# Mobile Computing

*Connecting to the Network*

## Introduction

Android applications in general are small but embedded a lot of functionality. One of the ways that mobile apps deliver such rich functionality is that they pull information from various sources including the other applications and the internet.

When grabbing information from the internet, HTTP is one of the common integration strategies. For example, you might have a Java servlet available on the Internet that provides services you want to leverage from one of your Android applications. How do you do that with Android? Android SDK is bundled with several different ways to doing that.

This tutorial explains a way of using **HttpURLConnection** to make HTTP calls in order to access the information in the internet.

## Application Configuration

### Permissions

In Android to perform the network operations it requires to obtain permission in the application manifest. The following permission tags should be in the application manifest in order to do the tutorial.

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

## Implementation

### HttpURLConnection

Android includes two HTTP clients: **HttpURLConnection** and Apache **HttpClient** both support HTTPS, streaming uploads and downloads, configurable timeouts, IPv6, and connection pooling. HttpURLConnection is recommended for applications targeted at Gingerbread and higher.

## Check the Network Connection

It is a best practice to check whether a network connection is available using getActiveNetworkInfo() and isConnected() before the app attempts to connect to the network. Otherwise the application will be crashed if there is no active internet connection.

```
ConnectivityManager connMgr = (ConnectivityManager)
        getSystemService(Context.CONNECTIVITY_SERVICE);
NetworkInfo networkInfo = connMgr.getActiveNetworkInfo();

if (networkInfo != null && networkInfo.isConnected()) {
    // fetch data
} else {
    // display error
}
```

### ***Display an Error with an Alert Dialog***

```
new AlertDialog.Builder(this)
    .setTitle("Connection Failure")
    .setMessage("Please Connect to the Internet")
    .setPositiveButton(android.R.string.ok, new DialogInterface.OnClickListener() {
         public void onClick(DialogInterface dialog, int which) {

         }
     })
    .setIcon(android.R.drawable.ic_dialog_alert)
    .show();
}
```

## Perform Network Operations on a Separate Thread

Input and output tasks like network operations can involve unpredictable delays and may cause to freeze the UI if performed in the main thread. To prevent this from causing a poor user experience, always perform network operations on a separate thread from the UI. The **AsyncTask** class provides one of the simplest ways to fire off a new task from the UI thread.

### ***AsyncTask***

The class AsyncTask contains two methods that is to be overridden;

1. doInBackground is called asynchronously by the thread and passes a String array as the argument. This method is mostly like the run() method in Java Thread class.

```
protected String doInBackground(String... urls)
```

2. onPostExecute is called when the doInBackground method returns and the returned String value is passed as the parameter of the onPostExecute method. This method can be used to utilize the output of

the asynchronous task. For an example, we can use this method to set the returned value to a View in the UI.

```
private class DownloadWebpageTask extends AsyncTask<String, Void, String> {


        @Override
        protected String doInBackground(String... urls) {
            // params comes from the execute() call
            try {
                return downloadUrl(urls[0]);
            } catch (IOException e) {
                return "Unable to retrieve web page. URL may be invalid.";
            }
        }


        // onPostExecute displays the results of the AsyncTask.
        @Override
        protected void onPostExecute(String result) {
            //textView.setText(result);
       }
    }
```

*Note: We have not made a connection yet. The above code snippet only shows how to implement an asynchronous task.*

## ***Connect and Download Data***

We call downloadUrl method inside the doInBackground method. The downloadUrl method actually connects to the network and sends the request. It is also responsible to catch the response from the network.

The following code use HTTP **GET** method for the request.

1. First create a URL object with the specified url in the application
2. Then create an HttpURLConnection object by using the factory method in the URL object.
3. Set the parameters if the HttpURLConnection object.
4. Make the connection by calling connect() method in the HttpURLConnection object.
5. Get the response code by calling getResponseCode() method in the HttpURLConnection object.
6. If the response is 200 (i.e. Success) then get the input stream of response data using getInputStream() method in the HttpURLConnection object.
7. Finally convert the data came through the input stream and return.

```
// Given a URL, establishes an HttpUrlConnection and retrieves
// the web page content as a InputStream, which it returns as
// a string.
private String downloadUrl(String myurl) throws IOException {
    InputStream is = null;
    // Only display the first 4000 characters of the retrieved
    // web page content.
    int len = 4000;

    try {
        URL url = new URL(myurl);
        HttpURLConnection conn = (HttpURLConnection) url.openConnection();
        conn.setReadTimeout(10000 /* milliseconds */);
        conn.setConnectTimeout(15000 /* milliseconds */);
        conn.setRequestMethod("GET");
        conn.setDoInput(true);
        // Starts the query
        conn.connect();
        int response = conn.getResponseCode();
        Log.d(DEBUG_TAG, "The response is: " + response);
        is = conn.getInputStream();

        // Convert the InputStream into a string
        String contentAsString = readIt(is, len);
        return contentAsString;

    // Makes sure that the InputStream is closed after the app is
    // finished using it.
    } finally {
        if (is != null) {
            is.close();
        }
    }
}
```

***<u>Convert the InputStream to a String</u>***

The input stream may contain images, texts or any type of data. If the input stream contains texts then it can be read using the following code. We set character coding for UTF-8 as it may contain Unicode characters as well. If you know that the responding data contains only ASCII data then you can specify it as well.

