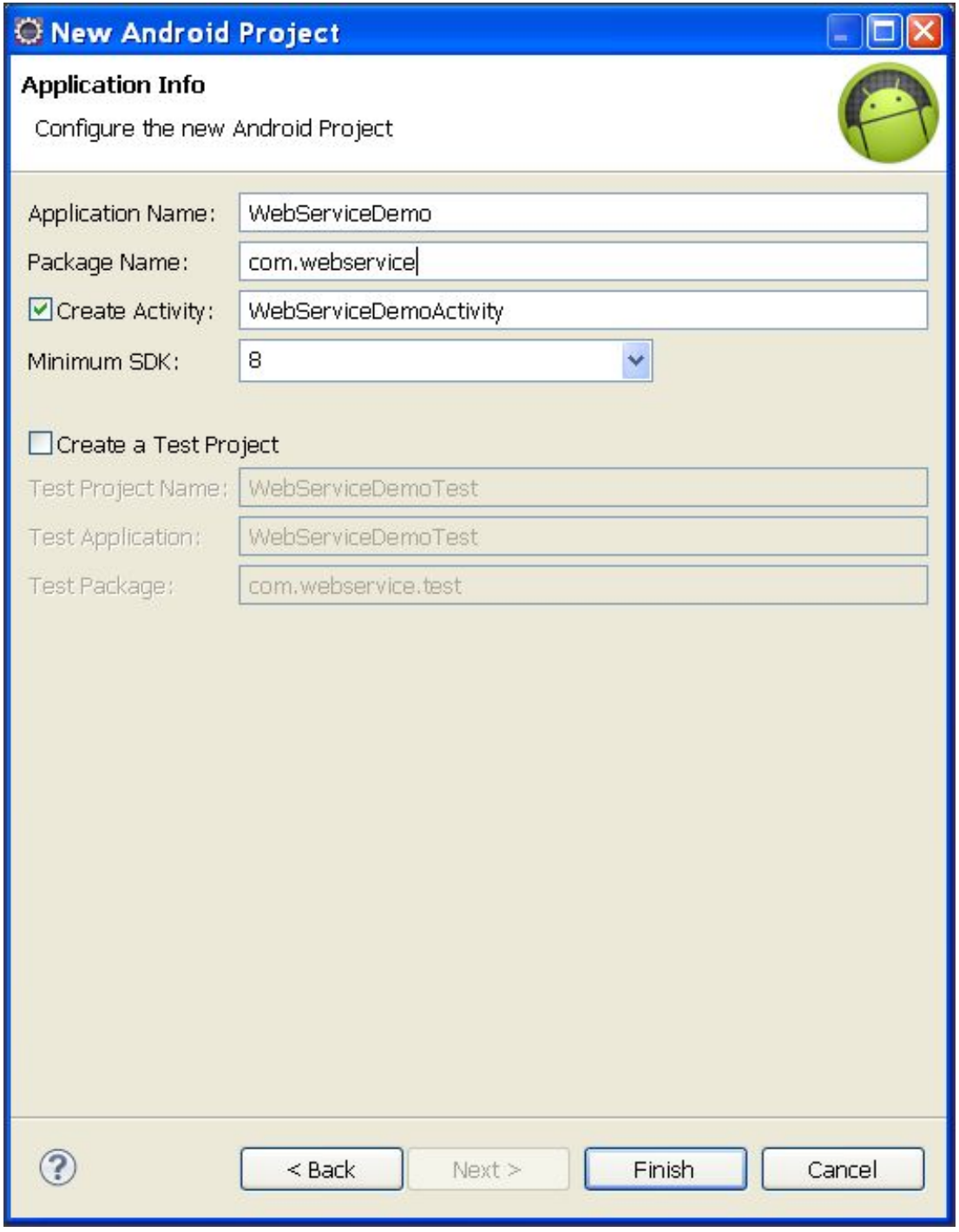


WHITEPAPERS AND BOOKS

- SharePoint 2010 Administration & Development
- Getting Started with Managed Metadata Service in SharePoint 2010
- Windows Phone 7 Hileleri
- Windows Phone Development Step by Step Tutorial
- Essentials of SharePoint 2010: Business Intelligence Capabilities
- Working with Directories in C#
- FileInfo in C#
- Programming List with C#
- Source Code: Graphics Programming with GDI+
- Programming Strings using C#

