Developer Manual

Chalmers On The Go – the Complete Chalmers Experience

This document provides guidance on further development of the Andriod application ChalmersOnTheGo – a map over the Chalmers area. The project is described and procedures for building, coding and releasing the app are discussed. Eclipse is recommended as developing tool since Eclipse provides a number of automated features, eliminating a lot of manual work. This manual describes procedures for Eclipse.

For ChalmersOnTheGo 1.0, Jelly Bean 4.0 and API 16

Fredrik Einarsson - Niklas Johansson - René Niendorf
Anders Nordin - Sofie Peters

Developer Manual

Table of Contents

1 Application description and structure

The ChalmersOnTheGo application features a map over Chalmers University of Technology. It contains usual map functions, such as navigating to determined destinations inside of the university area or searching for locations such as group rooms, lecture halls or school pubs. The application does also feature some other, less map related functions, such as a step counter and iCal synchronization for students' schedules. Note that the database containing information on the geographical sites is not complete, only containing enough data to test the applications functionality and performance.

The texts below depict the different classes in the ChalmersOnTheGo application, their mutual relationships and functions.

1.1 Package dat255.group5.chalmersonthego

This package contains all classes that the application consists of, except for those concerned with the database.

1.1.1 CalorieDialog

Creates and handles the CalorieDrinkingProgress feature.

1.1.2 CustomGoogleMaps
Defines customised view of Google Maps and handles the application's different map features such as clicking information window or checking map boundaries.

1.1.3 DirectionJSONParser
Handles JSON objects and polyline points.

1.1.4 MainActivity
Holds a Google Map. Calls on all the other classes, generally administrating what happens in the application.

1.1.5 MapWrapperLayout
Defines the customised look of the application's markers and similar using Android's RelativeLayout.

1.1.6 NavigationManager
Handles navigation on the campus map by drawing routes and communicating with and fetching data from Google Maps.

1.1.7 OnInfoWindowElemTouchListener
Handles the information window popping up on marked locations.

1.1.8 SpaceTokenizer
Exchanging comma (,) for space ( ) to oppose the Android feature of adding comma after search in navigation fields.

1.2 Package dat255.group5.database
This package contains the ways to create, open and close the database, as well as manipulating the data in it.

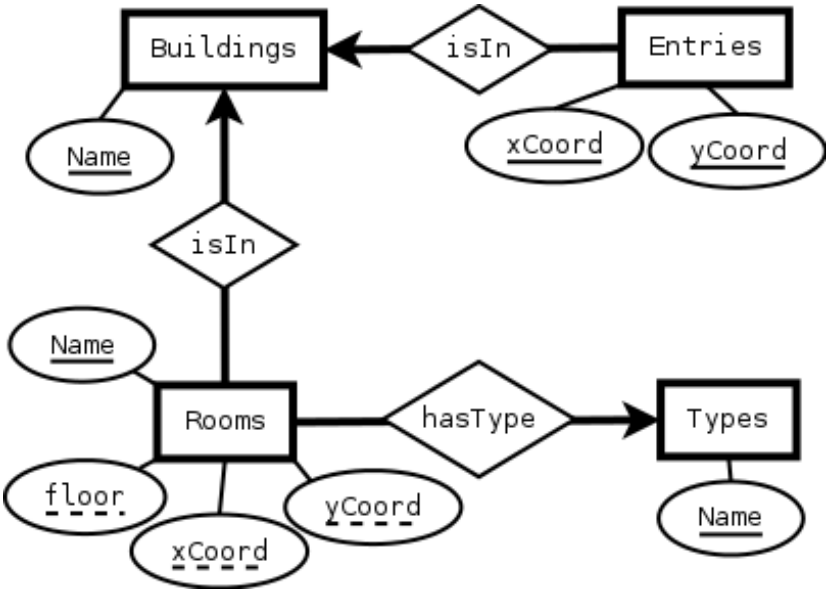
1.2.1 DAO
DAO – Data Access Object. A layer on top of the database, making opening, closing and manipulating the database via Java code possible.

1.2.2 DatabaseConstants
Stores all constants related to the database data.

1.2.3 DBHelper
Creates and holds the SQLite Database. Not to be called, preferably use the DAO instead.

1.2.4 InsertionsOfData
Class used to insert all data used in the application. This is where any new data can be added or existing data altered by using relational database insertions. Follow the existing pattern for easy insertion.

ER-diagram



1.2.5 SuggestionsContentProvider.java

Provides suggestions during search based on the contents in the database.

2 Build procedures

The text below describes needed libraries and plug-ins, and furthermore, procedures for building and running the ChalmersOnTheGo application, using Eclipse.

