COMP 90018 Mobile Computing Systems Programming

Tutorial on Android Development

Chu Luo, Eman Bin Khunayn

{chu.luo, eman.bin}@unimelb.edu.au

Welcome!

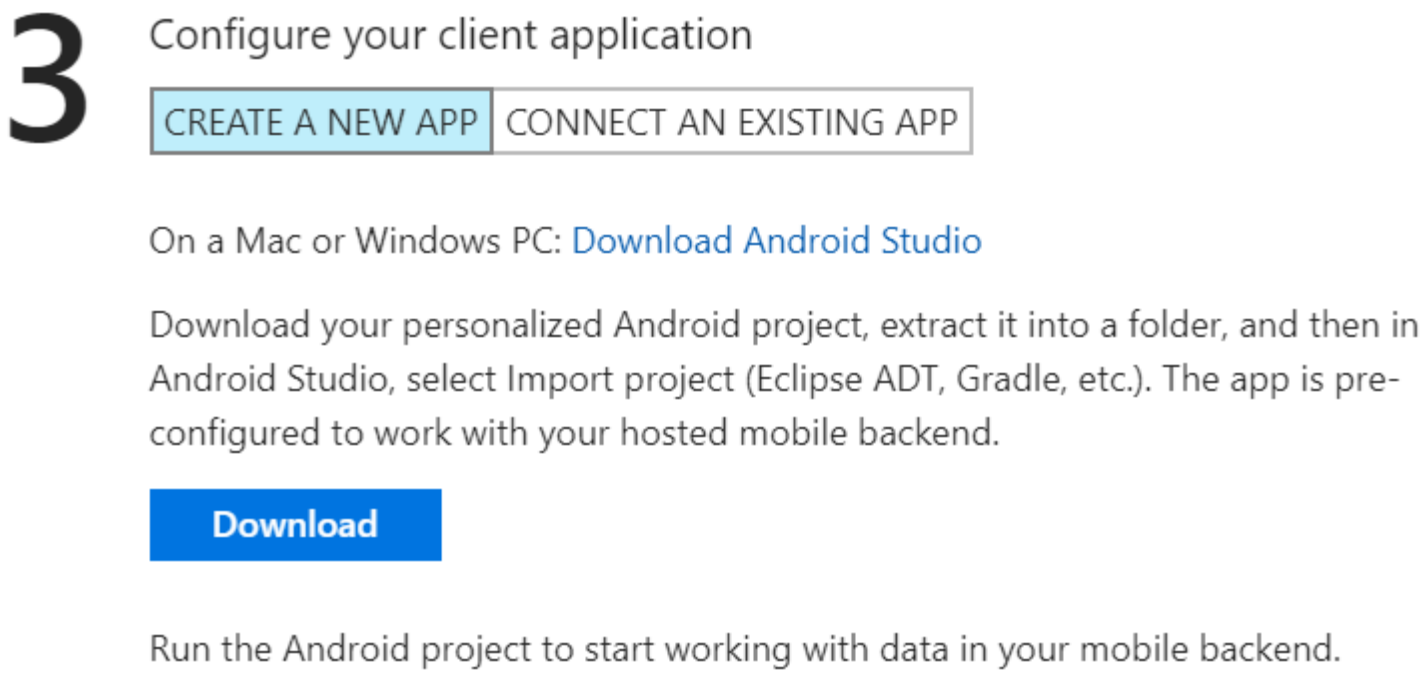
Outcomes of this tutorial:

1. Add Azure Connection in FrontEnd (Android App)

How to create a Android App with Azure FrontEnd?

Go to Android Studio!

