Porting Android (Java) applications to S60 5th Edition

Introduction

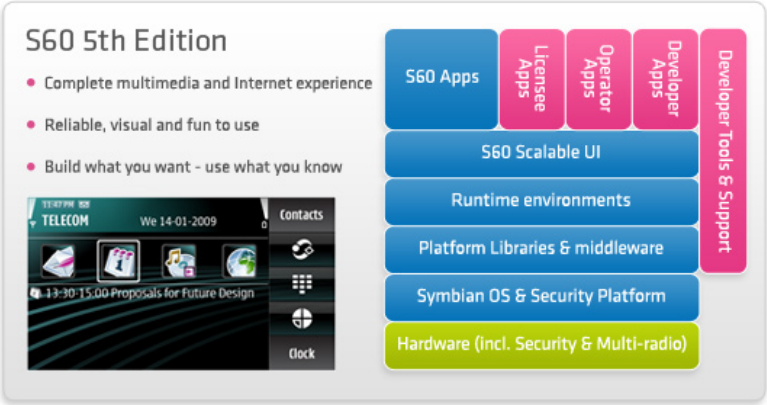
This porting guide should be used as a reference for Android developers porting existing software to S60 5th Edition.

Symbian is the market leader open operating system for mobile phones, designed to provide applications and services in a manner that is easy, efficient and secure.

Symbian supports global technology standards such as C/C++, Web (WRT), Qt, Java, Flash, Silverlight, Python and Ruby.

Overview of S60 5th Edition and Android

S60 5th Edition, built on Symbian OS 9.4, introduced a touch user interface, an enhanced sensor framework and platform services that enables Qt, WRT widgets and Flash developers access many features of the S60 platform, including GPS, contacts, calendar, messaging, audio and video through custom c++, javascript and actionscript APIs. S60 5th enables developers to write applications in a wide range of languages as described above. For a complete review of Symbian OS go to Wikipedia.



Android is built on the Linux Kernel and utilizes a custom virtual machine, commonly referred as Dalvik VM. Although the Android's kernel is based on Linux, developers can only use Java programming syntax to develop Android applications.

Java Micro Edition and Java SE applications are not compatible with Android due to the new format of bytecodes used by Dalvik virtual machine, this and many others facts force developers to rewrite Android applications to work on operating systems that already supports Java applications.

Application Lifecycle

Each runtime has a different approach to manage application lifecycle. To brief explain the key differences among the most popular runtimes we will summarize them bellow.

Symbian C++

TBD

Flash Lite

Flash lite is the Flash technology specifically developed for mobile phones and consumer electronics devices.

Symbian Web Runtime

(C) Copyright Nokia 2013. All rights reserved.

WRT Widgets runs on top of the Nokia web browser for S60. The application lifecycle can be compared to a common webpage, as an example, the onload event handler of the body tag is used when the document loads. WRT widgets also introduced a Javascript widget object to provide widget-specific APIs that perform fundamental operations on mobile.

The Javascript widget object onshow and onhide properties are an event handler for the events of when a widget is switching from the background to the foreground. In other words, when an opened widget gains focus or lost focus, the assigned callback function will be called.

Java

Java has become a popular programming environment on S60. Java applications typically designed to run on cellphones are called MIDlets. The Mobile Information Device Profile (MIDP) is the Java runtime environment for mobile devices, it specifies a lifecycle model for MIDlets. A MIDlet can be in any one of three states: ACTIVE, PAUSED or DESTROYED.

In the ACTIVE state the MIDlet is executing normally.

In the PAUSED state - memory and CPU consumption should be minimised. The MIDlet should not be holding any shared resource, and no non-essential threads running.

In the DESTROYED state the MIDlet has terminated and should be eligible for garbage collection.

Getting Started

The following topics will cover some basic aspects before you get started.

IDEs

Android
Android development is officially supported on Windows, Linux and Mac OS X through an Eclipse IDE plugin.

