LabW05 – Cloud Service

Objectives:

1. Understand how to use Azure cloud service

Tasks:

1. Query data from Cloud
2. Use Azure Mobile Apps service [Optional]
3. Backup your TodoList database using Azure database [Optional]

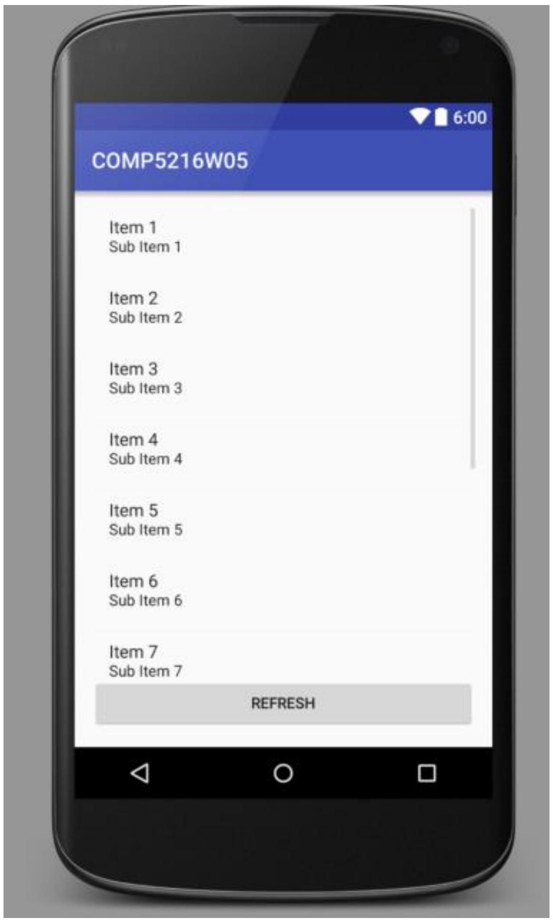
Task1: Query data from Cloud

Normally an app stores data locally on the phone (see LabW04). It does not allow data accessing from other clients, such as another phone and web site clients.

This task shows you how to connect your mobile app to a backend server and then query data from this backend server. You will create a simple app to query temperature data (captured by Raspberry Pi 3) that stores in the Azure Cloud and then display these data in a ListView.

1. Create a new empty Project named **COMP5216W05**

2. Edit your layout file
    - remove the “Hello World” TextView
    - add a **ListView** and a **Button** (Please refer to the screenshot below)



3. Copy the **TemperatureData.java** to “src” folder next to the MainActivity.

   This class is used for storing the temperature data that parsed from Azure Cloud Service

4. Copy **the ParseFromCloud.java** to “src” folder next to the MainActivity.

   This class is used for creating HTTP connections (Get/Post) to Azure Cloud Service and storing the parsed data to the TemperatureData.

   Use the following API to access data and display it in your app: http://usydcomp5216.azurewebsites.net/gettemp

5. Add a button click handler in your MainActivity

```
    public void onAddItemClick(View view) throws Exception {

        itemsAdapter.clear();

        ParseFromCloud temperatures = new ParseFromCloud();
        temperatures.gettemperature();
        ArrayList<TemperatureData> tds = temperatures.tds;

        for (TemperatureData td: tds) {
            String toAddString = td.getCreatedAt() + "\ntemperature is: " + td.getTemperature();
            if(toAddString != null && toAddString.length() > 0) {
                itemsAdapter.add(toAddString);
            }
        }
    }
```

6. Link the method to the button

   Edit the activity_main.xml layout file:

   1) Update the button text to “**REFRESH**”

   2) Add ‘android:onClick’ attribute to the method name ‘**onAddItemClick**’

7. Define the following variables in your MainActivity

```
//define variables
ListView listview;
ArrayList<String> items;
ArrayAdapter<String> itemsAdapter;

private final String USER_AGENT = "Mozilla/5.0";
```

8. Update the **onCreate** method in MainActivity

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

        //use "activity_main.xml" as the layout
        setContentView(R.layout.activity_main);
        if (android.os.Build.VERSION.SDK_INT > 9) {
            StrictMode.ThreadPolicy policy = new StrictMode.ThreadPolicy.Builder().permitAll().build();
            StrictMode.setThreadPolicy(policy);
        }

        //reference the "listview" variable to the id-"listview" in the layout
        listview = (ListView)findViewById(R.id.listview);

        //create an ArrayList of String
        items = new ArrayList<String>();

        //Create an adapter for the list view using Android's built-in item layout
        itemsAdapter = new ArrayAdapter<String>(this,android.R.layout.simple_list_item_1,items);

        //connect the listview and the adapter
        listview.setAdapter(itemsAdapter);
    }
```

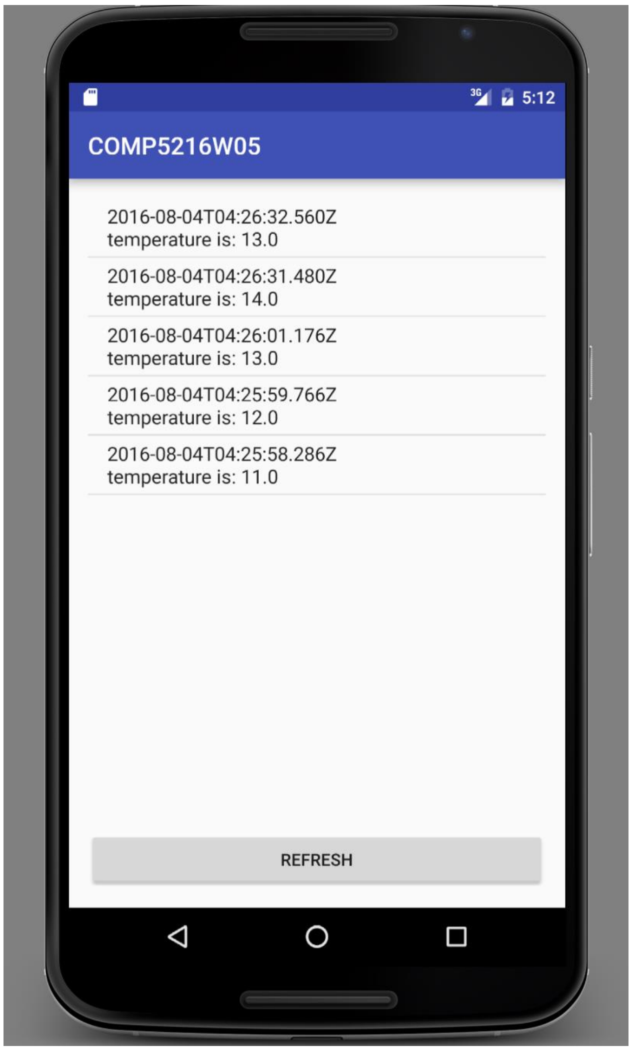
9. Add the following code in **AndroidMenifest.xml**

```
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="comp5216.sydney.edu.au.comp5216w05">

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

</manifest>
```

10. Run your app and click the REFRESH button. Once you finish the above steps correctly, you should be able to see something like this:



```
2016-08-04T04:26:32.560Z
temperature is: 13.0
2016-08-04T04:26:31.480Z
temperature is: 14.0
2016-08-04T04:26:01.176Z
temperature is: 13.0
2016-08-04T04:25:59.766Z
temperature is: 12.0
2016-08-04T04:25:58.286Z
temperature is: 11.0
```

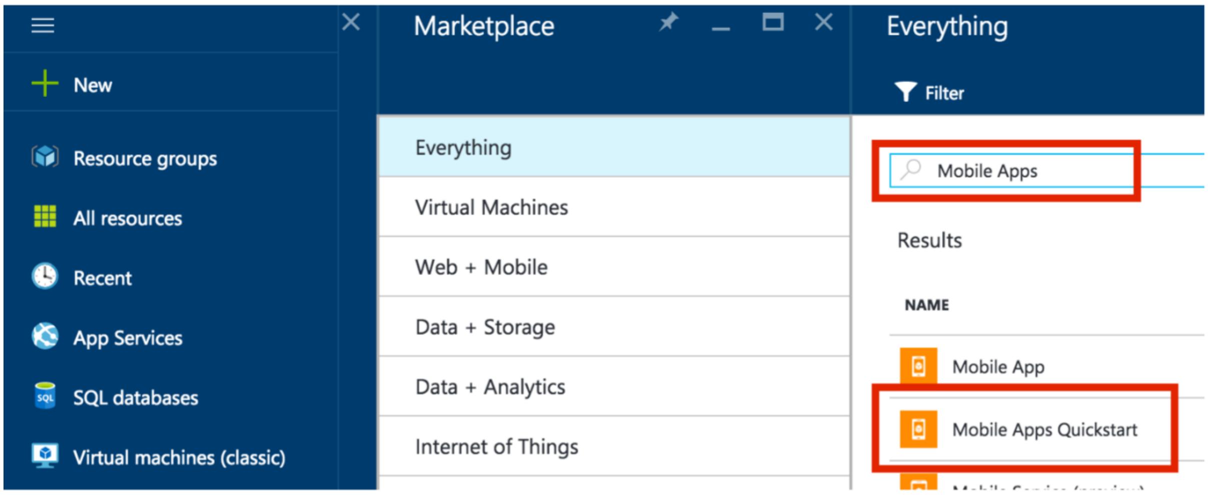
Task 2 [Optional]: Use Azure Mobile Apps services

This tutorial shows you how to add a cloud-based backend service to an Android mobile app by using an Azure mobile app backend. You will create both a new mobile app backend and a simple *Todo list* Android app that stores app data in Azure.

This tutorial is based on Microsoft Azure online tutorial: https://azure.microsoft.com/en-us/documentation/articles/app-service-mobile-android-get-started/

1. Go to https://azure.microsoft.com/en-us/ and use your SydneyUni account (e.g. abcd1234@uni.sydney.edu.au) to apply for a free Microsoft Azure Trial Pass.

2. Create a new Azure mobile app backend
   1) Log in at the Azure Portal.
   2) Click **+NEW** and type **Mobile Apps** in Search the marketplace. Select **Mobile Apps Quickstart** and click **Create**.

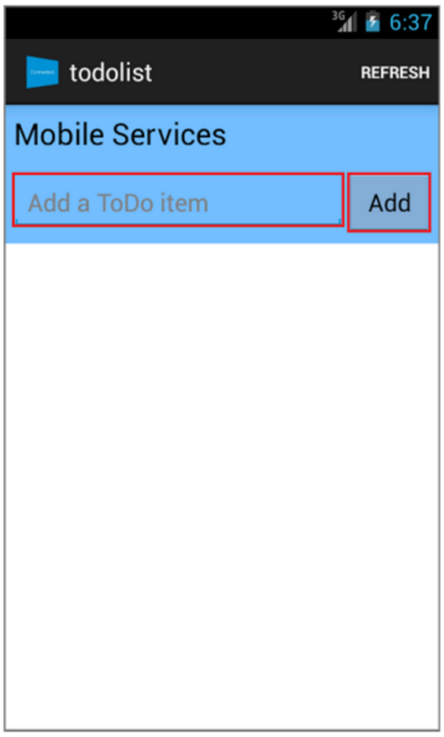


   3) For the **Resource Group**, select an existing resource group, or create a new one (using the same name as your app.)
   4) Click **Create**. Wait a few minutes for the service to be deployed successfully before proceeding.

3. Configure the server project
   1) Back in the Mobile App backend, click **Settings** > **Quick start** > your client platform.
   2) Under **Create a table API**, select Node.js for **Backend language**. Accept the acknowledgment and click **Create TodoItem table**. This creates a new TodoItem table in your database. Remember that switching an existing backend to Node.js will overwrite all contents! To create a .NET backend instead, follow these instructions.

4. Download and run the Android app
   1) Visit the Azure Portal. Click **Browse All** > **Mobile Apps** > the backend that you just created. In the mobile app settings, click **Quickstart** > **Android**). Under **Configure your client application**, click **Download**. This downloads a complete Android project for an app pre-configured to connect to your backend.

2) Open the project using Android Studio, using Import project (Eclipse ADT, Gradle, etc.). Make sure you make this import selection to avoid any JDK errors.
3) Press the **Run** 'app' button to build the project and start the app in the Android simulator.
4) In the app, type meaningful text, such as Complete the tutorial and then click the 'Add' button. This sends a POST request to the Azure backend you deployed earlier. The backend inserts data from the request is into the TodoItem SQL table, and returns information about the newly stored items back to the mobile app. The mobile app displays this data in the list.



5. The final content of your Activity would be similar to that in **ActivityWeek05_AzureMobileService.java** (in lab files). The only difference for your own Activity is a different Azure URL and a different application key.

```
mClient = new MobileServiceClient(
     "https://mobile-computing-tutorials.azure-mobile.net/",
     "fXKcgXHrdtIhmQXPLDkgEhqlxYwYjj11",
this).withFilter(new ProgressFilter());
```

6. The tutorial above uses a separate Android project. If you want to use Azure Mobile Service in your own project, e.g. the ToDoList project, the app needs to access the Internet, so add the “Internet” permission to the **AndroidManifest.xml**, inside the <manifest> tag but outside the <application> tag.

```
    <uses-sdk
        android:minSdkVersion="8"
        android:targetSdkVersion="21" />

    <!-- Week05 06 07 for Azure -->
    <uses-permission android:name="android.permission.INTERNET" />
```

   You will also need to copy the **mobileservices-1.1.5.jar** in the sample project into your own project’s **libs** folder.

Task 3 [Optional]: Backup your TodoList database using Azure database

Use the Azure Cloud Service in the ToDoList project used in W05 Lab. Rather than reading TodoItems from file or database, make it read and save TodoItems from the Azure cloud.

For more information, please visit the following link: https://azure.microsoft.com/en-us/documentation/services/sql-database/