Advanced Mobile Application Development
Week 14: Android and JSON

Network Connections

https://developer.android.com/reference/android/os/AsyncTask.html

Each Android application runs in its own process in a single thread which is called the **Main thread** or **UI thread.** Any operation which will block our main thread should be implemented in a separate thread. For example, Networking and receiving data from the Internet is a blocking operation. Prior to Ice Cream Sandwich, Android apps could open a network connection, and request network resources from the main UI thread. This led to such situations where the application would be completely unresponsive while waiting for a response. Now there are stricter rules regarding performing potentially expensive operations on the UI thread and network operations must be performed on another thread.

The **AsyncTask** class enables you to implement a background thread to perform operations asynchronously and not hold up the UI thread. The results of the background thread can then be published on the UI thread without having to manipulate threads and/or handlers. AsyncTasks should ideally be used for short operations (a few seconds at the most).

AsyncTask's generic types

To use the AsyncTask class you must subclass it and provide the three types used by an asynchronous task.

1. **Params**, the type of the parameters sent to the task
2. **Progress**, the progress type published during the task
3. **Result**, the type of the result returned on completion of the task.

Not all types are always used by an asynchronous task. To mark a type as unused, simply use the type Void.

AsyncTask's 4 steps

The AsyncTask subclass is automatically loaded on the UI thread and the task instance must be created on the UI thread.

When an asynchronous task is executed, the task goes through 4 steps:

1. **onPreExecute(),** invoked on the UI thread before the task is executed. This step is normally used to setup the task, for instance by showing a progress bar in the user interface.
2. **doInBackground(Params...),** invoked on the background thread immediately after onPreExecute() finishes executing. This step is used to perform background computation that can take a long time. The parameters of the asynchronous task are passed to this step. The result of the computation must be returned by this step and will be passed back to the last step.
3. **onProgressUpdate(Progress...),** invoked on the UI thread after a call to **publishProgress(Progress...).** The timing of the execution is undefined. This method is used to display any form of progress in the user interface while the background computation is still executing. For instance, it can be used to animate a progress bar or show logs in a text field.
4. **onPostExecute(Result),** invoked on the UI thread after the background computation finishes. The result of the background computation is passed to this step as a parameter.

Network Request
Android and Java have classes that enable you to download JSON files and then parse the data.

The **HttpURLConnection** class supports sending and receiving data over HTTP. We will use the openConnection() method to make a connection to a given URL.

The **InputStream** class gives us access to any stream of data. A stream is basically any set of characters. We will use the getInputStream() method to get the input stream for our JSON file. InputStream works only on bytes so we use the **InputStreamReader** class to convert a byte stream into a character stream which the **BufferedReader** class works with. The readLine() method reads one line at a time from the BufferedReader character stream. We can then convert the character stream to a String using the **StringBuilder** class which creates a mutable sequence of characters. We create a String from this sequence using the toString() method.

JSON
To parse the JSON content, you can use JSON standard API provided by Android SDK. All the APIs are based on 2 classes, both subclasses of **JSONRequest**:
JSONObject
- getJSONObject() returns a JSON object
    - { } tell you it's a JSON object
JSONArray
- getJSONArray() returns an array, usually in an object
    - [ ] tells you it's an array
The **JSONException** class handles all the exceptions related to JSON parsing.

For more complex parsing there are libraries such as Volley and Retrofit.

Social
We're going to use the Full Contact API to search email addresses and see their social networks.
Go sign up and get an API key.
https://www.fullcontact.com/developer/

Look at the JSON to figure out what data we want to use.
https://api.fullcontact.com/v2/person.json?email=aileen_pierce@yahoo.com&apiKey=8ea9fa44fe430665
Create a new Android project called Social with a minimum API of 21 using the Empty Activity template.

Our application requires internet access, so we must request the INTERNET permission in the AndroidManifest.xml file. Add the following before the application tag.