Symbian
Symbian has different approaches depending on what development framework you're using. If you are writing a C++ application you can use Qt and Qt Creator (or Carbide.C++ IDE for Symbian C++ development on Windows only). If you are writing Flash or WRT Widgets you can build them in both Windows and Mac OS X through the Adobe Flash IDE or Nokia Web Tools.

SDK

Android
The Android SDK provides light-weight tools to get started developing applications that run on Android-powered devices. At the time of this written the Android SDK 1.5, release 1 is available to download. This version supports multiple versions of the Android platform (Android 1.1, Android 1.5 and provides updated tools to let you deploy your application on any platform in the SDK, which helps you ensure forward-compitility and, if applicable, backward-compatibility.

Symbian OS 5th
As mentioned above, S60 platform supports a wide range of global technologies runtimes and tools to friendly help developers to get started developing content for S60 devices.

You can also download the all-in-one S60 Platform and Device SDKs for Symbian OS that best supports development of applications for the following runtimes:

- Symbian C++.
- Open C/C++.
- Java™ technology.
- Web Runtime (WRT).
- Python (using the PySymbian SDK plug-in).

For more information, visit the S60 platform and device SDKs' pages.

(C) Copyright Nokia 2013. All rights reserved.

Emulator

To use both SDK and emulator you need a Microsoft Windows XP SP2 machine. The emulator is only available for Windows XP and runs in a window on your computer. The S60 5th emulator supports touch UI emulation and produces tactile feedback in the form of sound. The S60 5th SDK best supports the following IDEs:

- Carbide.C++ v1.2, v1.3 and v2.0 Express, Professional, and Developer
- NetBeans IDE 6.0, 6.1 and 6.5
- Eclipse 3.3.1 with EclipseME 1.7.7 and MTJ

In addition the S60 5th Edition SDK has Web Runtime Platform Service APIs support for:

- Logging
- Calendar
- Messaging (SMS, MMS)
- Multimedia (media gallery)
- Contacts
- Location
- Landmarks
- Application Manager
- System Info
- Sensor

Applications

Now it's time to create your mobile application. First, you need to know how to package your applications for Symbian OS devices.

Application package

Android

The Android application package is an archive file with the *.apk extension bundled by the aapt tool and contains the compiled Java code along with any data and resource files required by the application.

Symbian OS 5th

Developers can use the .SIS extension to allow easy installation of applications. The Symbian OS installation file contains all the program files needed to be installed and is the standard way to distribute Symbian applications. These are usually produced using the makesis tool. As well as simply containing files, the SIS file can contain conditional statements which influence the installation e.g. device specific installations, language-specific installations, and user-selectable optional components.

The information in the SIS file is split up into two separate parts. The first part is the meta-data, describing the files that need to be installed. The second part of the SIS file contains all the actual file data. This enables software installation to be split into two phases, a decision and an installation phase. During the decision phase, the SIS file is examined and security checks are carried out in order to verify the install. The installation phase is only carried out if the verification is successful and is the process of copying the files to the device.

To make the packaging process easy many developers distribute free applications and online solutions to do that.

- Python

Ensymble is a GUI version for packaging Python scripts into SIS files.

- Flash Lite

Package Flash Lite content in a SIS by using the online packaging service provided by Nokia Developer and Kuneri.

- WRT

A WRT widget has a different approach from above. They use the .WGZ extension and doesn't require signing the package. For more info about this format go to Wgz.

- Java

Porting Android (Java) applications to Java ME on S60 5th Edition for a complete review.

Drawing graphics

All Java ME and Flash Lite source code are available at http://flashlitej2me.googlecode.com/.

- Symbian C++

Image Manipulation.

Showing GIF animations.

Using SVG in Symbian C++.

How to draw and update the screen directly by accessing the screen memory

(C) Copyright Nokia 2013. All rights reserved.

Avoid flickering with double buffering on Symbian.

- Flash Lite 1.1 and 2.x

Download sample.

