Quick Start demonstration

fork and edit tutorial (https://github.com/MobileFirst-Platform-Developer-Center/DevCenter/#fork-destination-box) | report issue (https://github.com/MobileFirst-Platform-Developer-Center/DevCenter/issues/new)

The purpose of this demonstration is to make you experience an end-to-end flow where IBM MobileFirst Platform Foundation SDK for Android is integrated into an Android project and used to retrieve data by using a MobileFirst adapter.

To learn more about creating projects and applications, using adapters and lots more, visit the Native Android Development (../) landing page.

Prerequisite: Make sure that you have installed the following software:

- MobileFirst Platform command line tool (download (file:////home/travis/build/MFPSamples/DevCenter/_site/downloads))
- Android Studio

---

1. **Create a MobileFirst back-end project and adapter.**
   - Create a back-end project in a location of your choice.

     ```
     mfp create MyProject
     cd MyProject
     ```

   - Add an HTTP adapter to the project.

     ```
     mfp add adapter MyAdapter -t http
     ```

2. **Deploy artifacts to the MobileFirst Server.**
   - Start the MobileFirst Server and deploy the adapter.

     ```
     mfp start
     mfp push
     ```

3. **Create an Android project in Android Studio.**

4. **Add the MobileFirst Android SDK to the Android Studio project**
   - In **Project > Gradle scripts**, select **build.gradle (Module: app)**.
   - After `apply plugin: 'com.android.application'`, add the following line:

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

- Inside `android`, add the following lines:

```
packagingOptions {
    pickFirst 'META-INF/ASL2.0'
    pickFirst 'META-INF/LICENSE'
    pickFirst 'META-INF/NOTICE'
}
```

- Inside `dependencies`, add the following lines:

```
compile group: 'com.ibm.mobile.foundation',
    name: 'ibmmobilefirstplatformfoundation',
    version: '7.1.0.0',
    ext: 'aar',
    transitive: true
```

- Add the following permissions to the `AndroidManifest.xml` file:

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

- Add the MobileFirst UI activity:

```
<activity android:name="com.worklight.wlclient.ui.UIActivity" />
```

- In Terminal, navigate to the root of the Android Studio project and add the required configuration files by running this command:

```
mfp push
```

- ## Implement MobileFirst adapter invocation.

  - ### Main `Activity` class

    Make sure that your MainActivity class extends the `Activity` class:

    ```
    public class MainActivity extends Activity {
    ...
    ```

    Add the following `import` statements:

```
import com.worklight.wlclient.api.*;
import android.util.Log;
import java.net.URI;
import java.net.URISyntaxException;
```

Add the following lines to the `onCreate` method:

```
super.onCreate(savedInstanceState);
setContentView(R.layout.activity_main);
final WLClient client = WLClient.createInstance(this);
client.connect(new WLResponseListener() {
    @Override
    public void onSuccess(WLResponse wlResponse) {
        URI adapterPath = null;
        try {
            adapterPath = new URI("/adapters/MyAdapter/getFeed");
        } catch (URISyntaxException e) {
            e.printStackTrace();
        }
        WLResourceRequest request = new WLResourceRequest(adapterPath,WLRe
sourceRequest.GET);
        request.send(new MyInvokeListener());
    }
    @Override
    public void onFailure(WLFailResponse wlFailResponse) {
        Log.i("MFPMyProject","Failed connecting to the MobileFirst Server: " + wlFailR
esponse.getErrorMsg());
    }
});
```

- `MyInvokeListener` class

  Add a new `MyInvokeListener` class.
  Add the following `import` statements:

  ```
  import com.worklight.wlclient.api.*;
  import android.util.Log;
  ```

  Paste the following lines:

```
public class MyInvokeListener implements WLResponseListener {
    @Override
    public void onSuccess(WLResponse wlResponse) {
        Log.i("MFPMyProject","Adapter invocation response: " + wlResponse.getResp
onseJSON());
    }
    @Override
    public void onFailure(WLFailResponse wlFailResponse) {
        Log.i("MFPMyProject", "Adapter invocation response: " + wlFailResponse.get
ErrorMsg());
    }
}
```

- ## Final configurations
  - Create an Android Virtual Device (AVD).
- ## Click Run.

  Review the LogCat view for the data retrieved by the adapter request.