September 6, 2010

Android Application Development

- A Guide for the Intermediate Developer

Degree Thesis in Computer Science

Benny Skogberg
Computer Science Department
School of Technology
Malmö University
SE-205 06 Malmö
Sweden
+46 (0)733 – 66 18 99
benny.skogberg@gmail.com

1. Abstract

The Degree Thesis in Computer Science focuses on development of Android applications to be used on a handheld device running the Android Operating System (OS). It covers the basics such as setting up a development environment, downloading appropriate tools and add-ons. The thesis serves as a guideline to intermediate developers, seeking solutions to problems not discussed in available textbooks.

Android is a platform consisting of an operating system and a Software Development Kit (SDK) for handheld devices. It originates from a small software company, acquired by Google and is now owned by Open handset Alliance (OHA), where Google is a member. A description of Android is included and its advantages and disadvantages are discussed. Resources and recommendations on Android development and Android versioning are presented. Strategies on development are also suggested throughout this thesis.

The development for an Android device and its data traffic characteristics is of interest, which is also included in this thesis. Connectivity and communication like connecting to Facebook and pulling data from an internet-connected web server, is discussed.

2. Resumé

Examensuppsatsen i datavetenskap innehåller utveckling av mjukvara för Android som ska användas av handhållna enheter som använder operativsystemet Android. Uppsatsen omfattar grunderna så som att skapa en utvecklingsmiljö samt att ladda ner lämpliga verktyg och tillägg. Uppsatsen kan användas som riktlinjer för genomsnittsprogrammeraren som söker efter problemlösningar som inte finns tillgänglig i litteraturen.

Uppsatsen innehåller en beskrivning av Android samt en diskussion om dess fördelar och nackdelar förs. Resurser och råd för Androidutveckling och Androidversioner presenteras. Uppsatsen innehåller också utvecklingsstrategier.

Utveckling för Android och dess egenskaper för datatrafik är intressanta, vilket också inkluderas i uppsatsen. Uppkoppling gentemot Facebook och annan kommunikation så som att hämta data från en webbserver behandlas i uppsatsen.

3. Contents

4. List of Abbreviations

3G International Mobile Telecommunications-2000 (IMT-2000) standards. The third generation standards family for Mobile Communication.

ADT Android Development Tool. A plug-in for Eclipse IDE

API Application Programming Interface

AVD Android Virtual Device. To be able to compile your solutions and run them as application on an emulator you need to setup at least one AVD

CPU Central Processing Unit

CRM Customer Relationship Management system

CSS Cascading Style Sheets

DDMS Dalvik Debug Monitor Server. Connects your IDE to your AVD emulator. Every Android application has its own Virtual Machine, not interfering with each other

HTML Hyper Text Markup Language

IDE Integrated Development Environment. In this thesis – Eclipse

J2SE Java Platform, Standard Edition

LBS Location-Based Service. Often used with a handheld devices' Global Position System receiver

MiB Mebibyte. 1 MiB is 2^{20} bytes equivalent to 1'048'576 bytes

OHA Open Handset Alliance

OS Operating System

SD Secure Digital. A Memory Card used in handheld devices to increase storage.

SDK Software Development Kit. A Software framework to use when developing applications.

XML Extensible Markup Language.

5. Introduction

The computer has been in constant evolution since the middle of the 20th century. Computers are continued to get smaller in size, using less power and performing more advanced calculations. In 2007 Apple released their iPhone to achieve the next goal in computing. This new type of communication tool, called Smartphone, is generally referred to as a phone, which is a poor labelling. A Smartphone is a handheld computer, which can place phone calls [5]. Although the term Smartphone was first used in 1992, Apple was the first company to release a Smartphone to a wider audience. This evolution is led by computer manufacturers and software companies and not handset manufacturers, which have controlled the market thus far [5].

One competitor to Apple iPhone OS is the Android OS. Android originates from a small software company, acquired by Google and is now owned by Open Handset Alliance (OHA), where Google is a member. OHA has over a hundred member companies such as mobile operators, semiconductor companies, handset manufacturers, software companies and commercialisation companies. Driven through the Apache License, anyone can use the Android Software Development Kit (SDK) to develop applications to run on the Android OS. Especially interesting for Android is its use of common non-proprietary techniques such as the Java programming language in combination with Extensible Markup Language (XML). This makes it open, simple and easy to use for a substantial part of the developer community.

The thesis reflects experiences of using XML, the Java Programming Language and Android SDK. Questions answered are the following:

- What is Android and how does it work using Linux, XML and Java?
- How does a developer install an Android application on an Android device, without using the Android market?
- What does a developer do to connect the Android application to Facebook Connect?
- How does a developer connect, and synchronize data, with an internet-connected server?

This thesis focuses on development of Android applications to be used on a handheld device running on the Android OS. The thesis covers the basics such as setting up a development environment, downloading appropriate tools and add-ons and shows how to get developers working with Android application development. In essence this thesis serves as a guideline to intermediate developers, seeking solutions to practical problems, who might find current literature a bit too shallow on giving answers to these questions, such as “How about the warning messages I get when compiling Android applications” and “The AVD does not start when I use a SD-card size of 2 MiB”.

Intermediate developers is in this thesis defined as developers having more than a year of software development experience at university level or more than a year software development experience as employees.

