Sending HTTP Requests in Android

Method 1 – Using AsyncTask

Android provides a class called AsyncTask, which can be used to perform time-consuming operations on a new thread. To use it, you need create a new class that extends AsyncTask. For example:

```
import java.io.BufferedReader;
import java.io.InputStreamReader;
import java.net.HttpURLConnection;
import java.net.URL;
import java.util.Map;

import android.content.Context;
import android.os.AsyncTask;

public class GetAsyncTask extends AsyncTask<Void, Void, String> {

    Context context;
    String url;
    Map<String, String> params;

    public GetAsyncTask(
            Context context,
            String url,
            Map<String, String> params) {
        this.context = context;
        this.url = url;
        this.params = params;
    }

    protected void onPreExecute() {
       // Perform any operations that need to be done before the async task
       // This function runs on the UI thread
    }

    @Override
    protected String doInBackground(Void... params) {

        String query_string = "?";
        int i = 0;
        for (Map.Entry<String, String> entry : params.entrySet()) {
             if (i > 0) query_string += "&";
             query_string += entry.getKey() + "=" + entry.getValue();
        }
        url = url + query_string;

        try {

            URL url_object = new URL(url);
            HttpURLConnection conn =
                (HttpURLConnection)url_object.openConnection();
            conn.setReadTimeout(15000);
```

```
            conn.setConnectTimeout(15000);
            conn.setRequestMethod("GET");

            int responseCode = conn.getResponseCode();
            if (responseCode == HttpURLConnection.HTTP_OK) {
                String line;
                BufferedReader br = new BufferedReader(
                   new InputStreamReader(conn.getInputStream()));
                while ((line = br.readLine()) != null) {
                    results += line;
                }
            } else {
                results = "";
            }

            return results;

        } catch (Exception e) {
            e.printStackTrace();
            return "";
        }
    }

    protected void onPostExecute(String results) {
        // This function runs on the UI thread
        // Perform any operation required after you have
       // received the response from the server
    }

}
```

Method 2 – Using Volley

Volley is an HTTP library that makes networking for Android apps easier and most importantly, faster. Volley is available through the open AOSP repository. For more detail see the guide on Android's developer site: http://developer.android.com/training/volley/index.html

To use Volley in your project, you should download the source code and then import Volley as an Android library module in your project:

Step 1: Clone the source code from Google's repository

```
git clone https://android.googlesource.com/platform/frameworks/volley
```

Step 2: Import Volley as an Android library module

- Choose [File] → [New] → [Import Module]
- Select the directory where Volley is downloaded and stored
- In your project, open `settings.gradle` (in the root directory of your project), and check if the

following line is present:

include ':app', ':volley'

- Go to the `build.gradle` file under the app directory, and add the following dependency:

compile project(":volley")

After importing the library, you can then use Volley in your project. The following example illustrates how to use Volley to send out a GET request.

```
String url = "http://api.openweathermap.org/data/2.5/weather";
url = url + "?q=London,uk&appid=44db6a862fba0b067b1930da0d769e98";

// Request a string response
StringRequest stringRequest = new StringRequest(
    Request.Method.GET,
    url,
    new Response.Listener<String>() {
        @Override
        public void onResponse(String response) {
            // Handle response here
            // This runs on the UI thread
        }
    },
    new Response.ErrorListener() {
        @Override
        public void onErrorResponse(VolleyError error) {
            // Error handling
            // This runs on the UI thread
        }
    });

// Add the request to the queue
Volley.newRequestQueue(this).add(stringRequest);
```

The following example illustrates how to use Volley to send out a POST request. To send out a POST request, you have to override the getParams method:

```
String url = "http://www.abc.com/api/login";

StringRequest postRequest = new StringRequest(
    Request.Method.POST,
    url,
    new Response.Listener<String>() {
        @Override
        public void onResponse(String response) {
            // Handle response here
        }
    },
```

```
    new Response.ErrorListener() {
        @Override
        public void onErrorResponse(VolleyError error) {
            // Handle error here
        }
    }) {
        @Override
        protected Map<String, String> getParams() {
            Map<String, String>  params = new HashMap<>();
            params.put("username", "abc");
            params.put("password", "123456");
            return params;
        }
    };

// Add the request to the queue
Volley.newRequestQueue(this).add(postRequest);
```

Method 3 – Using the Android Asynchronous HTTP Client by James Smith

Android Asynchronous HTTP Client (http://loopj.com/android-async-http/) is a library developed by James Smith to allow easy implementation of HTTP requests. To use it in your project, add the following dependency into the `build.gradle` file under the app directory:

`compile 'com.loopj.android:android-async-http:1.4.9'`

For details of how this HTTP client can be used, you can refer to the documentation on the Website. A simple example of using the library to perform a GET request is show below. To perform a POST request, simply call the post() method instead of the get() method of the AsyncHTTPClient.

```
// Create a new aysnc task HTTP client
AsyncHttpClient client = new AsyncHttpClient();

// Prepare the parameters
RequestParams request_params = new RequestParams();
request_params.put("param1", "value1");
request_params.put("param2", "value2");
request_params.put("param3", "value3");

// Calls the get() method of the client
// The HTTP request is automatically performed in a new thread
client.get(
    url,
    request_params,
    new TextHttpResponseHandler() {
        @Override
        public void onFailure(
            int statusCode, Header[] headers,
```

```
            String response, Throwable throwable) {
            // Perform any action needed after a 'failure' response
            // is received
            // This function runs on the UI thread
        }
        @Override
        public void onSuccess(
            int statusCode, Header[] headers, String response) {
            // Perform any action needed after a 'success' response
            // is received
            // This function runs on the UI thread
        }
    }
);
```