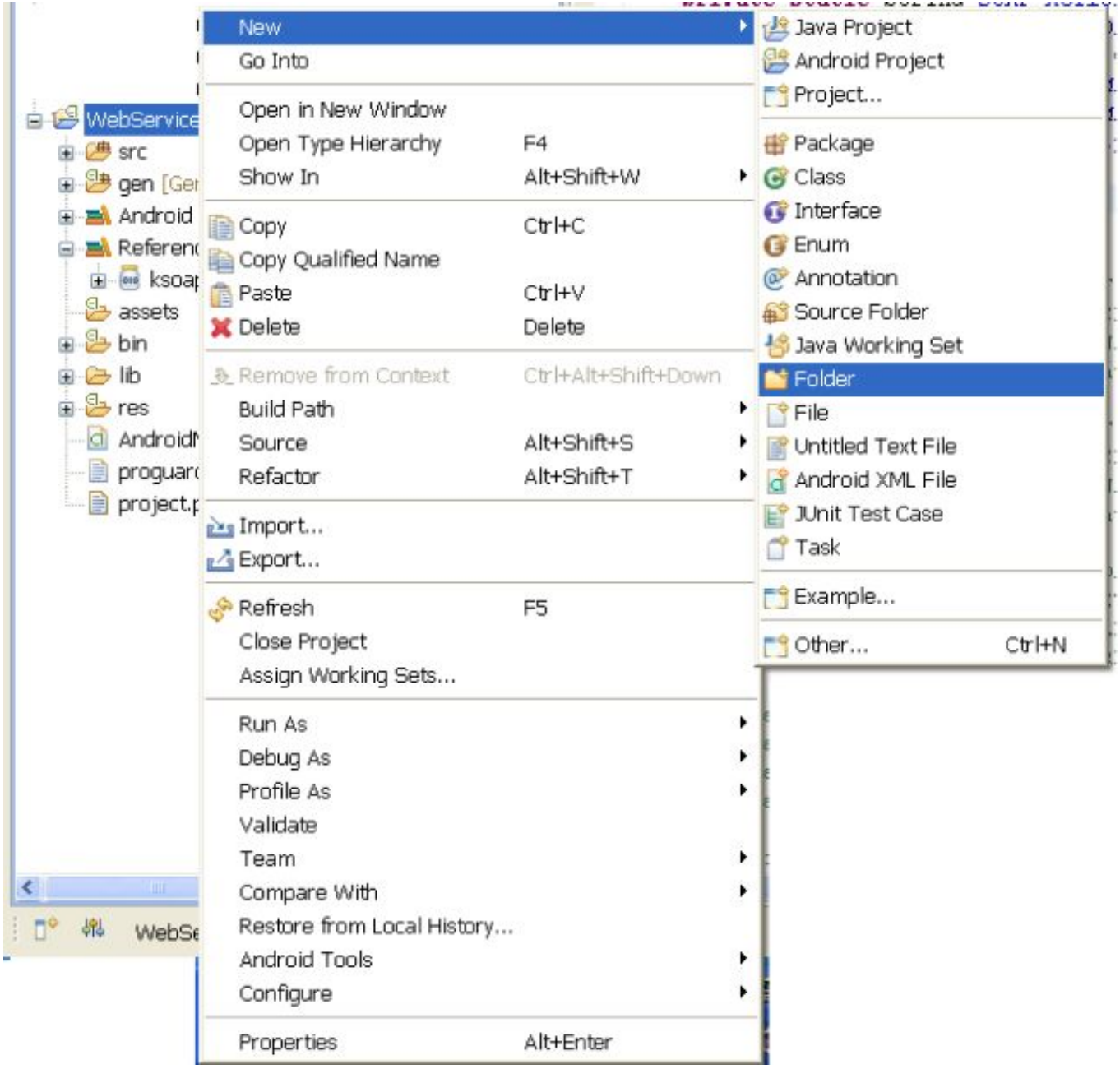
View All

POLL | RESULT | ALL POLLS

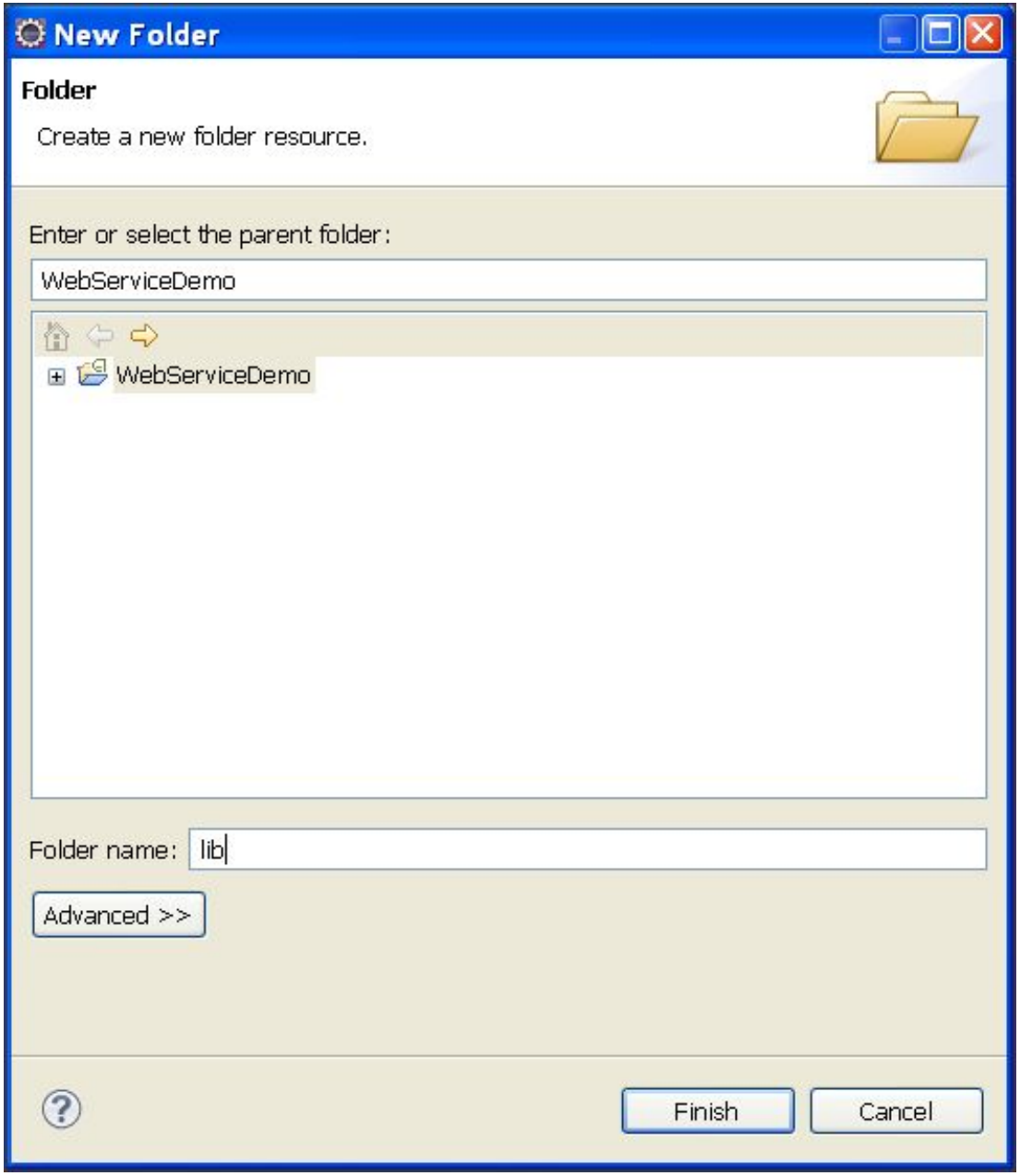


Step 2:

Now right-click on your "WebServiceDemo" project and select "New -> Folder"



Now, give it a name it "lib". We need to add a SOAP library into this directory.