App Service on Azure Portal

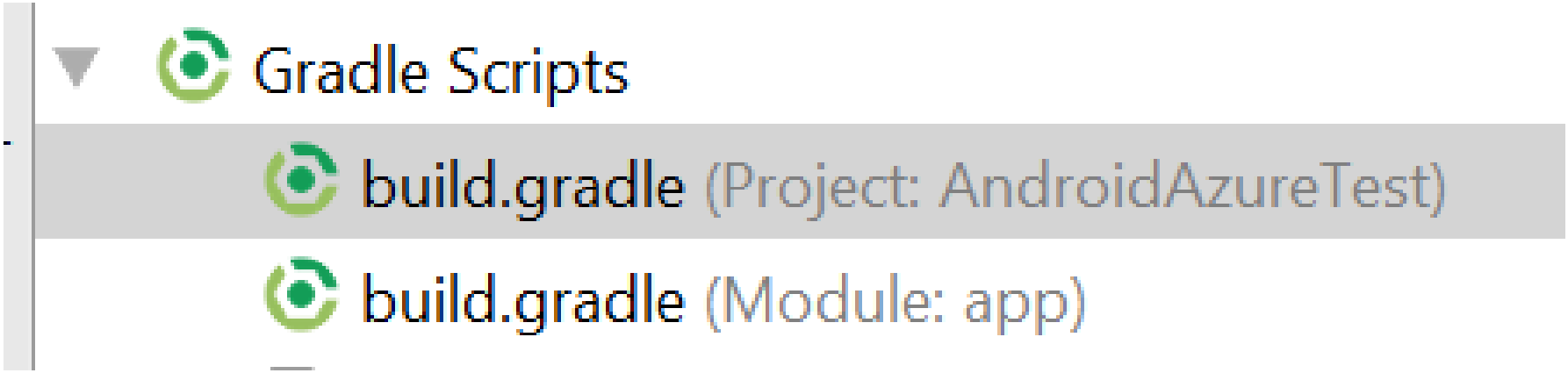


Download this example app for analysis

There are some Microsoft Instructions with many mistakes. Just check the concepts. Its code has bugs

https://docs.microsoft.com/en-us/azure/app-service-mobile/app-service-mobile-android-how-to-use-client-library

Locate your build.gradle files



You have a Project and Module one

FrontEnd

1. Add this code to the *Project* level **build.gradle** file inside the *buildscript* tag:

text

```
buildscript {
    repositories {
        jcenter()
    }
}
```

This should be already there

FrontEnd

2. Add this code to the *Module app* level **build.gradle** file inside the *dependencies* tag:

```text
compile 'com.microsoft.azure:azure-mobile-android:3.3.0'
```

Add this. If not working, use compile 'com.microsoft.azure: azure-mobile-android:3.3.0@aar'

FrontEnd

```
dependencies {
    compile fileTree(dir: 'libs', include: ['*.jar'])
    compile 'com.google.code.gson:gson:2.3'
    compile 'com.google.guava:guava:18.0'
    compile 'com.squareup.okhttp:okhttp:2.5.0'
    compile 'com.microsoft.azure:azure-mobile-android:3.2.0@aar'
    compile 'com.microsoft.azure:azure-notifications-handler:1.0.1@jar'
}
```

Also add these. Same as the example app.

Internet Permission

Enable internet permission

To access Azure, your app must have the INTERNET permission enabled. If it's not already enabled, add the following line of code to your **AndroidManifest.xml** file:

XML Copy

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

Just do it

Azure Mobile provides 4 things:

1. Data Access and Synchronization
2. Call Custom APIs
3. Authentication and Authorization
4. Push Notification Registration with Notification Hubs.

To do any of the 4, you need:

1. a MobileServiceClient object
2. URL of your back-end

The steps are here. Its code has bugs

1. https://docs.microsoft.com/en-us/azure/app-service-mobile/app-service-mobile-android-how-to-use-client-library#create-a-client-connection

Initialisation in MainAcitivty onCreate

```
/**
 * Mobile Service Client reference
 */
private MobileServiceClient mClient;

// Create the Mobile Service Client instance, using the provided
// Mobile Service URL and key
mClient = new MobileServiceClient(
        yourServiceURL,
        this).withFilter(new ProgressFilter());
```

Don’t have to use this

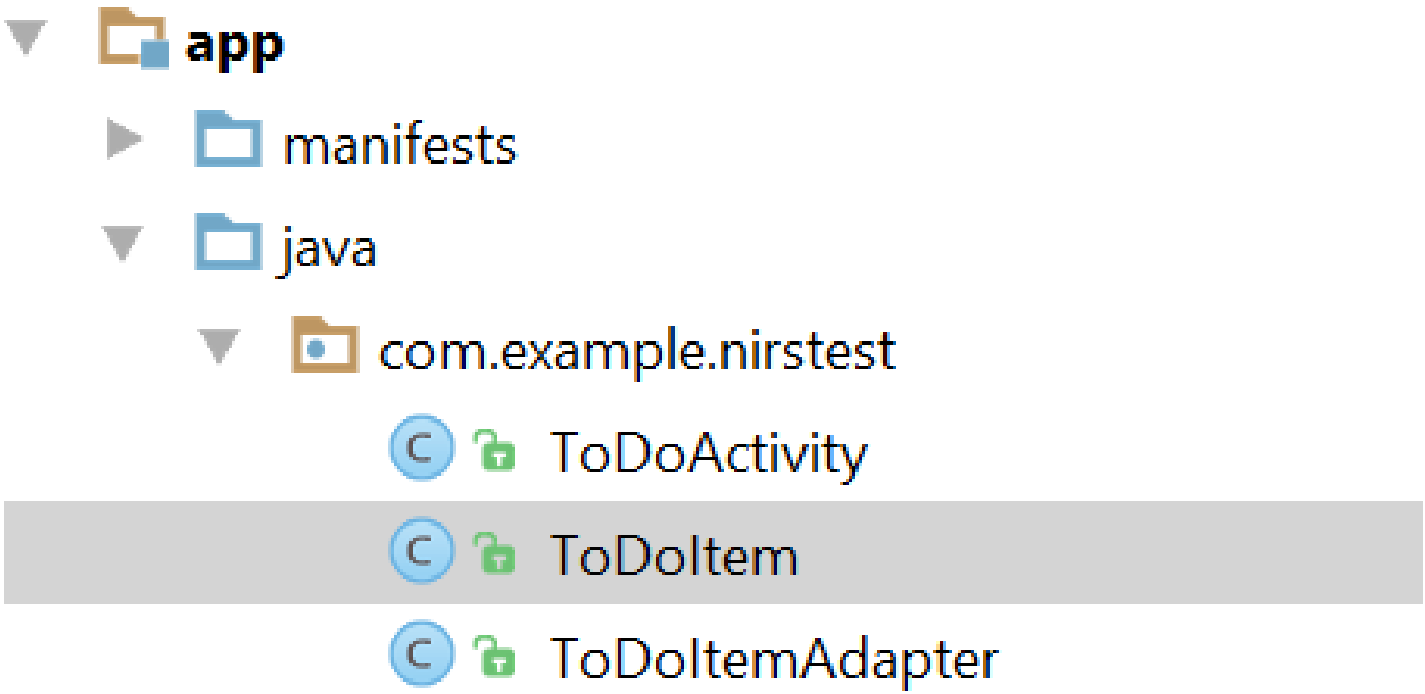
URL is a String like (http/https up to you) :"https://a.azurewebsites.net";

