Android Application Programming

Tutorial 1: Introduction to Android and Eclipse

Introduction

In this tutorial, you will setup your Android development environment and go through the Hello World tutorial on the Android Developer website at http://developer.android.com/resources/tutorials/hello-world.html. You will finally learn about using the Eclipse debugger and logging errors.

Setup Your Environment

The best way to develop applications for Android is using the Eclipse IDE and the Android plug-in for Eclipse called Android Development Tools (ADT). You can test your applications by running them on an Android Virtual Device (AVD), aka Android emulator.

To setup your environment, follow each of the steps below in order. Detailed installation instructions are available here: http://developer.android.com/sdk/installing.html

1. Download and install JDK 7
   http://www.oracle.com/technetwork/java/javase/downloads/index.html

2. Download and install Eclipse IDE for Java EE Developers (version 3.7 - Indigo)
   http://www.eclipse.org/downloads/

3. Download the Android SDK and unzip it onto your hard drive in a location of your choosing.
   http://developer.android.com/sdk/index.html

4. Download and install the Android Development Tools (ADT) for Eclipse
   http://developer.android.com/sdk/eclipse-adt.html

Now your environment is ready for development.

Hello World Tutorial

Run through the "Hello World" tutorial at http://developer.android.com/resources/tutorials/hello-world.html which takes you through creating an Android Virtual Device (AVD), creating an Android project, writing some code to display "Hello, Android", and running the application in the emulator.

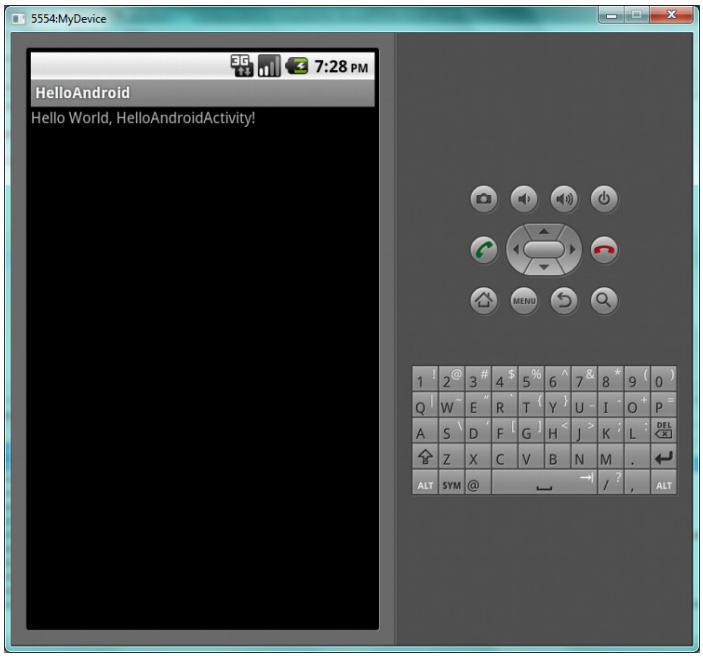
When you run an Android application, Eclipse will first start an emulator if one isn't already running. If you have multiple AVD configurations, it's usually a good idea to *first* start your emulator of choice so you can control which emulator you want to use.

It may take a few minutes for your emulator to appear because of a lengthy initialization period… you'll have to be patient. You can watch the status of the emulator from Eclipse's Console window. It will look something like the lines displayed below which show how the .apk file is first installed and then launched:

```
[2011-10-22 14:22:34 - HelloAndroid] ------------------------------
```

```
[2011-10-22 14:22:34 - HelloAndroid] Android Launch!
[2011-10-22 14:22:34 - HelloAndroid] adb is running normally.
[2011-10-22 14:22:34 - HelloAndroid] Performing com.example.HelloAndroidActivity activity launch
[2011-10-22 14:22:34 - HelloAndroid] Automatic Target Mode: using existing emulator 'emulator-5554'
running compatible AVD 'MyDevice'
[2011-10-22 14:22:34 - HelloAndroid] Uploading HelloAndroid.apk onto device 'emulator-5554'
[2011-10-22 14:22:34 - HelloAndroid] Installing HelloAndroid.apk...
[2011-10-22 14:22:38 - HelloAndroid] Success!
[2011-10-22 14:22:38 - HelloAndroid] Starting activity com.example.HelloAndroidActivity on device
emulator-5554
[2011-10-22 14:22:40 - HelloAndroid] ActivityManager: Starting: Intent {
act=android.intent.action.MAIN cat=[android.intent.category.LAUNCHER]
cmp=com.example/.HelloAndroidActivity }
```

Once the application is started, you will see it running like the figure on the right. Note that this is the default Android 2.2 AVD skin. If you use a different version of Android, your emulator will look somewhat different.