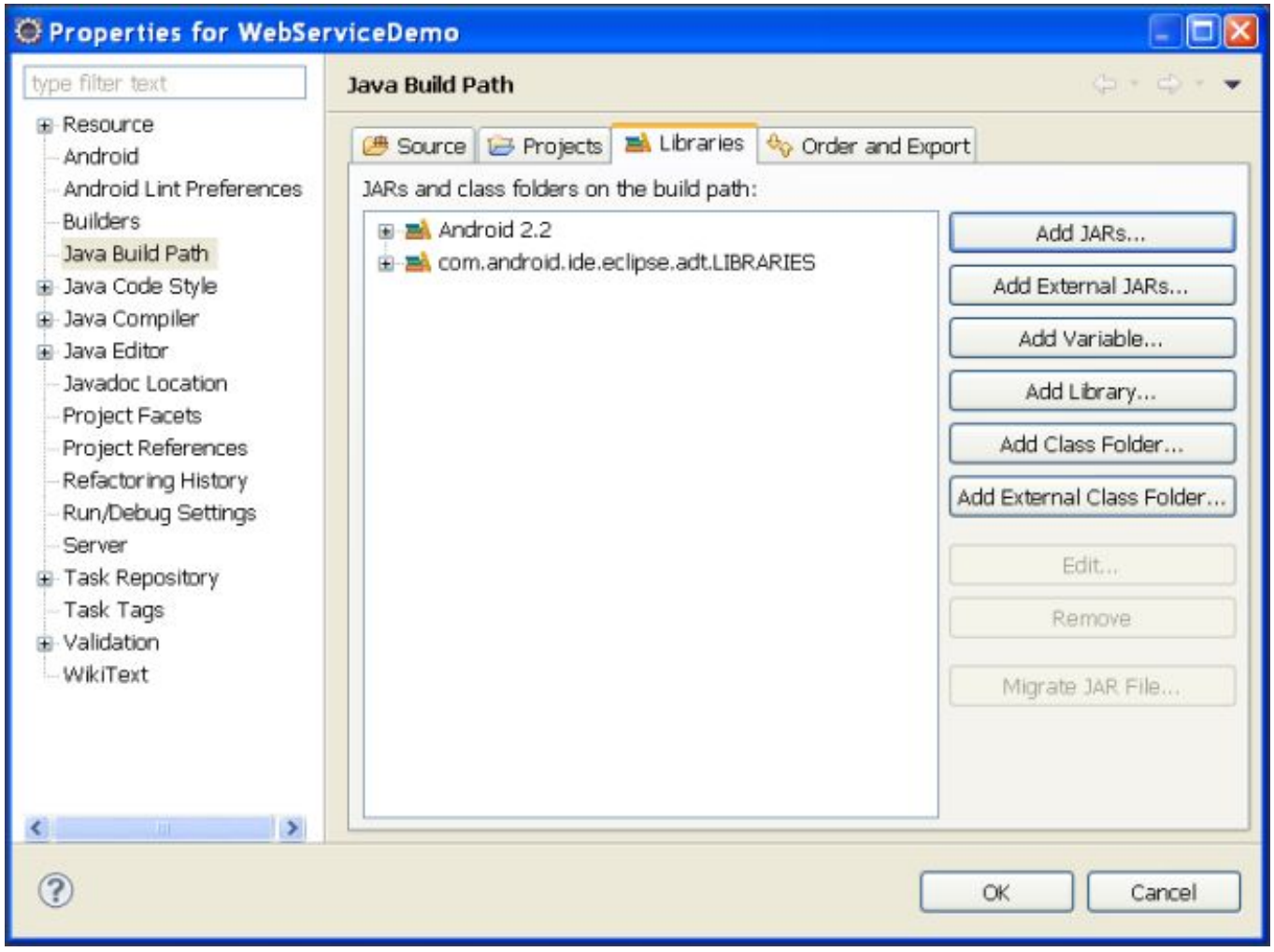


Step 3:

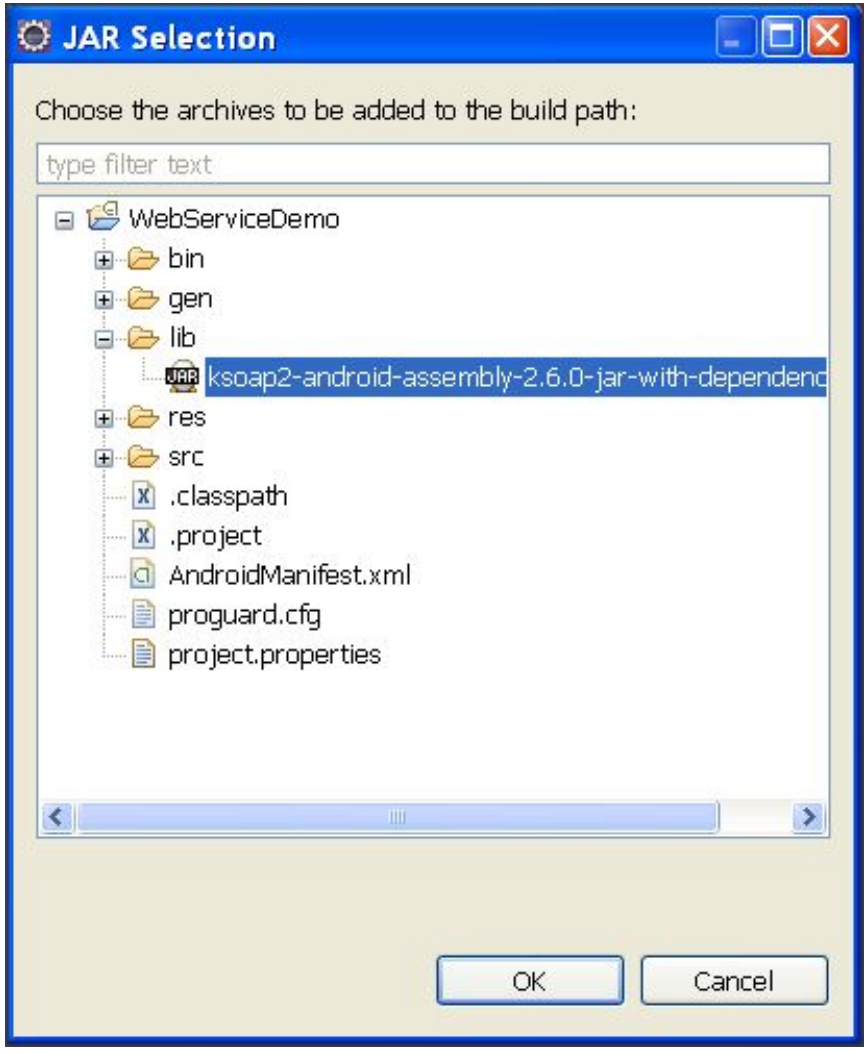
Now download the attached library named "ksoap2-android-assembly-2.6.0-jar-with-dependencies.jar". Copy that file and paste it into the "lib" directory.

After copying, do the following steps:

1. Right-click on the project.
2. Go "Build Path -> Configure Build Path"



3. Now, click on "Add Jars" and select ".jar" file from "project -> lib" directory.



4. Click on "Ok" to finish the procedure of adding library to Android application.

Step 4:

Next we need to create a layout of screen. To do so, go to "WebServiceDemo -> res -> layout -> main.xml"
Open this xml file in editing mode, and place below code.

Main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
```

```
android:layout_height="match_parent"
android:orientation="vertical" >

<TextView
    android:id="@+id/textView1"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Fahrenheit"
    android:textAppearance="?android:attr/textAppearanceLarge" />

<EditText
    android:id="@+id/txtFar"
    android:layout_width="match_parent"
    android:layout_height="wrap_content" >

    <requestFocus />
</EditText>

<TextView
    android:id="@+id/textView2"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Celsius"
    android:textAppearance="?android:attr/textAppearanceLarge" />

<EditText
    android:id="@+id/txtCel"
    android:layout_width="match_parent"
    android:layout_height="wrap_content" />

<LinearLayout
    android:id="@+id/linearLayout1"
    android:layout_width="match_parent"
    android:layout_height="wrap_content" >

    <Button
        android:id="@+id/btnFar"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_weight="0.5"
        android:text="Convert To Celsius" />

    <Button
        android:id="@+id/btnCel"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
```

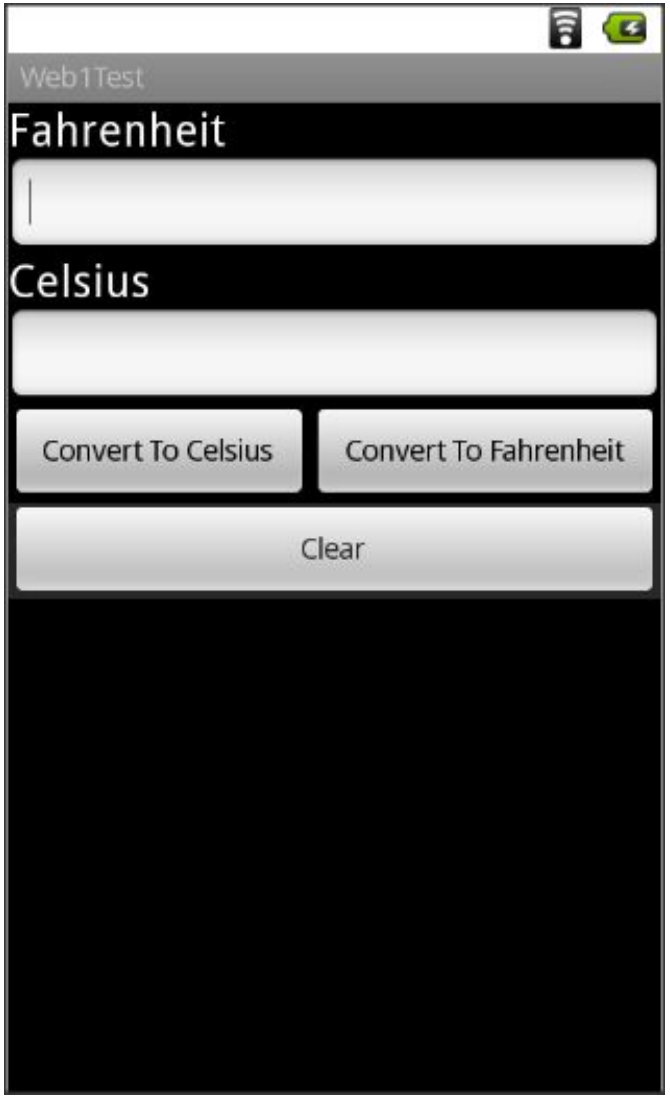
```
            android:layout_weight="0.5"
            android:text="Convert To Fahrenheit" />

    </LinearLayout>


    <Button
        android:id="@+id/btnClear"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="Clear" />

