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 6 SE development environment

Android SDK targets

- Minimum SDK: **16**
- Target SDK: **16**

(SDK targets are motivated in *Project Plan*)

License

Licensed under the Apache License, Version 2.0 (the "License").

The Apache 2.0 license is a free and commonly known open source license. It is a flexible license which can be combined with other licenses, this makes it easy to work with. It also allows patents which might be of interest to the client in the future. Other than that it protects us from being held liable for errors and problem the application might cause ahead.

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 project has to be run as an Android Application in Eclipse. In order for this to work, the device has to be connected via Android Debug Bridge (ADB).

To uninstall the TensionCamApp on an Android device, standard procedures should be followed. I.e. using the device's default tool for app-removal is recommended.

To be able to send the picture to the analysis program there has to be a web server which the program lies on. The solution that has been used for this project is an Apache Tomcat web server that has been used locally. In order for the devices to be able to communicate with the web server the Spring MVC project has to be deployed on the web server. To make the project deployable it has to go through two steps:

1. Run the Pom.xml as "Maven clean".
2. Run the Pom.xml as "Maven install".

To be able to connect the device to the local web server two additional steps has to be followed:

1. Create a hot spot on the device.
2. Connect the local server to this hot spot.

Release procedure

This section describes the steps taken before every release of the TensionCam 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/app-signing.html. Releases will be built, signed and compiled using Eclipse ADT. In order to build and install, access to a private key is necessary. The key is stored in a keystore and in order to get access to the full path, authors need to be contacted.

Building a release package

Use Eclipse ADT to compile and sign an apk-file for release:

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. When signing you will need to provide the key alias and password stored in keystore.
3. Complete the Export Wizard and your application will be compiled, signed, aligned, and ready for distribution.

Organizing the distribution directory

After having built a release package, it should be organized in the distribution directory: **dist** in root direvtory.

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

   a. V0.1

   b. V0.2

   .

   .

   .

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

3. Rename the application package to TensionCamApp-<version>.apk. Examples:

   a. TensionCamApp_V0.1.apk

   b. TensionCamApp_V0.2.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
  - Coming features
- A test report document.

Tests

Automatic tests are included in a separate project folder called TensionCamAppTester. To run the tests using Eclipse ADT, you should mark the project folder and choose to run it as "Android JUnit Test". Eclipse will then automatically run included tests and display the test results.

The tests can also be tested individually. To do so; expand the project folder in the package explorer, mark desired test, and choose to run it as "Android JUnit Test".

Note that the main project 'TensionCamApp' needs to be included in the build path for the 'TensionCamAppTester'.

Architecture

The application code resides in different packages and over all can the architecture be divided in three parts

- Spring MVC project (`Analyze`) for web server (package `net.codejava.analyze`)
- Android application project (`TensionCamApp`) (packages for `activities`, `utils`, `views`, `controllers`)
- External software (analyzing program) provided by the client

The reasons for using a web server are several. The main reason is 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.

Android application project

Packages:

`activitis`, `utils`, `views`, `controllers`

External libraries:

HttpClient 4.2.5

HttpCore 4.2.4

Motivation for libraries:

Enables connecting to and communicating with the web server through POST request.

Helpers are provided in the utils package. The controller consits of a class that also function as a model

Spring MVC Project

Packages:

net.codejava.analyze

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.

