COMP4521 EMBEDDED SYSTEMS SOFTWARE

LAB 5: USING WEBVIEW AND USING THE NETWORK

INTRODUCTION

In this lab we modify the application we created in the first three labs. Instead of using the default browser, we will learn to create a WebView in our application to open the webpages. The second main objective is to learn to fetch .xml files over the Internet, and extract the information with local parser.

OBJECTIVES

- Learn to use Webview
- Learn to fetch files over the Internet.
- Learn to use a Progress Dialog

USING WEBVIEW

1. Locate the SDK of Android for Eclipse (if necessary). In Eclipse folder, run eclipse.bat, set your workspace to D:\Temp, click **Window**-> **Preferences**-> **Android**, and choose the SDK location (must be D:\comp4521\android-sdk-windows) as where you have put the Android SDK.
2. Create an Android Virtual Device (AVD) (if necessary).
3. Download the existing project from course website and import it into Eclipse. We will start from the application finished in Week 4.
4. Create a new layout including a webview. Create a new file **/res/layout/webwindow.xml**, and use following code as its content:

```
<?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"
    android:background="@color/ustblue"
    >
  <LinearLayout
    android:orientation="vertical"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:layout_weight="1">
```

```
    <TextView
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:text="@string/CourseCode"
    android:gravity="center_vertical|center_horizontal"
    android:textColor="@color/ustblue"
    android:background="@android:color/white"
    android:textSize="6pt"
    android:textStyle="bold"/>

    <TextView
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:text="@string/CourseTitle"
    android:gravity="center_vertical|center_horizontal"
    android:background="@android:color/white"
    android:textColor="@color/ustblue"
    android:textSize="6pt"
    android:textStyle="bold"/>

    <WebView android:id="@+id/webView"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent">
    </WebView>

</LinearLayout>

</LinearLayout>
```

Notice how a WebView is defined.

5. Create the class file for the webwindow activity. Below is the source code of the webwindow class. Read the comments carefully. The class implements a simple web browser using the methods offered by Android system.

```
public class webwindow extends Activity {
    /** Called when the activity is first created. */

 final Activity activity = this;

 WebView browser;

 @Override
   public void onCreate(Bundle savedInstanceState) {
       super.onCreate(savedInstanceState);

       // Turn the App title bar into a progress bar
       this.getWindow().requestFeature(Window.FEATURE_PROGRESS);

       setContentView(R.layout.webwindow);

       browser = (WebView) findViewById(R.id.webView);

       // get a handle to the browser settings
       WebSettings websettings= browser.getSettings();
       websettings.setSupportZoom(true);
       websettings.setJavaScriptEnabled(true);
       websettings.setBuiltInZoomControls(true);
       websettings.setDefaultZoom(WebSettings.ZoomDensity.FAR);

       // Enables the title bar to show the loading progress of the web page
       browser.setWebChromeClient(new WebChromeClient() {
          public void onProgressChanged(WebView view, int progress)
          {
             activity.setTitle("Loading...");
             activity.setProgress(progress * 100);
```

```
            if(progress == 100)
                activity.setTitle(R.string.app_name);
            }
        });

        browser.setWebViewClient(new WebViewClient() {
            @Override
            public void onReceivedError(WebView view, int errorCode, String description,
String failingUrl)
            {
                // Handle the error
            }

            // If I click on a link in the webview, the resulting page will also be loaded
into the same webview
            @Override
            public boolean shouldOverrideUrlLoading(WebView view, String url)
            {
                view.loadUrl(url);
                return true;
            }
        });

        // Get pointer to the intent and then get the extra parameter (URL) sent by the
calling activity
        Intent i = getIntent();
        String url = i.getStringExtra("URL");

        // Load the webview with the URL
        browser.loadUrl(url);
    }

  // Makes the back key to act as a go back key for the webview
  @Override
  public boolean onKeyDown(int keyCode, KeyEvent event) {
      if ((keyCode == KeyEvent.KEYCODE_BACK) && browser.canGoBack()) {
          browser.goBack();
          return true;
      }
      return super.onKeyDown(keyCode, event);
  }
}
```

