Special Interest Group on Innovative Software 2012-2013
Workshop on Android app development

Session 3: Communication between app and server, QR Code Scanning & Location Service

Contents

Prepared by Mr. Felix Tam, Committee Member of the Special Interest Group (SIG) on Innovative Software 2012-2013, Department of Computer Science, Hong Kong Baptist University.

1 Detect location and manage Maps objects

1.1 Detect user current location

1. We will base on the example **com.example.googlemapsv2** that we created in session 2.

2. Add the follow markups to **AndroidManifest.xml** above <application> tag to declare the use of permission of location access.
   <uses-permission android:name="android.permission.ACCESS_FINE_LOCATION" />
   <uses-permission
   android:name="android.permission.ACCESS_COARSE_LOCATION" />

   Note:
   ACCESS_FINE_LOCATION : Access Precise location from GPS
   ACCESS_ COARSE_LOCATION : Access Approximate location from Wi-Fi and cellular.

3. Replace the contents in **MainActivity.java** with the following codes:

```
package com.example.googlemapsv2;
import com.google.android.gms.maps.CameraUpdateFactory;
import com.google.android.gms.maps.GoogleMap;
import com.google.android.gms.maps.SupportMapFragment;
import com.google.android.gms.maps.model.CameraPosition;
import com.google.android.gms.maps.model.CameraPosition.Builder;
import com.google.android.gms.maps.model.LatLng;

import android.content.Context;
import android.location.Location;
import android.location.LocationListener;
import android.location.LocationManager;
import android.os.Bundle;
import android.support.v4.app.FragmentActivity;

public class MainActivity extends FragmentActivity {
```

```
        private GoogleMap googleMap;
        private LocationManager locationManager;
        private LocationListener locationListener;
        private Builder cameraPositionBuilder;
        private CameraPosition cameraPosition;

        @Override
        protected void onCreate(Bundle savedInstanceState) {
            super.onCreate(savedInstanceState);
            setContentView(R.layout.activity_main);

            googleMap = ((SupportMapFragment)
getSupportFragmentManager().findFragmentById(R.id.map)).getMap();

            cameraPositionBuilder = new CameraPosition.Builder();
            cameraPositionBuilder.zoom(17);

            locationManager = (LocationManager)
this.getSystemService(Context.LOCATION_SERVICE);
            locationListener = new LocationListener() {
                @Override
                public void onLocationChanged(Location location) {
                    cameraPositionBuilder.target(new
LatLng(location.getLatitude(), location.getLongitude()));
                    cameraPosition = cameraPositionBuilder.build();

        googleMap.animateCamera(CameraUpdateFactory.newCameraPosition(
cameraPosition));
                }

                @Override
                public void onStatusChanged(String provider, int status, Bundle
extras) {}

                @Override
                public void onProviderEnabled(String provider) {}

                @Override
```

```
                public void onProviderDisabled(String provider) {}

            };
            locationManager.requestLocationUpdates(
  LocationManager.NETWORK_PROVIDER, 0, 0, locationListener);

  locationManager.requestLocationUpdates(LocationManager.GPS_PROVIDER, 0,
  0, locationListener);
        }
}
```

Code explanation:

- We use **((SupportMapFragment) getSupportFragmentManager().findFragmentById(R.id.map)).getMap()** to get the map from the map fragment.
- We use a **Builder** object to build a camera view which contains target, zoom, bearing and tilt information
- We use the **LocationManager** to get the latitude and longitude from the device.
- We use the **LocationListener** to listen for the change of latitude and longitude.
- When the app acquired the location, it will use the **Builder** to build the **CameraPosition**. Afterwards, we will use the **CameraUpdateFactory** to construct the **CameraUpdate** object and supply it to **animateCamera()** of **GoogleMap.**

4. Now test the app in the device and remember to activate your internet connection. You will soon see the map and the view will be moved to your current location after a while.

1.2 Manage Maps objects

You can add different objects to Google Maps including markers, polylines and polygons. In our workshop we will focus on how to add a default marker and customize it later on. We will continue to base on the example **com.example.googlemapsv2** that we just completed in section 1.1.

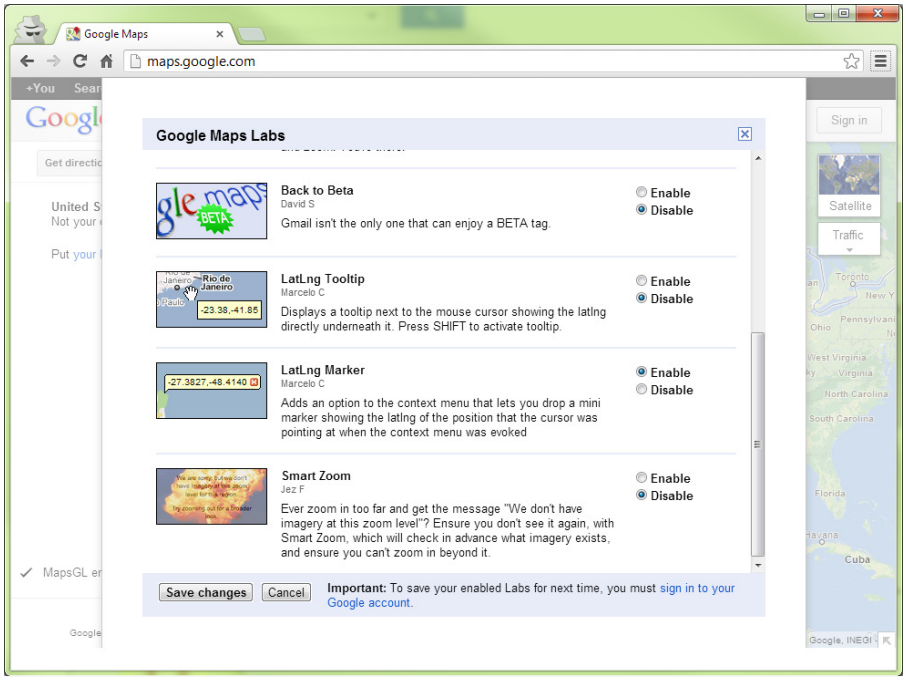
Note: Import any necessary packages using Eclipse Quick-fix feature when you are prompted for errors due to un-resolved classes.