- Python

```
from graphics import Image
import appuifw
import e32

def exit():
    lock.signal()
    appuifw.app.set_exit()

# Define o corpo da aplicação
appuifw.app.title = u'Load Image'
appuifw.app.screen = 'large' #normal, large, full

# seta o canvas
appuifw.app.body = canvas = appuifw.Canvas()
# define um evento de saída da aplicação
appuifw.app.exit_key_handler = exit

# obtem a largura e altura do canvas
cWidth, cHeight = canvas.size

# carrega a image
logoDir = "e:\\Python\\forumnokia.gif"
logo = Image.open(logoDir)

# obtem a largura e altura da image
imgWidth, imgHeight = logo.size

# desenha a imagem no centro do canvas
canvas.blit(logo, target=(cWidth/2 - imgWidth/2, cHeight/2 - imgHeight/2))

lock = e32.Ao_lock()
lock.wait()
```

- Java ME

```
     /**
    * Paint the ball on Screen
    *
    * @param graphics the graphics object to draw on.
    */
    public void paint(Graphics g) {
        // x and y are the coordinates of the top corner of the game's display area:
        int x = g.getClipX();
        int y = g.getClipY();

        // w and h are the width and height of the display area:
        int w = g.getClipWidth();
        int h = g.getClipHeight();

        // clear the screen (paint it white):
        g.setColor(0xffffff);
        g.fillRect(x, y, w, h);

        // set the x and y positions
        int xpos = (int)(Math.cos(this.angulo) * raio) +  ((this.getWidth()/2) - this.diametro/2);
        int ypos = (int)(Math.sin(this.angulo) * raio) +  (this.getHeight()/2);

        g.setColor(0xFF0000);
        //g.drawArc(xpos, ypos, this.diametro, this.diametro, 0, this.circum);

        this.angulo++;
        if(this.angulo >= 360)
                this.angulo = 0;
    }
```

- WRT

How to load images dynamically on WRT Widgets.

Controlling Keys and Softkeys

- Symbian C++

TouchUI: Event from hardware to software.

- Flash Lite

```
// seta o conteúdo para tela inteira
fscommand2("FullScreen", true);

// habilita o uso das softkeys em seu projeto mobile
fscommand2("SetSoftKeys", "Left", "Right");

// define um ouvinte dos eventos de tecla
var objOuvinte:Object = new Object();

// define a função onKeyDown
objOuvinte.onKeyDown = function() {
 getKey("Pressed");
}

// define a função onKeyUp
objOuvinte.onKeyUp = function() {
 getKey("Released");
}

// escreve no textfield o conjunto tecla pressionada/ação realizada
function getKey(pAction:String):Void {
 switch(Key.getCode()) {
  case 53:
   fscommand2("Quit");
  case Key.ENTER:
```


(C) Copyright Nokia 2013. All rights reserved.

```
    txtTeclas.text = "Enter Key " + pAction;
    break;
   case Key.UP:
    txtTeclas.text = "Up Key " + pAction;
    break;
   case Key.DOWN:
    txtTeclas.text = "Down Key " + pAction;
    break;
   case Key.LEFT:
    txtTeclas.text = "Left Key " + pAction;
    break;
   case Key.RIGHT:
    txtTeclas.text = "Right Key " + pAction;
    break;
   case ExtendedKey.SOFT1:
    txtTeclas.text = "SoftKey Left " + pAction;
    break;
   case ExtendedKey.SOFT2:
    txtTeclas.text = "SoftKey Right " + pAction;
    break;
   default :
    txtTeclas.text = "Key " + Key.getAscii() + " " + pAction;
  }
 }

 // adiciona um observador ao eventos de teclado
 Key.addListener(objOuvinte);

 // criacao de um objeto textfield dinamicamente
 this.createTextField("txtTeclas",
       this.getNextHighestDepth(),
       0,
       0,
       Stage.width,
       Stage.height);

 txtTeclas.text = "Application started\nPress '5' to quit.";
```

- Python

Archived:How to use Keys in PySymbian.

