



**It's happening now!**  
A two-day parade of epic deals



**prime day**



## Calling RESTful services from your Android app

By **William J. Francis**  in **Software Engineer**, in **Mobility**  on March 20, 2012, 12:55 AM PST

Mobile developer William J. Francis demonstrates how easy it is to consume a RESTful service from your Android device.

RESTful services are well suited for providing content to small footprint devices like smartphones and tablets. In fact, if you've interacted with any cloud-based APIs in the last couple of years, there is a strong chance that API was exposed via a REST interface. There are a number of documents available on the web explaining what constitutes a REST service and how to implement one. There are also a number of write-ups explaining how to consume REST services from a client. Yet a common request I see in Android forums is for examples of Java-based Android specific REST consumers.

Recently my son and I released an update for [our game](http://www.cheesejedi.com/) (<http://www.cheesejedi.com/>), which included an online leaderboard. The server code is pretty simple — it keeps track of the five best scores on each of the 40 levels. Because of its light-weight and relative ease of implementation, I exposed the leaderboard to my app (and anyone else who is interested) via a RESTful HTTP GET call. For instance to see the top five players on level #1, just pop this URL into any browser: [http://www.cheesejedi.com/rest\\_services/get\\_big\\_cheese?level=1](http://www.cheesejedi.com/rest_services/get_big_cheese?level=1) ([http://www.cheesejedi.com/rest\\_services/get\\_big\\_cheese?level=1](http://www.cheesejedi.com/rest_services/get_big_cheese?level=1)).

### Figure A



([https://tr1.cbsistatic.com/hub/i/2015/05/07/8a16b0d6-f4ac-11e4-940f-14feb5cc3d2a/http\\_get\\_02.png](https://tr1.cbsistatic.com/hub/i/2015/05/07/8a16b0d6-f4ac-11e4-940f-14feb5cc3d2a/http_get_02.png))

Calling a REST endpoint from within an Android application requires pushing an HTTP request to the background thread and then parsing the results on the UI thread. It's not very difficult once you see it laid out in front of you, but getting started can sometimes be a bit of a hurdle.

This tutorial includes a pared down version of the code actually used in my production application. Hopefully it will prove useful to you in getting started with your own Android REST consumers. You can follow along with the step-by-step tutorial, or [download the entire project to import into Eclipse](http://b2b.cbsimg.net/downloads/Weilage/http_get.zip) ([http://b2b.cbsimg.net/downloads/Weilage/http\\_get.zip](http://b2b.cbsimg.net/downloads/Weilage/http_get.zip)).

1. Create a new Android project in Eclipse. Target SDK 1.6 or greater, and be sure to rename the startup activity to Main.java.
2. In your /res/layout folder, create a main.xml resource that will define a button and an edit box.

main.xml

```
<?xml version="1.0" encoding="utf-8"?>
```

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
```

```
    android:layout_width="fill_parent"
```

```
    android:layout_height="fill_parent"
```

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

```
    <TextView
```

```
        android:layout_width="fill_parent"
```

```
        android:layout_height="wrap_content"
```

```
        android:text="Http GET Demo"/>
```

```
    <Button
```

```
android:layout_width="wrap_content"
```

```
android:layout_height="wrap_content"
```

```
android:layout_gravity="center"
```

```
android:text="GET"
```

```
android:id="@+id/my_button"/>
```

```
<EditText
```

```
android:layout_margin="20dip"
```

```
android:layout_width="fill_parent"
```

```
android:layout_height="wrap_content"
```

```
android:minLines="15"
```

```
android:maxLines="15"
```

```
android:textSize="12sp"
```

```
android:editable="false"
```

```
android:id="@+id/my_edit"/>
```

```
</LinearLayout>
```

### 3. Because our application will be making HTTP calls, we will have to declare the uses internet permission in the manifest.

AndroidManifest.xml

