Advanced Programming

Assignment 2: Hackathon Spec (Android Client-Server App)

Deadline: 21:00 1st April 2020

Introduction

During the hackathon you are going to develop a client-server application, an android app that connects to a (provided) server to retrieve data for it, and display the results.

When visiting places that you aren't quite so familiar with, it can sometimes be difficult to figure out how to get around via public transport. You are going to create an android app that can sense its location using GPS, and find out the names and distance to the nearest railway stations. You will be working with the android geolocation API, and the standard java/android networking and JSON parsing libraries.

I have found a list of the location of every railway station in the UK [1]. The locations in this list aren't quite as accurate as those in the official government lists [2], but the locations in the government statistics are expressed in British National Grid format, rather than the international latitude/longitude format used by GPS devices, and android.

The App

Your android client will supply its current location to the server by placing it the query string of a URL that it shall request from the server via HTTP. The server will supply data back in a JSON format, and the android app will decode this data and display it to the user. As in the labs, your app should be designed for use on a Google Pixel 3 phone, running Android 28, with a minimum supported version of Android 23.

Running The Server

The server has been made available as a single jar, available on Moodle. Once you have extracted both files from the zip, you can run the server either by double-clicking the jar file, or by running from the command line, in the directory into which you have saved the jar file.

```
java -jar server.jar
```

The server will start on port 8080. Because only one application can use an open port at once, you will not be able to start a second copy of the server until you have closed the first. If necessary, you can add an alternative port number to the command above to work on an alternative port.

To check that the server is working, and to view and experiment with the format of the parameters passed via the URL query string, you can load the following URL in your browser:

http://localhost:8080/stations?lat=53.472&lng=-2.244

The location supplied to the server above is my office, so the stations returned in the JSON data should all be familiar. The parameters in the URL are as follows:

lat The latitude of the location to search using

lng The longitude of the location to search using

The JSON data returned by the server consists of an array of objects, with each object containing the data for one individual nearby railway station. Each JSON object contains three keys, with the station name mapped to a string and the location to two double values, as depicted below.


```
[
  {
    "StationName": "Manchester Oxford Road",
    "Latitude": 53.4739,
    "Longitude": -2.2419
  },
  {
    "StationName": "Deansgate",
    "Latitude": 53.4743,
    "Longitude": -2.2515
  },
  {
    "StationName": "Manchester Piccadilly",
    "Latitude": 53.4775,
    "Longitude": -2.2313
  },
  {
    "StationName": "Salford Central",
    "Latitude": 53.4829,
    "Longitude": -2.2548
  },
  {
    "StationName": "Manchester Victoria",
    "Latitude": 53.4876,
    "Longitude": -2.2423
  }
]
```


Figure 1: JSON Format of Returned Data

Try a few different latitudes and longitudes in your browser, to get a feel for how you can manipulate the server into giving you the results for various different locations.

Geolocation Functionality (25 Marks)

Your app needs to track the device's location, so that when the user clicks the search button the app will know what latitude and longitude to supply to the server. Use the android Location service and the LocationListener interface to have your app receive updates every time the device's GPS chip informs the phone of a new location. **DO NOT** use the FusedLocation API, or the Google Maps, MapBox, or other third party API to retrieve the location, as these sometimes provide non-GPS locations, even when you specifically request one.

You will need to make sure your app declares the correct permissions, performs runtime checking, requests location updates and stores the current location in a variable so that it is immediately available when the user clicks the search button. **DO NOT** use getLastKnownLocation() when the user clicks the button, as if the GPS was not enabled until now, you won't get an up-to-date location.

Fetching and Displaying Data (25 Marks)

You will need to use a URLConnection to fetch the data from the server, use the built-in JSON decoding library to decode the data, and design a sensible user interface in which to display the results. More advanced solutions will calculate the distance to each of the returned stations, and display this to the user as well. **DO NOT** use any third party networking libraries when implementing your work (e.g. Volley or the Apache HTTP library).

You will need to use a slightly different URL in an android program to access the server running on your computer. Because the emulator is essentially a full computer running inside an application, localhost inside the emulator refers to the phone/tablet itself! Instead, you'll need to use the special IP address 10.0.2.2 to refer to the host computer, e.g.:

http://10.0.2.2:8080/stations?lat=53.472&lng=-2.244

You will need to substitute the user's current latitude and longitude, obtained from the device's GPS in the previous section, into the URL above to retrieve the railway stations closest to the user's current location. This is easily achieved using standard string concatenation (+) and variables.

If you have not (or have not yet) implemented threading, remember the StrictMode workaround you used in previous labs, which prevents android from throwing an exception if you attempt to perform a networking operation in the application's main thread. If your app throws a NetworkOnMainThreadException, then its because you forgot this!

Try to avoid situations where UI events can "orphan" the network request, which is relatively expensive in both computational and battery power.

A simple solution could display the results in a single TextView combined with a ScrollView, more advanced solutions may use a ListView with an Adapter, or even build a layout of many TextViews dynamically. You will be assessed on how well the data is decoded and displayed, and how well designed the interface is, so keep this in mind when deciding how much time to dedicate to this.

Code Quality (10 Marks)

Your work will be assessed on how well architected your app is, how well laid out and commented the code is, and how easy it would be for others to extend or reuse your code. Keep this in mind when working on your app. What should be put in which method? How best to lay out the code so that it is readable and maintainable? How best to help a later reader with comments? You should write sensible and easy to understand code, that is commented with JavaDoc comments.

Displaying Results on a MapBox Map (20 Marks)

The app would be considerably more useful if it displayed the nearby train stations to the user on a MapBox map. Ideally, the map should be zoomed to an appropriate level so that all the results fit on

screen simultaneously, whilst showing as much map detail as possible. The strongest solutions will use a custom map marker and name the station in the associated info window when clicking the marker. This section is intended to offer some additional challenge, going a little beyond what was covered in lectures: don't worry if you don't get to it.

Threading (20 Marks)

The StrictMode workaround mentioned earlier is an ugly hack, and is not meant for use in production code. The correct way to handle networking in android is to use a background thread to perform the request, and to perform the UI updates back in the app's main thread. The AsyncTask class makes this more convenient. Modify your networking code to use an AsyncTask, and remove the StrictMode workaround.

Submission

Submission on this assignment shall be via Moodle. You will need to make a zip of the Android Studio project directory. Also contained in this zip you will need to include some screenshots, taken in the official android emulator, of your app displaying the results of some specified searches in the following locations, to demonstrate that the app is working properly.

1. London City Airport, (Lat: 51.50508, Long: 0.04497)
2. Kyle of Localsh (Lat: 57.28288, Long: -5.71895)
3. Spital in the Street (Lat: 53.40126, Long: -0.55046)

Plagiarism and Duplication of Material

I, the Faculty, and the University all take academic malpractice very seriously. The work you submit for this assignment must be your own, completed without any significant assistance from others. Be particularly careful when helping friends to avoid them producing work similar to your own. I will be running all submitted work through an automated plagiarism checker, and I am generally vigilant when marking. The penalties for academic malpractice can be severe. Please refer to the guidance at http://www.celt.mmu.ac.uk/plagiarism/ for further information.

References

[1] Railway Codes. Railway Station Data. Available: http://www.railwaycodes.org.uk/stations/station0.shtm

[2] data.gov.uk. National Public Transport Access Nodes (NaPTAN) – Datasets. Available: https://data.gov.uk/dataset/naptan