</LinearLayout>
```

This will create simple following screen.



Step 5:

Now, find out any Web Service, make sure that you can view its WSDL file by writing "?wsdl" after that address.

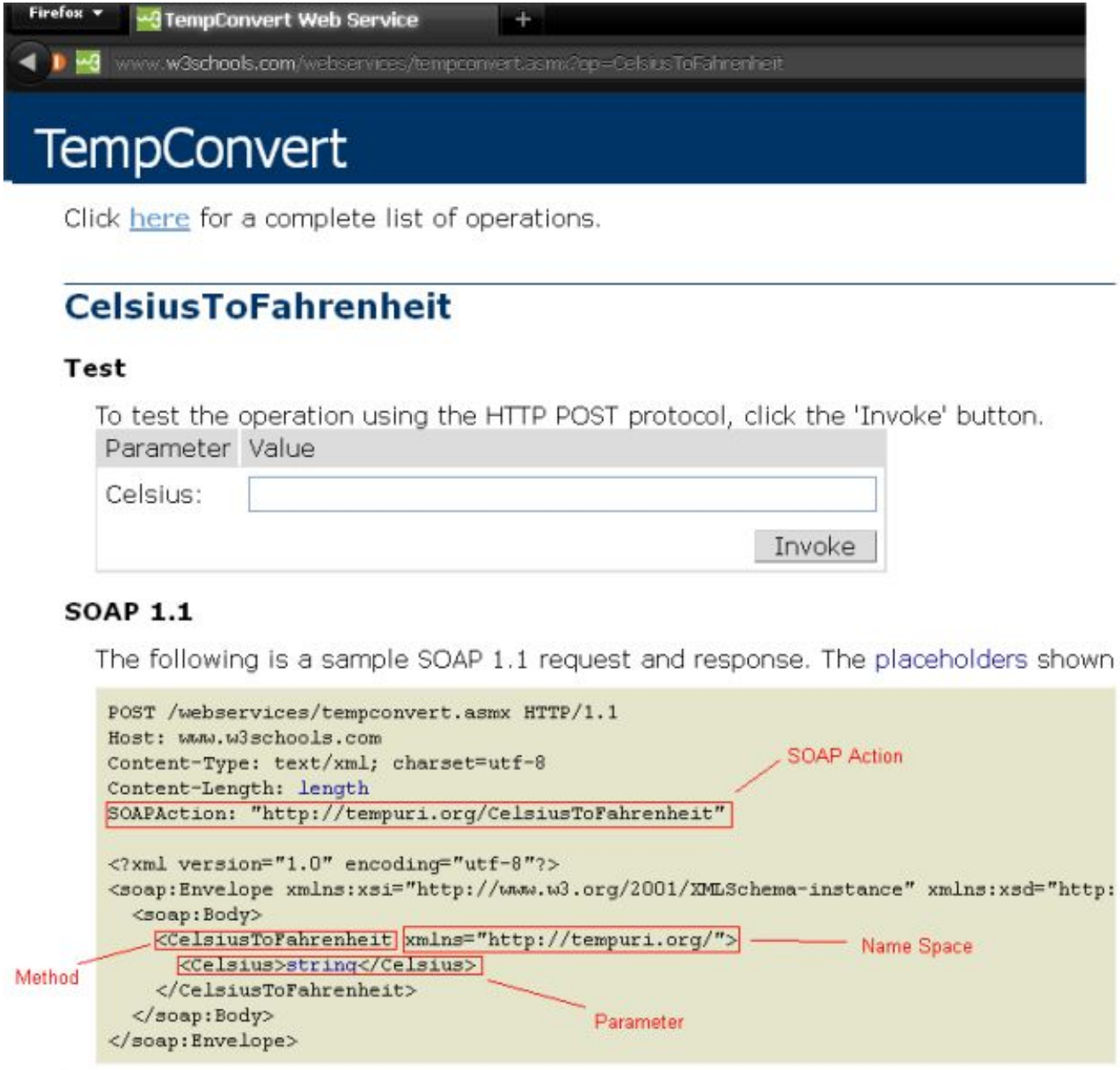
For example, you have a web service like http://www.w3schools.com/webservices/tempconvert.asmx", so to view the WSDL file, simply write "?wsdl" after this address like: http://www.w3schools.com/webservices/tempconvert.asmx?WSDL".

You are now done with the Web Service part from the internet. Now you need to extend some

portion of the WSDL file. Open the first link in your browser, it will display 2 conversions for you:

- CelsiusToFahrenheit
- FahrenheitToCelsius

Select anyone of them, and you will see following screen:



For CelsiusToFahrenheit:

1. SOAP_ACTION = "http://tempuri.org/CelsiusToFahrenheit";
2. NAMESPACE = "http://tempuri.org/";

3. METHOD_NAME = "CelsiusToFahrenheit";
4. URL = "http://www.w3schools.com/webservices/tempconvert.asmx?WSDL";

For FahrenheitToCelsius:

1. SOAP_ACTION = "http://tempuri.org/FahrenheitToCelsius";
2. NAMESPACE = "http://tempuri.org/";
3. METHOD_NAME = " FahrenheitToCelsius ";
4. URL = "http://www.w3schools.com/webservices/tempconvert.asmx?WSDL";

Step 6:

You need to understand some classes before proceeding to use a Web Service.

1. **SoapObject (**

   A simple dynamic object that can be used to build SOAP calls without implementing KvmSerializable. Essentially, this is what goes inside the body of a SOAP envelope - it is the direct subelement of the body and all further sub elements. Instead of this class, custom classes can be used if they implement the KvmSerializable interface.

   Constructor:

   SoapObject (java.lang.String namespace, java.lang.String method)

2. SoapSerializationEnvelope

   This class extends the SoapEnvelope with Soap Serialization functionality.

   Constructor:

   SoapSerializationEnvelope (int version)

   Fields:

Type	Field	Description

boolean	dotNet	Set this variable to true for compatibility with what seems to be the default encoding for .Net-Services.

Methods:

Return Type	Method Name	Description
void	**setOutputSoapObject**(java.lang.Object soapObject)	Assigns the object to the envelope as the outbound message for the soap call.

3. **HttpTransportSE (org.ksoap2.transport.HttpTransportSE)**

   A J2SE based HttpTransport layer.

   Constructor:

   HttpTransportSE(java.lang.String url)

   Method:

Return Type	Method	Description
void	**call**(java.lang.String SoapAction, SoapEnvelope envelope)	set the desired soapAction header field

Step 7:

Open your "WebServiceDemo -> src -> WebServiceDemoActivity.java" file and enterr following code:

WebServiceDemoActivity.java

