**TensionCamApp: Developer Manual**

**Latest revision:** 2013-05-20

A quick-start guide and overlook of the TensionCamApp Android application designed for G-coder Systems AB.

**Getting started**

git clone git:// github.com/lisarythenlarsson/TensionCamApp

**Dependencies**

* Android SDK
* An actual (preferred) or virtual Android device
* Java XX XX (6 SE) development environment

**Android SDK targets**

● Minimum SDK: **16**

● Target SDK: **16 (ändra i eclipse?)**

(SDK targets are motivated in *Project Plan*)

**Building and installing**

A build.xml is included in the root directory which may be used for building the project. The default output directory is bin in the project root. By default, Eclipse is set to build automatically. To build the project manually, go to *Project* in the menu bar.

To install the TensionCamApp on an Android device, the device has to be connected via Android Debug Bridge (ADB) and then installed

To build and install the WatchMe.apk package file on a connected Android device, run the following

on the command line in the project directory:

$ ant clean debug install

To uninstall the application from the device:

$ ant uninstall

That will install the application in debug mode. Other modes are available: instrument and release.

To view all ant targets, run

$ ant -p

**Release procedure**

This section describes the steps taken before every major release of the WatchMe application.

**Requirements**

To build an application package in release mode, it needs to be signed with a certificate. Refer to this

Android guide on signing applications for release: http://developer.android.com/tools/publishing/appsigning.

html. A keystore also needs to be present somewhere in your system, which path should be

DAT255

specified in an ant.properties file in the project root.

**Building a release package**

Use Ant to build an .apk file for release:

$ ant clean release

You will be prompted for the passphrase of your keystore in this process. The built package will be

put in the directory bin/WatchMe-release.apk.

**Organizing the distribution directory**

After having built a release package, it should be organized in the distrbution directory (dist in the

project root).

1. Create a new directory in dist named with the version number. Examples:

a. v0.1alpha

b. v0.2

c. v0.4

d. v0.6beta

e. v0.3rc1

2. Move the WatchMe.apk package from the bin directory to the newly created release directory

3. Rename the application package to WatchMe-<release>.apk. Examples:

a. WatchMe-v0.1alpha.apk

b. WatchMe-v0.2.apk

c. WatchMe-v0.6beta.apk

**Release requirements**

Every release’s directory include the following:

● An application package (see above).

● A release notes document with the following headings (if applicable):

○ New features

○ Changed features

○ Removed features

○ Known bugs (refer to bug ID)

○ Coming features

● A test report document. See existing reports for templates.

**Tests**

Automatic tests are included in a separate project embedded within the application project, and is

called WatchMeTest. To run the included tests, head to the command line and use ant from the

WatchMeTest directory:

$ ant test

Make sure to test the newest application package by running ant clean debug install before

running any tests.

**Test libraries**

DAT255

For GUI testing, the Robotium (http://code.google.com/p/robotium/) library is used (version 3.5.1).

To run the Robotium test code, please download and put the Robotium JAR package in the

WatchMeTest/libs directory. It’s already referred to that location in .classpath.

**Architecture**

The functionality of the application is built up by three different parts. A web server together with a STS (Spring Tool Suite) project, external software for picture analysis and the android project. Kan vi motivera val här? The reasons for using a web server are several. The main reasons though are the following:

* Less redundancy since there is no need for each device (cell phone) to carry the analyze program. This minimizes the storage need for the application. It also enables the analyzing program to be run on a more powerful machine.

The application code resides in packages organized by area, such as database (for data provider

interactions), net (for HTTP and IMDb connections), activity (for Android Activities), etc. Various

helpers are in the utils package.

The domain model consists of two classes: Movie and Tag.

**Content Provider**

**Package:**

The application code

**External libraries:**

HttpClient 4.2.5

HttpCore 4.2.4

**Motivation for libraries:**

These two libraries enables connecting to and communicating with the web server through POST request.

database

The data source is backed by an SQLite database which a Content Provider is using to interface with

the application code.

**Web server Package**

**External libraries:**

Commons FileUpload 1.3

**Motivation for libraries:**

Enables the client to send picture (file) to the web server through a POST request whit multipart/form-data, in this case the picture.

**HTTP and the IMDb API connection**

**Package**

net

**API endpoint:**

http://api.themoviedb.org/2.1/

**Response format:**

JSON

The API calls to the IMDb API is handled by a middle layer class called HttpRetriever which

basically performs an HTTP GET request for an arbitrary URL (using the Apache HTTP library). The

top layer is the IMDBHandler class which uses a HttpRetriever to make calls to the IMDb API

service.

**Notifications and Services**

**Package**

notifications

The movie notification feature is handled by a system of alarm tasks, clients and Android Services.

The main task for the system is to schedule Android notifications on a certain date (e.g. a movie’s

release date, set by the user).

DAT255

To use notifications with the Movie model, create an instance of the NotificationClient

class, connect to the underlying notification service with connectToService(), and use

setMovieNotification(Movie movie) to set a notification for a movie object.