2.1 Set up libraries and project

1. Install Google Play services from the SDK manager and import as a new project.
2. Check that the ChalmersOnTheGo application is a Google Play project in properties
3. Add google-play-services_lib as library

2.1.1 Detailed instructions

SDK Manager

1. Download the Android SDK from http://developer.android.com/sdk/index.html
2. Launch the SDK Manager.
    a. From Eclipse, select **Window** > **Android SDK Manager**.
    b. On Windows, double-click the SDK Manager.exe file at the root of the Android SDK directory.
    c. On Mac or Linux, open a terminal and navigate to the tools/ directory in the Android SDK, then executeandroid sdk.
1. Scroll to the bottom of the package list, select **Extras > Google Play services**, and install it. The Google Play services SDK is saved in your Android SDK environment at <android-sdk-folder>/extras/google/google_play_services/.
2. Copy the <android-sdk-folder>/extras/google/google_play_services/libproject/google-play-services_lib library project into the source tree where you maintain your Android app projects. In Eclipse, import the library project into your workspace. Click **File > Import**, select **Android > Existing Android Code into Workspace**, and browse to the copy of the library project to import it.

Set up a project using the Google Play Services SDK:

1. Reference the library project in your Android project. **Note:** You should be referencing a copy of the library that you copied to your source tree - you should not reference the library from the Android SDK directory.
    a. To add a reference to a library project, follow these steps:
    b. Make sure that both the project library and the application project that depends on it are in your workspace. If one of the projects is missing, import it into your workspace.
    c. In the **Package Explorer**, right-click the dependent project and select **Properties**.
    d. In the **Properties** window, select the "Android" properties group at left and locate the **Library** properties at right.
    e. Click **Add** to open the **Project Selection** dialog.
    f. From the list of available library projects, select a project and click **OK**.
    g. When the dialog closes, click **Apply** in the **Properties** window.
    h. Click **OK** to close the **Properties** window.
    i. As soon as the Properties dialog closes, Eclipse rebuilds the project, including the contents of the library project.

Access the Google Maps servers with the Maps API

You have to add a Maps API key to your application. To do this, you will need to register a project in the Google APIs Console, and get a signing certificate for your app.

1. Retrieve information about your application's certificate, more information on: https://developers.google.com/maps/documentation/android/start#displaying_certificate_information
2. Register a project in the Google APIs Console and add the Maps API as a service for the project, more information on: https://developers.google.com/maps/documentation/android/start#creating_an_api_project
3. Once you have a project set up, you can request one or more keys, more information on: https://developers.google.com/maps/documentation/android/start#obtaining_an_api_key
4. Finally, you can add your key to your application and begin development, more information on: https://developers.google.com/maps/documentation/android/start#adding_the_api_key_to_your_application

Specify settings in manifest file

An Android application that uses the Google Maps Android API needs to specify the following settings in its manifest file, AndroidManifest.xml:

- Permissions that give the application access to Android system features and to the Google Maps servers.
- Notification that the application requires OpenGL ES version 2. External services can detect this notification and act accordingly. For example, Google Play Store won't display the application on devices that don't have OpenGL ES version 2.
- The Maps API key for the application. The key confirms that you've registered with the Google Maps service via the Google APIs Console.

More information on how to do specify the settings on:
https://developers.google.com/maps/documentation/android/start#specify_settings_in_the_application_manifest

For more information on Android developing:
http://developer.android.com/tools/workflow/index.html
For more information on Google Play Services: http://developer.android.com/google/play-services/index.html

2.2 Building

If you are developing in Eclipse, the ADT plugin incrementally builds your project as you make changes to the source code. Eclipse outputs an .apk file automatically to the bin folder of the project, so you do not have to do anything extra to generate the .apk. If you are developing in a non-Eclipse environment, you can build your project with the generated build.xml Ant file that is in the project directory. The Ant file calls targets that automatically call the build tools for you.

More information on: http://developer.android.com/tools/building/index.html

2.3 Running

To run an application on an emulator or device, the application must be signed using debug or release mode. Eclipse signs the application for you in debug mode when you build your application, which makes running the app while developing possible. For information on release mode, see section 3 Release procedures.

More information on: http://developer.android.com/tools/building/index.html

2.4 Testing

To create a test project:

1. Import > New android project,
2. Browse to the chalmersOnTheGo library and place yourself on the test folder
3. Click ok
4. Click on Properties in your new ChalmersOnTheGoTest project
5. Click on Build path > Projects > Add ChalmersOnTheGo

More information on the Androis testing framework:
http://developer.android.com/tools/testing/testing_android.html

3 Release procedures

Before you begin, make sure that the Keytool utility and Jarsigner utility are available to the SDK build tools. Both of these tools are available in the JDK. More information on:
http://developer.android.com/tools/publishing/app-signing.html

If you are using Eclipse with the ADT plugin, you can use the Export Wizard to export a *signed* APK (and even create a new keystore, if necessary). The Export Wizard performs all the interaction with the Keytool and Jarsigner for you, which allows you to sign the package using a GUI instead of performing the manual procedures to compile, sign, and align, as discussed above. Once the wizard has compiled and signed your package, it will also perfom package alignment withzipalign.

To create a signed and aligned APK in Eclipse:

1. Select the project in the Package Explorer and select **File > Export**.

2. Open the Android folder, select Export Android Application, and click **Next**.
   The Export Android Application wizard now starts, which will guide you through the process of signing your application, including steps for selecting the private key with which to sign the APK (or creating a new keystore and private key).
3. Complete the Export Wizard and your application will be compiled, signed, aligned, and ready for distribution

More information on releasing on:
http://developer.android.com/tools/publishing/publishing_overview.html