1½ Hour Android Bootcamp

Michiel Helvensteijn

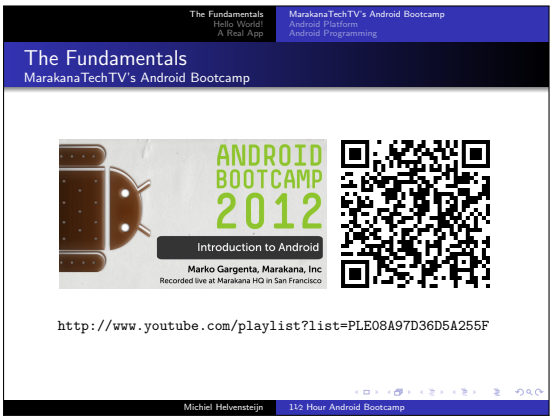
Challenges in Computer Science Seminar, 19-2-2013

Introduction

I was invited to give a presentation for the subject Challenges in Computer Science Seminar at Leiden University: a brief demonstration of how to develop an Android app. I'm afraid that even with this summary sheet, you will not be able to get the full benefit of the presentation, because most of it was an actual demonstration inside Eclipse. I just write these sheets for completeness sake. However, I can highly recommend MarakanaTechTV's Android Bootcamp Series at `http://www.youtube.com/playlist?list=PLE08A97D36D5A255F`

1 The Fundamentals

1.1 MarakanaTechTV's Android Bootcamp

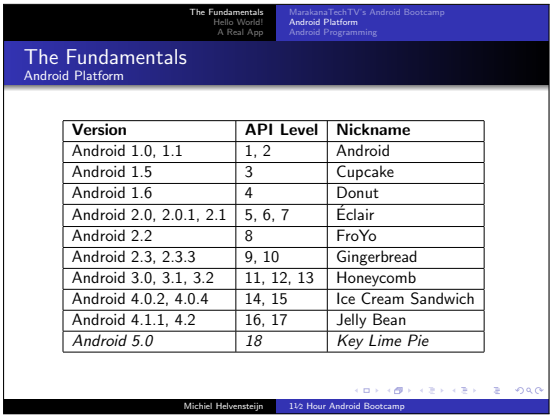


This is an excellent series of videos that actually taught me a lot of the basics (though that was an earlier iteration of the Bootcamp). They gave me the idea to use the 'live programming' format for this presentation.

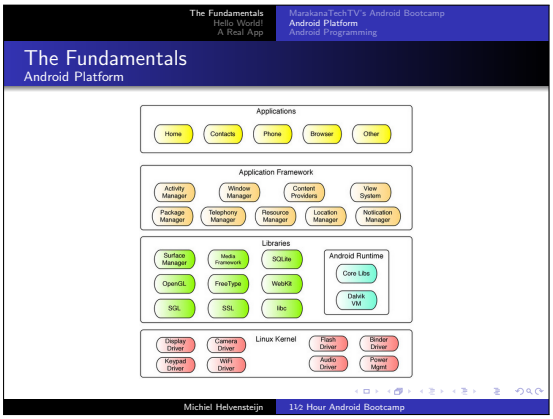
Just one thing: those videos can be... painfully slow. And 1½ hours will probably be way too fast. I give this presentation only to give you a taste of how things look and feel.

Oh, yeah, and I'm stealing a few slides from them (just a few).

1.2 Android Platform



The API level is the thing developers actually care about. You want to *target* a level as high as possible to be able to offer the best new features; but you want to *support* a level as low as possible, gracefully degrading your app so that people with old phones can still use it.



This is very cool. Android is built upon the *Linux Kernel*. Only the kernel, mind you.

The *Libraries* on top of it are all open source and optimized for a mobile environment. We *don't* care about caching as much, because everything is flash memory. We *do* care about battery life.

On top of that, we have the *Application Framework*. This is the only thing app developers have access to, and this is what corresponds to a specific 'API Level'.

