About This Article

This Article gives an overview on networking options available for Android applications development and about basic Android networking skills. It provides to learn how to leverage Android's array of networking options for Application development. Explore a real-world application that requires networking when used with an environmental monitoring system. A sample code snippet for practice is provided in this Article.

Scope:

This article is intended for novice users who want brief understanding on Android programming. This articleguides you step by step on developing Network based Applications in Android. It assumes that the user has already installed Android and necessary tools to develop application. It also assumes that the user to be familiar with Java or familiar with Object Oriented Programming concepts.

Introduction

Network programming plays an important role in wireless application development. Android contains the Apache HttpClient Library and this library is the preferred way of performing network operations in Android. Android also allows accessing the network via the standard Java Networking API (java.net package). Even if java.net package is used it internally uses the Apache library. To access the internet your application requires the "android.permission.INTERNET" Permission

Network Related Packages in Android

Below are the some of the Network Related package in Android SDK

- **java.net**
  Provides networking-related classes, including stream and datagram sockets, Internet Protocol, and generic HTTP handling. This is the multipurpose networking resource. Experienced Java developers can create applications right away with this familiar package.
- **java.io**
  Though not explicitly networking, it's very important. Classes in this package are used by sockets and connections provided in other Java packages. They're also used for interacting with local files (a frequent occurrence when interacting with the network).
- **java.nio**
  Contains classes that represent buffers of specific data types. Handy for network communications between two Java language-based end points.
- **org.apache.***
  Represents a number of packages that provide fine control and functions for HTTP communications. You might recognize Apache as the popular open source Web server.
- **android.net**
  Contains additional network access sockets beyond the core java.net.* classes. This package includes the URI class, which is used frequently in Android application development beyond traditional networking.
- **android.net.http**
  Contains classes for manipulating SSL certificates
- **android.net.wifi**
  Contains classes for managing all aspects of WiFi (802.11 wireless Ethernet) on the Android platform. Not all devices are equipped with WiFi capability, particularly as Android makes headway in the "flip-phone" strata of cell phones from manufacturers like Motorola and LG.

- **android.telephony.gsm**
  Contains classes required for managing and sending SMS (text) messages. Over time, an additional package will likely be introduced to provide similar functions on non-GSM networks, such as CDMA, or something like android.telephony.cdma.

Android Networking

Using Apache HttpClient Library

In Apache HttpClient library below are the packages useful for Network Connections

- org.apache.http.HttpResponse
- org.apache.http.client.HttpClient
- org.apache.http.client.methods.HttpGet
- org.apache.http.impl.client.DefaultHttpClient

```
HttpClient httpclient=new DefaultHttpClient();
```

To retrieve the information from the server use the HttpGet class constructor

```
HttpGet request=new HttpGet(“http://developer.samsung.com”);
```

Then pass the HttpGet object in method execute() of HttpClient class to retrieve HttpResponse object

```
HttpResponse response = client.execute(request);
```

Then read the response received

```
BufferedReader rd = new BufferedReader
                                        (new
InputStreamReader(response.getEntity().getContent()));
    String line = "";
    while ((line = rd.readLine()) != null) {
        Log.d(“output: ”,line);
    }
```

Sample Example

Given below is the sample code snippet for using Apache HttpClient library for Network Connection Add the permission "android.permission.INTERNET" to "AndroidManifest.xml" to allow your application to access the internet.

AndroidManifest.xml

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
      package="com.apache"
      android:versionCode="1"
      android:versionName="1.0">
    <uses-sdk android:minSdkVersion="8" />
<uses-permission android:name="android.permission.INTERNET"></uses-permission>
    <application android:icon="@drawable/icon" android:label="@string/app_name">
        <activity android:name=".ApacheConnection"
                  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>
```

Create the project "com.apache" with the activity "ApacheConnection". Change the layout "main.xml" to the following.

Main.xml

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    >
    <TextView
        android:text="Enter URL"
        android:id="@+id/textView1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content">
    </TextView>

    <EditText
        android:id="@+id/editText1"
        android:layout_width="match_parent"
        android:text="http://developer.samsung.com"
        android:layout_height="wrap_content">
    </EditText>

    <Button
        android:text="Click Here"
        android:id="@+id/button1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content">
     </Button>

    <EditText
        android:id="@+id/editText2"
        android:layout_width="match_parent"
        android:layout_height="fill_parent">
     </EditText>

</LinearLayout>
```

Now create a code in java which will allow viewing the HTML code.