Two classes cannot be imported automatically. Please import them yourself.

```
import android.webkit.WebChromeClient;
import android.webkit.WebViewClient;
```

6. Modify the action of the menu buttons in menu.java. Recall that we used a class offered by Android, the *Uri.parse* to open webpages. Since we have implemented a WebView by ourselves, now we will learn how to open a webpage inside it. First we need a new variable of type Intent in the *onItemClick* class.

```
Intent browse;
```

This intent will pass the URL from the menu activity to the webwindow activity. For example, now the event of clicking on button 2 will be like:

```
 case 2:
      browse = new Intent(menu.this, webwindow.class);
```

```
        browse.putExtra("URL","http://course.cse.ust.hk/comp4521/Description.html");
        startActivity(browse);
    break;
```

Finish the rest by yourself.

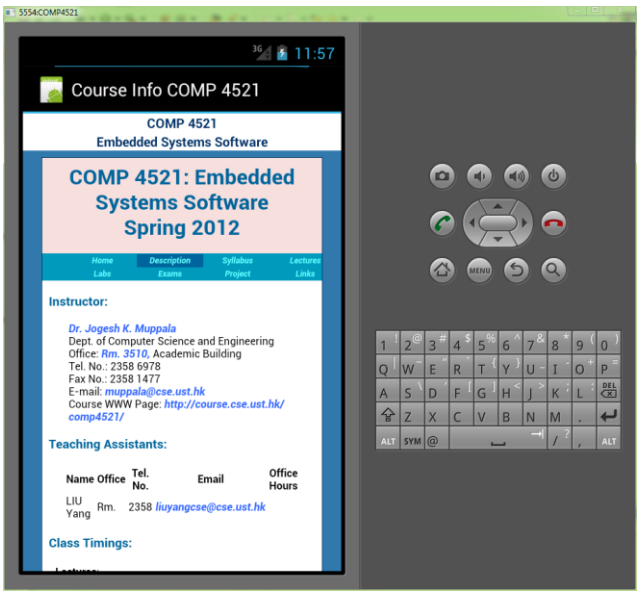
7. Modify AndroidManifest.xml. This time we not only create a declaration for the new activity; we also need to declare necessary permissions so that our application can directly access the network.

```
<activity android:name=".webwindow"
          android:label="@string/app_name">
</activity>
```

Necessary permissions. By now all of the activities we have created use the default permissions. For webviews, we need extra permission to enable the activity to access the internet. They can be placed before the application body in AndroidManifest.xml.

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

8. Run the application. Now the webpages will be opened in our own WebView. Notice that when you click on any links in the WebView or use the “Go Back” button on the emulator, we will still stay in our own application.



FETCHING AND PARSING XML FILES FROM THE INTERNET

1. Since we will learn to fetch necessary .xml files from the web, we no longer need the local .xml files, the two files under **/res/xml**.
2. Modify the two table layouts. For **contacts.xml**, we will use a single TableLayout instead of two Table Layouts for instructor and TA. The tablelayout will be the only component in the scrollview.

```
<TableLayout
android:id="@+id/contactsTable"
android:layout_width="fill_parent"
android:layout_height="fill_parent"
android:stretchColumns="*">
</TableLayout>
```

   We also simplify the ScrollView in timetable.xml to:

```
<ScrollView
android:id="@+id/ScrollViewTimeTable"
android:layout_width="fill_parent"
android:layout_height="fill_parent"
android:scrollbars="vertical">

<TableLayout
android:id="@+id/timeTable"
android:layout_width="fill_parent"
android:layout_height="fill_parent"
android:stretchColumns="*">
</TableLayout>

</ScrollView>
```