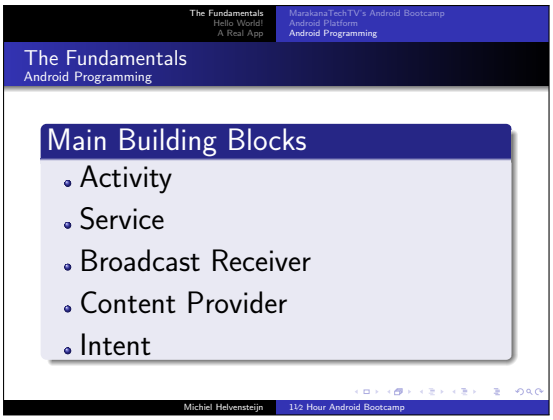
On top of that are your apps. And Google's *apps*. And any app you download from Google Play.

1.3 Android Programming

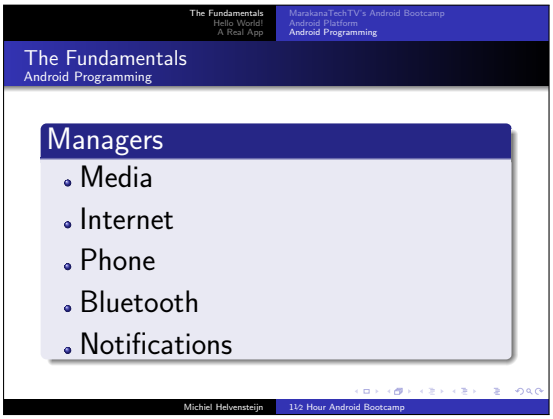
Android programming is Java programming. Most of what you can do with Java, you can do with Android. But the 'way of working' can sometimes be quite different.

You don't have to worry about creating a `main()` method, for instance. That's all taken care of by the Application Framework. It includes an *event loop*, and it is up to you to 'react' to many different kinds of events by extending existing (abstract) classes and then overwriting specific methods.

You can actually program for Android without any IDE. Everything is available on the command line. But Eclipse makes the go work infinitely more smoothly. They've put quite some work into the Android Eclipse plugin. You should probably use it.



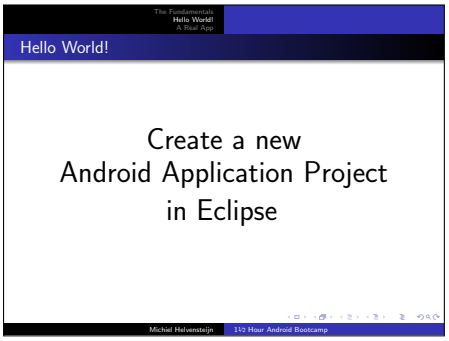
Activities are visible screens on the phone. They have layouts, buttons, text and that kind of thing. A *Service* is a unit that can run code in the background without necesserily having anything visible (by default they don't run in a separate thread, though). A *Broadcast Receiver* can pick up all kinds of signals from the system, other apps - or itself - and react to them in some way (like notifying other parts of the app). A *Content Provider* makes data from your app available to other apps, so they can all work with the same data-set. All of these communicate with each other by sending *Intents*. In this presentation we only concern ourselves with activities, services and intents.

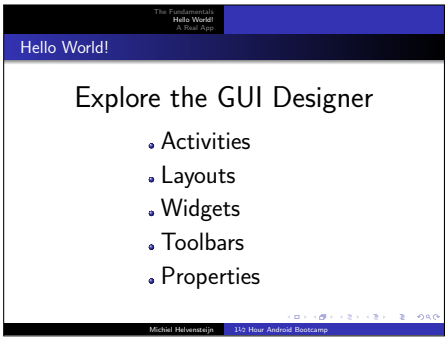


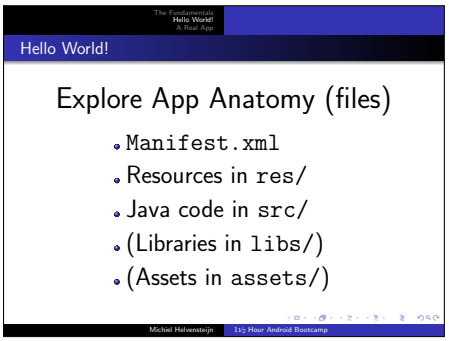
Android has a common pattern of exposing the functionality of the device and operating system to app developers: *Managers*. This slide lists just a few example aspects that can be controlled through Managers.