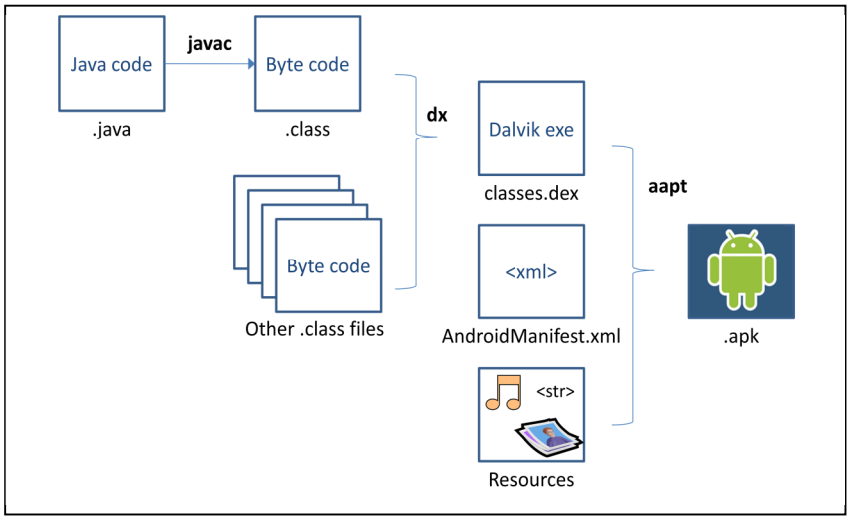


You can use the emulator as you would a real Android device by using your mouse to click on the screen and using your keyboard to type.

When you make changes to your program, **do not close the emulator**. Just edit your program, save it (which makes Eclipse re-build it), and re-run your new program (Ctrl-F11); the emulator will replace the previous program with the new one and execute it (you'll have to be patient… it can take up to a minute to reload).

The Android Package

The .apk file, the Android Package, is the application file that was installed and executed on the emulator. The .apk file is produced by Eclipse in a number of steps as illustrated in the figure below.



1. The Java source code is compiled into Java bytecode (.class files) as is traditionally done in Java programming.
2. The dx tool converts the Java bytecode into Android bytecode (.dex files), a compact format that can be executed by a Dalvik virtual machine.
3. Finally, the Android Asset Packaging Tool (aapt) is used to combine .dex files with the AndroidManifest.xml and other resources making up your application. It produces the .apk which is really just a zip file.

You can find .apk files on the Web and install them directly to your emulator or Android device although you must be careful about installing un-trusted .apk files that could contain malicious code.

XML Layout

Continue following the tutorial where it describes how to change from a "programmatic" layout to an XML-based layout. In this section, you'll learn about main.xml, strings.xml, and the auto-generated R class. Specifying the layout of your Android apps with XML is considered a "best practice" and is ideally how you will design all your applications in the future.

Using the Debugger

Continue following the tutorial where it describes how to debug your project in Eclipse. In this part of the tutorial, you'll introduce a `NullPointerException` that will cause your program to terminate abruptly. You'll also add a breakpoint so that you can step through your code.

When Eclipse pauses at a break point, you'll notice that Eclipse has shifted around some sub-windows and introduced new ones. This orientation is called the "Debug" perspective, and this is the ideal perspective to use when debugging. The perspective you were seeing before debugging is the "Java" perspective.

When stepping through your code, use **F6** to step to the next line, **F5** to step into a function call, and **F8** to resume execution. This might take some getting used to if you are transitioning from Visual Studio.

When debugging, sometimes you'll notice that Eclipse tries to display source code that it cannot find and will display a "Source not found" message. Press F8 to continue when this happens. Sometimes you'll need to press F8 several times because Eclipse will break at several locations. In this example, eventually you'll notice that Eclipse will not continue running your app when pressing F8, and that's because the modal dialog box complaining about the error in the emulator must be dismissed before Eclipse can continue to execute any code.

