



Contents

- [1 Introduction](#)
- [2 Overview of Android Platform](#)
- [3 Overview of Qt S60](#)
- [4 Examples](#)
- [5 API Mappings](#)
 - ◆ [5.1 File I/O](#)
 - ◆ [5.2 Networking](#)
 - ◆ [5.3 Media](#)
- [6 Android vs Qt for Symbian](#)
- [7 References](#)

Introduction

As an initiative of [The Open Handset Alliance](#) and [Google](#), the Android has been developed as the first open and free platform for mobile phones. The project have been developed since November 5, 2007, when Google Inc., Intel, T-Mobile, Sprint, HTC, Qualcomm and Motorola decided to archive an ambitious goal: provide services so consumers would face a far better user experience.

An important point of Android platform is that handset manufacturers and wireless operators can provide suitable versions of Android for their products and services. This characteristic directly impacts on lower cost and innovative products. Then, Android can be considered a new and potential option on mobile platforms for smartphones.



Porting_Android_(Java)_applications_to_Qt_for_Symbian

On the other hand, Qt has been considered as a powerful component on this newest mobile programming arena. Qt is most notably used in Desktop applications, such as KDE, Opera, Skype and VirtualBox, but recently it has been ported to Nokia mobile platforms, such as S60 and maemo. Initially developed by Trolltech since 1991, Qt is released on two different licenses (open source and commercial), which should make Qt more suitable for non-GPL open source projects and for commercial users. In June 2008, Nokia acquired Trolltech to enable the acceleration of cross-platform software strategy for mobile devices and desktop applications. On September 29, 2008 Nokia renamed Trolltech to Qt.

This article provides information that guide the development to port Android applications to Qt for Symbian.

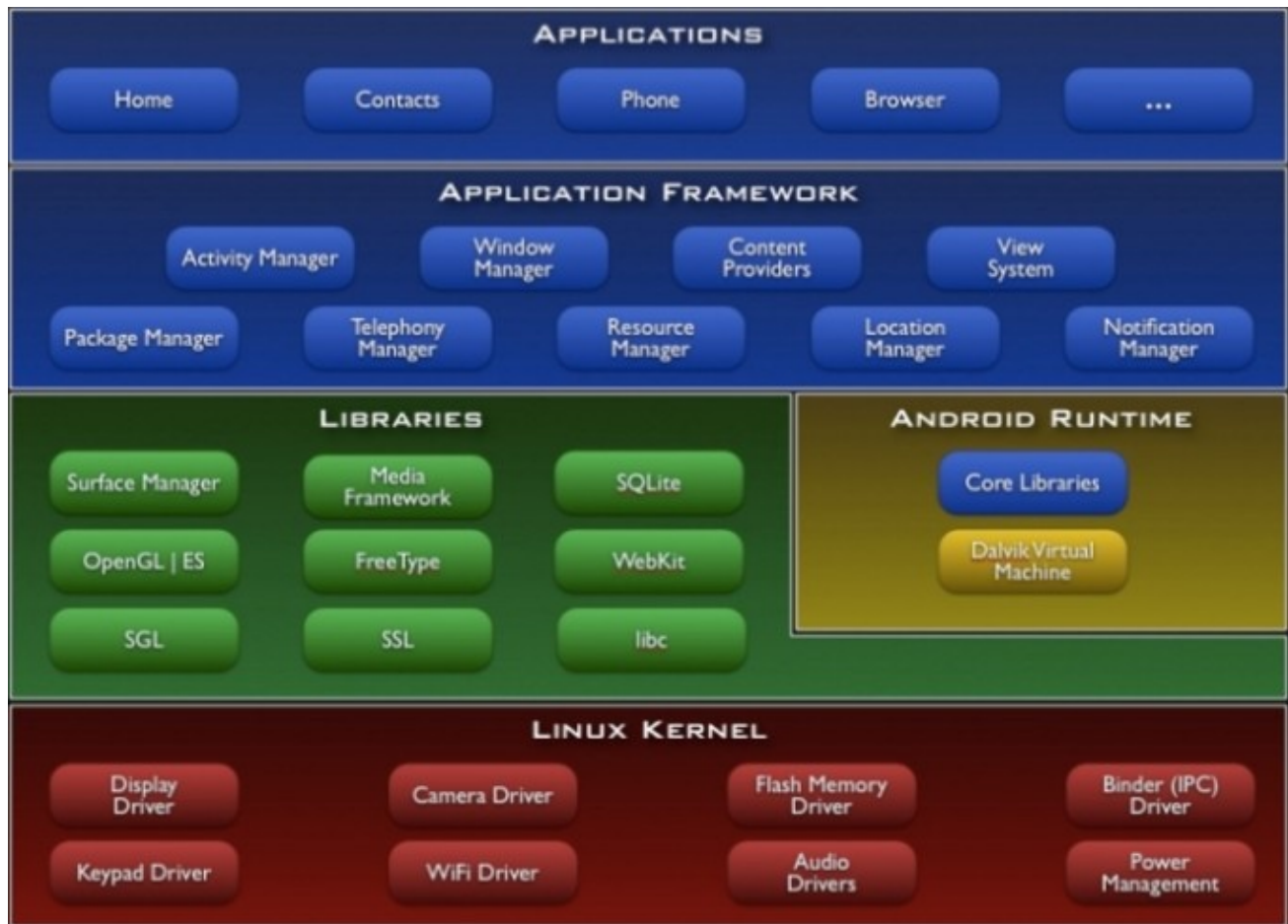
Overview of Android Platform

Android applications are develop with Java programming language (Dalvik virtual machine) and device services, such as touchscreen and storage features, can be access through Google services API. It is possible to run applications written in C or any other language, but the application have to be compiled to native code and run, but this development path is not officially supported by Google.

Since October 2008, Android has been available as open source project (Apache License). Then, handset manufacturers and wireless operators are free do add closed and proprietary extensions to their products.

Although Android is based on a Linux kernel, according to Google, it is not a Linux operating system. In addition, it does not have a native windowing system, nor supports the full set of standard Linux libraries, including GNU C Library. This characteristic make it difficult to reuse existing Linux applications or libraries. Android does not use standard Java APIs, such as J2SE and J2ME. This prevents compatibility among Java applications written for those platforms and those for the Android platform. Android only reuses the Java language syntax, but does not provide the full-class libraries and APIs bundled with J2SE or J2ME.

The next picture shows the current Android architecture (from Android Developer's Guide).



The system has access to mobile phone resources through system drivers, such as camera, display, WiFi and keypads drivers. Then, the next layer consists of libraries and runtime system of Android. Finally, Android provides a set of libraries (Application Framework) so it is possible to extend them and create new applications.

Android makes it possible to reuse other components of other applications. For example, if you need to reuse a component that provides a suitable scroller and made it available to others, you can invoke such component to do the work for you. Therefore, the system has been designed to start an application process when any part of it is needed, and instantiate the Java objects for that part. Then, Android does not provide an entry point, such as main function, but essential components: activity, services, broadcast receives and content providers.

Activities represent screens on an Android application. From an activity, you can display buttons, labels, menus and much more. All activities subclasse *android.app.Activity* class. **Services** do not have visual appearance, but they runs on background. For example, a service play a music while the user perform other tasks. Each service extends the *android.app.Service* base class. **Broadcast receives** are components that receive and reacts to different broadcast announcements, for example, a message that the battery is low. All receivers extend the *android.content.BroadcastReceive* base class. A **content provider** are responsible to provide application's data available to other applications. Then, with content providers, it is possible to share data between different applications. All content providers extend the *android.content.ContentProvider* base class.

Android development environment consists of: Android SDK, Android source code and, optionally, IDEs that helps programming Android applications a lot quicker. Android software development kit (SDK) consists of

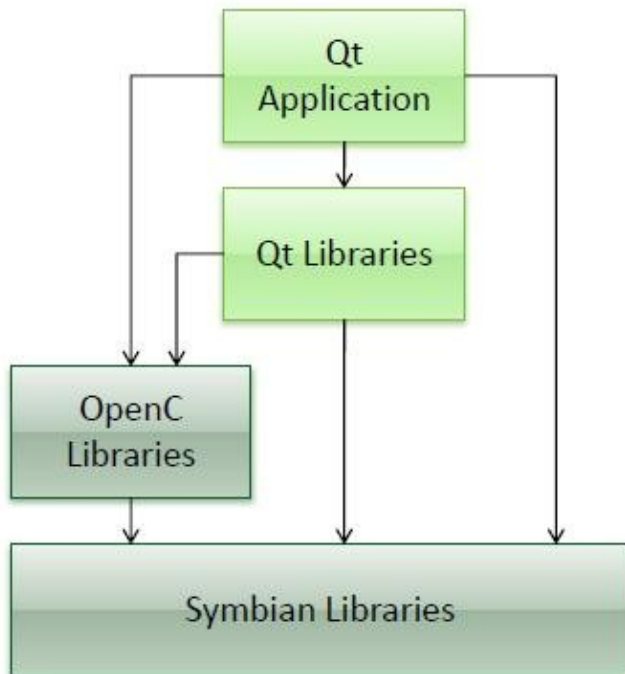
libraries and tools, including an emulator to run applications. The Android SDK is available for Windows, Mac OS X, and Linux. There are different IDEs that provides support for Android development, such as Eclipse (Eclipse plug-in for Android).

Overview of Qt S60

Qt applications are developed with C++ with several non-standard extensions implemented by an additional pre-processor that generates standard C++ code before compilation. Qt also provides bindings for several other programming languages, like Python, Ruby and Perl.

Qt is notably used because of its GUI widget, but also provides a set of non-GUI related features: SQL database access, XML parsing, thread management, network support and a unified cross-platform API for file handling. Qt has being successfully ported to S60 Nokia platform. Qt for Symbian will enable you to create advanced applications with innovative user experiences while getting to market quickly?. Compared with Android, Qt is not supposed to be used as platform for mobile phones, but an interesting and easy-to-use programming framework that certainly brings a considerable contribution to mobile programing.

Considering the fact that Qt has been recently ported to S60 platform, there are still some APIs that are not implemented such as camera and contacts modules. However, it is possible to access such system services thought callbacks to Symbian applications.



In order to develop Qt Symbian applications, you need to install Qt S60 Nokia SDK, with consists libraries and tools (including the emulator for S60 environment). In addition, Carbide++ IDE (based on Eclipse framework) provides a full feature environment that helps a lot the developers. You can found interesting articles on Forum Nokia Wiki about Qt for Symbian (see [Category Qt for Symbian](#)).

Qt for Symbian applications are easily built on Carbide IDE and they can be launched on S60 emulator or even on your device (you have to install Qt for Symbian libraries first). The entry point of any Qt application

is the method main (remember that Android application starts from an Activity instance).

Examples

This is a simple "Hello World" application on Android.

```
package helloandroid;

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

public class HelloAndroid extends Activity {

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        TextView tv = new TextView(this);
        tv.setText("Hello World");
        setContentView(tv);
    }
}
```

As explained before, the entry point of any Android application is an Activity (more strict, the method `android.app.Activity#onCreate(Bundle)`). There's no label class on Android, but text view that can be also used as labels. Then, we create a text view and we set the contents of Activity with the text view just created.

Now, lets see how is the same application, but on Qt framework.

```
#include <QApplication>
#include <QLabel>

int main(int argc, char *argv[])
{
    QApplication app(argc, argv);
    QLabel label("Hello, world!");
    label.show();
    return app.exec();
}
```

It simply opens an application and shows a label with "Hello World" string. You can check that the main Qt class (`QApplication`) is properly initialized with the given arguments from main arguments. As described before, the entry point of Qt applications are the main function of your Qt main class. Then, it creates a simple label and show its content. Finally, the application return the code from execution of Qt application. Another important point is that such piece of code can also be built and launch on maemo platform (obviously, the graphical result is different, since you're using a different platform with different widgets).

The following example shows how we can insert a menu on Android and Qt applications.

On Android, the menus are defined on method `android.app.Activity#onCreateOptionsMenu(Menu)`. The callbacks of menu items are defined on method `android.app.Activity#onOptionsItemSelected(MenuItem)`.

```
package helloandroid;
```

Porting_Android_(Java)_applications_to_Qt_for_Symbian

```
import android.app.Activity;
import android.os.Bundle;
import android.widget.TextView;
import android.widget.Menu;
import android.widget.MenuItem;

public class HelloAndroid extends Activity {

    private final int MENU_QUIT = 1;

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        TextView tv = new TextView(this);
        tv.setText("Hello World");
        setContentView(tv);
    }

    /* Creates the menu items */
    public boolean onCreateOptionsMenu(Menu menu) {
        menu.add(0, MENU_QUIT, 0, "Quit");
        return true;
    }

    /* Handles item selections */
    public boolean onOptionsItemSelected(MenuItem item) {
        switch (item.getItemId()) {
            case MENU_QUIT:
                quit();
                return true;
        }
        return false;
    }
}
```

On Qt for Symbian, it is a little bit easier :). You just need to define a QAction. Then, insert it on the application menu bar. Finally, the callback is defined on method *connect* (used to define any callback on Qt).

```
#include <QApplication>
#include <QAction>

int main(int argc, char *argv[])
{
    QApplication app(argc, argv);
    QAction exitAction = new QAction(tr("&Exit"),this);

    // Add action direct to menubar
    menuBar()->addAction(exitAction);
    connect(exitAction, SIGNAL(triggered()),this, SLOT(close()));
}
```

API Mappings

File I/O

- To read from or write to files or other devices.
 - ◆ Android classes (from package **java.io**): File, FileReader, FileWriter, BufferedReader, BufferedWriter
 - ◆ Qt classes: QFile, QTemporaryFile, QBuffer, QProcess, QTcpSocket, QUdpSocket, QDataStream, QTextStream

Networking

- Socket communication.
 - ◆ Android classes (from package **java.net**): Socket, InetAddress, ServerSocket
 - ◆ Qt classes: QTcpSocket, QUdpSocket
- HTTP/FTP communication.
 - ◆ Android classes (from package **java.net**): HttpURLConnection, URL
 - ◆ Qt classes: QNetworkAccessManager, QUrl, QUrlInfo
 - ◆ Android classes (from package **org.apache.http.***): HttpClient, HttpGet, HttpPost, HttpResponse
 - ◆ Qt classes: QHttpHeader, QHttpRequestHeader, QHttpResponseHeader
 - ◆ Qt class: QFtp (no correspondence with Android)

Media

- To read from or write to files or other devices.
 - ◆ Android classes (from package **android.media**): MediaPlayer, MediaRecorder
 - ◆ Qt classes: AudioOutput, MediaController, MediaNode, MediaObject, MediaSource, ObjectDescription, Path, VideoPlayer

Android vs Qt for Symbian

Android was designed as a platform for mobile phones. Therefore, it provides access to different system resources, such as touch screen, camera, and telephony. On the other hand, Qt is a cross-platform application and UI framework initially designed for Desktop environments. Then, Qt does not yet provide mechanism to access resources of the mobile phone, yet. The following list shows some important differences of Android and Qt for Symbian.

- Android provides access to **PIM (Contacts, Calendar)** but Qt for Symbian doesn't provide it **yet**;
- Android provides access to **Telephony** but Qt for Symbian doesn't provide it **yet**;
- Android provides access to **Messaging** but Qt for Symbian doesn't provide it **yet**;
- Android provides access to **Camera** but Qt for Symbian doesn't provide it **yet**;

However there is a [Mobile Extensions for Qt for Symbian](#) available as technology preview.

- On Android, it is possible to load application UI from a XML description file (Maemo also support this feature if you use Glade to build your application's UI). However, Qt for S60 doesn't support such feature;

Porting_Android_(Java)_applications_to_Qt_for_Symbian

- Android is based on Java and Qt for Symbian is based on C++;
- Qt is cross-platform. That is, chances are that an application can be executed on Qt for Symbian and on Qt for Maemo, and even on Windows desktop.

References

- Qt
 - ◆ [Qt Site](#)
 - ◆ [Qt for Symbian Developer's Library](#)
 - ◆ [Foundations of Qt® Development \(Book\)](#)
 - ◆ [C++ GUI Programming with Qt 4 \(Book\)](#)
 - ◆ [The Book of Qt 4: The Art of Building Qt Applications \(Book\)](#)
- Android
 - ◆ [Wiki](#)
 - ◆ [Official Site](#)
 - ◆ [Developers Guide](#)
 - ◆ [Professional Android Application Development \(Book\)](#)