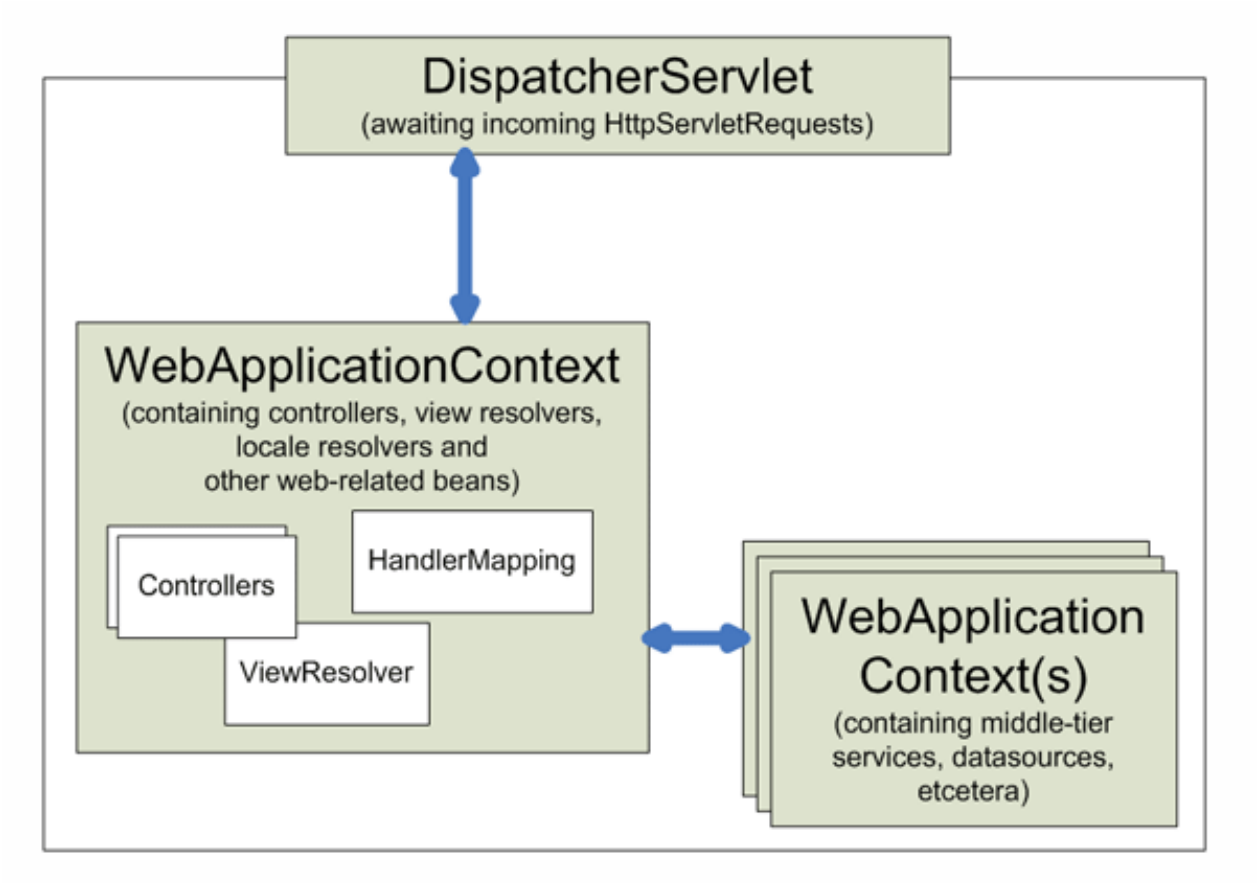

Figure 1. Context hierarchy in Spring Web MVC

Source: http://static.springsource.org/spring/docs/2.0.x/reference/mvc.html

Overview

TensionCamApp-project

doInBackground(String ... params)

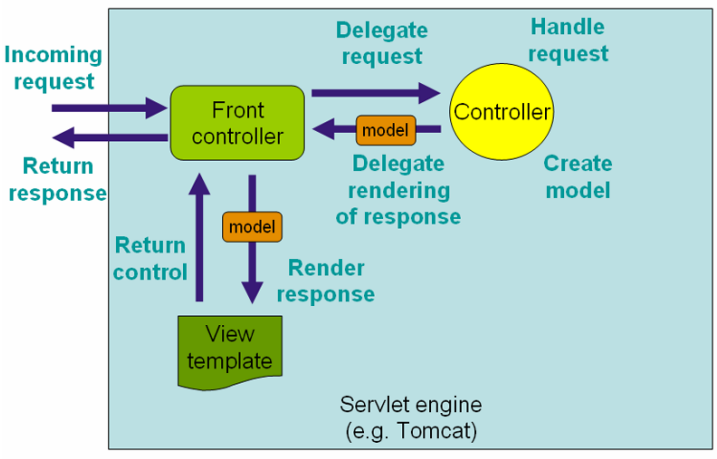

Figure 2. The requesting processing in Spring Web MVC.

Source: http://static.springsource.org/spring/docs/2.0.x/reference/mvc.html

The android application project communicates with the servlet engine through an incoming request in the doInBackground(String... params) –method in SendTask.class . The communication is done in a separate thread from application's main thread. doInBackground(String... params) – method also receives the returns method and calls onPostExecute(String result) which sets the result in the application.