1. Inside the **onCreate(),** add the following codes after getting the Google Maps instance.

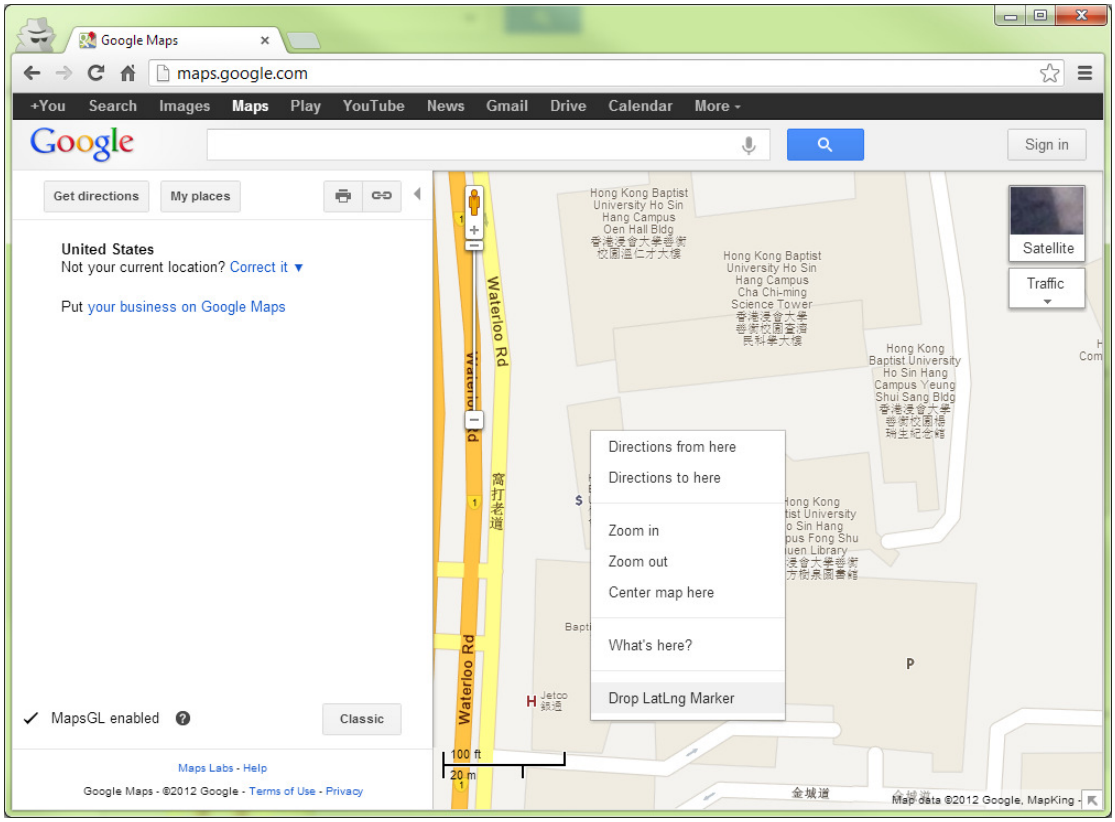
```
MarkerOptions markerOptions = new MarkerOptions();
markerOptions.position(new LatLng(22.340405, 114.179580));
markerOptions.title("RRS 638, Sir Run Run Shaw Building HKBU");
googleMap.addMarker(markerOptions);
```

2. Now run the app and you will get a default marker placed on our campus. You can click on the marker and see the title of the marker.

3. To get the latitude and longitude of a specific location, go to http://maps.google.com/ and click on **Maps Labs** at the bottom left corner. Find out **LatLng Marker** and enable it. Click **Save changes** to proceed.



4. Now you can right click on the map anywhere and choose **Drop LatLng Marker**.



5. You will get a balloon with latitude and longitude information. Copy and paste them when creating a new **LatLng** object if necessary.

6. In the following steps, we will change the default marker to our department logo

and supply a snippet to it. Download the department logo from http://www.comp.hkbu.edu.hk/~sigis/android/public/csd_logo.png

7. Copy the image to **drawable-hdpi** folder of your project and set the icon for the **markerOptions** object using the following code:

   markerOptions.icon(BitmapDescriptorFactory.fromResource(
   R.drawable.csd_logo));

8. Also, use the **snippet()** method to set the snippet of the **markerOptions** object. For example: markerOptions.snippet("Department of Computer Science");
   Note: You should call the icon() as well as snippet() before adding the marker using the addMarker() method of GoogleMap.

9. Run the app and you should get the following result.

The complete project (sample codes) is available at **googlemapsv2_s3.zip**

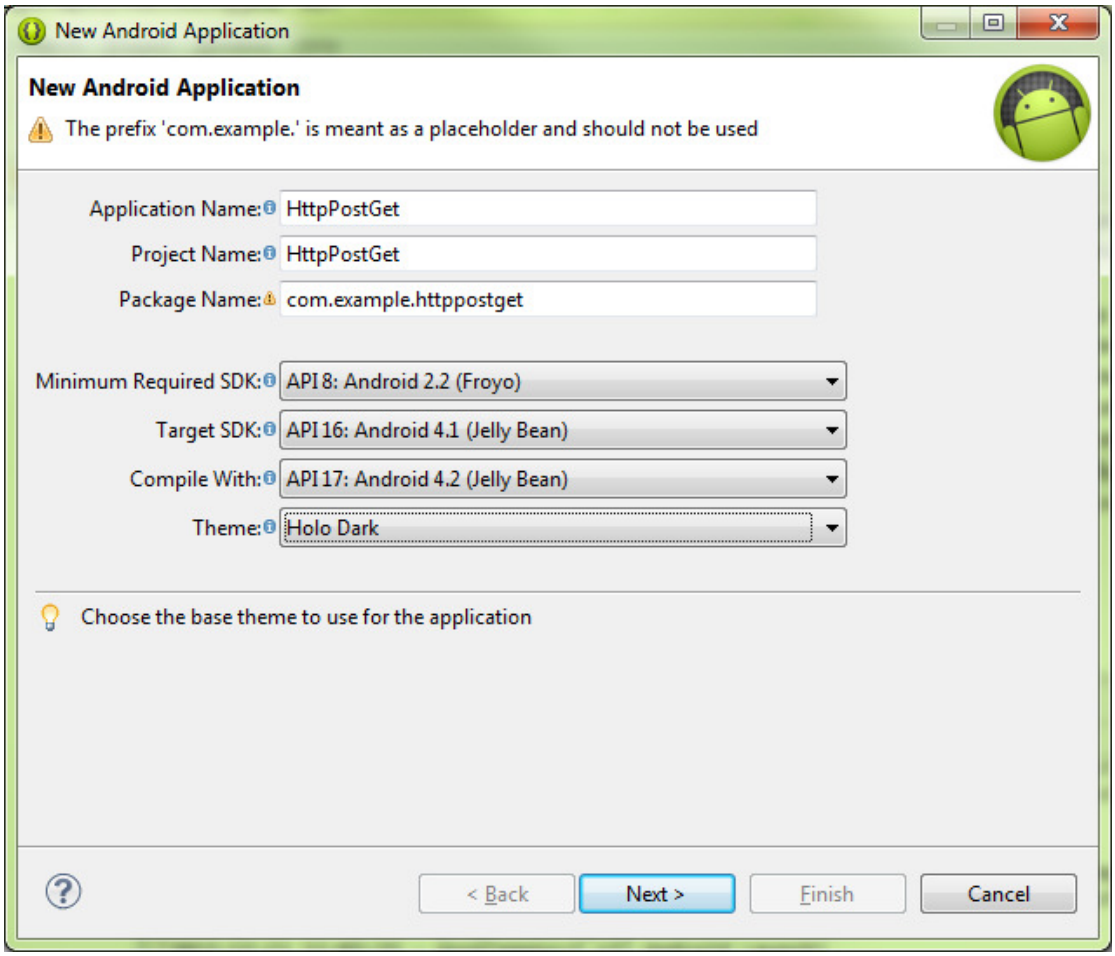
2 Communicate with server

To communicate with server using an Android device, we may make use of either HTTP POST or GET to send data and retrieve the data in XML format.

2.1 Send and receive data using HTTP POST/GET

1. In this tutorial, we have a PHP script (http://www.comp.hkbu.edu.hk/~sigis/android/public/handler.php) which accepts a string and then returns the name followed by current date and time. The variable name of the string for POST and GET are post_string and get_string respectively. The source code of the PHP script is included at the end of this section.

2. Create a new project called **HttpPostGet** with **Create activity** option checked in **Configure Project** step**.** Click next in the rest of steps and refer to the screenshot below for the information at the first step.