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

Layout
Our layout will have an EditText for the email address to search with **android:id="@+id/emailText"**
A button to initiate the search by sending the API query with **android:id="@+id/queryButton"**
Beneath these, there's a ProgressBar that's initially set to be invisible, becomes visible when the "Search" Button is clicked, and disappears again when the data is about to be displayed.

```
android:id="@+id/progressBar"
style="?android:attr/progressBarStyle"
android:indeterminate="true"
android:visibility="gone"
```

Then an ImageView for the person's photo android:id="@+id/imageView"
A TextView for their name android:id="@+id/nameView"

Then another TextView for the list of social networks android:id="@+id/responseView"

I added the following strings to strings.xml
```
<string name="app_name">Social</string>
<string name="email_hint">Enter email address</string>
<string name="button_title">Search</string>
```

MainActivity
Add the following to your class:
```
private EditText emailText;
private TextView responseView;
private TextView nameView;
private ProgressBar progressBar;
private ImageView photoImageView;
private static final String API_KEY = "8ea9fa44fe430665";
private static final String API_URL = "https://api.fullcontact.com/v2/person.json?";
```

Update onCreateView() to get access to the views we need and set up a listener for the click event on the search button.
```
responseView = findViewById(R.id.responseView);
nameView = findViewById(R.id.nameView);
emailText = findViewById(R.id.emailText);
progressBar = findViewById(R.id.progressBar);
photoImageView = findViewById(R.id.imageView);

Button queryButton = findViewById(R.id.queryButton);
queryButton.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        String email = emailText.getText().toString();
        new RetrieveFeedTask().execute(email);
    }
});
```

You will have an error on RetrieveFeedTask because we haven't created that yet.
RetrieveFeedTask is our class that is going to subclass AsyncTask which allows you to perform background operations and publish results on the UI thread without having to manipulate threads and/or handlers.

RetrieveFeedTask extends AsyncTask<String, Void, String>. We will implement 3 of its methods.

In onPreExecute(), we set the ProgressBar to become visible, and we clear the contents of the TextView. This is done before the task begins running.

doInBackground() expects Params, which is Void in this particular case. We then set up and open a HttpURLConnection to make an API request. The URL is built from the supplied email, and our API_KEY. Then we read the complete response string, using a BufferedReader and a StringBuilder.

onPostExecute() hides the ProgressBar once more, and displays the fetched response in the layout.

```java
class RetrieveFeedTask extends AsyncTask<String, Void, String> {

    private Exception exception;

    //invoked on the UI thread before the task is executed
    protected void onPreExecute() {
        progressBar.setVisibility(View.VISIBLE);
        responseView.setText("");
        nameView.setText("");
        photoImageView.setImageResource(android.R.color.transparent);
    }

    //invoked on the background thread immediately after onPreExecute() finishes executing
    protected String doInBackground(String... args) {
        String email = args[0];
        try {
            URL url = new URL(API_URL + "email=" + email + "&apiKey=" + API_KEY);
            HttpURLConnection urlConnection = (HttpURLConnection) url.openConnection();
            try {
                BufferedReader bufferedReader = new BufferedReader(new
InputStreamReader(urlConnection.getInputStream()));
                StringBuilder stringBuilder = new StringBuilder();
                String line;
                while ((line = bufferedReader.readLine()) != null) {
                    stringBuilder.append(line).append("\n");
                }
                bufferedReader.close();
                return stringBuilder.toString();
            } finally {
                urlConnection.disconnect();
            }
        } catch (Exception e) {
            Log.e("ERROR", e.getMessage(), e);
            return null;
        }
    }
```

The syntax String... args is similar to String[] args but the difference is how you can call it. Along with calling the function and passing an array it can also be called with a list of strings (multiple strings separated by comma) or with no arguments at all.

```java
//invoked on the UI thread after the background computation finishes
//response is the result of doInBackground
protected void onPostExecute(String response) {
    if (response == null) {
        response = "THERE WAS AN ERROR";
    }
    progressBar.setVisibility(View.GONE);

    //parse JSON object
    try {
        JSONObject object = (JSONObject) new JSONTokener(response).nextValue();
        JSONObject contact = object.getJSONObject("contactInfo");
        String name = contact.getString("fullName");
        nameView.setText(name);

        JSONArray photoarray = object.getJSONArray("photos");
        JSONObject photos = photoarray.getJSONObject(0);
        String photo = photos.getString("url");
        new DownloadImageTask(photoImageView).execute(photo);

        JSONArray socialprofiles = object.getJSONArray("socialProfiles");
        for (int i = 0; i < socialprofiles.length(); i++) {
            JSONObject socialprofile = socialprofiles.getJSONObject(i);
            String social = socialprofile.getString("type");
            String url = socialprofile.getString("url");
            responseView.append(social + " \t" + url + "\n");
        }
    } catch (JSONException e) {
        e.printStackTrace();
    }
}
```

(DownloadImageTask will give you an error at this point.)