```
<?xml version="1.0" encoding="utf-8"?>
```

```
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
```

```
package="com.authorwjf.http_get"
```

```
android:versionCode="1"
```

```
android:versionName="1.0" >
```

```
<uses-sdk android:minSdkVersion="4" />

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

<application

    android:icon="@drawable/ic_launcher"

    android:label="@string/app_name" >

    <activity

        android:name=".Main"

        android:label="@string/app_name" >

        <intent-filter>

            <action android:name="android.intent.action.MAIN" />

            <category android:name="android.intent.category.LAUNCHER" />

        </intent-filter>

    </activity>

</application>

</manifest>
```

#### 4. The Main.java file in our /src directory contains the standard on create override and implements an on click listener for the button.

Main.java

```
package com.authorwjf.http_get;
import java.io.IOException;
import java.io.InputStream;
import org.apache.http.HttpEntity;
import org.apache.http.HttpResponse;
import org.apache.http.client.HttpClient;
import org.apache.http.client.methods.HttpGet;
import org.apache.http.impl.client.DefaultHttpClient;
import org.apache.http.protocol.BasicHttpContext;
import org.apache.http.protocol.HttpContext;
import android.app.Activity;
import android.os.AsyncTask;
import android.os.Bundle;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.Button;
import android.widget.EditText;
public class Main extends Activity implements OnClickListener {
    @Override

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

        findViewById(R.id.my_button).setOnClickListener(this);
    }

    @Override

    public void onClick(View arg0) {
        Button b = (Button)findViewById(R.id.my_button);

        b.setClickable(false);
        new LongRunningGetIO().execute();
    }

}
```

5. Our Main.java file references something called LongRunningGetIO; this is something we need to implement as well, and for the sake of our example, we will do so as an inner class inside our Main. The inner class extends AsyncTask. If you aren't familiar with how it works, you can get up to speed by reading my post on [Long Running I/O](https://www.techrepublic.com/blog/app-builder/using-androids-async-task-to-handle-long-running-io/670)

(<https://www.techrepublic.com/blog/app-builder/using-androids-async-task-to-handle-long-running-io/670>).

Main.java

```
private class LongRunningGetIO extends AsyncTask <Void, Void, String> {  
protected String getASCIIContentFromEntity(HttpEntity entity) throws IllegalStateException,  
IOException {  
InputStream in = entity.getContent();
```

```
StringBuffer out = new StringBuffer();  
int n = 1;  
while (n>0) {  
byte[] b = new byte[4096];  
n = in.read(b);
```