Since Android is a new software environment, there are few previous studies on this topic. Other authors have made related work in books and articles, but a study like this has, too my knowledge, never been done before. Sharon P. Hall's and Eric Anderson's article "Operating Systems for Mobile Computing" compares different mobile operating systems coming to the conclusion that Android is uniquely positioned to facilitate all features of a Smartphone device [5]. Toshihiko Yamakami comes to a similar conclusion stating "… foundation engineering and open source software significantly impacts the software engineering in the mobile platform software" [10], based on his understanding that no single vendor has the resources to develop all software (OS and applications). Jorge Agüero et al. explain that the Java Android Library is similar to the Java Standard Edition in their article "Towards on embedded agent model for Android Mobiles" [2].

Books [1] [4] [7] [9] on Android development also cover the setup of development environments on different operating systems, but they fail to discuss the essential and fundamental aspect such as role of the Secure Digital card (SD) size in the installation process. The author of this thesis has reviewed such books and has noticed gaps and shortages in various areas. The literature [1] [4] [7] [9] also has a tendency to discuss straight forward development and not the programmers' everyday quest on finding errors. The thesis focuses on useful information for developers which is hard to come by in books and articles. As the handheld Smartphone is primarily a communication device, great attention will be paid on telecommunication connectivity.

The thesis will not cover any financial aspects of Android. Although this thesis is not intended to serve as a tutorial for developing Android applications, it will however present some examples of code. This thesis has a focus on intermediate developers who wants to get an overview of Android development and find useful information in setting up the development environment and connectivity with Android as well as finding answers to compiling errors they might run into.

6. Materials & Methods

6.1 Literature Selection

The selection of books for this thesis is not based on a random selection. The author only stated one mandatory criteria of selecting books on Android development, the use of Android version 1.5.

- Android Programming Tutorials [7] by Murphy M. L. was selected to get the author quickly up to speed on developing Android applications. It was the first search result using the search string "Android tutorial" on the online bookstore Adlibris [11] December 18, 2009.
- Android Wireless Application Development [4] by Conder S. and Darcey L. was selected to get the author a more deep understanding of Android fundamentals and features. It was the

first search result using the search string “Android development” on the online bookstore Amazon [12] January 5, 2010.

- Pro Android Games [9] by Silva V. was selected to increase knowledge of graphic design, since the gaming industry relies on graphics. It was the first search result using the search string “Android games” on the online bookstore Adlibris [11] January 24, 2010.
- Unlocking Android [1] by Ableson F. et al. was selected to get the author a second book of Android development fundamentals. It was the second search result using the search string “Android development” on the online bookstore Amazon [12] January 24, 2010. The first search result had already been purchased [4].

The selection of articles [2] [5] [10] used in this thesis is based on author’s evaluation from searching in The ACM Digital Library [16] in December 2009. The criterion in the evaluation was the three most relevant articles discussing development for Android handheld devices. From a selection of 10-15 articles using search keywords; Android, Smartphone, OS and development in various combination the author found these three articles most relevant.

- New Perspectives on XML [3] by Carey P. is literature taken from course Distributed Information Systems at Malmö University [24].
- Complete Java 2 Certification [6] by Heller P. and Roberts S. is literature taken from the course Advanced Java Programming at Gothenburg University [22].
- Head First Java [8] by Sierra K. and Bates B. was selected by the highest user ratings by participant of the course Advanced Java Programming at Gothenburg University [22].

6.2 Method

This is an empirical qualitative study, based on reading above mentioned literature and testing their examples. Tests are made by programming according to books and online resources, with the explicit goal to find best practices and a more advanced understanding of Android.

One use case is presented in this thesis as a “Hello World!”-application, explaining what happens behind the scenes. The other use case is a developed application which is presented at a conceptual level.

6.3 What is Android?

> "Android was built from the ground up with the explicit goal to be the first open, complete, and free platform created specifically for mobile devices."
>
> - Ableson F. et al., Unlocking Android, page 4.

Android is an open system, and is free to use by anyone. A handset manufacturer can use Android if they follow the agreement stated in the Software Development Kit. There are no restrictions or requirement for the handset manufacturer to share their extensions with anyone else, as there are in other open source software, if they leave the Linux kernel as is. The Linux kernel is under a different and more restricted license than Android.

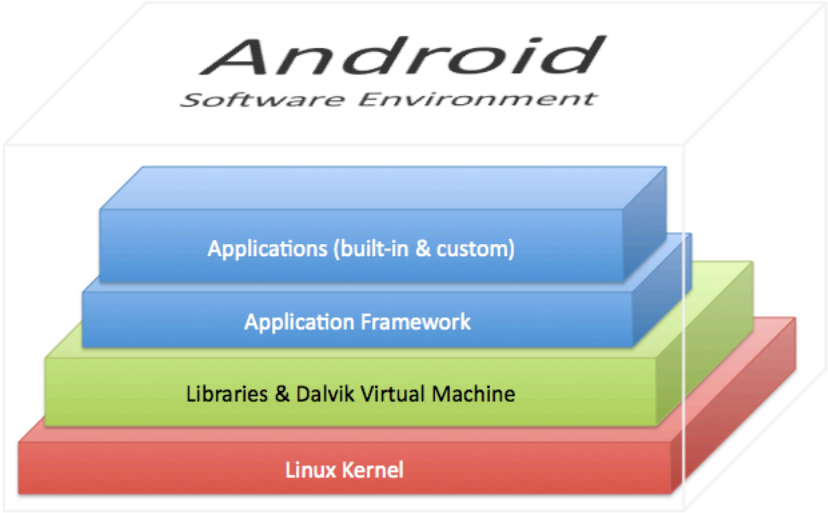


Figure 1: Android Software Environment contains a Linux Kernel, Libraries, Dalvik Virtual Machine, Application Framework and built-in and custom applications. Illustration made by Benny Skogberg.

