MakeMyRun - Development Introduction

Tools, and basic knowledge of the following required

- Emma
- Eclipse
- Ant
- Android SDK
- Java
- Stan for Eclipse

Note: All guidance for a command line tool is described for a bash terminal, specifically for a linux or Mac environment.

Initial setup

1. Fetch git repository from https://github.com/Salking/MakeMyRun
2. Set up Eclipse with the Android SDK. A guide on how it's easiest done can be found at http://developer.android.com/tools/sdk/eclipse-adt.html
    a. Make sure you install the Google API v. 15 through the Android SDK Manager. You also need to make sure that the Eclipse project is using the correct API when building the application. This is done through the Project properties -> Android page and there checking Google API v. 15.
3. Import the root folder from the git repository as a project called MakeMyRun
4. Import the MakeMyRunTest folder from git repository root as a separate Eclipse project.

API description

The application is built around the two following Google APIs:

We rely heavily on the **Google Maps API for Android** while developing this application. Their documentation can be found here https://developers.google.com/maps/documentation/android/

To make sure our computed routes follows walkable paths (i.e. no highways) we use **Google Directions** restful web API.Their documentation is found here https://developers.google.com/maps/documentation/directions/

As a new developer it is essential that you know of these two APIs.

Setting up keystore

In order to get the Google Maps API for Android to work correctly, i.e. allow the MapView created inside our Activity to render the actual map provided by google you need a keystore file. This should be placed somewhere outside of your git repository so that you do not push it to our public repo accidentally.

The file is acquired from a senior developer through mail and shall not be distributed outside the development team.

Once the file is on your computer you do **one** of the following

1. In Eclipse (**NOTE**: This solution does not provide command line builds with correct signature)
    - Open Window -> Preferences -> Android -> Build and point the custom keystore path to the keystore acquired for this project.
2. Overwrite your old debug.keystore file with the newly acquired keystore file. Path should be ~/.android/debug.keystore

Build

In Eclipse

Assuming the project source has been imported properly and the compiler recognizes all libraries and does not prompt you with any other errors in the Project (which is a lot of ifs, but these are easiest handled face to face, if there are problems here ask a senior developer for help).

Right-click the project from Eclipse's project manager and choose **Debug as... Android Application**. We build in debug mode because the Google Maps API requires us to do so.

Tests are run by right-clicking the MakeMyRunTest project and choosing Run as... Android JUnit Test (or Debug as..., both works since we do not need the Maps to render properly here).

From command line

An initial setup to be able to run tests is needed.

1. Open up a terminal
2. cd into where you cloned the git repository, for example "*/home/myuser/PiFive/MakeMyRun"*
3. In this exact sequence run
    a. *android update project --path ´´pwd´´*
    b. *android update test-project -m ´´pwd´´ -p MakeMyRunTest*

From project root folder you can now do ***ant debug install*** to build the application in debug mode and then install it to either a running emulator or a plugged in Android device.

From MakeMyRunTest folder you can now do ***ant emma debug install test*** to build the application and the test project and run tests on an emulator, **not** an ordinary Android device. This also produces a **coverage report** in MakeMyRunTest/bin/coverage.html

Release

- First **build an apk file** using the Eclipse method. Run the apk on a real Android Device and check that the Maps are rendered correctly, click around to make sure nothing fatal has happened while building.
- Copy the MakeMyRun.apk file into dist/
- Update all documentation files located in dist/, update the actual documentation on the project's Google Drive and then download and commit them through git.
- Once everything is in place and committed create a tag for the release.
    - *git tag -a "version.subversion.bugfix" -m "Releasing new version x.x.x"*

- *git push --tags*

Brief source code walkthrough

The implemented functionality can be viewed in the User's Manual and is easiest grasped by testing the app personally. Very shortly it is about generating routes for the user to inspect and then run the path we've generated for her.

We use a tool called STAN to view an analysis of the source structure. This tool is also very helpful when you wish to get an overview of the whole project. Make sure you have STAN integrated to your Eclipse environment and right-click the MakeMyRun project -> Run as... -> Structure analysis.
This will open up the STAN view where you can click around to get familiar with the different packages.

For detailed description please refer to each class' Javadoc