ECE 1778 - Creativity and Programming for Mobile Devices
February 2016
Programming Assignment P4

Threads and Databases

The goal of this assignment is to learn the how to offload non-user interface tasks into separate threads, and to cover the basics of a simple on-device database.

1 Reading

Read the following sections from the course texts, if you are developing on Android:

i. Chapter titled "The WebView Widget" of the **The Busy Coder's Guide to Android Development**, version 6.9.
ii. Chapter titled "Dealing with Threads" of the **The Busy Coder's Guide to Android Development**, version 6.9.
iii. Chapter titled "SQLite Databases" of the **The Busy Coder's Guide to Android Development**, version 6.9.

For iPhone: from **Beginning iOS 7 Development** by Mark, Nutting, LaMarche and Olsson and the web**,** read:

i. Review the UIWebview Class – at this http://tinyurl.com/p5lzqnw.
ii. Lookup the method initWithContentsOfURL in the iPhone Documentation..
iii. Chapter 15 ("Grand Central Dispatch, Background Processing, and You").
iv. Chapter 13 ("Basic Data Persistence"), the section on Using SQLite3.

2 Assignment

NOTE: As in previous assignments, when writing your code for this assignment, please be sure to follow 'Braiden Brousseau's Guide To Quality Apps' that was given as part of Assignment P1. Part of your grade will be assigned based on following those guidelines.

You are to write an application that creates a small SQL database that is populated from one or more 'information files' located at a URL on the Internet. This information file will contain a list of names, a short description of each person, and a *local* (to that location in the internet) file path to a picture of the person. The information file format is given below. The program will then create a small 'dossier' for each person in the file, allowing the user to quickly learn about these people. Should the user want to learn more about an individual they are able to launch an Internet search for that person.

The application will have two primary functions: populating the database, and viewing the dossiers. A separate thread must be spawned while retrieving data over the Internet and populating the database. Here is the specification in more detail:

Populating the Database

- The user should be able to enter a URL where the file is located, and this location should default to the following string:
  http://www.eecg.utoronto.ca/~jayar/PeopleList.txt
- Upon pressing a button the app should download this information file and populate a database with its contents. This must be done on a newly **spawned worker thread**, and not on the main UI thread. While this is happening, the UI should display some kind of 'I know you're waiting' visual – such as a progress bar or spinning wheel. Once the database is populated, the first thread should send a message to the main UI process that it has finished (and then it should terminate), and the UI should both stop displaying the wait motif, and should emit a 'toast' (Android) or an 'alert' (iPhone) that indicates the database is loaded. At this point a second button ('Search') should become active.
- It should be possible to populate the database multiple times with different lists of people to create a database containing the content of all the retrieved information files combined. However, do not add duplicates of the same person.
- There should be a button that takes the user to the results section where the current people in the database can be viewed. This should not be possible if the database is empty or still in the processes of downloading a new file.
- There should be a button 'clear' which clears the whole database – i.e. makes it empty.

Viewing the Dossier of Each Person

- The dossier for each person in the database should be presented to the user. This should include the name, picture and biographical information.
- In the presentation of each dossier, there must be a way to indicate that the user doesn't wish to see this person again. If that indication is made, then this person should subsequently be removed from the database.
- If the user wants more information they should be able to launch a web browser with a search query of the person's name. This can be done by launching the default web browser of the device with appropriate parameters. Such a Google search can be done by simply loading a URL that looks something like www.google.ca/search?q=this+is+my+search although feel free to use any other search engine, or search method you find useful.
- The user should be able to return to the database population screen.

Viewing Results Special Instructions for iPhone Developers

The user should be able to quickly load the next entry or the previous entry but other than that you have the freedom to choose how you implement the user interactions.

Viewing Results Special Instructions for Android Developers

When the search button is touched a new activity should be launched which implements a ViewPager, which is a fragment-based android UI framework. (When creating a new Android Studio project or Activity you can select "Tabbed Activity" and then select under navigation type "swipe views" for a code skeleton for this type of interaction.

You will then create a single Fragment class with the layout for the dossier that accepts input arguments. For each name in the database you will create an instance of this fragment which will populate with the data for that person and associate it with a section of the ViewPager. Thus as you scroll to the right from page to page you should see successive dossiers. Pressing the 'back' button should cause the app to go back to the first activity with the 'populate' and 'search' buttons.

Information File Format

The file to be retrieved will have the following format with each entry on a new line. This file: http://www.eecg.utoronto.ca/~jayar/PeopleList.txt is an example of the general format:

Name1 <newline>
Bio1 <newline>
LocalFilePathToPicture1 <newline>
Name2 <newline>
Bio2 <newline>
LocalFilePathToPicture2 <newline>
Name3 <newline>
Bio3 <newline>
LocalFilePathToPicture3 <newline>
…

You can assume that a <newline> character delineates each piece string of information. Note that the local path to file name for picture is on the same server and in the same location as the file with the list itself.

For All Developers

Due date: Monday February 22nd, 6pm, marked out of 10, 0.5 marks off every hour late. Submit your solution on Blackboard portal, **Programmer Assignments** link and **Assignment P4**.

What to submit:

1. Android developers: a zip file containing your final Android application file (.apk); use your student number as the filename. Also submit the complete Android Studio project directory in a separate zip file.
2. iPhone developers: you must submit the complete project directory, including source, in a zip file. Use your student number as the filename. Please do your development on the 7.2 version of the SDK, and make sure that you haven't included any files by reference. In fact, please test your submitted zip file before sending it in.