- Java ME

Using on-screen keypad in MIDlets in S60 5th Edition devices.

- WRT

Controlling the softkeys in Symbian Web Runtime.

How to interpret key events in WRT widgets?.

Off-line storage

- Symbian C++

CS000926 - Using SQL API for attaching and detaching database.

Archived:Using SQL API for creating non-secure and secure databases on Symbian.

- Flash Lite

Archived:Como utilizar o Shared Objects para armazenar informações com Flash Lite. You can download the source code here.

- Python

Archived:How to save SMS list using PySymbian

Archived:Database in PySymbian.

- Java ME

Record sorting in RMS.

```
     /**
     * Open the record store
     *
     * @param data - string data to be saved.
     */
      public void start() {
         if(!this.isLoaded) {
             try {
                 // Create a new record store
                 this.recordStore_rs = RecordStore.openRecordStore(REC_STORE, true);
                 this.isLoaded = true;

                 // Save 'Java RMS' string data
                 save("Java RMS");
                 // Read saved data
                 read();
                 // Print on Screen
                 print();
                 // Close record store
                 close();
             } catch (RecordStoreException ex) {
                 System.out.println(ex.toString());
             }
         }
     }

     /**
     * Paint the ball on Screen
     *
     * @param data - string data to be saved.
     */
     public void save(String data) {
         byte[] rec = data.getBytes();
```


(C) Copyright Nokia 2013. All rights reserved.

```
        try {
            this.recordStore_rs.addRecord(rec, 0, rec.length);
        } catch (Exception e) {
            System.out.println(e.toString());
            this.dataStored_str = e.toString();
        }
    }

    /**
     * Read saved data and record store details
     *
     */
    public void read() {
        try {
            this.dataStored_str = "Name of record: " + recordStore_rs.getName() + "\n";
            this.dataStored_str += "Number of records: " + recordStore_rs.getNumRecords() + "\n";
            this.dataStored_str += "Size: " + recordStore_rs.getSize() + "\n";
            this.dataStored_str += "Size available: " + recordStore_rs.getSizeAvailable() + "\n";

            if(recordStore_rs.getNumRecords() > 0)   {
                RecordEnumeration re = recordStore_rs.enumerateRecords(null, null, false);

                while(re.hasNextElement()) {
                    String data = new String(re.nextRecord());
                    this.dataStored_str += "Data saved: " + data + "\n";
                }
            }
        } catch (Exception e) {
            System.out.println(e.toString());
            this.dataStored_str = e.toString();
        }
    }

    /**
     * Close record store 'database'
     *
     */
    public void close() {
        try {
            recordStore_rs.closeRecordStore();
        } catch (Exception e) {
            System.err.println(e.toString());
        }
    }
```

- WRTh

Storing settings in Symbian Web Runtime.

Porting Android applications to S60 5th Edition

In this part of the guide we will discuss some similarities and differences porting Android applications to S60 5th Edition.

Porting GUI Applications to S60 5th Edition

Android

An Android user interface is composed of objects called Views. Each element in your UI layout is a View, such as a button, image or a text, each of these objects is a subclass of the View class.

Android provides an alternate XML-based UI construction model, instead of "programmatic" UI layouts. The general structure of an Android XML layout file is simple: it's a tree of XML elements, wherein each node is the name of a View class.

Symbian OS 5th

Native Symbian C++ development doesn't not provide XML based UI layout. In contrast, WRT developers can use their web design skills to construct UI based on HTML, CSS and JavaScript.

Technical References

The technical references used in this article are:

Twitter

- @symbiandevco
- @forumnokia
- @felipeandrade

Websites

- http://developer.android.com/index.html
- http://library.developer.nokia.com/

(C) Copyright Nokia 2013. All rights reserved.

(C) Copyright Nokia 2013. All rights reserved.