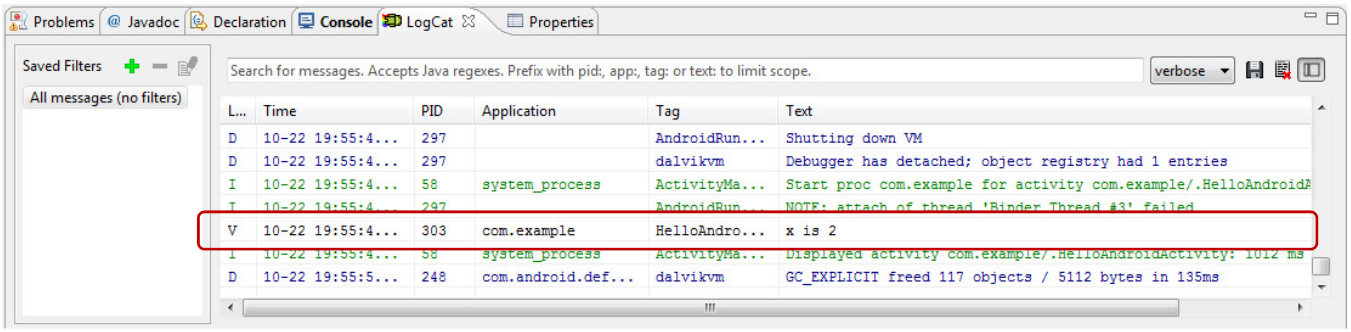
Logging Messages

Rather than using the debugger to debug your code, you may find it more helpful to use the Android logging class (`android.util.Log`) to send messages to the **LogCat**. The LogCat should be visible in one of the tabs near the bottom of the Eclipse window. If you don't see the LogCat, select **Window > Show View > Other…, Android > LogCat** to make it visible.

To log messages, first import `android.util.Log` and declare a log tag (a unique identifier for your log messages). Add a call to `Log.v()` in your program like so:

```
package com.example.helloandroid;

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

public class HelloAndroid extends Activity {

    private static final String LOG_TAG = "HelloAndroid_tag";

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

        int x = 2;
        Log.v(LOG_TAG, "x is " + x);
    }
}
```

Now run your program. When the `onCreate()` method runs, the `Log.v()` call will output "`x is 2`" to the LogCat as shown below.

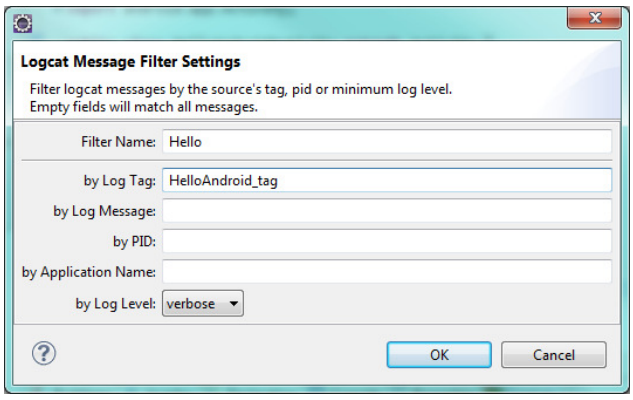


Note that there are different levels of log messages:

Verbose:	`Log.v()`
Debug:	`Log.d()`
Information	`Log.i()`
Warning	`Log.w()`
Error	`Log.e()`

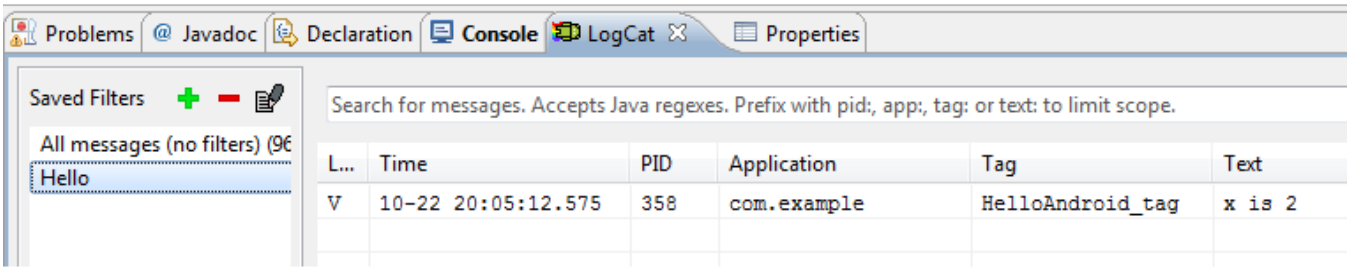
Typically you'll want to use the appropriate method to log the severity of the error message.

Because the log messages can easily get lost in the numerous messages shown in the LogCat, it's a good idea to create a filter so only the messages logged by our app are displayed.



First click the green + next to Saved Filters (left side of the LogCat window). This will launch a dialog box as shown to the right. Enter **Hello** for the Filter Name and **HelloAndroid_tag** as the Log Tag and press OK.

Now when "Hello" is selected from the list of filters, you should only be able to see the messages logged by the app as shown below. If you'd like to see all the messages, click "All messages".



Conclusion

You should now be familiar with setting up an emulator and running an Android app on the emulator. You also know how to use the Eclipse debugger to step through code and can sending debug information to the LogCat. In the next tutorial, you will create a more sophisticated Android application that plays tic-tac-toe.