Android is a software environment and not a hardware platform, which includes an OS, built on Linux kernel-based OS hosting the Dalvik virtual machine. The Dalvik virtual machine runs Android applications as instances of the virtual machine. Android contains a rich user interface, application framework, Java class libraries and multimedia support [15]. Android also comes with built-in applications containing features such as short message service functionality (messaging), phone capabilities and an address book (contacts).

6.3.1 Dalvik Virtual Machine and Android Applications

Every Android application runs on its own Dalvik virtual machine and developed applications are not subordinate to built-in applications. If a built-in application is poor or lacks handy features, the open source community is most likely to build a better one. One example is the built-in Messaging

application, which can be replaced by Handcent SMS (downloaded from Android market, version 3.0.4) since it displays the message received on the Android device directly, without using the notification bar scroll-down facility.

The downside of this freedom is that users have to be very careful when uninstalling applications. They could easily destroy parts of Android, making it unable to reboot. Another downside of Android is its openness, which makes it more exposed to malware [25].

6.3.2 Android Structure, Java and XML

Android is also the Java class library used to build applications for the Android software environment. Java on Android makes use of XML for variables such as strings and integers. XML is also used to control the layout and style of an application [1] [4]. This separate layer convention is similar to Hyper Text Markup Language (HTML) for content and Cascading Style Sheet (CSS) for style. There are differences due to different languages and functionalities as HTML is not a programming language as Java is. However, from a conceptual level this comparison can be made. As far as practical aspects are concerned, Java defines the button's functionality, while XML defines the buttons' text, colour, font-weight and size.

6.3.3 Android Versions

Android comes in many forms and shapes. From the first open public release in October 21 2008, Android used the version 1.1. Since then there have been versions 1.5, 1.6, 2.0, 2.0.1 to the recently released 2.1. To use Google Application Programming Interface (API), such as Google Maps, developer has a similar branch of versions, called Google API from Android version 1.5.

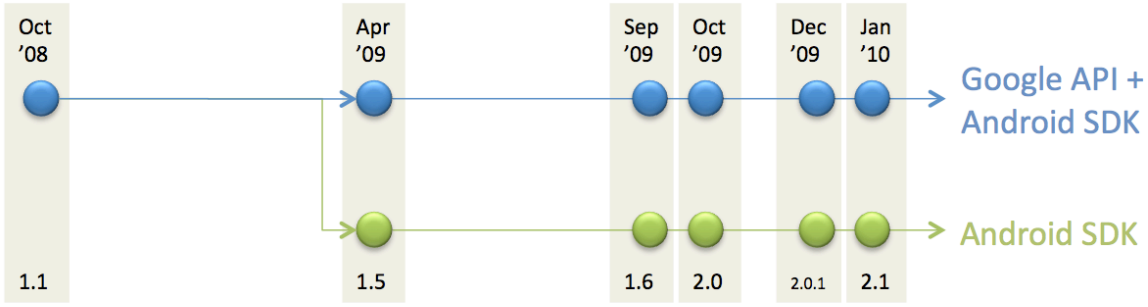


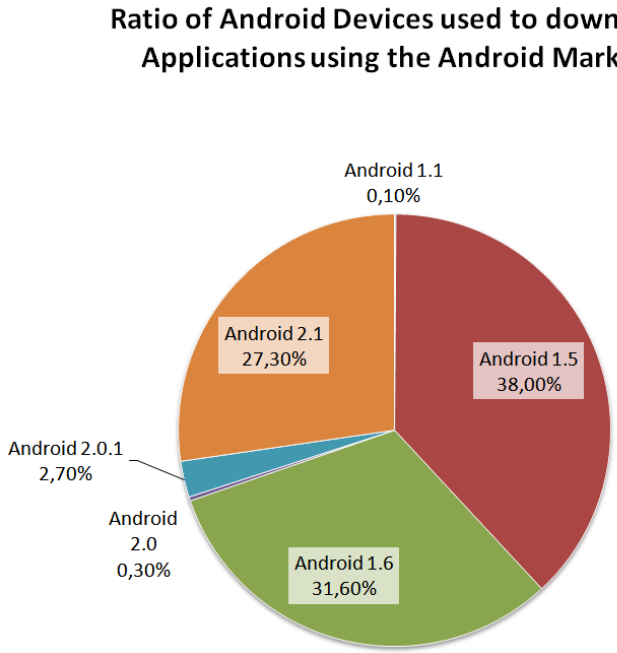
Figure 2: A timeline starting at the first open public release of Google API and Android SDK, the split of the two in April 2009 until its latest release version 2.1 in January 2010. [30]

From a developer's point of view, this is important. Considering the fact that there are six different versions of Android today and most certainly even more to come in the future, a developer needs to master the different versions of Android. The challenge in building an application that works well on all different versions is one of the main issues with Android. One approach is to use the lowest version possible for your application, making it accessible to higher versions. The other approach is

to target each version of Android, but this approach is costly. Depending on stakeholder needs in terms of time to market and financial means, choice of Android version is important.

The RunKeeper application, takes the latter approach, and was recently released to the Android market in version 2.0. However, RunKeeper themselves state that the application will be supported by earlier versions "in the near future" [27]. The design decision is in any case depending on a solid and thought business decision, which impacts the whole development process. If not, the developer is advised to address this to the stakeholders and project management.

To keep track of the currently used versions of Android, one option is to use the Android website and turn your attention to the resources tab and navigate to Platform versions. There the people interested can find by which Android version users have downloaded Android applications. It is a rather blunt tool, showing you the last two weeks of download preceding a certain displayed date in the Android Market. The pie diagram gives a quick overview by which versions Android users download applications on a global scale. If your target market is not global, consider carrying out a survey of your own, matching your target audience.