Downloading an image also should be done in another thread so we create another subclass of AsyncTask for that.

```java
private class DownloadImageTask extends AsyncTask<String, Void, Bitmap> {
    ImageView bmImage;

    public DownloadImageTask(ImageView bmImage) {
        this.bmImage = bmImage;
    }
    protected Bitmap doInBackground(String... urls) {
        String urldisplay = urls[0];
        Bitmap mIcon = null;
        try {
            InputStream in = new java.net.URL(urldisplay).openStream();
```

```
      mIcon = BitmapFactory.decodeStream(in);
   } catch (Exception e) {
      e.printStackTrace();
   }
   return mIcon;
}


protected void onPostExecute(Bitmap result) {
   bmImage.setImageBitmap(result);
}
}
```

We use the BitmapFactory class to decode the input stream and create a Bitmap that we can use to set the image in our ImageView.

Your app should work at this point.

Volley
https://developer.android.com/training/volley
Volley is an HTTP library that makes networking for Android apps easier and faster.
Add the following dependency to your app's build.gradle file:
implementation **'com.android.volley:volley:1.1.1'**

In MainActivity add a RequestQueue.
private RequestQueue **queue**;

Use Volley for the networking and the JSON parsing remains the same.
```
private void requestData(String email){
   progressBar.setVisibility(View.VISIBLE);
   responseView.setText("");
   nameView.setText("");
   photoImageView.setImageResource(android.R.color.transparent);

   queue = Volley.newRequestQueue(this);
   //create URL for request
   String url = API_URL + "email=" + email + "&apiKey=" + API_KEY;

   // Request a string response from the provided URL.
   StringRequest stringRequest = new StringRequest(Request.Method.GET, url,
         new Response.Listener<String>() {
            @Override
            public void onResponse(String response) {
               parseJSON(response);
            }
         }, new Response.ErrorListener() {
      @Override
      public void onErrorResponse(VolleyError error) {
         Log.e("ERROR", error.getMessage(), error);
      }
```

```java
        });

        // Add the request to the RequestQueue
        queue.add(stringRequest);
}

private void parseJSON(String response){
    //response should be a String with JSON objects
    if (response == null) {
        response = "THERE WAS AN ERROR";
    }
    progressBar.setVisibility(View.GONE);

    //parse JSON object
    try {
        JSONObject object = (JSONObject) new JSONTokener(response).nextValue();
        JSONObject contact = object.getJSONObject("contactInfo");
        String name = contact.getString("fullName");
        nameView.setText(name);

        JSONArray photoarray = object.getJSONArray("photos");
        JSONObject photos = photoarray.getJSONObject(0);
        String imageURL = photos.getString("url");
        requestImage(imageURL);

        JSONArray socialprofiles = object.getJSONArray("socialProfiles");
        for (int i = 0; i < socialprofiles.length(); i++) {
            JSONObject socialprofile = socialprofiles.getJSONObject(i);
            String social = socialprofile.getString("type");
            String url = socialprofile.getString("url");
            responseView.append(social + " \t" + url + "\n");
        }
    } catch (JSONException e) {
        e.printStackTrace();
    }
}

private void requestImage(String imageURL){
    // Initialize a new ImageRequest
    ImageRequest imageRequest = new ImageRequest(imageURL, // Image URL
            new Response.Listener<Bitmap>() { // Bitmap listener
                @Override
                public void onResponse(Bitmap response) {
                    // Do something with response
                    Log.d("IMAGE", "in onresponse");
                    photoImageView.setImageBitmap(response);
                }
            },
            0,
```

```java
            0,
            null,
            null,
            new Response.ErrorListener() { // Error listener
               @Override
               public void onErrorResponse(VolleyError error) {
                  Log.e("ERROR", error.getMessage(), error);
               }
            }
   );

      // Add ImageRequest to the RequestQueue
      queue.add(imageRequest);
}
```

Update the button's onClick() method:

```java
@Override
public void onClick(View v) {
   String email = emailText.getText().toString();
   requestData(email);
}
```