ApacheConnection.java

```
package com.apache;

import java.io.BufferedReader;
import java.io.InputStreamReader;

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 android.app.Activity;
import android.os.Bundle;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.Button;
import android.widget.TextView;

public class ApacheConnection extends Activity {

        Button bt;
        TextView textView1;
        TextView textView2;

        /** Called when the activity is first created. */
        @Override
        public void onCreate(Bundle savedInstanceState) {
                super.onCreate(savedInstanceState);
                setContentView(R.layout.main);
                bt = (Button) findViewById(R.id.button1);
                textView1 = (TextView) findViewById(R.id.editText1);
                textView2 = (TextView) findViewById(R.id.editText2);
            bt.setOnClickListener(new OnClickListener() {
                      @Override
                public void onClick(View v) {
                    // TODO Auto-generated method stub
                    textView2.setText("");

                         try {
                            /*Apache HttpClient Library*/
                    HttpClient client = new DefaultHttpClient();
                    HttpGet request = new HttpGet(textView1.getText().toString());
                    HttpResponse response = client.execute(request);
```

```
                    /* response code*/
                    BufferedReader rd = new BufferedReader(
                           new InputStreamReader(response.getEntity().getContent()));
                            String line = "";
                    while ((line = rd.readLine()) != null) {
                           textView2.append(line);
                            }
                  } catch (Exception exe) {
                       exe.printStackTrace();
                           }
                }
                 });

        }
}
```

Using Java Networking API

In Java Networking API below are the packages useful for Network Connections

- •java.net.URL
- •java.net.URLConnection

By creating the URL instance specify the location of the resource on the internet

```
URL url=new URL("http://developer.samsung.com");
```

Then use the URLConnection class for reading or writing

```
URLConnection conn = url.openConnection();
```

Then read the response received

```
BufferedReader rd = new BufferedReader
                                        (new
InputStreamReader(response.getEntity().getContent()));
    String line = "";
    while ((line = rd.readLine()) != null) {
         System.out.println(line);
    }
```

Sample Example

Given below is the sample code snippet for using Java.net packages for Network Connection Add the permission "android.permission.INTERNET" to "AndroidManifest.xml" to allow your application to access the internet.

AndroidManifest.xml

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
   package="com.net"
```

```
   android:versionCode="1"
   android:versionName="1.0">
  <uses-sdk android:minSdkVersion="8" />
  <uses-permission android:name="android.permission.INTERNET"></uses-permission>
    <application android:icon="@drawable/icon" android:label="@string/app_name">
     <activity android:name=".NetworkingProject" 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>
```

Create the project "com.net" with the activity "NetworkingProject". Change the layout "main.xml" to the following.

Main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    >
    <TextView
        android:text="Enter URL"
        android:id="@+id/textView1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content">
    </TextView>

    <EditText
        android:id="@+id/editText1"
        android:layout_width="match_parent"
        android:text="http://developer.samsung.com"
        android:layout_height="wrap_content">
    </EditText>

    <Button
        android:text="Click Here"
        android:id="@+id/button1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content">
     </Button>
```

```
    <EditText
         android:id="@+id/editText2"
         android:layout_width="match_parent"
         android:layout_height="fill_parent">
    </EditText>
</LinearLayout>
```

Now create a code in java which will allow viewing the HTML code.

NetworkingProject.java

```
package com.net;

import java.io.BufferedReader;
import java.io.InputStreamReader;
import java.net.URL;
import java.net.URLConnection;

import android.app.Activity;
import android.os.Bundle;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.Button;
import android.widget.TextView;

public class NetworkingProject extends Activity {

         Button bt;
         TextView textView1;
         TextView textView2;

         /** Called when the activity is first created. */
         @Override
         public void onCreate(Bundle savedInstanceState) {
                  super.onCreate(savedInstanceState);
                  setContentView(R.layout.main);
                  bt = (Button) findViewById(R.id.button1);
                  textView1 = (TextView) findViewById(R.id.editText1);
                  textView2 = (TextView) findViewById(R.id.editText2);

             bt.setOnClickListener(new OnClickListener() {

                 @Override
                  public void onClick(View v) {
```

```
                        // TODO Auto-generated method stub
                textView2.setText("");
                 try {
                /*Java Networking API*/
                        URL url = new URL(textView1.getText().toString());
                        URLConnection conn = url.openConnection();
                        /*Read the Response*/
                        BufferedReader rd = new BufferedReader(new
                                        InputStreamReader(conn.getInputStream()));
                String line = "";
                while ((line = rd.readLine()) != null) {
                   textView2.append(line);
                }
                } catch (Exception exe) {
                exe.printStackTrace();
                }
              }
        });

        }
}
```

Output

Figure below shows the output of above sample examples

Figure 1: Network Connection Output

ref:http://developer.samsung.com/android/technical-docs/Using-Network-Connections-in-Android