ıre 3: Ratio of Android Devices used to download Applications using Android Market March 29 to April 12 2010 [14].

Even though the pie chart is a rough tool to use, the developer can get a quick overview of which versions to target or to leave out. Developing a global application today the highest version to use would be Android 1.5, since almost 4 of 10 devices downloading applications from the market are using this version. Android version 1.1 is almost obsolete since only one out of a thousand use this version. Android versions 2.0 and 2.0.1 have not got that many downloads either, and can rely on version 1.6.

The author suggests that a good strategy to develop an application published in the market today would be to first release a version 1.5 followed by versions 1.6 and 2.1, leaving out version 1.1, 2.0 and 2.0.1. If the stakeholder only wants one Android application, version 1.5 would be a good choice.

6.4 The Development Environment

Android SDK makes use of Java programming language, similar to Java Standard Edition (J2SE), called Java Android Library. This is an advantage to developers familiar with programming languages originating from the programming language family C. The syntax is the same as Java in terms of operands, selections, iterations, file handling and more. The more specific Android classes and packages use other names that are not similar to Java editions, such as the Activity Class and the View Class [4].

To develop an Android application, developers need to make sure that the development environment has a Java version 5 or above. Today Java 6 is stable and there is really no reason not to use it. Download and install a Java version appropriate for the developing computer OS, since Java is OS independent, developer could choose developing environment operating system freely [1].

6.4.1 Android SDK

The Android SDK is mandatory to Android developers. It contains all the packages, application framework and class libraries the developer needs to develop an Android application. The developer should download and install the Android SDK and set up the PATH environment variable. The PATH variable is used by the developing computer's OS to locate the Android SDK and is essential for developing software anywhere in the developing machine's file system [4].

6.4.2 Eclipse IDE

Even though Eclipse Integrated Development Environment, IDE, is not necessary for Android developers, the author recommends developers to use it. Today it is the only IDE which supports Android development, and makes development more straight forward and thus - quicker. It is possible, as in most other programming languages, to use a simple text editor for development. However using a text editor is time consuming and sometimes frustrating for developers when the

lack of a semi colon (;) or a misspelling is not highlighted. Furthermore Eclipse has a handy Android Development Tool (ADT) plug-in which is a powerful extension to Eclipse. It lets the developer use all the basic tools any full featured IDE uses [1] [4]. To mention a few, adding breakpoints and check variables at a specific time within the programming code, access Dalvik Debug Monitor Server (DDMS) and use the project wizard are some of the features.

6.4.3 Android Virtual Device

Every version of Android has a different Android Virtual Device (AVD) which you have to set up first. The general idea is to develop to the lowest Android version possible, to get the developed application as accessible as possible to a wider range of Android users. Not every Android user updates version frequently and there are some handset manufacturers that do not support new versions on previously sold Smartphones. The conclusion for the developer is to make informed decisions on appropriate Android versions, never using a version too high if not necessary. Developer should keep in mind an Android application in a lower version is forward compatible to a higher version [4].

The AVD can be set up using console application or the developers recently installed Eclipse ADT plug-in. For convenience, developers are advised to set up all possible virtual devices from the very beginning. This approach makes development effort simpler later on, switching from one version of virtual machine to another, especially in test purposes. The ADV is an emulator which actually contains the specific Smartphone OS [1]. This is convenient if the developer does not have a device of their own. Nonetheless the developer should be careful to release an application that has not been tested on an actual handset. The more different handsets and versions the developer tests their application on, verifying functionality and purpose of application, the better. Releasing a fully functional, well-tested and stable application to the Android market will increase user ratings getting the developed application a better market placing. The default view of the Android market application suggestions are presented in descending order by popularity, based on the number of downloads and user ratings.

6.4.4 Secure Digital Card

Within the literature [1] [4] [7] [9] there is not mentioned what size the SD-card should have. The author tried with 2 Mebibyte (MiB) and failed. After substantial searches, the author later found out that anything less than 8 MiB would fail. One MiB represents the value of 2^{20} bytes which equals 1’048’576 bytes of digital storage. Therefore the author suggests that the developer to use the double size 16 MiB to avoid trouble with the AVD in the development mode. This size corresponds to 16.384 Megabytes (MB).

6.5 “Hello World!” as Test of IDE

Following the steps in the previous section, the developer is set up to start developing her first Android application. The first application in any programming language is the “Hello World”-

application which basically prints out the text “Hello World!” to the user interface, in this case – the AVD emulator.

Android uses XML and Java in close conjunction, which might feel confusing at first. The files and folder structure is very similar to any other Java application, with a few differences.

AndroidManifest.xml is placed in the root folder (Figure 4) and is a description of the application, its permissions and capabilities [4]. Permissions are used to enlighten the user of the application and what capabilities the application uses. If the application requires Internet access or GPS, the user will be notified when installing or updating the application. It is mandatory to specify what capabilities the application uses and is one of the things a developer agrees upon when downloading the Android SDK [1] [4].

FirstApp.java is located in the java source folder src/apt/tutorial meaning that the package name is apt.tutorial. When building applications that would be released to the Android market, the naming convention is com.android.<company name>. The java source file (or files) contains all the functionality of the application [1].

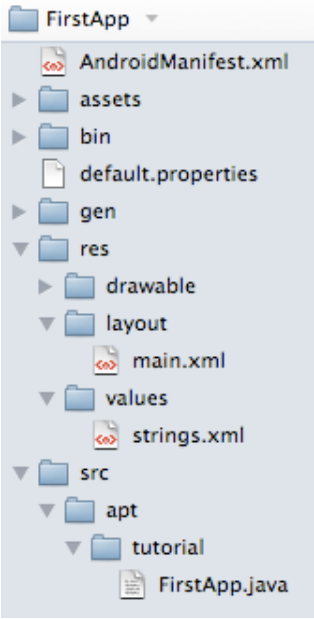


Figure 4: Folder structure of the Hello World application in Android.