3. Replace the contents in **activity_main.xml** with the following markups:

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical" >
    <EditText
        android:id="@+id/userInput"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"/>
    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:orientation="horizontal" >
        <Button
            android:id="@+id/postSend"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:text="Send w/ POST"
            android:layout_weight="1"/>
        <Button
            android:id="@+id/getSend"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:text="Send w/ GET"
            android:layout_weight="1"/>
    </LinearLayout>
    <TextView
        android:id="@+id/result"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"/>
</LinearLayout>
```

4. Replace the contents in **MainActivity.java** with the following codes:

```
package com.example.httppostget;
import java.net.URLEncoder;
import java.util.ArrayList;
```

```
import org.apache.http.HttpResponse;
import org.apache.http.NameValuePair;
import org.apache.http.client.entity.UrlEncodedFormEntity;
import org.apache.http.client.methods.HttpGet;
import org.apache.http.client.methods.HttpPost;
import org.apache.http.impl.client.DefaultHttpClient;
import org.apache.http.message.BasicNameValuePair;
import org.apache.http.protocol.HTTP;
import org.apache.http.util.EntityUtils;

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.TextView;

public class MainActivity extends Activity {

    private Button sendWithPostBtn;
    private Button sendWithGetBtn;
    private TextView inputTv;
    private TextView resultTv;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        sendWithPostBtn = (Button)findViewById(R.id.postSend);
        sendWithGetBtn = (Button)findViewById(R.id.getSend);
        inputTv = (TextView)findViewById(R.id.userInput);
        resultTv = (TextView)findViewById(R.id.result);

        sendWithPostBtn.setOnClickListener(sendWithPostBtnListener);
        sendWithGetBtn.setOnClickListener(sendWithGetBtnListener);
```

```
	}

	private OnClickListener sendWithPostBtnListener = new OnClickListener(){
		@Override
		public void onClick(View arg0) {
			new SentWithAsyncTask().execute("POST");
		}
	};

	private OnClickListener sendWithGetBtnListener = new OnClickListener(){
		@Override
		public void onClick(View arg0) {
			new SentWithAsyncTask().execute("GET");
		}
	};

	private String sendWtihPost(String inputStr){
		HttpPost request = new
HttpPost("http://www.comp.hkbu.edu.hk/~sigis/android/public/handler.php");
		ArrayList<NameValuePair> params = new ArrayList<NameValuePair>();
		params.add(new BasicNameValuePair("post_string", inputStr));
		String result = "";
		try{
			request.setEntity(new UrlEncodedFormEntity(params,
HTTP.UTF_8));
			HttpResponse httpResponse = new
DefaultHttpClient().execute(request);
			if(httpResponse.getStatusLine().getStatusCode() == 200){
				result = EntityUtils.toString(httpResponse.getEntity());
			}
		}catch(Exception e){
			e.printStackTrace();
		}
		return result;
	}

	private String sendWtihGet(String inputStr){
		String result = "";
```

```
            try {
                HttpGet request = new HttpGet(
  "http://www.comp.hkbu.edu.hk/~sigis/android/public/handler.php?get_string=" +
  URLEncoder.encode(inputStr, "UTF-8"));
                HttpResponse response = new
  DefaultHttpClient().execute(request);
                result = EntityUtils.toString(response.getEntity());
            }catch(Exception e) {
                e.printStackTrace();
            }
            return result;
        }

        private class SentWithAsyncTask extends AsyncTask<String, Void, String> {
            @Override
            protected String doInBackground(String... params) {
                if(params[0].equals("POST")){
                    return sendWtihPost(inputTv.getText().toString());
                }else{
                    return sendWtihGet(inputTv.getText().toString());
                }
            }
            @Override
            protected void onPostExecute(String result) {
                super.onPostExecute(result);
                resultTv.setText(result);
            }
        }
}
```

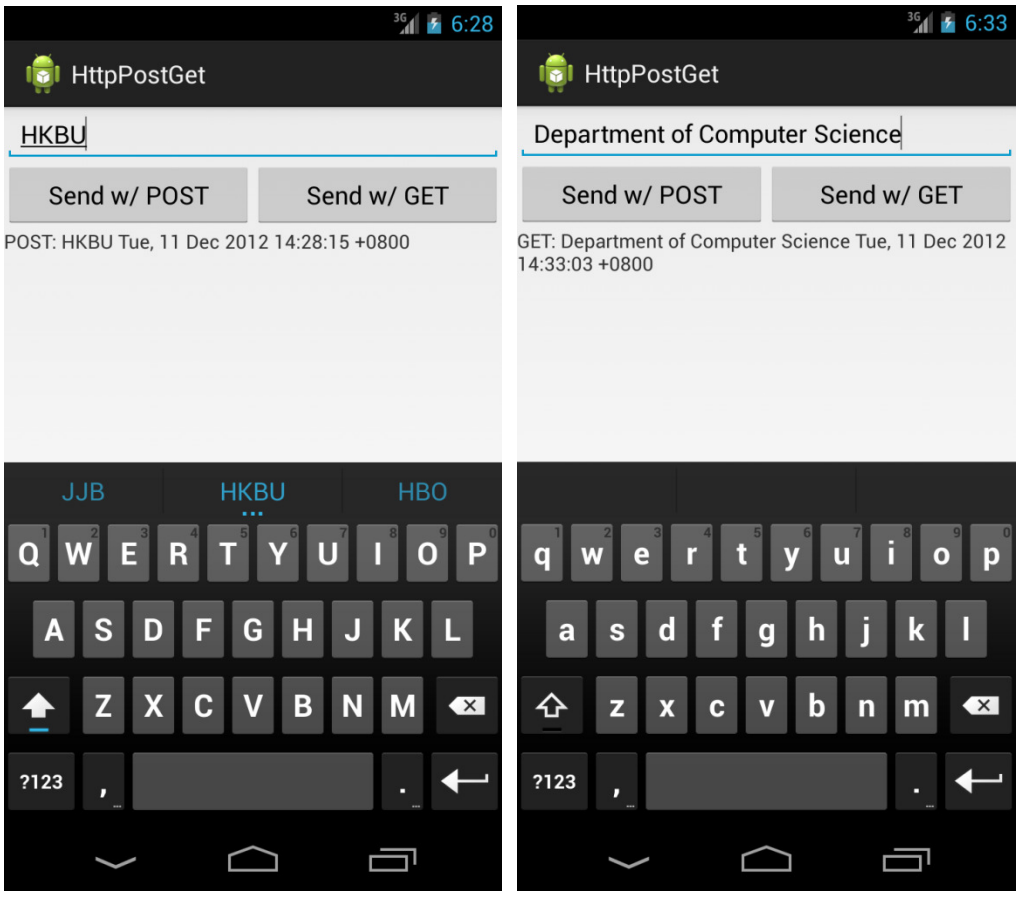
Code explanation:

1. We create two methods **sendWithPost()** and **sendWithGet()** for sending HTTP request using POST and GET respectively.
2. For **sendWithPost(),** we construct a **HttpPost** object with the target URL.
3. We create a **NameValuePair** object and put our name-value pair to an **ArrayList**.
4. We construct an **UrlEncodedFormEntity** with the **NameValuePair ArrayList** and use UTF-8 as an encoding scheme.

5. We set the **UrlEncodedFormEntity** to the **HttpPost** object using **setEntity()**.
6. We create a **DeafultHttpClient** object to execute the **HttpPost** request.
7. We create a **HttpResponse** object to receive the response from **DeafultHttpClient.**
8. When the response code equals to **HTTP 200**, we will get the result from **HttpResponse** object by using **getEntity()** method.
9. We convert the **HttpEntity** object by using the **toString()** in **EntityUtils** imported from **org.apache.http.util.EntityUtils.**
10. For **sendWtihGet(),** the flow is similar to **sendWithPost()** but this time we add the input directly to the URL as an query string.
11. We use **AsyncTask** to execute the either **sendWithPost()** or **sendWithGet()** to avoid running network operation in the main thread.
12. We add two **OnClickListeners** for the two buttons respectively in order to trigger the **AsyncTask**.

5. Remember to add **<uses-permission android:name="android.permission.INTERNET"/>** to AndroidManifest.xml

6. Run the app try to input something, press the buttons to get the result below.



7. Source code of **handler.php**:

```
<?php
    if(isset($_POST['post_string'])){
        echo "POST: " . $_POST['post_string'] . " " . date("r");
    }else if($_GET['get_string']){
        echo "GET: " . $_GET['get_string']    . " " . date("r");
    }else{
        echo "No data" . " " . date("r");
    }
?>
```

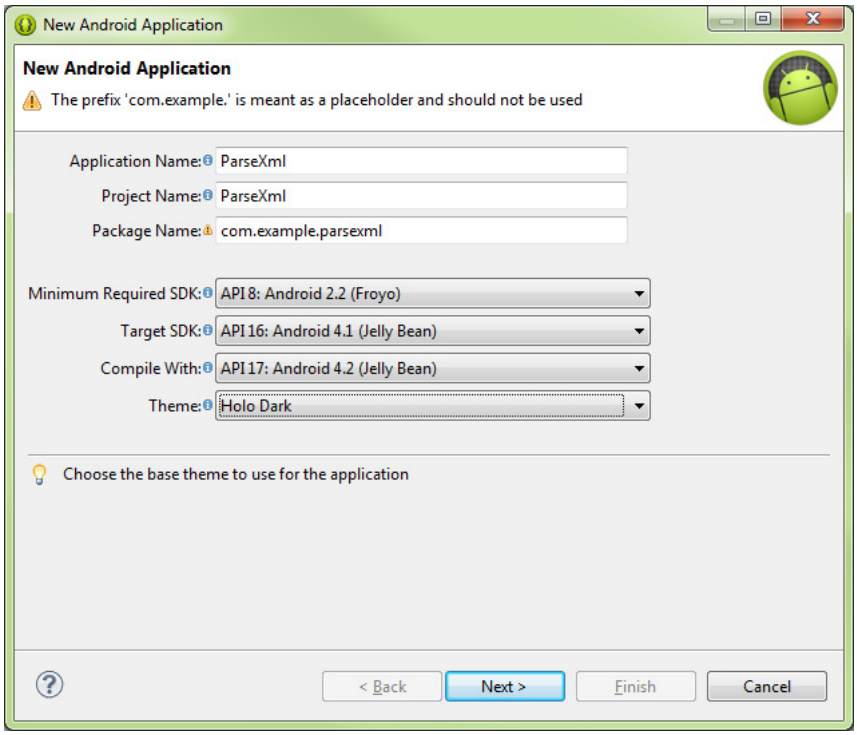
The complete project (sample codes) is available at **HttpPostGet.zip**

2.2 Retrieve data by parsing XML

The previous example is only feasible if the data structure is simple. If the data structure we retrieved is complex, it is recommended to wrap it in XML format. There are two methods to parse XML in Android: **DOM parser** and **SAX parser**. In this example, we will use DOM parser to parse our XML document located at http://www.comp.hkbu.edu.hk/~sigis/android/public/books.xml. Below is the content of our **books.xml**:

```
<?xml version="1.0" encoding="utf-8"?>
<result>
    <book>
        <name isbn="1234">Book 1</name>
        <author>Author 1</author>
        <publisher>Publisher 1</publisher>
    </book>
    <book>
        <name isbn="4567">Book 2</name>
        <author>Author 2</author>
        <publisher>Publisher 2</publisher>
    </book>
    <book>
        <name isbn="7890">Book 3</name>
        <author>Author 3</author>
        <publisher>Publisher 3</publisher>
    </book>
</result>
```

1. Create a new project called **ParseXml** with **Create activity** option checked in **Configure Project** step**.** Click next in the rest of steps and refer to the screenshot below for the information at the first step.



2. Replace the content in **activity_main.xml** with the following markups:

```
<?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" >

    <Button
        android:id="@+id/parseXML"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="Parse XML" />

    <TextView
        android:id="@+id/result"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"/>
</LinearLayout>
```

3. Replace the contents in **MainActivity.java** with the following codes:

```
package com.example.parsexml;

import java.net.URL;
import java.net.URLConnection;
import java.util.ArrayList;
import java.util.HashMap;

import javax.xml.parsers.DocumentBuilder;
import javax.xml.parsers.DocumentBuilderFactory;

import org.w3c.dom.Document;
import org.w3c.dom.Element;
import org.w3c.dom.Node;
import org.w3c.dom.NodeList;
import org.xml.sax.InputSource;

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.TextView;

public class MainActivity extends Activity {

    private Button parseXmlBtn;
    private TextView resultTv;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        parseXmlBtn = (Button)findViewById(R.id.parseXML);
        resultTv = (TextView)findViewById(R.id.result);
        parseXmlBtn.setOnClickListener(parseXmlBtnListener);
    }
```

```
    private OnClickListener parseXmlBtnListener    = new OnClickListener(){
        @Override
        public void onClick(View arg0) {
            new ParseWithAsyncTask().execute();
        }
    };

    private ArrayList<HashMap<String, String>> loadXml(){
        HashMap<String, String> map;
        ArrayList<HashMap<String, String>> bookList = new
ArrayList<HashMap<String, String>>();
        try {
            URL url = new
URL("http://www.comp.hkbu.edu.hk/~sigis/android/public/books.xml");

            URLConnection urlConn = url.openConnection();
            urlConn.setConnectTimeout(10000);
            urlConn.connect();

            DocumentBuilderFactory dbf =
DocumentBuilderFactory.newInstance();
            DocumentBuilder db = dbf.newDocumentBuilder();
            Document doc = db.parse(new InputSource(url.openStream()));
            doc.getDocumentElement().normalize();
            NodeList nodeList = doc.getElementsByTagName("book");

            for (int i = 0; i < nodeList.getLength(); i++) {
                Node node = nodeList.item(i);
                map = new HashMap<String, String>();

                Element fstElmnt = (Element) node;
                NodeList list;
                Element element;

                list = fstElmnt.getElementsByTagName("name");
                element = (Element) list.item(0);
                list = element.getChildNodes();
```

```
                        if(list.getLength()>0){
                            map.put("name", ((Node)list.item(0)).getNodeValue());
                        }else{
                            map.put("name", "");
                        }
                        map.put("isbn", element.getAttribute("isbn"));


                        list = fstElmnt.getElementsByTagName("author");
                        element = (Element) list.item(0);
                        list = element.getChildNodes();
                        if(list.getLength()>0){
                            map.put("author",
((Node)list.item(0)).getNodeValue());
                        }else{
                            map.put("author", "");
                        }

                        list = fstElmnt.getElementsByTagName("publisher");
                        element = (Element) list.item(0);
                        list = element.getChildNodes();
                        if(list.getLength()>0){
                            map.put("publisher",
((Node)list.item(0)).getNodeValue());
                        }else{
                            map.put("publisher", "");
                        }