```
if (n>0) out.append(new String(b, 0, n));  
}
```

```
return out.toString();  
}
```

@Override

```
protected String doInBackground(Void... params) {  
HttpClient httpClient = new DefaultHttpClient();  
HttpContext localContext = new BasicHttpContext();  
HttpGet httpGet = new HttpGet("http://www.cheesejedi.com/rest_services/get_big_cheese.php?  
puzzle=1");  
String text = null;  
try {  
HttpResponse response = httpClient.execute(httpGet, localContext);
```

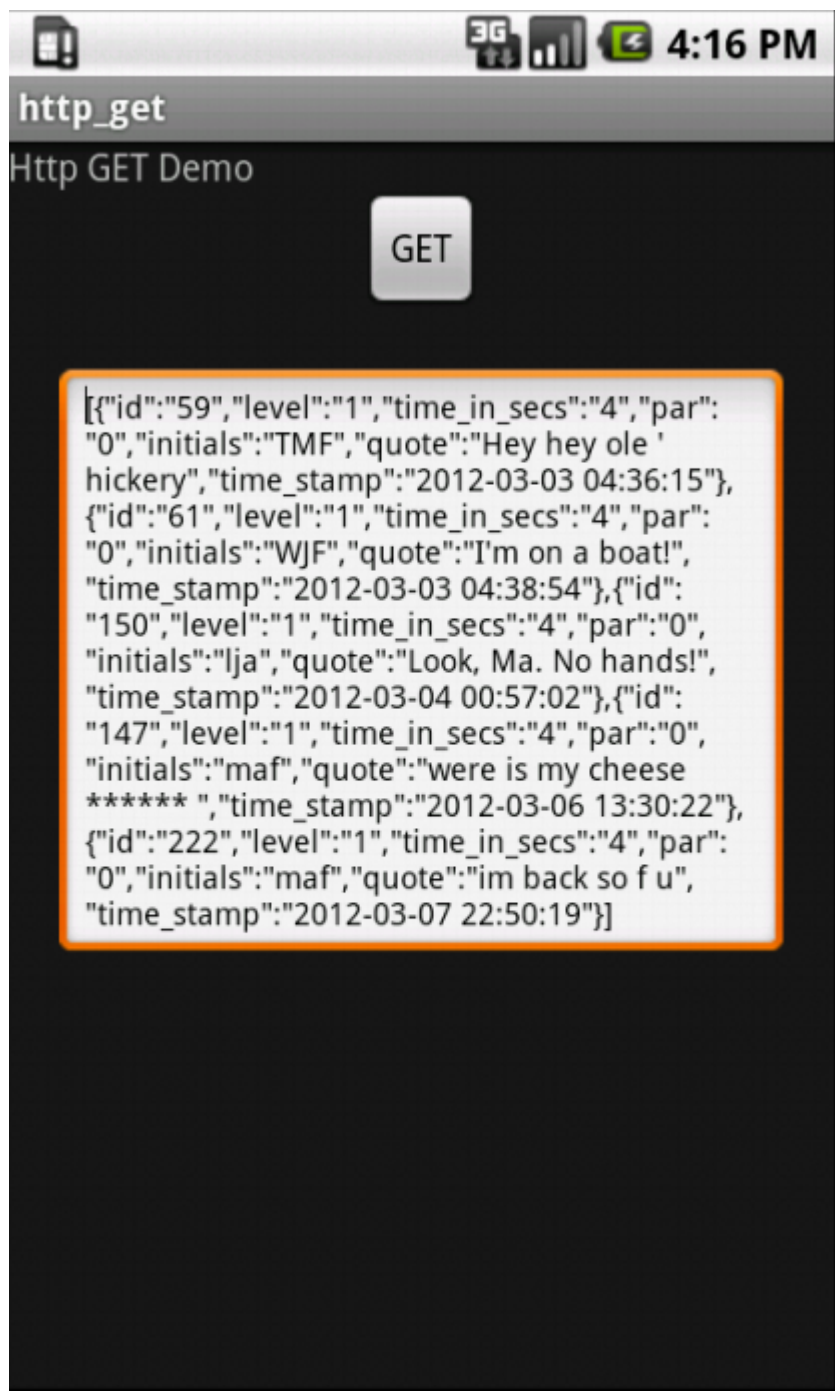
```
HttpEntity entity = response.getEntity();
```

```
text = getASCIIContentFromEntity(entity);
```

```
} catch (Exception e) {  
return e.getLocalizedMessage();  
}  
  
return text;  
}  
  
protected void onPostExecute(String results) {  
if (results!=null) {  
EditText et = (EditText)findViewById(R.id.my_edit);  
  
et.setText(results);  
  
}  
  
Button b = (Button)findViewById(R.id.my_button);  
  
