EDDIE@BLOG

MONDAY, MAY 10, 2010

Comsuming WCF Services With Android

It seems processing XML is too heavy for mobile devices. Android did not provide any tool to help consuming SOAP web service. But as Android bundled with org.apache.http and org.json packages, it is relative simple to consume RESTful WCF services.

The following sections describe the steps to create RESTfule WCF services and the Android client to consume the services.

First, I created a service contract with two GET and one POST operations. Since the Android client will transfer data in JSON objects, I specify JSON as request and response format. In order to support more than one parameter, I set *BodyStyle* to *WrappedRequest*.

```
namespace HttpWcfWeb
{
    [ServiceContract(Namespace = "http://services.example.com")]
    public interface IVehicleService
    {
        [OperationContract]
        [WebGet(
            UriTemplate = "GetPlates",
            BodyStyle = WebMessageBodyStyle.WrappedRequest,
            ResponseFormat = WebMessageFormat.Json,
            RequestFormat = WebMessageFormat.Json)]
        IList<string> GetPlates();

        [OperationContract]
        [WebGet(UriTemplate = "GetVehicle/{plate}",
            BodyStyle = WebMessageBodyStyle.WrappedRequest,
            ResponseFormat = WebMessageFormat.Json,
            RequestFormat = WebMessageFormat.Json)]
        Vehicle GetVehicle(string plate);

        [OperationContract]
        [WebInvoke(
            Method = "POST",
            UriTemplate = "SaveVehicle",
            BodyStyle = WebMessageBodyStyle.WrappedRequest,
            ResponseFormat = WebMessageFormat.Json,
            RequestFormat = WebMessageFormat.Json)]
        void SaveVehicle(Vehicle vehicle);
    }
}
```

Next, I defined the composite object will be transferred between server and Android client. It is simple but enough to prove we will be able to transfer complex objects.

```
namespace HttpWcfWeb
{
    [DataContract]
    public class Vehicle
    {
        [DataMember(Name = "year")]
        public int Year
        {
            get;
            set;
        }

        [DataMember(Name = "plate")]
        public string Plate
        {
            get;
            set;
        }

        [DataMember(Name = "make")]
        public string Make
        {
            get;
            set;
        }

        [DataMember(Name = "model")]
        public string Model
        {
```

BLOG ARCHIVE

▶ 2011 (1)

▼ 2010 (4)

▼ May (1)

Comsuming WCF Services With Android

▶ January (3)

LABELS

Android (1)
C# (1)
Diff (1)
Networking (1)
RESTful (1)
Silverlight (1)
TFS (1)
WCF (1)
WPF (1)
XPS (1)

```
            get;
            set;
        }
    }
}
```

Now, expose the WCF service via *webHttp* behavior in *web.config*.

```
<system.serviceModel>
  <behaviors>
    <endpointBehaviors>
      <behavior name="httpBehavior">
        <webHttp />
      </behavior>
    </endpointBehaviors>
    <serviceBehaviors>
      <behavior name="">
        <serviceMetadata httpGetEnabled="true" />
        <serviceDebug includeExceptionDetailInFaults="false" />
      </behavior>
    </serviceBehaviors>
  </behaviors>
  <serviceHostingEnvironment multipleSiteBindingsEnabled="true" />
    <services>
      <service name="HttpWcfWeb.VehicleService">
        <endpoint address=""
                  behaviorConfiguration="httpBehavior"
                  binding="webHttpBinding"
                  contract="HttpWcfWeb.IVehicleService" />
    </service>
  </services>
</system.serviceModel>
```

It you are using Visual Studio's Development Server to test the WCF service, you may need to deploy the service to IIS. This is due to the Development Server only serve request from local machine, and the Android client won't be able to access the service hosted on it.

Further, if you are using host name (e.g. computer name) in the URL of the service, you may have to setup the DNS in you device or emulator, so that it can resolve the host name. Simply go to Settings -> Wireless Control -> Mobile Networks -> Access Point Names, click on the one that is in use, fill in *Proxy* and *Port* with your DNS server.

Now, I have my WCF service ready, and I am going to build the Android client to consume the WCF service.