                        bookList.add(map);
                    }
                    return bookList;
                } catch (Exception e) {
                    e.printStackTrace();
                    return new ArrayList<HashMap<String, String>>();
                }
            }

            private class ParseWithAsyncTask extends AsyncTask<Void, Void,
```

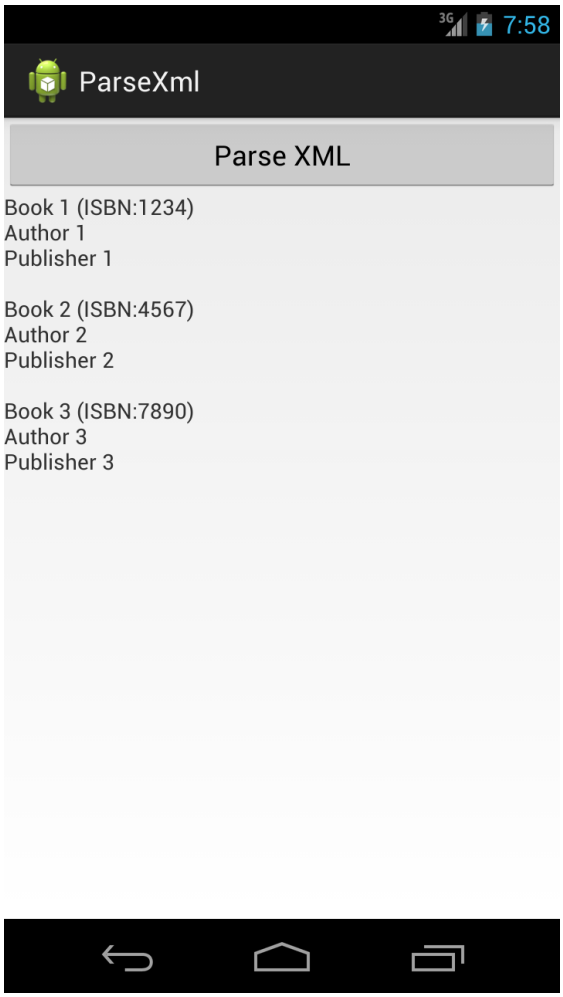
```
ArrayList<HashMap<String, String>>> {
        @Override
        protected ArrayList<HashMap<String, String>> doInBackground(Void...
params) {
            return loadXml();
        }
        @Override
        protected void onPostExecute(ArrayList<HashMap<String, String>>
result) {
            super.onPostExecute(result);
            String tvText = "";
            for(int i=0; i<result.size(); i++){
                HashMap<String, String> map = result.get(i);
                tvText += map.get("name") + " (ISBN:" + map.get("isbn") +
")" + "\n" + map.get("author") + "\n" + map.get("publisher") + "\n\n";
            }
            resultTv.setText(tvText);
        }
    }
}
```

Code explanation:

1. We create the **loadXml()** to load the XML document through Internet.
2. In the **loadXml()**, we will store the book information in an **ArrayList** of **HashMaps**. Each **HashMap** holds four keys of a book including the **name**, **isbn**, **author** and **publisher**.
3. We create a **URLConnection** object and open the connection to the URL of the XML document.
4. We use a **DocumentBuilderFactory** to create a **DocumentBuilder.**
5. We use the **DocumentBuilder** to parse the XML document and return a **Document** object.
6. We build the **NodeList** containing all books using **doc.getElementsByTagName("book")**
7. We use a for loop to loop through the whole book **NodeList**.
8. We extract the children (name, author and publisher) from the parent (book) using **getElementsByTagName("<name/author/publisher>")**
9. Since each child contains only one node (itself) and we use **((Node)list.item(0)).getNodeValue()** to get the node value.

10. Since the **isbn** is stored as an attribute of the **name** element, we use element.getAttribute("isbn") to retrieve the attribute value.
11. We store the node values and attribute of each book using **HashMap.**
12. We put all the **HashMaps** (books) into an **ArrayList**. The **loadXml()** will return the **ArrayList** after going through the whole tree.
13. We use an **AsyncTask** to run the **loadXml()** in a separate thread.
14. When the **AsyncTask** completed its task, it will loop through the **ArrayList** and put the result inside the **TextView**.