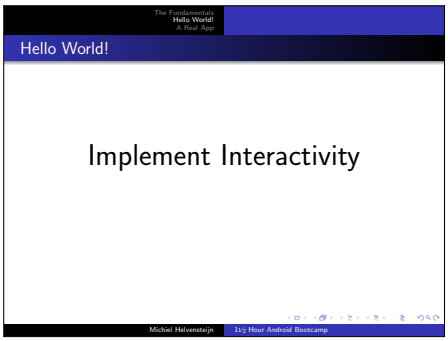
As an Android developer, there is a single website that will be most important to you. The official API reference on the Android Developer website.

2 Hello World!



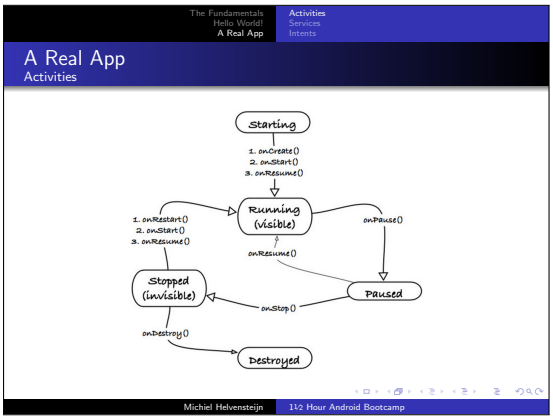




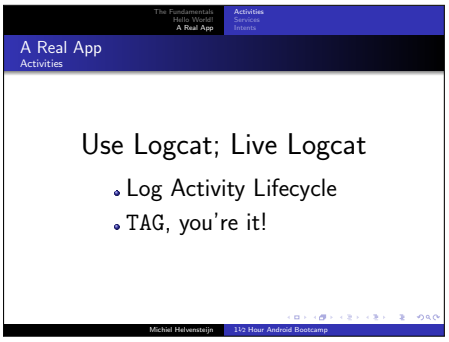


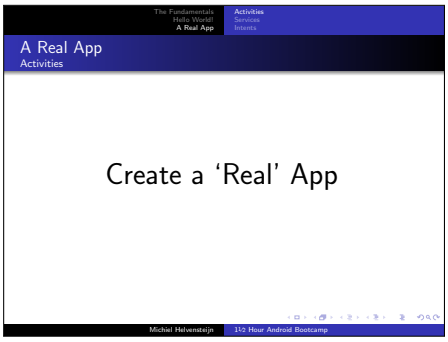
3 A Real App

3.1 Activities

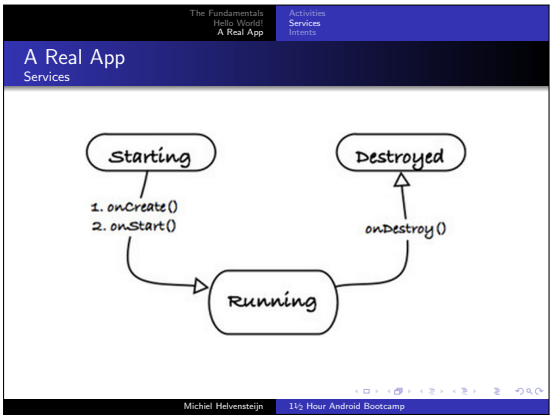


Activities have a well-defined *lifecycle*. App developers cannot directly control this lifecycle (merely hint / request), but can dictate how their activity behaves on transitions between phases. For this purpose there is a set of methods that you can overwrite when you inherit from the `Activity` class. This way you can always request or release resources, save data or do anything else to make sure your activity behaves properly no matter what the user or the operating system does.

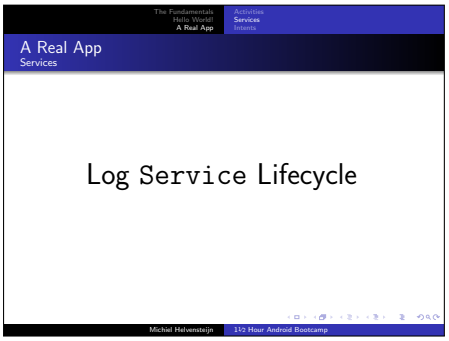


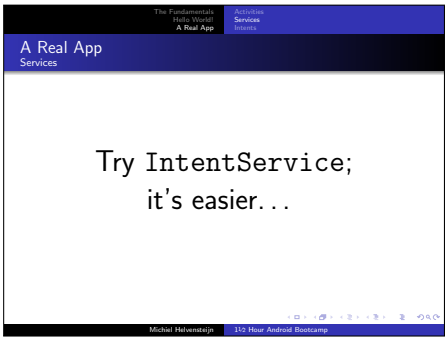


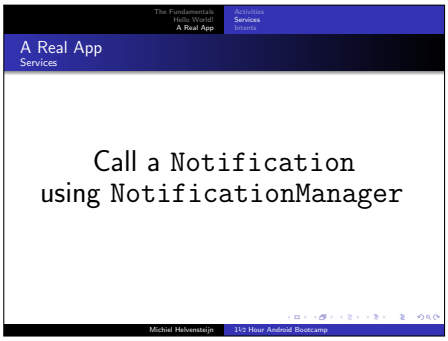
3.2 Services

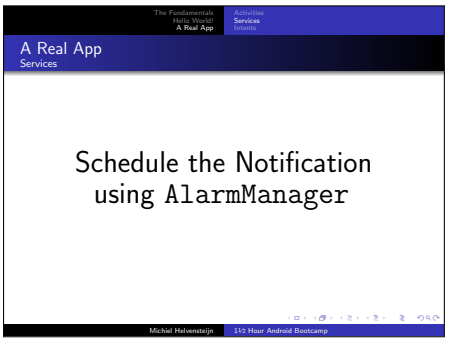


Like activities, services also have a lifecycle, but a much easier one. They are created, started and then destroyed. Nonetheless, it is important that you prepare your service to be killed at any time when the OS decides that it needs to kill it to reallocate resources.

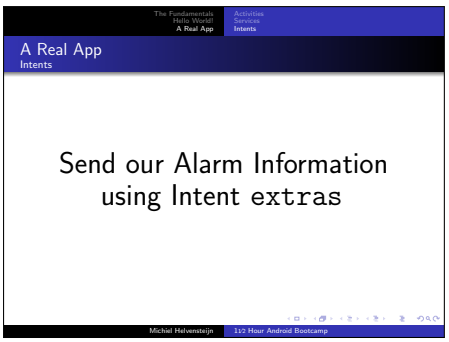


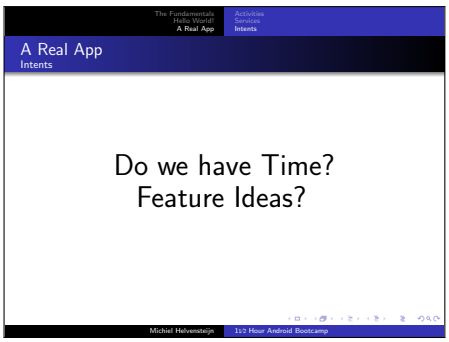






3.3 Intents





Conclusion

Thanks for reading!

intentionally (almost) empty