During initialization, the *Activity* will invoke *IVehicleService.GetPlates* method to populate the *Spinner*. When the *Load Vehicle* button is clicked, the vehicle will be loaded from the *IVehicleService.GetVehicle* method and the *EditText* views will be populated. On the other hand, *Save* button will wrap the data entered and post to *IVehicleService.SaveVehicle* method.

The code the initialize the UI I created.

```
public class MainActivity extends Activity {

    private final static String SERVICE_URI = "http://lt0.studio.entail.ca:8080/VehicleService.s

    private Spinner plateSpinner;
    private EditText makeEdit;
    private EditText plateEdit;
    private EditText yearEdit;
    private EditText modelEdit;

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

        plateSpinner = (Spinner)findViewById(R.id.plate_spinner);
        makeEdit = (EditText)findViewById(R.id.make_edit);
        plateEdit = (EditText)findViewById(R.id.plate_edit);
        yearEdit = (EditText)findViewById(R.id.year_edit);
        modelEdit = (EditText)findViewById(R.id.model_edit);
    }

    @Override
    public void onResume() {
        super.onResume();

        // Invoke IVehicleService.GetPlates and populate plateSpinner
        refreshVehicles();
    }
}
```

The *refreshVehicles* method will be invoked when the activity is resumed or a new vehicle is saved. It send a GET request to the WCF service and retrieves a list of plates in JSON string, and the response string is parsed by *JSONArray*.

```
private void refreshVehicles() {
    try {

        // Send GET request to <service>/GetPlates
        HttpGet request = new HttpGet(SERVICE_URI + "/GetPlates");
```

```
        request.setHeader("Accept", "application/json");
        request.setHeader("Content-type", "application/json");

        DefaultHttpClient httpClient = new DefaultHttpClient();
        HttpResponse response = httpClient.execute(request);

        HttpEntity responseEntity = response.getEntity();

        // Read response data into buffer
        char[] buffer = new char[(int)responseEntity.getContentLength()];
        InputStream stream = responseEntity.getContent();
        InputStreamReader reader = new InputStreamReader(stream);
        reader.read(buffer);
        stream.close();

        JSONArray plates = new JSONArray(new String(buffer));

        // Reset plate spinner
        ArrayAdapter<String> adapter = new ArrayAdapter<String>(this, android.R.layout.simple_sp
        adapter.setDropDownViewResource(android.R.layout.simple_spinner_dropdown_item);
        for (int i = 0; i < plates.length(); ++i) {
            adapter.add(plates.getString(i));
        }
        plateSpinner.setAdapter(adapter);

    } catch (Exception e) {
        e.printStackTrace();
    }
}
```

The *onLoadVehicleClick* method is the event handler for *Load Vehicle* button. Just like *refreshVehicles* method, It send a GET request to the WCF service and retrieve the vehicle information by plate number. But instead of *JSONArray*, it used *JSONObject* to parse the response data, since the WCF service is returning an vehicle object.

```
public void onLoadVehicleClick(View button) {
    try {
        // Send GET request to <service>/GetVehicle/<plate>
        DefaultHttpClient httpClient = new DefaultHttpClient();
        HttpGet request = new HttpGet(SERVICE_URI + "/GetVehicle/" + plateSpinner.getSelectedIte

        request.setHeader("Accept", "application/json");
        request.setHeader("Content-type", "application/json");

        HttpResponse response = httpClient.execute(request);

        HttpEntity responseEntity = response.getEntity();

        // Read response data into buffer
        char[] buffer = new char[(int)responseEntity.getContentLength()];
        InputStream stream = responseEntity.getContent();
        InputStreamReader reader = new InputStreamReader(stream);
        reader.read(buffer);
        stream.close();

        JSONObject vehicle = new JSONObject(new String(buffer));

        // Populate text fields
        makeEdit.setText(vehicle.getString("make"));
        plateEdit.setText(vehicle.getString("plate"));
        modelEdit.setText(vehicle.getString("model"));
        yearEdit.setText(vehicle.getString("year"));

    } catch (Exception e) {
        e.printStackTrace();
    }
}
```