4. Remember to add **<uses-permission android:name="android.permission.INTERNET"/>** to AndroidManifest.xml
5. After pressed the button, the result should look like:



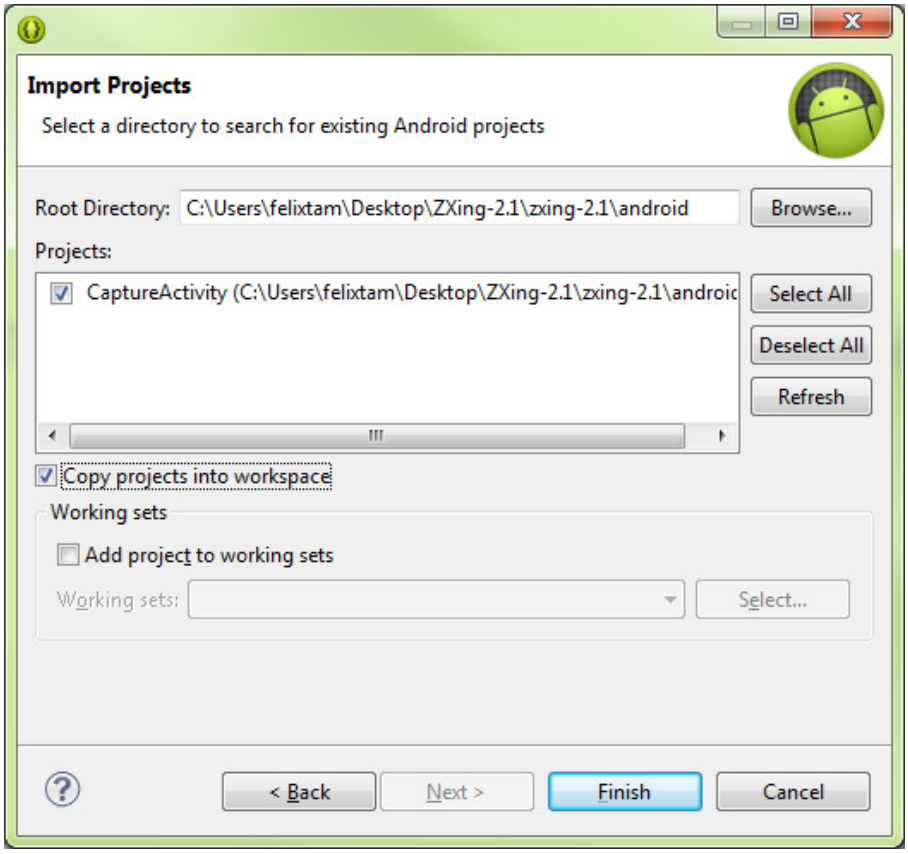
The complete project (sample codes) is available at **ParseXml.zip**

3 Scan QR code

- In Android, we can make use of an external library called ZXing ("Zebra Crossing") to scan and decode QR Codes. We can download the library here: http://zxing.googlecode.com/files/ZXing-2.1.zip

3.1 Prepare the ZXing library

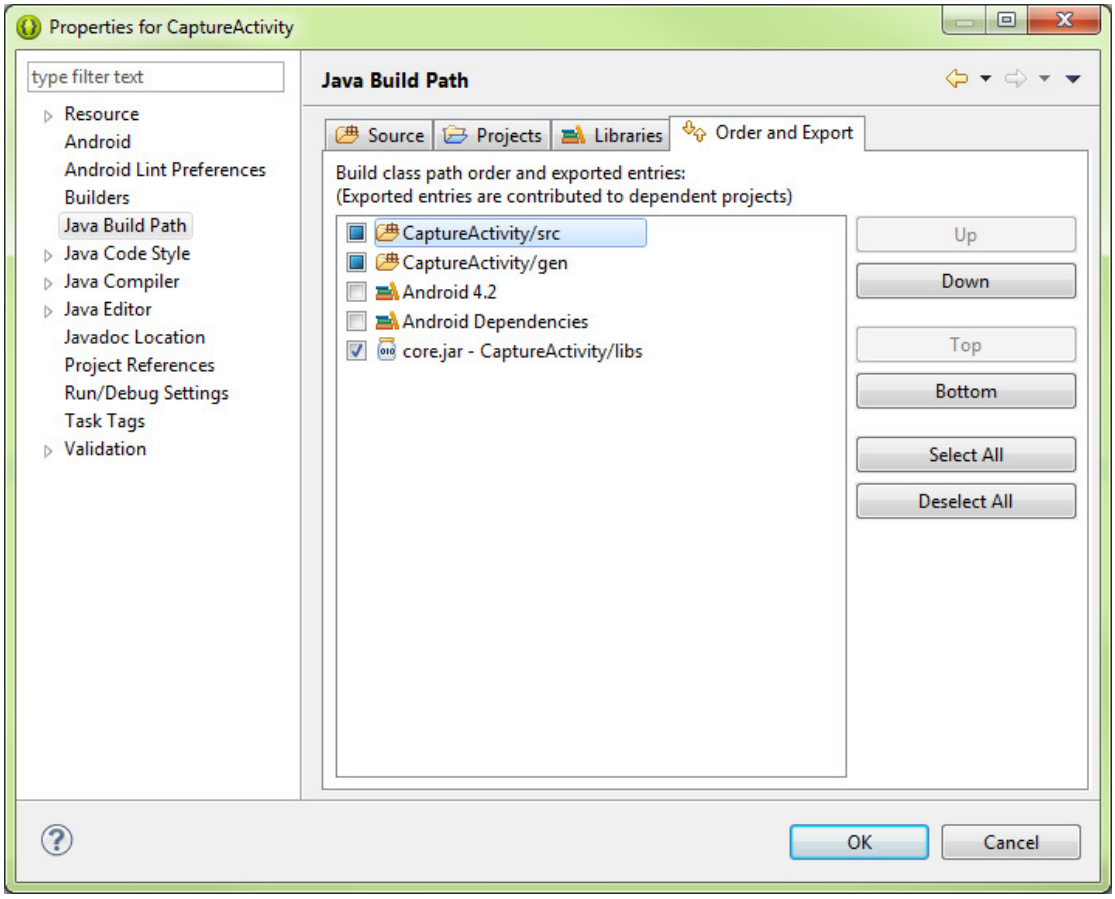
1. Extract the zip file.
2. In Eclipse, click **File > New > Other > Android > Android Project from Existing Code**.
3. Click **Browse** to locate the **android** folder and check the option **Copy projects into workspace**.



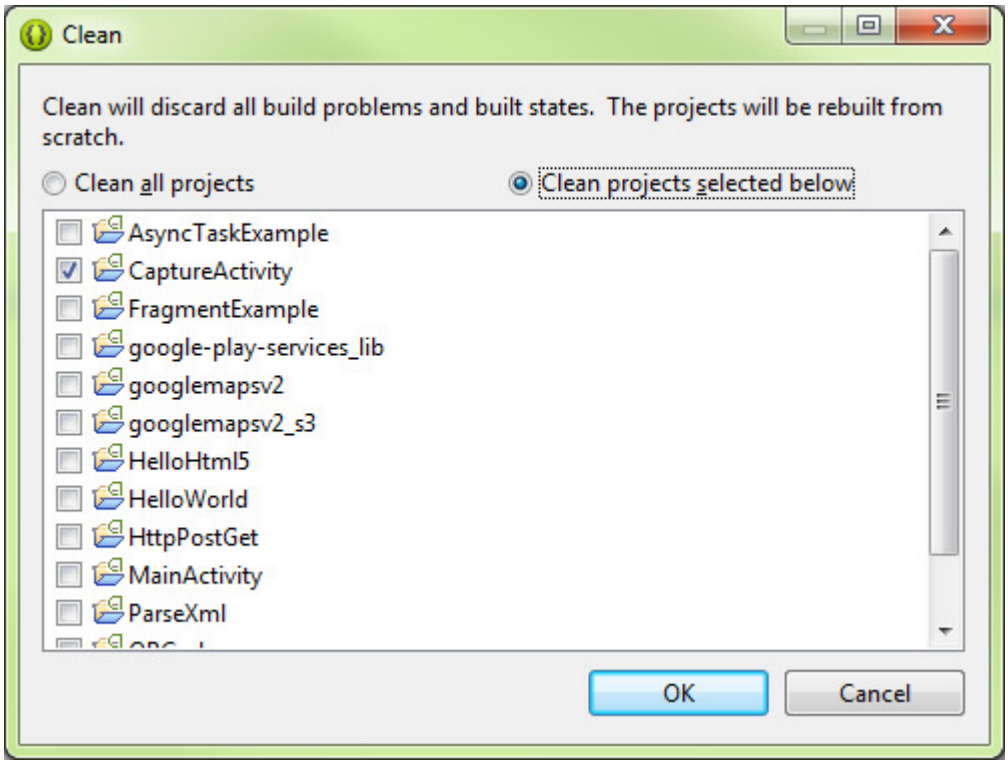
Click **Finish** to import the project.

4. Create a folder called **libs** in the root of **CaptureActivity** project. Copy the **core.jar** from the **core** folder (inside the ZXing directory) and paste it to **libs** folder.
5. In the Package Explorer, right click the core.jar and select **Build Path > Add to Build Path.**
6. Right click the **CaptureActivity** project and select **Properties > Android**, check the option **Is Library**.

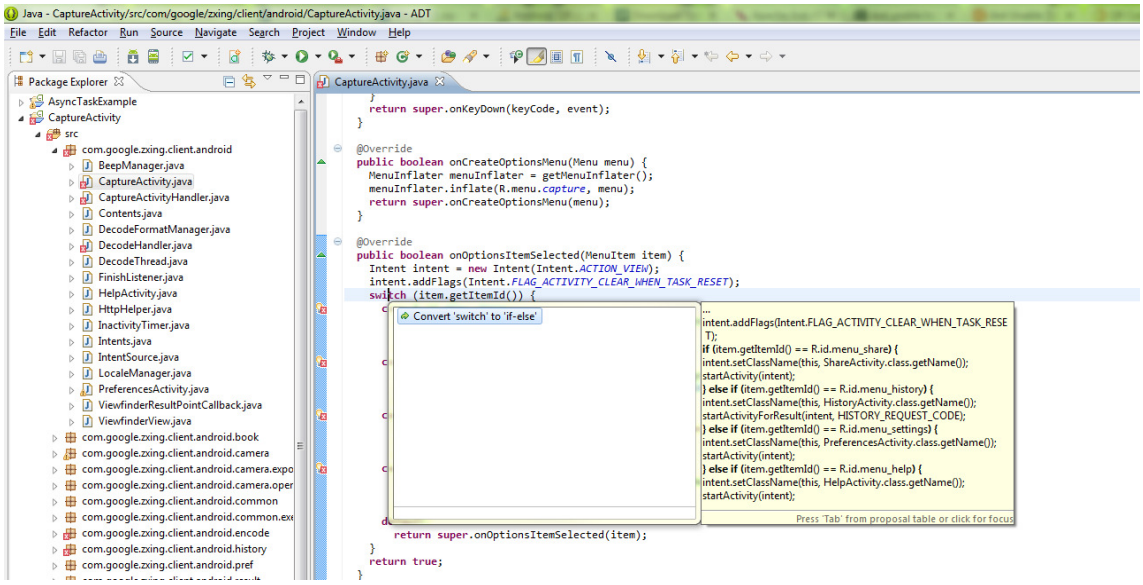
7. Go to **Java Build Path > Order and Export**, select **core.jar** and click the **OK** button.



8. Select **Project > Clean > Clean projects selected below** and choose **CaptureActivity.** You will receive errors in some java files afterwards.



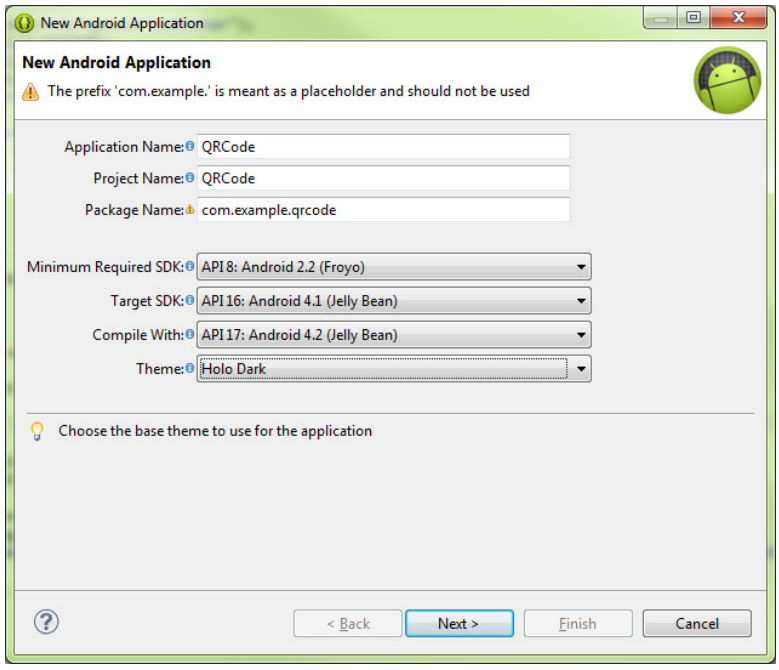
9. Open the java files which contain errors go to the line shown in Eclipse. Click on the **switch** and press Ctrl + 1. This is convert the switch statement into if statement.



10. Repeat step 9 for all similar cases. Now you have finished to configure ZXing library for your app.

3.2 Create the QR code scanning app

1. Create a new project called **QRCode** with **Create activity** option checked in **Configure Project** step**.** Click next in the rest of steps and refer to the screenshot below for the information at the first step.



2. Replace the contents in **activity_main.xml** with the following markups:

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:orientation="vertical" >
    <Button
        android:id="@+id/scanBtn"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="Scan" />
    <TextView
        android:id="@+id/scanResult"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="Scan result: " />
    <TextView
        android:id="@+id/scanResultFormat"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="Scan result format: " />
</LinearLayout>
```