The resource folder, res, contains three different folders; drawable, for images and pictures; layout, for the positioning and size of different elements such as buttons, text fields and the screen view, and values for element names and colour.

The AndroidManifest.xml (Figure 5) is a simple XML file with an XML schema namespace called android. It defines the java package, which icon to represent the application (in this case a file

called icon.png is located in the res/drawable folder) and the application name at line 6. Application name and icon represents the application in the Android device application list. On line 16 the minimum SDK version is specified, in this case the SDK version 3, which is equivalent to Android 1.5. When the application is started, wrapped by element intent-filter, "action android: name" (line 10) is where the Android device finds which file defines the application setup, in this case main.xml.

```
1  <?xml version="1.0" encoding="utf-8"?>
2  <manifest xmlns:android="http://schemas.android.com/apk/res/android"
3        package="apt.tutorial"
4        android:versionCode="1"
5        android:versionName="1.0">
6      <application android:icon="@drawable/icon" android:label="@string/app_name">
7          <activity android:name=".FirstApp"
8                    android:label="@string/app_name">
9              <intent-filter>
10                 <action android:name="android.intent.action.MAIN" />
11                 <category android:name="android.intent.category.LAUNCHER" />
12             </intent-filter>
13         </activity>
14
15     </application>
16     <uses-sdk android:minSdkVersion="3" />
17
18 </manifest>
```

Figure 5: AndroidManifest.xml from the Hello World application [7].

The main.xml file (Figure 6), also specifying android namespace defines a linear layout, the orientation (line 3) the layout width (line 4) and layout height (line 5). Orientation can be vertical, horizontal or dependent on which way the Android device itself is oriented horizontal or vertical. The TextView, lines 7-11, is an element, which is used to fill with text, in this case pointing to the attribute hello in the element string.

```
1  <?xml version="1.0" encoding="utf-8"?>
2  <LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
3      android:orientation="vertical"
4      android:layout_width="fill_parent"
5      android:layout_height="fill_parent"
6      >
7  <TextView
8      android:layout_width="fill_parent"
9      android:layout_height="wrap_content"
10     android:text="@string/hello"
11     />
12 </LinearLayout>
```

Figure 6: main.xml from the Hello World application [7].

Android uses XPath [3] to locate the attribute value hello (line 3) of element string in the file strings.xml (Figure 7). The value of this element is Hello World! On line 4, the attribute value app_name is specified as First App referenced by AndroidManifest.xml.

```xml
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <string name="hello">Hello World!</string>
    <string name="app_name">First App</string>
</resources>
```

Figure 7: strings.xml from the Hello World application [7].

The java-file FirstApp.java (Figure 8) is specified as an ordinary java-file beginning with the package apt.tutorial (line 1). On line 3 and 4 this application imports the Activity class, and the Bundle class. The class FirstApp (lines 6-14) inherits the Activity class. Line 8 and 9 onCreate-method overrides the Activity class onCreate-method, taking the Bundle type variable savedInstanceState as argument.

The saved instance state is a reference [6] to a place in memory where the application ran previously. To this application it has no real value, but in lager applications it is crucial in terms of multitasking. Android devices have limited resources, and are unable to run unlimited number of applications at the same time. To mimic true multitasking, the application saves the instance state to memory, when sent to background. The OS needs not to kill the application instantly, but if the user continues to open other applications to a point close to the resources limits, the OS terminates the application. If the user returns to a killed application, its instance is still saved in memory; thus, the OS starts the application again and grabs the saved instance state giving the impression the application were never killed [13].

Continuing on line 11 the Activity class's onCreate-method is called with the savedInstanceState as argument. Last on line 12, the content view is set with argument R.layout.main. R stands for the resource folder res; layout is the subfolder and main the XML-file (Figure 6).

```java
package apt.tutorial;

import android.app.Activity;
import android.os.Bundle;

public class FirstApp extends Activity {
    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState)
    {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
    }
}
```

Figure 8: FirstApp.java from the Hello World application [7].

When the developer runs the application, the emulator most appropriate for the minSdkVersion starts as an actual Android device. It takes the same time as starting an actual Android device, so the developer is advised to keep the emulator running when developing applications for Android,

minimizing development time. It is possible to uninstall the application from the emulator when the test is complete.

Be advised that Eclipse will generate two equal and unfortunate warnings, which from a developer's perspective is annoying and unimportant. The warnings say "Warning once: This application, or a library it uses, is using NSQuickDrawView that has been deprecated. Apps should cease use of QuickDraw and move to Quartz". The developer should not be alarmed by this deprecated message; it is just a warning that your environment is using a technique which would not be supported later on. However – running the application on a real Smartphone device does not generate these errors.

If the setup of development environment is correct, the "Hello World"-application will be up and running as is shown on the emulator in Figure 9.

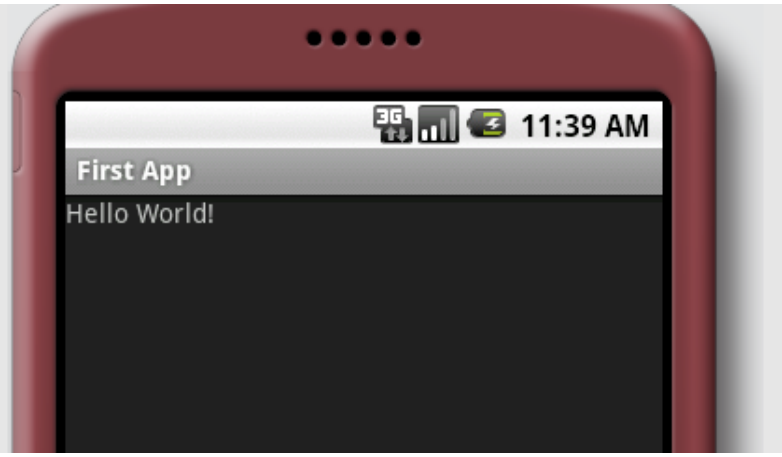


Figure 9: Emulator view when the compiled application runs from the Hello World application [7].

6.6 Test Application

The purpose of the test application is to synchronize locally stored information with information stored on a remote server. Information edited on the Android device will be synchronized with the server, and synchronized from the server to other Smartphone devices. The registered user shares data and synchronizes its information in the background with pre-determined intervals, thus always having accurate and updated information. The user does not need to know when information was last edited since all information is updated according to the pre-determined time interval.

This application has the following main features:

1. Connect to an internet-connected server which stores information
2. Sign in to a server using Facebook connect
3. Sign in to a server with username and password

To connect to the server the Smartphone needs to have access to the Internet through 3G. While connected to the server, the application should synchronize current application information with the information on the server where the most recent changes are updated, overwriting the older data.

To be able to use Facebook connect, a user needs to have a Facebook account. The Android application needs to facilitate a Facebook Developer application, set up within Facebook. Setting up the application on Facebook is done using the Facebook Developer guidelines [18]. The Facebook application is a framework where the developer can embed content from another server. Once installed, the Facebook application needs not to change since the actual application, framed by Facebook, is hosted on another server. Connecting to Facebook Connect lets the user sign in with their Facebook account, sending an authentication token to the host server (the developer controlled server), and verifying the user to the account on the internet-connected server. Once signed in, the synchronization process starts. Users not having a Facebook account or users unwilling to use a Facebook account for signing in, should also be able to sign in to the server using a username and a password.

The information container features are extended with more smart functions. It should make use of user controlled settings, like synchronization interval and notifications. The Android application will extend using instant, letter by letter, search result view finding contacts immediately. An option can be using a Location Based Service (LBS) to be able to use 'friends last seen' functionality using Google Maps.

6.7 Developing Strategy

There are different techniques and approaches for developing an application in general. The author prefers to take on the requirement valued as the most difficult one first. The reason for doing so is to get a clear view of whether it is possible at all to meet stakeholder's requirements. If the developer succeeds, the developer has a head start and can be certain that the whole project is possible to develop. If the developer fails, the developer can, at an early stage, address the stakeholder of the difficulties and not lull the stakeholder into a possible false security. If the author finds different requirements having the same value of difficulty, the one most valuable to the stakeholder is chosen.

There are other strategies too, such as a "quick win". A "quick win" is a function having a high value to the stakeholder, and one which is easy to develop. In this case one could develop the user interface first, implementing only a part of the application but having the look and feel of the application at hand. This has apparently a high value to the stakeholder, is easy to implement, and gives a feeling for the application's end result [31]. However, this strategy gives false hope to the stakeholder and an unwanted pressure on the developer to finish the application. To use this strategy, the author suggests only showing the features and functions that are fully developed from the user interface to the server and from the server to the user interface.

When developing applications, constant communication with the stakeholder is also an essential part of the strategy, letting her see the process as open and uncovered. The developer gains trust and needs not to explain to the stakeholder if the developer should fail. The stakeholder should always be made aware of the development roadmap.

7. Result

7.1 Facebook Connect

Complementing the username/password login task the application needs to make use of the Facebook connect feature. Today it's seen on sites within the location-based social media sphere, such as Foursquare [20] and Gowalla [23]. The general idea with Facebook connect on a handheld device is to have your sign-in process just a click away, to improve usability. It can be a daunting task typing long usernames and obscure passwords, especially on a small-screen Smartphone with a touch control keyboard.

7.1.1 Facebook Application

On Facebook, the developer needs to set up a Facebook application which starts with adding the application Facebook Developer to the developers account. With Facebook Developer attached to the developer's account the developer can develop numerous of Facebook Applications. The author created his own application [19] within Facebook using a guide [17]. The Facebook application linked to the author's own host [28]. This application just displays two images side by side, and has no useful value of its own. But if the developer wants to add more content or features, Facebook allows it. Used as a Facebook application, the user not signed in to Facebook is unable to see the application. Also, the user has to give the application permission to pull data from the users account.

At this development stage you have your Facebook Application in place as a frame to your own host web page. Three different Facebook users also tested this setup, to verify the application worked as planned.

The third step is to access the Facebook Application using the Android Smartphone, to complete the Facebook connect function. Facebook Connect for Android is hosted by Google Code [21] and is a port of the Facebook Connect for iPhone. A port is a way of changing a software to make it usable for other software environments. In this case it only uses the variables API key and API secret to connect you to the correct Facebook application. The variable GetSessionCookie is also defined by Facebook Connect for Android, but not used in this application. If you were to use the actual Facebook Application to more people than just sign in users, the variable GetSessionCookie is required to keep the Smartphone connected to the Facebook Application.

7.1.2 Facebook Connect for Android

To get the Facebook Connect for Android to work properly, developers need to disable any IDE presets, such as JUnit test, because this interferes with the Eclipse Android project when compiling the application. Equally important is the fact that the developers need to make sure they use Java version 6 when developing a Facebook Connect application. Mac OS X (10.5) has Java version 5 as standard, which requires that the developers need to manually update to Java version 6.

When compiling the Android project, the emulator starts and the application is installed on the emulator. Unlocking the emulator, the application starts, looking like an actual Android handheld device. On start up a Facebook Connect button is visible and when activated you get the usual sign-in form asking for username and password, since this emulator has not been connected to Facebook before. Signing in with user credentials, the user is now actually connected to the Facebook application and has the option to post a message on the user's Facebook Wall using the developed Android application.

A conceptual view of the Facebook Connect for Android is in three layers. At the top layer, the user handles her Smartphone which connects to the Facebook, the middle layer. The Facebook Application uses your user credentials and connects to the host, the bottom layer. The application images are not present, since the Facebook Connect does not support the Android WebView library.

The developer, as always, needs to use the correct Android version, and the correct AVD emulator version. In January 10, 2010 Facebook Connect for Android was updated from version 1.5 to 1.6.

7.2 Networking

When the Facebook Connect is working properly, the next step is to connect to the server. In a general way, this is the same as connecting to any other server using the Hyper Text Transfer Protocol (HTTP). Having your user credentials through Facebook Connect the application just needs to connect to the server.

Since we are accessing data that is supposed to be concealed to the public and only accessible to authorized users, we need to secure our network connection using the Secure Socket Layer (SSL). Using SSL minimizes the risk of any unauthorized use of the private data.

It is possible to build the application in a simple sequential way; however this technique makes the application slow. When connecting to a resource on the Internet, the probability of failure increases. First, you have the Android device connected to a 3G-network, which might not have the best signal strength at all times. Secondly the Internet resources could be occupied with numerous requests from other sources or all together unavailable. A recommended approach for the developer is to make use of threads when connecting to sources on the Internet [4]. Threading in programming is used to make an application do several different things simultaneously, such as handling user interface requests and at the same time download data from the Internet.

From a conceptual view, an application can work with multiple threads at the same time, keeping the User Interface idle and functional to the user. It is also recommended to give the user the

option to cancel current operation, when the user chooses to do so for whatever reason. The downside, from a developer's perspective, is that it is impossible to force a thread to start. When calling a thread's start-method, the developer makes the thread eligible to run. The thread scheduler determines when the Central Processing Unit (CPU), has enough capacity to handle the thread just started [6].

When running threads, it is a good idea to have an event listener attached to the Internet connected thread, since we want the user to be able to cancel the operation. This technique is also used in the opposite direction. We want to tell the user when we have pulled all the required data from the server, thus we make use of callbacks [8].

The author has experienced a lot of Android applications, downloaded from the Android market, used on an Android device which has been forced to quit by the Android OS. The main reason for this is not known (since source code of applications is concealed), but one technique to minimize this kind of behaviour is to make use of exception handling where this is appropriate. Validating user input is not appropriate for exception handling, but using exceptions connecting to a resource to the Internet is. The developer should make reasonable judgment on exceptions, since it slows down the application. The general advice is to use exceptions only where the application is likely to crash for reasons that are out of control and cannot be hindered by simpler programming constructs. Therefore, we implement exception handling using try-catch statements on connections to the Internet.

7.3 Data Synchronization

Having an authorized user through Facebook Connect and downloaded data from the server, we need to recognise which data is most accurate. All locally stored data is contained in a collection of objects. The same goes for the downloaded data from the Internet.

The next step is to compare the objects from the local list of objects to the downloaded list of objects. Each object has a unique identity and a timestamp, which determines which object was edited last. Comparing time-stamps of the same identity determines which object has precedence over the other. If the object on the server is newer, the local object is replaced by the one from the server.

Iterating through the list in the above-described fashion, keeps the data accurate and synchronized with the data on the server. Since there is only one user who updates her data, there is no need to check if another user made any change.

Since server storage is set up in a way that only allows downloading all objects, improvements can be made. A suggestion is to allow the Android application to compare each object's identity and timestamp before downloading the entire object. This would not only increase usability in terms of a faster application, but also decreases network load.

8. Discussion

Android Software Environment was first released as an open source project in October 2008. The reason for making the source code available to the open community is not revealed even though the goal is stated on Open Handset Alliance website [26]: "Android was built from the ground up with the explicit goal to be the first open, complete, and free platform created specifically for mobile devices". Yamakami [10] takes the approach towards the cost of keeping and maintaining proprietary mobile operating systems. The author extends Yamakami ideas to include an even closer and tight connection with the users of Android. There are many potential developers, hobbyists and professionals, who use Android in their daily lives. Instead of waiting for the next release of preferred handset manufacturers' improvements, the developer can herself add features to existing applications or write new ones, if the developer feels something is missing. "Time to market" is just a handmade application away.

The openness of Android makes it more exposed to various malicious codes, than controlled proprietary mobile operating systems. If a substantial number of Android devices are infected by virus or other malicious code, it could jeopardize the whole Open Handset Alliance's effort to stay open. There are reports of bank fraud from applications downloaded from the Android Market, which is not as guarded as Apples AppStore. With the growing popularity of Android, criminals is more likely to pay more attention to Android applications and Android Marked [29].

Due to the rapid pace of new releases of Android versions, developers need to pay attention to the development of versions as well as the current downloads made. Existing applications need to be updated to newer versions, as users might not feel comfortable with older features on newer Android devices. As in other software environments, it is most likely that Android will also mature as a programming language, making it more interesting for the corporate world and not just the consumer market. A sales person might want to have an application connecting to the company's Customer Relationship Management system (CRM) having access to all to marketing valuable data within hands reach. A Supply Chain Manager might want to check a warehouse inventory levels or the one time delivery report. Possibilities are endless with Smartphone's.

From a developer's perspective, the use of common programming techniques is recommended. The use of Java in combination with XML makes programming for Android fairly simple for the intermediate developer. The advantage is the possibility to use skills and techniques learned from other programming languages. The disadvantage with Android from the developer's perspective is the rapid pace of new versions, which puts a strain on the developer to continuously follow the evolution of Android. Android is still in its early stages, and when it matures no one can tell.

9. Conclusion

This thesis has shown what Android is and how it works with XML in combination with Java. It has described how to setup a development environment and the emulator (AVD). It has showed what

textbooks [1] [4] [7] [9] did not focus on, such as how the binding between XML and Java work on Android.

Issues of versioning of Android and its rapid evolvement in terms of new SDK's have been discussed as well as how developers should address these issues. The thesis has also discussed the advantages and disadvantages with Android software environment, and has shown several tips for the intermediate developer.

The author's understanding is that Android and its SDK is a feature of the future. In part due to the fact that it will be fun for must phone users to be able to add, develop and equip their phones with new features and personal preferences, it will open a new era in sharing open source software components, such free utilities and games, for mobile devices. Furthermore, Android has recently arrived, and the author is convinced that the development environment and tools will be improved and enhanced in the future, making the development process effective.

10. References

[1] Ableson F. and Sen R., Unlocking Android, 2009. Manning Publications Co., ISBN 978-1-933988-67-2

[2] Agüero J., Rebollo M., Carascosa C. and Julián V., Towards on embedded agent model for Android mobiles, 2008. International Conference on Mobile and Ubiquitous Systems, Article no 37, ISBN 978-963-9799-27-1

[3] Carey P., New Perspectives on XML, 2007 2nd edition. Thomson Course Technology, ISBN 978-1-4188-6064-6

[4] Conder S. and Darcey L., Android Wireless Application Development, 2009. Addison-Wesley, ISBN 978-0-321-62709-4

[5] Hall S. P. and Anderson E., Operating Systems for Mobile Computing, December 2009. Journal of Computing Sciences in Colleges, ISSN:1937-4771

[6] Heller P. and Roberts S., Complete Java 2 Certification, 2005 5th edition. Sybex, ISBN 0-7821-4419-5

[7] Murphy M. L., Android Programming Tutorials, 2009. CommonsWare, ISBN 978-0-9816780-2-3

[8] Sierra K. and Bates B., Head First Java, 2005 2nd edition. O'Reilly, ISBN 978-0-596-00920-5

[9] Silva V., Pro Android Games, 2009. Apress, ISBN 978-1-4302-2647-5

[10] Yamakami T., Foundation-based Mobile Platform Software Engineering: Implications to Converge to Open Source Software, 2009. ACM International Conference Proceeding Series; Vol. 403. ISBN:978-1-60558-710-3

10.1 Internet References

[11] Adlibris bokhandel, http://www.adlibris.com, retrieved December 18, 2010

[12] Amazon, http://www.amazon.com, retrieved January 5 and 24, 2010

[13] Android, Androidology - Part 2 of 3 - Application Lifecycle, http://developer.android.com/videos/index.html#v=fL6gSd4ugSI, retrieved May 4, 2010

[14] Android, Platform versions, http://developer.android.com/resources/dashboard/platform-versions.html, retrieved April 30, 2010

[15] Android. What is Android? http://developer.android.com/guide/basics/what-is-android.html, retrieved March 4, 2010

[16] Association for Computing Machinery, http://portal.acm.org, retrieved December 18, 2010

[17] Facebook, Creating a Platform Application, http://wiki.developers.facebook.com/index.php/Creating_a_Platform_Application, retrieved February 17, 2010

[18] Facebook, Facebook developers, http://developers.facebook.com/, retrieved February 17, 2010

[19] Facebook and Skogberg B., MyFBDevApp, http://apps.facebook.com/myfbdevapp/, retrieved February 17, 2010

[20] Foursquare, http://foursquare.com/login, retrieved April 30, 2010

[21] Google, Facebook Connect for Android, http://code.google.com/p/fbconnect-android/, retrieved February 17, 2010
[22] Gothenburg University, Advanced Java Programming, http://www.utbildning.gu.se/ViewPage.action?siteNodeId=427372&languageId=100000&contentId=-1&courseId=info.uh.gu.TIG075 , retrieved January 30, 2010
[23] Gowalla, http://gowalla.com/, retrieved April 30, 2010
[24] Malmö University, Distributed Information System, http://www.edu.mah.se/DA164A/syllabus/, retrieved April 30, 2010
[25] McAllister, N., Android malware: How open is too open? Infoworld, http://www.infoworld.com/d/developer-world/android-malware-how-open-too-open-784, retrieved April 30, 2010
[26] Open Handset Alliance, Overview, http://www.openhandsetalliance.com/oha_overview.html, retrieved April 30, 2010
[27] RunKeeper for Android, http://runkeeper.com/android, retrieved April 30, 2010
[28] Skogberg B., MyFBDevApp, http://www.skogberg.eu/myfbdevapp/facebook-platform/myapp/, retrieved February 17, 2010
[29] Virus Experts, Banking malware found on Android Marketplace, http://www.virusexperts.org/security-news/banking-malware-found-on-android-marketplace/, retrieved April 30, 2010
[30] Wikipedia, Android (operating system), http://en.wikipedia.org/wiki/Android_(operating_system), retrieved April 30, 2010

10.2 Lecture References

[31] Hagert, G., Lecture in Information Design: Quality, March 4, 2008