3. Modify the source code of contacts.java. Our objective is to fetch the XML files from the Internet instead of reading local files. To avoid the device being stuck when trying to fetch a file from the web, we create a subclass of AsyncTask named ContactsTask, which will do the fetching and parsing job in the background. We need to declare two variables in contacts.java. ProgressDialog is a class offered by Android showing messages of an ongoing progress.

```
ContactsTask contactprocessing;

ProgressDialog waitDialog;
```

   Since the real work is done by the background thread, the onCreate() method can be very simple:

```
super.onCreate(savedInstanceState);
setContentView(R.layout.contacts);
contactprocessing = new ContactsTask();
contactprocessing.execute();
```

4. Now we define the ContactsTask class. It's an extension of AsyncTask:

```
private class ContactsTask extends AsyncTask<Object, String, Boolean>
```

Two variables are required: one points to the TableLayout showing the information, one includes the information got from the .xml.

```
TableLayout contactsTable;
XmlPullParser contactinfo;
```

ContactsTask includes 6 methods:

onPreExecute() executes before the background starts, showing the progress dialog;

onProgressUpdate() inserts a new row to the TableLayout as soon as a piece of information is extracted from the .xml;

onPostExecute() passes the result from the background thread, and remove the progress dialog from the screen;

doInBackground() fetches the XML from the web, and calls processcontacts() to parse the file;

processcontacts() parses the XML file. It is almost the same as what we have used in Week 4, except that it uses publishProgress() to update the main UI because now we are doing the parsing in a separate thread.