Define your data class

1. https://docs.microsoft.com/en-us/azure/app-service-mobile/app-service-mobile-android-how-to-use-client-library#data-operations

Fix bugs if any. Better refer to Your example app

Example data class definition



In your example app

Data access from Front to Back end

```
/**
 * Mobile Service Table used to access data
 */
private MobileServiceTable<ToDoItem> mToDoTable;


// Get the Mobile Service Table instance to use

mToDoTable = mClient.getTable(ToDoItem.class);
```

Use the MobileServiceTable, Same as your example app

How to manage Azure data then?

Avoid Data Access on UI thread

```
// Insert the new item
AsyncTask<Void, Void, Void> task = new AsyncTask<Void, Void, Void>(){
    @Override
    protected Void doInBackground(Void... params) {
        try {
            final ToDoItem entity = addItemInTable(item);
```

Example app uses AsyncTask

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

Avoid Data Access on UI thread

```
private AsyncTask<Void, Void, Void> runAsyncTask(AsyncTask<Void, Void, Void> task) {
    if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.HONEYCOMB) {
        return task.executeOnExecutor(AsyncTask.THREAD_POOL_EXECUTOR);
    } else {
        return task.execute();
    }
}
```

executeOnExecutor can improve Performance via parallel tasks

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

Insert operation for Azure DB

```
public ToDoItem addItemInTable(ToDoItem item) throws E
    ToDoItem entity = mToDoTable.insert(item).get();
    return entity;
```

Example app uses this to add an item on Azure SQL database

Update operation for Azure DB

```
public void checkItemInTable(ToDoItem item)
    mToDoTable.update(item).get();
}
```

Example app uses this to make an item as complete on Azure SQL database

Retrieve operation for Azure DB

```
private List<ToDoItem> refreshItemsFromMobileServiceTable()
    return mToDoTable.where().field("complete").
            eq(val(false)).execute().get();
}
```

This returns items (which *complete* value are false) on Azure SQL database

Detele operation for Azure DB

Delete data in a mobile app

The following code shows how to delete data from a table by specifying the data object.

Java

```
mToDoTable
    .delete(item);
```

You can also delete an item by specifying the **id** field of the row to delete.

Java

```
String myRowId = "2FA404AB-E458-44CD-BC1B-3BC847EF0902";
mToDoTable
    .delete(myRowId);
```

https://docs.microsoft.com/en-us/azure/app-service-mobile/app-service-mobile-android-how-to-use-client-library#deleting

There are more things that an Android app can do with Azure

https://docs.microsoft.com/en-us/azure/app-service-mobile/app-service-mobile-android-how-to-use-client-library

More to explore:

1. Implement Offline Sync (between your phone SQLite and Azure DB)
2. Call a custom API (any actions beyond CRUD for the server to do)

More to explore:

3. Add authentication to your app (to identify your users and provide each personalised service)

Demo …

Exercise:

1. Let your BackEnd and FrontEnd communicate

2. E.g., CRUD your Azure database from Android

3. Custom an API (for advanced actions)

More learning directions:

1. Finish your assignment with sensors and Azure

See you next week

COMP 90018

Tutorial on Android Development

Chu Luo, Eman Bin Khunayn

{chu.luo, eman.bin}@unimelb.edu.au