b.setClickable(true);  
}  
  
}
```

The demo should be ready for you to try. Upload the APK to an emulator and tap the button. Providing you have an Internet connection and the server isn't overloaded with requests, you should get something back. The demo doesn't do any parsing on the result string, but it should be clear how easy it would be to encompass the results in a JSON array and display them in a meaningful manner.

## Figure B



([https://tr1.cbsistatic.com/hub/i/2015/05/07/8a6b8087-f4ac-11e4-940f-14feb5cc3d2a/http\\_get.png](https://tr1.cbsistatic.com/hub/i/2015/05/07/8a6b8087-f4ac-11e4-940f-14feb5cc3d2a/http_get.png))

Congratulations — you've just consumed a RESTful service from your Android device!

WHITE PAPERS, WEBCASTS, AND DOWNLOADS



## The Elephant in the Room: Why Cloud Communications Providers Dont Like to Talk About Compliance

White Papers from 8x8, Inc.

[DOWNLOAD NOW](#)

## How to Solve the Enterprise Communications Crisis with an Open Cloud Strategy

White Papers from 8x8, Inc.

[DOWNLOAD NOW](#)

## Cisco Umbrella Case Study: How cloud security keeps patient and employee data safe at Memorial Hermann Health System

White Papers from Cisco

[DOWNLOAD NOW](#)

## Tech Pro Research: IT Budget Research Report 2019

White Papers from TechRepublic.com

[CONTINUE](#)

## Avast Business Patch Management Subscription

Downloads from Avast Business

[LEARN MORE](#)



---

## EDITOR'S PICKS



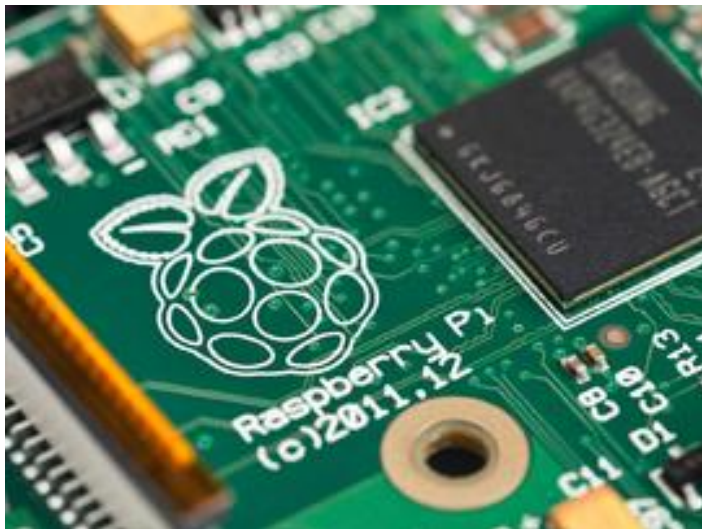
**Python is eating the world: How one developer's side project became the hottest programming language on the planet**



**How iRobot used data science, cloud, and DevOps to design its next-gen smart home robots**



## Beyond the PC: Lenovo's ambitious plan for the future of computing



**Inside the Raspberry Pi: The story of the \$35 computer that changed the world**

## Straight up: How the Kentucky bourbon industry is going high tech



**Smart farming: How IoT, robotics, and AI are tackling one of the biggest problems of the century**



### By William J. Francis

William J Francis began programming computers at age eleven. Specializing in embedded and mobile platforms, he has more than 20 years of professional software engineering under his belt, including a four year stint in the US Army's Military Intelligence...

[MOBILITY](#)[SECURITY](#)[HARDWARE](#)[SOFTWARE](#)[APPLE](#)[ANDROID](#)[GOOGLE](#)[MOBILITY ON ZDNET](#) ➔

## Recommended

Promoted Links by Taboola

### Which 2019 Pickup Truck Is Worth Buying?

Kelley Blue Book

### You're Probably Overpaying at Amazon – This Genius Trick Will Fix That

Honey

### Hawaii Will Pay You to Install Solar if You Live Near Honolulu

Energy Bill Cruncher Solar Quotes

### 5G and sports: Study reveals younger audiences will pay more for a better experience

## **Infographic: Why email is your weakest security link**

### **News, Tips, and Advice for Technology Professionals**

---

#### **EDITOR'S PICKS**

**Python is eating the world: How one developer's side project became the hottest programming language on the planet**

**How iRobot used data science, cloud, and DevOps to design its next-gen smart home robots**

**Beyond the PC: Lenovo's ambitious plan for the future of computing**

**Straight up: How the Kentucky bourbon industry is going high tech**

#### WHITE PAPERS, WEBCASTS, AND DOWNLOADS

8x8's Enterprise Engagement Management Platform: Moving Toward an Integrate...

White Papers From [8x8, Inc.](#)

**DOWNLOAD NOW**

Securing Students' Safety Everywhere

White Papers From [Cisco](#)

**DOWNLOAD NOW**

Ransomware: How healthcare organizations can stay ahead of attacks

White Papers From [Cisco](#)

**DOWNLOAD NOW**

Don't Risk It. Higher Education Cybersecurity 101

White Papers From [Cisco](#)

**DOWNLOAD NOW**