When *Save* button is clicked, *onSaveVehicleClick* method will be invoked. It simply gather all text fields into a *JSONObject* and post it to the WCF service. Noticed that the all the data was wrapped into an object named *vehicle*, WCF will pass this object as parameter *vehicle*.

```
public void onSaveVehicleClick(View button) {

    try {

        Editable make = makeEdit.getText();
        Editable plate = plateEdit.getText();
        Editable model = modelEdit.getText();
        Editable year = yearEdit.getText();

        boolean isValid = true;

        // Data validation goes here

        if (isValid) {

            // POST request to <service>/SaveVehicle
            HttpPost request = new HttpPost(SERVICE_URI + "/SaveVehicle");
            request.setHeader("Accept", "application/json");
            request.setHeader("Content-type", "application/json");

            // Build JSON string
            JSONStringer vehicle = new JSONStringer()
                .object()
                    .key("vehicle")
                        .object()
                            .key("plate").value(plate)
                            .key("make").value(make)
```

```
                            .key("model").value(model)
                            .key("year").value(Integer.parseInt(year.toString()))
                        .endObject()
                    .endObject();
            StringEntity entity = new StringEntity(vehicle.toString());

            request.setEntity(entity);

            // Send request to WCF service
            DefaultHttpClient httpClient = new DefaultHttpClient();
            HttpResponse response = httpClient.execute(request);

            Log.d("WebInvoke", "Saving : " + response.getStatusLine().getStatusCode());

            // Reload plate numbers
            refreshVehicles();
        }

    } catch (Exception e) {
        e.printStackTrace();
    }
}
```

Finally, add the internet permission into *AndroidManifest.xml*, to allow the sample application access WCF service.

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

And the demo is ready to go.

POSTED BY EDDIE LIN

REACTIONS: useful (0) interesting (0) cool (0)

LABELS: ANDROID, RESTFUL, WCF

11 COMMENTS:

merrjep.com said...

Have you also created svc-file also?
do you have the samplecode?

SEPTEMBER 16, 2010 AT 7:35 PM

merrjep.com said...

How would it differ if I was using Linq to SQL-classes?

Do I need to create IVehicleService and add Datamember?

SEPTEMBER 16, 2010 AT 7:47 PM

dryale said...

Your examples have been so very helpful, I can't thank you enough. I'm having some trouble passing/receiving collections. I see you show example of receiving an iList as JSONArray. Can you show example of sending JSONArray > iList using HttpPost?

SEPTEMBER 16, 2010 AT 10:34 PM

Martin said...

You have saved my week, and perhaps even my reputation ;)

This example was AWESOME. Help me LOADS.

OCTOBER 20, 2010 AT 9:59 AM

Anonymous said...

I try to follow your tutorial but get some error this is happen when
I try to create JSONObject by sending buffer as this statement

JSONObject t = new JSONObject(new String(buffer));

when above statement is executed it gonna throw exception and go to catch. I found it is "JSONObject text must begin with '{' at Character 1 of ...

Any help or suggestion
Thanks

NOVEMBER 17, 2010 AT 9:56 AM

Oattie said...

I try to test this tutorial on C# Console Application. I initiated the Webservice instance but it throws an

InvalidOperationException exception.

"Could not find default endpoint element that references contract 'HttpWcfWeb.IVehicleService' in the ServiceModel client configuration section ". What I did is it's exactly the same as the tutorial.

I think it cannot find the endpoint in the Web.config file. I have no idea to go furture.

Is there anyone got the same exception?

NOVEMBER 18, 2010 AT 2:49 AM

Anonymous said...

Thanks Eddie, this was such a helpful resource, cheers.

NOVEMBER 23, 2010 AT 4:58 AM

Ferns said...

Hi ,

How do i send an array of "Vehicles" to the Service ? I tried sendding it as an JSONArray to my C# method which takes List as an paramter nothing happens ?? Any idea ??

FEBRUARY 16, 2011 AT 7:20 AM

Rock said...

Hi,
Thanx a lot.. the tutorial was really very helpfull

JANUARY 24, 2012 AT 4:10 AM

Anonymous said...

can get me course android, please

MARCH 16, 2012 AT 6:24 AM

katakit.ma said...

http://www.katakit.ma

APRIL 7, 2013 AT 7:51 PM

POST A COMMENT

Enter your comment...

Comment as: Google Accou

Publish Preview

LINKS TO THIS POST

Create a Link

Newer Post Home Older Post

Subscribe to: Post Comments (Atom)

Eddie Lin