3. Replace the contents in **MainActivity.java** with the following codes:

```
package com.example.qrcode;
import android.os.Bundle;
import android.app.Activity;
import android.content.Intent;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.Button;
import android.widget.TextView;

public class MainActivity extends Activity {

    private TextView scanResultTv;
    private TextView scanResultFormatTv;
    private Button scanBtn;
```

```
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        scanResultTv = (TextView)findViewById(R.id.scanResult);
        scanResultFormatTv =
 (TextView)findViewById(R.id.scanResultFormat);
        scanBtn = (Button)findViewById(R.id.scanBtn);
        scanBtn.setOnClickListener(scanBtnListener);
    }

    private OnClickListener scanBtnListener = new OnClickListener(){
        @Override
        public void onClick(View arg0) {
            Intent intent = new
 Intent("com.google.zxing.client.android.SCAN");
            intent.putExtra("SCAN_MODE", "QR_CODE_MODE");
            intent.setPackage("com.example.qrcode");
            startActivityForResult(intent, 0);
        }
    };

    @Override
    public void onActivityResult(int requestCode, int resultCode, Intent intent) {
        if (requestCode == 0) {
            if (resultCode == RESULT_OK) {
                scanResultTv.setText("Scan result: " +
 intent.getStringExtra("SCAN_RESULT"));
                scanResultFormatTv.setText("Scan result format: " +
 intent.getStringExtra("SCAN_RESULT_FORMAT"));
            }
        }
    }
}
```

Code explanation:

1. In the **scanBtnListener()**, we create an **Intent** object and specific the action to **com.google.zxing.client.android.SCAN.** We will register this action to

com.google.zxing.client.android.CaptureActivity in AndroidManifest.xml in the next step.

2. We supply additional information to the intent which sets the **SCAN_MODE** to **QR_CODE_MODE**.
3. We also limit the scope of our intent to our own package **com.example.qrcode** by using **setPackage()**. This can prevent other activities in other apps with the same intent filter action to intercept the intent message.
4. We pass the intent object and use the **startActivityForResult()** to launch ZXing activity.
5. We override the **onActivityResult()** to handle the intent sent from the ZXing activity.
6. After the QR code is scanned and recognized, it will finish itself and return the scan result to our **MainActivity**.
7. If the **resultCode** equals to **RESULT_OK,** we will display the results in our **TextViews** including the value and the data type.
8. There is a bug in the ZXing library that will cause **Force Close** during the first run. Simply go to the **CaptureActivity.java** of the **CaptureActivity** package and comment out the call of **showHelpOnFirstLaunch()**.