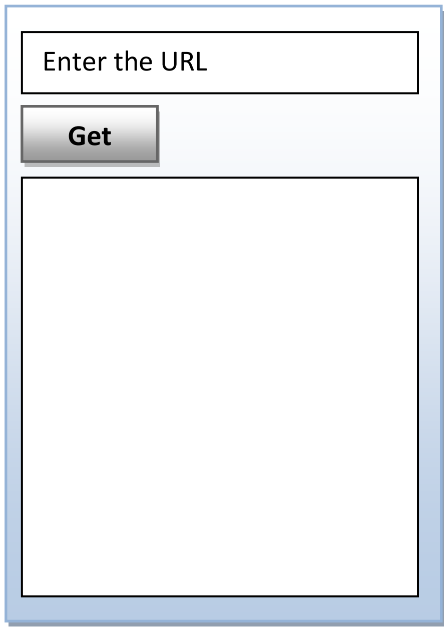
```
// Reads an InputStream and converts it to a String.
public String readIt(InputStream stream, int len) throws IOException,
UnsupportedEncodingException {
    Reader reader = null;
    reader = new InputStreamReader(stream, "UTF-8");
    char[] buffer = new char[len];
    reader.read(buffer);
    return new String(buffer);
}
```

## The Full Code Integration

```
public class HttpExampleActivity extends Activity {
    private static final String DEBUG_TAG = "HttpExample";
    private EditText urlText;
    private TextView textView;

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        urlText = (EditText) findViewById(R.id.myUrl);
        textView = (TextView) findViewById(R.id.myText);
    }


    public void myClickHandler(View view) {
        String stringUrl = urlText.getText().toString();
        ConnectivityManager connMgr = (ConnectivityManager)
            getSystemService(Context.CONNECTIVITY_SERVICE);
        NetworkInfo networkInfo = connMgr.getActiveNetworkInfo();
        if (networkInfo != null && networkInfo.isConnected()) {
            new DownloadWebpageTask().execute(stringUrl);
        } else {
            textView.setText("No network connection available.");
        }
    }

    private class DownloadWebpageTask extends AsyncTask<String, Void, String> {
        @Override
        protected String doInBackground(String... urls) {

            // params comes from the execute() call: params[0] is the url.
            try {
                return downloadUrl(urls[0]);
            } catch (IOException e) {
                return "Unable to retrieve web page. URL may be invalid.";
            }
        }
        // onPostExecute displays the results of the AsyncTask.
        @Override
        protected void onPostExecute(String result) {
            textView.setText(result);
       }

       private String downloadUrl(String myurl) throws IOException {
            InputStream is = null;
            int len = 500;

            try {
                 URL url = new URL(myurl);
                 HttpURLConnection conn = (HttpURLConnection) url.openConnection();
                 conn.setReadTimeout(10000 /* milliseconds */);
                 conn.setConnectTimeout(15000 /* milliseconds */);
                 conn.setRequestMethod("GET");
                 conn.setDoInput(true);
                 // Starts the query
                 conn.connect();
                 int response = conn.getResponseCode();
                 Log.d(DEBUG_TAG, "The response is: " + response);
                 is = conn.getInputStream();
```

```
                    String contentAsString = readIt(is, len);
                    return contentAsString;


                } finally {
                    if (is != null) {
                    is.close();
                }
            }

         public String readIt(InputStream stream, int len) throws IOException,
UnsupportedEncodingException {
                 Reader reader = null;
                 reader = new InputStreamReader(stream, "UTF-8");
                 char[] buffer = new char[len];
                 reader.read(buffer);
                 return new String(buffer);
         }
     }
}
```

## Create the Following Application



Create the above UI and implement the action of the button "Get" to request from the URL and display the response in the text view at the bottom.

## Exercise

Create a client application to Madura online dictionary. Your application should query the http://www.maduraonline.com with the given English word and then fetch the relevant Sinhala words from the responded webpage.

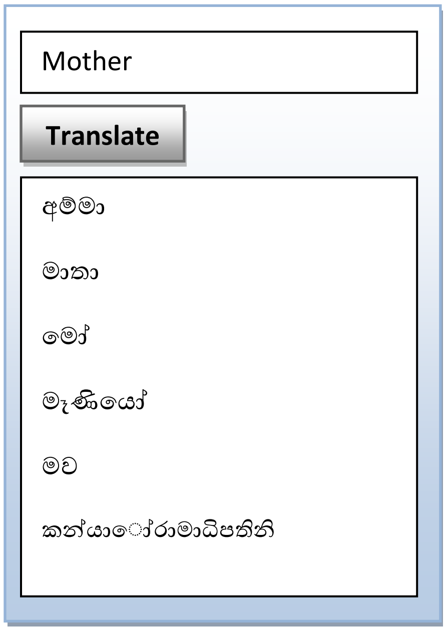
The query URL can be formed as the following;

*"http://www.maduraonline.com?find="+English_Keyword*

You can use regular expression to extract the list of relevant Sinhala translations from the html code that is sent from the web server. Since the list of words contained in an HTML table you can use the following regular expression to extract the set of Sinhala words.

*(<td>)(.+?)(</td>)*

The user interface should be as the following.



```
අම්මා
මාතා
මෝ
මෑණියෝ
මව
කන්‍යාෝරාමාධිපතිනි
```