insertContactRow() inserts a new row in the TableLayout when a piece of message is successfully extracted.

```
        @Override
        protected void onPreExecute() {
             // TODO Auto-generated method stub
             super.onPreExecute();

             waitDialog = ProgressDialog.show(contacts.this, "Contacts", "Loading
...");
        }

        @Override
        protected void onPostExecute(Boolean result) {
             // TODO Auto-generated method stub
             super.onPostExecute(result);

             waitDialog.dismiss();
        }



        @Override
        protected void onProgressUpdate(String... values) {
             // TODO Auto-generated method stub
             super.onProgressUpdate(values);

             if (values.length == 1){
                  String strvalue = values[0];

                  insertContactRow(contactsTable, strvalue);

             }
             else{
                  final TableRow newRow = new TableRow(contacts.this);
                  TextView noResults = new TextView(contacts.this);
                  newRow.addView(noResults);
```

```
                contactsTable.addView(newRow);
            }
        }


        @Override
        protected Boolean doInBackground(Object... params) {

          //Get pointers to the table layout in the contacts.xml file
          contactsTable = (TableLayout) findViewById(R.id.contactsTable);

          // Open a XML resource parser to parse the contactinfo.xml file
          // XmlResourceParser contactinfo = getResources().getXml(R.xml.contactinfo);

          try{
              URL xmlUrl = new
URL("http://course.cse.ust.hk/comp4521/xml/contactinfo.xml");
              contactinfo = XmlPullParserFactory.newInstance().newPullParser();
              contactinfo.setInput(xmlUrl.openStream(), null);
          }catch (XmlPullParserException e){
              contactinfo = null;
          }catch (IOException e){
             contactinfo = null;
          }

          // Now construct the information for the instructor and TA from the parsed XML
file
          try {
              // process the contacts xml file to set up the information on the activity
screen
              processcontacts(contactsTable, contactinfo);
          } catch (Exception e) {
              Log.e(DEBUG_TAG, "Failed to load Contacts", e);
          }
              return null;
        }



        /**
         * Churn through an XML score information and populate a {@code TableLayout}
         *
         * @param contactsTable
         *          The {@code TableLayout} to populate
         *
         * @param contact
         *          A standard {@code XmlResourceParser} containing the scores
         * @throws XmlPullParserException
         *           Thrown on XML errors
         * @throws IOException
         *           Thrown on IO errors reading the XML
         */
        private void processcontacts(final TableLayout contactsTable,
                  XmlPullParser contact) throws XmlPullParserException,
                  IOException {
              int eventType = -1;
              boolean bFoundContacts = false;
              // Find records from XML
              while (eventType != XmlResourceParser.END_DOCUMENT) {
                   if (eventType == XmlResourceParser.START_TAG) {
                        // Get the name of the tag (eg contact, instructor or assistant)
                        String strName = contact.getName();
                        if (strName.equals("instructor")) {
                             bFoundContacts = true;
                             publishProgress("Instructor");
                             String name = contact.getAttributeValue(null, "name");
                             publishProgress("     " + name);
                             String office = contact.getAttributeValue(null, "office");
```

```
                              publishProgress("    Office: " + office);
                              String tel = contact.getAttributeValue(null, "tel");
                              publishProgress("    Tel: " + tel);
                              String email = contact.getAttributeValue(null, "email");
                              publishProgress("    Email: " + email);
                              String web = contact.getAttributeValue(null, "web");
                              publishProgress("    Web: " + web);
                              publishProgress("    ");

                         }
                         if (strName.equals("assistant")) {
                              bFoundContacts = true;
                              publishProgress("Teaching Assistant");
                              String name = contact.getAttributeValue(null, "name");
                              publishProgress("    " + name);
                              String office = contact.getAttributeValue(null, "office");
                              publishProgress("    Office: " + office);
                              String tel = contact.getAttributeValue(null, "tel");
                              publishProgress("    Tel: " + tel);
                              String email = contact.getAttributeValue(null, "email");
                              publishProgress("    Email: " + email);
                              publishProgress("    ");
                         }
                    }
                    eventType = contact.next();
               }
               // Handle no records available
               if (bFoundContacts == false) {
                    publishProgress();
               }
          }

          /**
           * {@code insertContactRow()} helper method -- Inserts a new contact information
row {@code
           * TableRow} in the {@code TableLayout}
           *
           * @param contactTable
           *          The {@code TableLayout} to add the contact information to
           * @param strValue
           *          The value of text string to be inserted into the row
           * @param mask
           *          specifies what regex I need to look for in the string in order to
Linkify it. mask <= 0 implies no need to Linkify.
           */
          private void insertContactRow(final TableLayout contactTable, String strValue)
{
               // create a new table row and populate it
               final TableRow newRow = new TableRow(contacts.this);
               TextView textView = new TextView(contacts.this);
               textView.setText(strValue);
               Linkify.addLinks(textView, Linkify.ALL);
               if (strValue == "Instructor" || strValue == "Teaching Assistant"){
                    textView.setTextSize(18);
                    textView.setTextColor(getResources().getColor(R.color.ustgold));
               }
               newRow.addView(textView);
               contactTable.addView(newRow);
          }
```

5. Modify the source code of time_table.java. Similarly, we need two new variables:

```
TimeTableTask timetableprocessing;

ProgressDialog waitDialog;
```

Please modify the onCreate() method, using the onCreate() in contacts.java as the example.