```
import org.ksoap2.SoapEnvelope;
```

```
import org.ksoap2.serialization.SoapObject;
import org.ksoap2.serialization.SoapSerializationEnvelope;
import org.ksoap2.transport.HttpTransportSE;
import android.app.Activity;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.Toast;

public class WebServiceDemoActivity extends Activity
{
   /** Called when the activity is first created. */
     private static String SOAP_ACTION1 = "http://tempuri.org/FahrenheitToCelsius";
     private static String SOAP_ACTION2 = "http://tempuri.org/CelsiusToFahrenheit";
     private static String NAMESPACE = "http://tempuri.org/";
     private static String METHOD_NAME1 = "FahrenheitToCelsius";
     private static String METHOD_NAME2 = "CelsiusToFahrenheit";
     private static String URL = "http://www.w3schools.com/webservices/tempconvert.asmx?
WSDL";

     Button btnFar,btnCel,btnClear;
     EditText txtFar,txtCel;

   @Override
   public void onCreate(Bundle savedInstanceState)
   {
      super.onCreate(savedInstanceState);
      setContentView(R.layout.main);

      btnFar = (Button)findViewById(R.id.btnFar);
      btnCel = (Button)findViewById(R.id.btnCel);
      btnClear = (Button)findViewById(R.id.btnClear);
      txtFar = (EditText)findViewById(R.id.txtFar);
      txtCel = (EditText)findViewById(R.id.txtCel);

      btnFar.setOnClickListener(new View.OnClickListener()
      {
            @Override
            public void onClick(View v)
            {
                //Initialize soap request + add parameters
```

```
            SoapObject request = new SoapObject(NAMESPACE, METHOD_NAME1);

            //Use this to add parameters
            request.addProperty("Fahrenheit",txtFar.getText().toString());

            //Declare the version of the SOAP request
            SoapSerializationEnvelope envelope = new
SoapSerializationEnvelope(SoapEnvelope.VER11);

            envelope.setOutputSoapObject(request);
            envelope.dotNet = true;

            try {
                HttpTransportSE androidHttpTransport = new HttpTransportSE(URL);

                //this is the actual part that will call the webservice
                androidHttpTransport.call(SOAP_ACTION1, envelope);

                // Get the SoapResult from the envelope body.
                SoapObject result = (SoapObject)envelope.bodyIn;

                if(result != null)
                {
                    //Get the first property and change the label text
                    txtCel.setText(result.getProperty(0).toString());
                }
                else
                {
                    Toast.makeText(getApplicationContext(), "No
Response",Toast.LENGTH_LONG).show();
                }
            } catch (Exception e) {
                e.printStackTrace();
            }
            }
        });

    btnCel.setOnClickListener(new View.OnClickListener()
    {
            @Override
            public void onClick(View v)
            {
```

```
            //Initialize soap request + add parameters
            SoapObject request = new SoapObject(NAMESPACE, METHOD_NAME2);

            //Use this to add parameters
            request.addProperty("Celsius",txtCel.getText().toString());

            //Declare the version of the SOAP request
            SoapSerializationEnvelope envelope = new
SoapSerializationEnvelope(SoapEnvelope.VER11);

            envelope.setOutputSoapObject(request);
            envelope.dotNet = true;

            try {
                HttpTransportSE androidHttpTransport = new HttpTransportSE(URL);

                //this is the actual part that will call the webservice
                androidHttpTransport.call(SOAP_ACTION2, envelope);

                // Get the SoapResult from the envelope body.
                SoapObject result = (SoapObject)envelope.bodyIn;

                if(result != null)
                {
                    //Get the first property and change the label text
                    txtFar.setText(result.getProperty(0).toString());
                }
                else
                {
                    Toast.makeText(getApplicationContext(), "No
Response",Toast.LENGTH_LONG).show();
                }
            } catch (Exception e) {
                e.printStackTrace();
            }
            }
        });

    btnClear.setOnClickListener(new View.OnClickListener()
    {
            @Override
            public void onClick(View v)
```

```
            {
                txtCel.setText("");
                txtFar.setText("");
            }
        });
    }
}
```

Step 8:

Now, open your "WebServiceDemo -> android.manifest" file. Add the following line before the <application> tag:

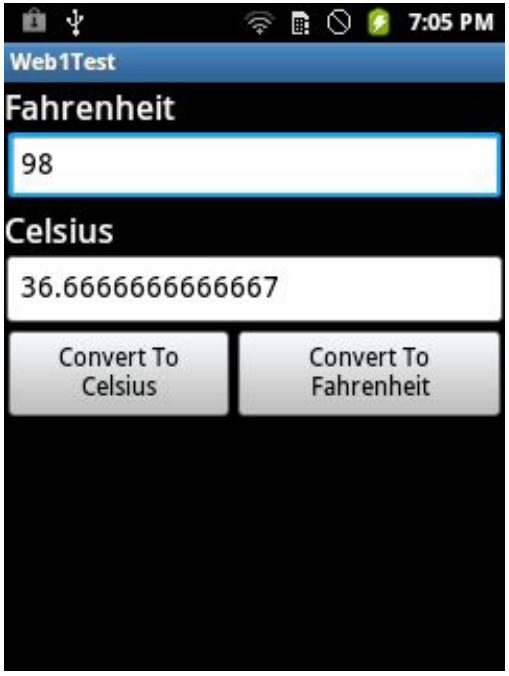
```
<uses-permission android:name="android.permission.INTERNET" />
```

This will allow the application to use the internet.

Step 9:

Run your application in the Android Cell. You will get the following outcome:

Note: In the emulator, we need to fix a proxy, so try the application in an Android Cell.



Summary

In this brief tutorial, we learned about Web Services, SOAP envelopes, WSDL files, HTTP transport, and how to use the in an Android application.

Article Extensions

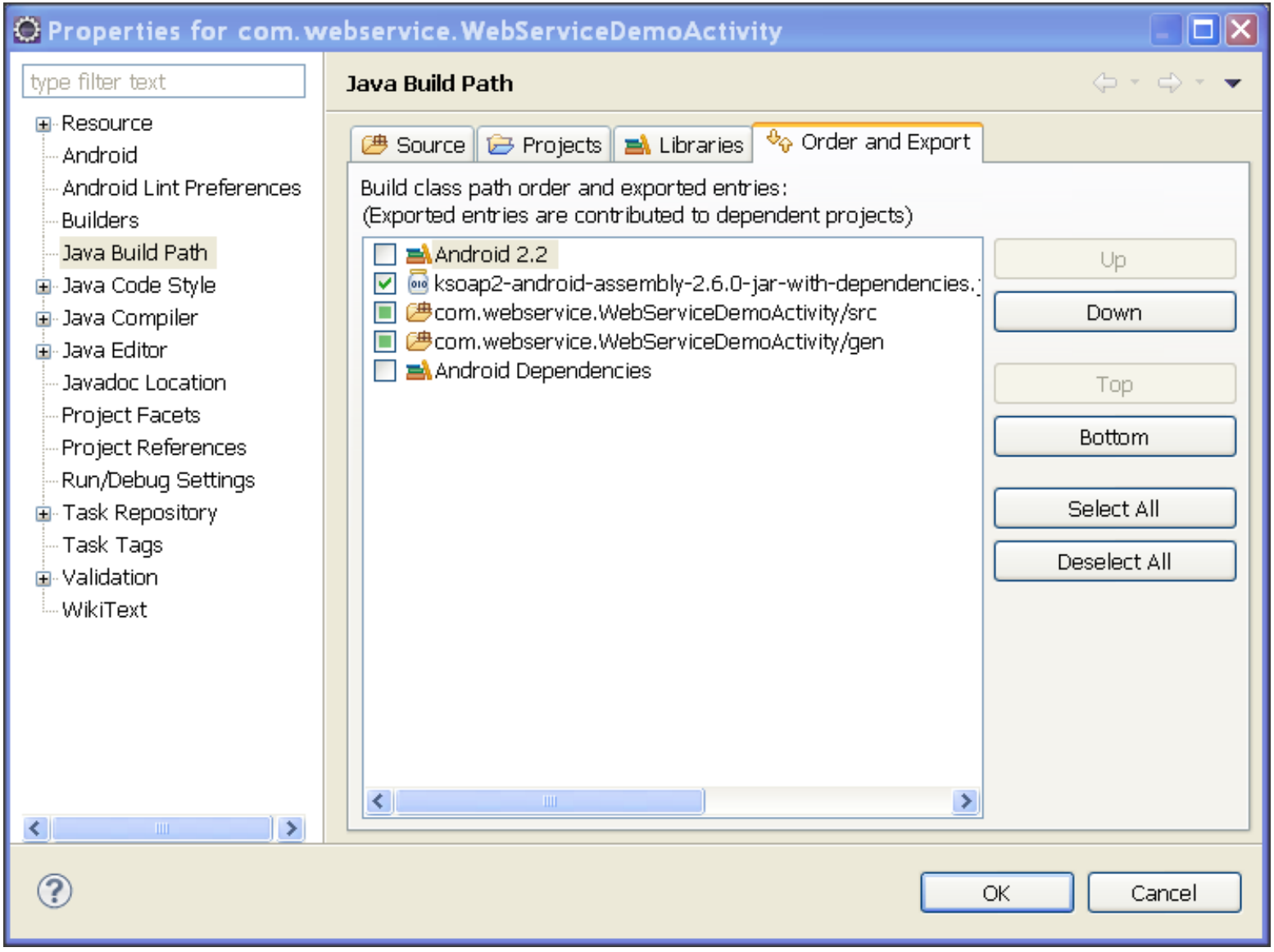
Contents added by Rajesh Pisode on Dec 11, 2012
Really Nice dude.....
Keep posting such kind of blogs....

Contents added by Chintan Rathod on Nov 21, 2012
Hi Friends,

I searched for the actual error that has Eclipse problem for "NoClassFound".
I have fixed this by following.

Right click "project" -> Properties -> Java Build Path -> Order And Explort

---



Now, make sure that, you have "Checked" that external added "jar" file as you can see in the snap.
Make sure that order of that external jar file also same like displayed in snap.

Now, Run your project and enjoy.

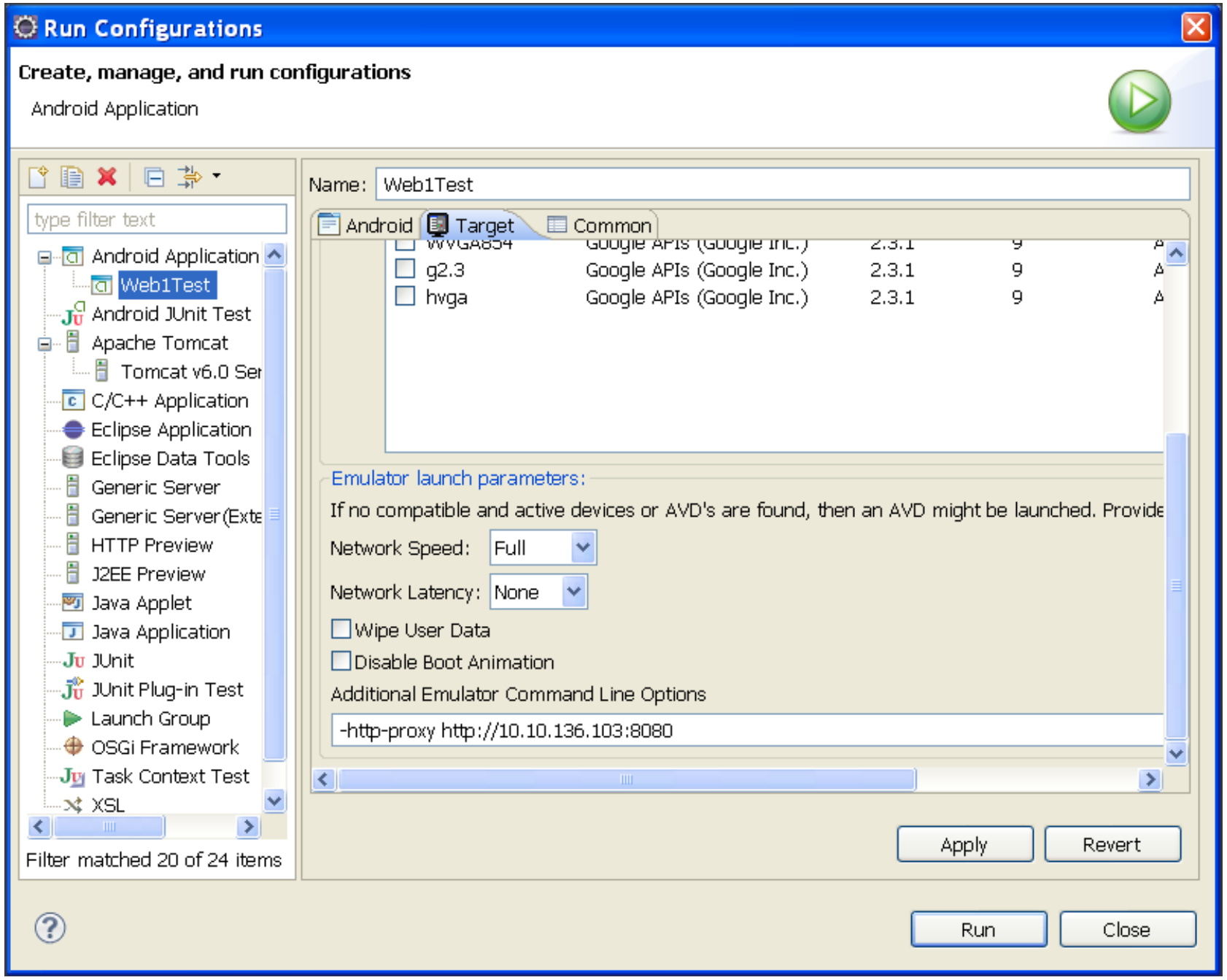
Thanks & Regards

Chintan Rathod

Contents added by Chintan Rathod on Oct 23, 2012
Following are steps for **How to configure Proxy in "Run Configuration" when you are using emulator**

Step 1:

Right click on "project" -> Run As -> Run Configuration



Proxy -> your internet access IP address
port -> default 8080

Then click on "Apply".

Now, you can try to run web application in emulator, it will not give you any error regarding network connection.

Develop Mobile Applications Using PhoneGap and Kendo UI: Part 1
Tags: Android development, Kendo UI, Mobile Applications Kendo UI, Mobile Applications using PhoneGap, PhoneGap

Prev

Android Installation Steps
Tags: Android Development, Android Installation, Eclipse, Eclipse Installation

Next

RELATED ARTICLES

- Difference Between Calling a Web Service using jQuery and AJAX
- Chapter 33: Advanced Web Services
- Calling a web service into another web service application
- Developing SOAP Web Services with PHP
- SOAP Client in Windows XP
- Chapter 32: Web Service Standards and Extensions
- Web Service Optimization
- Simple Calculator Using Mono for Android
- Passing Data From One Activity to Another Activity in Android
- Working With Text Controls in Mono for Android

POST COMMENT

Post Comment

The application crashes when it is run. Even after setting the proxy as specified. Any suggestions

Posted by Vijay Kumar on Jan 23, 2013

can you send me your code => rathod[dot]chintan[at]yahoo[dot]com

Posted by Chintan Rathod on Jan 23, 2013

please help me ;(

Posted by arslan tariq on Jan 22, 2013

hello arslan here @anyone i have made this on android 4.1 every thing went fine.. but only clear button is working ... both other buttons are not working ;(why??? tell me what to do ??? where i will see the out put??

Posted by arslan tariq on Jan 22, 2013

hello arslan here @anyone i have made this on android 4.1 every thing went fine.. but only clear button is working ... both other buttons are not working ;(why??? tell me what to do ??? where i will see the out put??

Posted by arslan tariq on Jan 22, 2013

View All Comments

COMMENT USING

6 comments

Pavan Tilak · Juniour Android Developer at Dharani Info Technologies Pvt. Ltd.

good tutorial

Reply · 1 · Like · December 23, 2012 at 5:05am

Trần Duy Thanh

Thanks
excellent

Reply · Like · January 11 at 9:00am

Melbourne Lopes · Android Developer at Biz Technologies Pvt. Ltd.

Thanks buddy.....very nice tutorial....:)

Reply · Like · January 9 at 1:02am

Ravi Verma · Software Developer at VSK technologies pvt ltd.

It's really good..

Reply · Like · January 8 at 9:58pm

Nikos Stavropoulos · Univercity of Piraeus

very good.

Reply · 2 · Like · December 23, 2012 at 6:55am

View 1 more

Facebook social plugin