4. Add the following markups to **AndroidManifest.xml** (As a child of **<application>**):

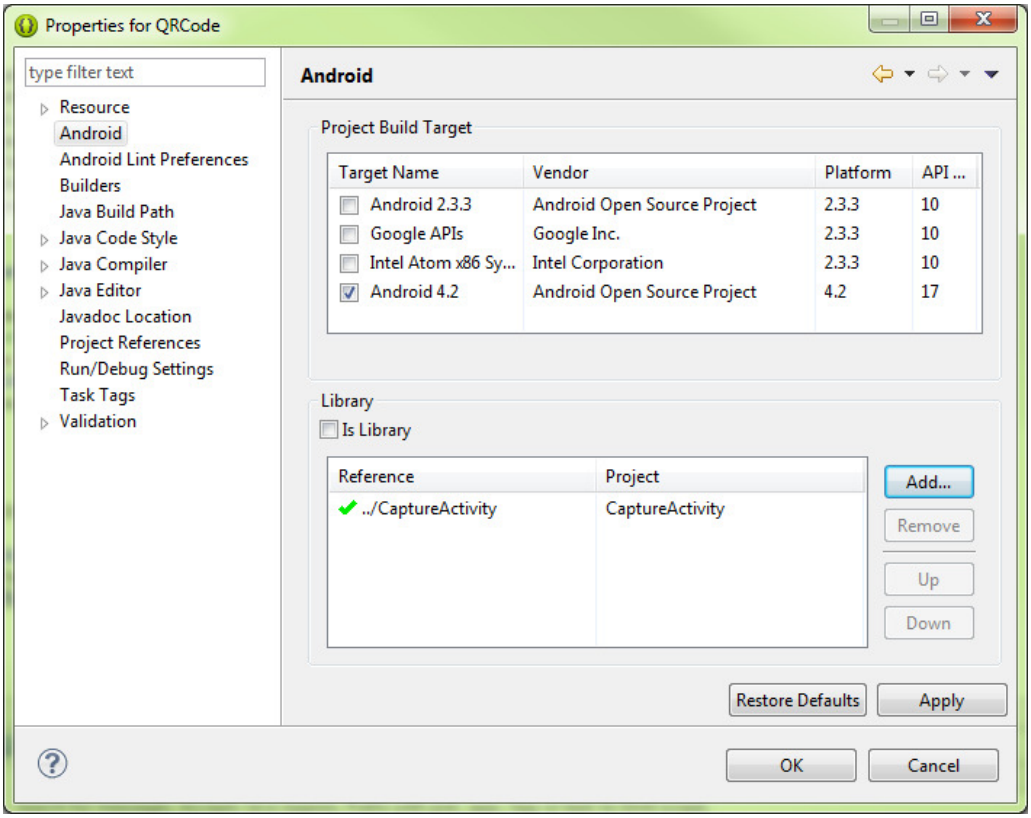
```
<activity android:name="com.google.zxing.client.android.CaptureActivity"
                android:screenOrientation="landscape"
                android:configChanges="orientation|keyboardHidden"
                android:theme="@android:style/Theme.NoTitleBar.Fullscreen"
                android:windowSoftInputMode="stateAlwaysHidden">
                <intent-filter>
                    <action android:name="android.intent.action.MAIN"/>
                    <category android:name="android.intent.category.DEFAULT"/>
                </intent-filter>
                <intent-filter>
                    <action
android:name="com.google.zxing.client.android.SCAN"/>
                    <category android:name="android.intent.category.DEFAULT"/>
                 </intent-filter>
        </activity>
```

5. Declare the use of camera permission in **AndroidManifest.xml** (Above the

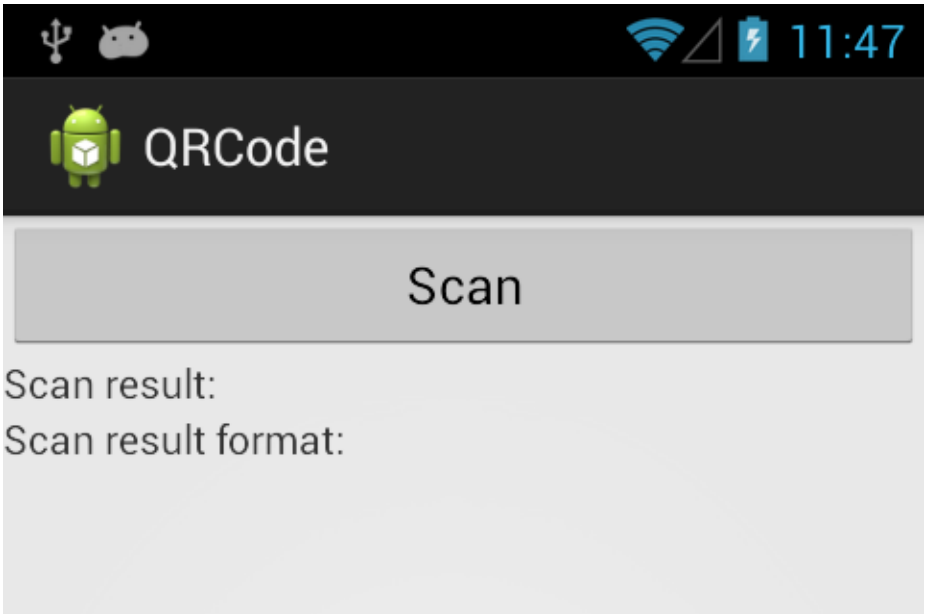
<application>) with the following markup:

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

6. Right click the QRCode project in Package Explorer, select **Properties > Android > Add > CaptureActiivty**.

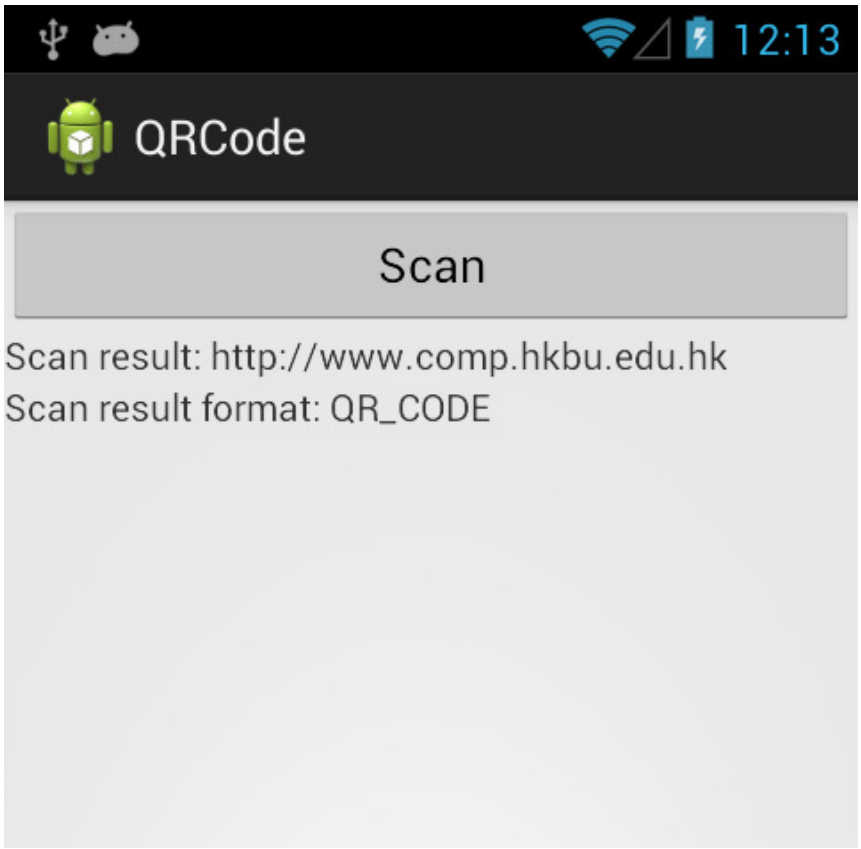


7. Run the app on a real device and try press the scan button.



8. Go to http://qrcode.kaywa.com/ and try to generate a QR code. Point your device camera to the QR code.

9. After the app recoginzed the QR code, it will automatically go back and display the result.



The complete project (sample codes) is available at **QRCode.zip**

4 Reference and learning resources

- Official Android developer website:
  http://developer.android.com/index.html
- Many common questions raised by other developers previously:
  http://stackoverflow.com/
- Many Android tutorials with complete source code:
  http://www.anddev.org/
- ZXing ("Zebra Crossing") Project:
  http://code.google.com/p/zxing/
- Special Interest Group on Innovative Software homepage:
  http://www.comp.hkbu.edu.hk/~sigis/