The TimeTableTask class is as follows. It's very similar to ContactsTask.

```
private class TimeTableTask extends AsyncTask<Object, String, Boolean> {
  TableLayout timeTable;
    XmlPullParser timetableinfo;

  @Override
       protected Boolean doInBackground(Object... params) {

       //Get pointers to the two table layouts in the contacts.xml file
       timeTable = (TableLayout) findViewById(R.id.timeTable);

       try{
            URL xmlUrl = new
URL("http://course.cse.ust.hk/comp4521/xml/time_table.xml");
            timetableinfo = XmlPullParserFactory.newInstance().newPullParser();
           timetableinfo.setInput(xmlUrl.openStream(), null);
       }catch (XmlPullParserException e){
            timetableinfo = null;
       }catch (IOException e){
         timetableinfo = null;
       }


       try {
          processtimetable(timeTable, timetableinfo);
       } catch (Exception e) {
          Log.e(DEBUG_TAG, "Failed to load Time Table", e);
       }

       // TODO Auto-generated method stub
            return null;
       }

       @Override
       protected void onPostExecute(Boolean result) {
            // TODO Auto-generated method stub
            super.onPostExecute(result);
            waitDialog.dismiss();
       }

       @Override
       protected void onPreExecute() {
            // TODO Auto-generated method stub
            super.onPreExecute();
            waitDialog = ProgressDialog.show(time_table.this, "Time Table", "Loading
...");
       }

       @Override
       protected void onProgressUpdate(String... values) {
            // TODO Auto-generated method stub
            super.onProgressUpdate(values);

            if (values.length == 1){
                 String strvalue = values[0];

                 insertTimeTableRow(timeTable, strvalue);

            }
            else{
                 final TableRow newRow = new TableRow(time_table.this);
                 TextView noResults = new TextView(time_table.this);
                 newRow.addView(noResults);
                 timeTable.addView(newRow);
```

```
            }
        }

        /**
     * Churn through an XML score information and populate a {@code TableLayout}
     *
     * @param lectureTable
     *          The {@code TableLayout} to populate
     * @param timetable
     *          A standard {@code XmlResourceParser} containing the scores
     * @throws XmlPullParserException
     *           Thrown on XML errors
     * @throws IOException
     *           Thrown on IO errors reading the XML
     */
    private void processtimetable(final TableLayout timeTable,
              XmlPullParser timetable) throws XmlPullParserException,
              IOException {
         int eventType = -1;
         boolean bFoundTimeTable = false;
         // Find records from XML
         while (eventType != XmlResourceParser.END_DOCUMENT) {
              if (eventType == XmlResourceParser.START_TAG) {
                   // Get the name of the tag (eg timetable, lecture or lab)
                   String strName = timetable.getName();
                   if (strName.equals("lecture")) {
                        bFoundTimeTable = true;
                        publishProgress("Lectures");
                        String days = timetable.getAttributeValue(null, "days");
                        publishProgress("    " + days);
                        String time = timetable.getAttributeValue(null, "time");
                        publishProgress("    Time: " + time);
                        String room = timetable.getAttributeValue(null, "room");
                        publishProgress("    Room: " + room);
                        publishProgress("    ");

                   }
                   if (strName.equals("lab")) {
                        bFoundTimeTable = true;
                        publishProgress("Lab Sessions");
                        String days = timetable.getAttributeValue(null, "days");
                        publishProgress("    " + days);
                        String time = timetable.getAttributeValue(null, "time");
                        publishProgress("    Time: " + time);
                        String room = timetable.getAttributeValue(null, "room");
                        publishProgress("    Room: " + room);
                        publishProgress("    ");
                   }
              }
              eventType = timetable.next();
         }
         // Handle no records available
         if (bFoundTimeTable == false) {
              final TableRow newRow = new TableRow(time_table.this);
              TextView noResults = new TextView(time_table.this);
              newRow.addView(noResults);
              timeTable.addView(newRow);
         }
    }

    /**
     * {@code insertTimeTableRow()} helper method -- Inserts a new time table row {@code
     * TableRow} in the {@code TableLayout}
     *
     * @param timeTable
     *          The {@code TableLayout} to add the time table information to
     * @param strValue
     *          The value of the  text string to be inserted into the row
     */
```

```
private void insertTimeTableRow(final TableLayout timeTable, String strValue) {
      final TableRow newRow = new TableRow(time_table.this);
      TextView textView = new TextView(time_table.this);
      textView.setText(strValue);
      if (strValue == "Lectures" || strValue == "Lab Sessions"){
            textView.setTextSize(18);
            textView.setTextColor(getResources().getColor(R.color.ustgold));

      }
      newRow.addView(textView);
      timeTable.addView(newRow);
}
   }
```

6. Run the application. Since the XML files fetched from the web server is the same as the local files we used, we will get the same results. However, when updating the screen, a progress dialog will be shown on the screen: