Using Remote Connections in a Mobile App

Today's lab is about creating HTTP connections and grabbing info held on a remote server.

- Part 1 is an illustrate example of using HTTP connections.
- Part 2 is a more realistic example, retrieving data in JSON format.

PART 1 –Example of remote network connection from an app to a server

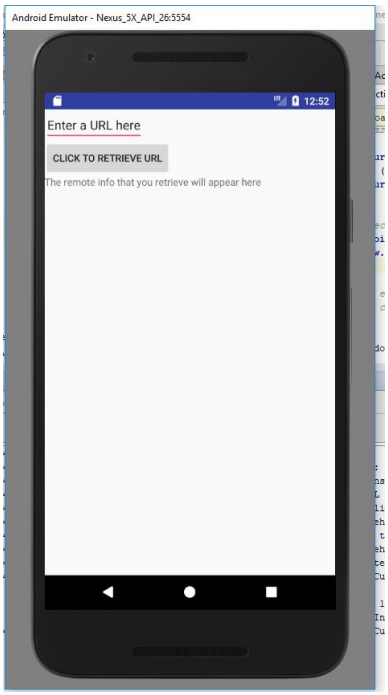
The purpose of this part is to get you using the `HttpURLConnection` class to connect to a remove server. This example connects to a server (based on the URL that you will type in), and retrieves the HTML at the URL.

The activity code in mainActivity.java under "Lab 10 Network Activity".

1. Create an XMl Layout screen to look like below – with an edittext, button and textview (use the IDs that you will need to get the activity working, as shown in the "findViewbyID methods in the activity).

2. Add code to your activity (in the onClick()) so that when the button is clicked, the activity checks for a network connection and runs the network processing tasks (the "slow" task in DownloadWebPageTask).

3. Don't forget to set your manifest file permissions: ACCESS_NETWORK_STATE and INTERNET



Answer the following questions about the code

- What does the Asynchronous task doing in the code? Why is it used?
- What is the method in the Asynch task that executes the task it was set up for?

Part 2 – JSON

Read Slides 20 – 27 in the Network lecture notes.

Apps often retrieve data from a remote source, and usually the data format is JSON. So, in this part, we'll retrieve JSON data from a remote location, and parse it/ manipulate it into usable cotent.

Get sample JSON Test data from a test site

If you were writing an app that retrieves JSON data from a remote database, you would have some sort of server side program (e.g. PHP script) doing queries on a remote database, and sending the result back as JSON data.

Since we don't have all the server side infrastructure here.. we'll use a JSON Test site instead that has been set up to hold some "fake" JSON test data that represents a data about a set of tasks. Click on the link below, you'll see if returns Json data bout "Todos" (Or type the URL into your Part 1 screen):

http://jsonplaceholder.typicode.com/todos

***Parse* the JSON data from the test site**

Edit the activity in part (1) so that it goes always to the URL at below (saves retyping it all the time): http://jsonplaceholder.typicode.com/todos

Add JSON parsing code to your activity in Part 1 so that you capture the JSON key value pairs into separate strings as shown below i.e. so that you have parsed the JSON data. Discard the USER ID data coming back.. so that you just have Task ID, description and complete status. Assign the parsed JSON data to your TextView on your screen