Eduonix Android Test_Framework_Application

Introducing the Eduonix Android Test_Framework_Application

As professional developers we must be always asking ourselves how can we build better software

As professional Android application developers we must be always asking ourselves how can we build better Android applications

Software Engineers perform research to answer these questions, professional Android developers write blogs to share their experience in trying to build better Android applications, lets build on the research and experience they provide.

Principles for successful Android Application development

1. Create a **prototype** for the app you're going to create. A prototype is an incomplete, unrefined version of the app. Build it so that it does just enough to illustrate the core value-add of the app. Then, show it to your prospective users, be quiet, and listen. Incorporate their feedback into your eventual design

2. Principle 1 is the first step in a process called **Incremental Development.**
Build your app one piece at a time, testing as you go. This process allows you to see results sooner (and have fewer software bugs) when you follow it.

3. **KISS** "Keep it simple, stupid!" The number of bugs will be proportional to the size and complexity of the code. Simple code means less bugs

4. The **UNIX philosophy** Essentially it comes down to: Write code that does one thing and does it well. Write code in blocks that perform a task well and reuse the blocks where you can.

5. **Rapid Application Development** With elaborate software development planning procedures, people will specify requirements and suggest solutions often change their mind if they see a possible solution in action. Therefore, the "in action" part is often the first thing to do.

So lets build an app that acts as a development template that we can reuse to construct prototypes for Rapid Application Development, that allows testing, code reuse and we will call it the **Eduonix_Android_Test_Framework_Application**.

Lets use the most uptodate and trending technologies in the Android space for the principles outlined above

Grade build tool
Roboelectric testing framework
Junit testing framework

High Level Outcomes
Learn Gradle, learn structure android project, learn unit testing.

Low Level Outcomes
Android sdk, dependency management, manifest. xml views and java controllers(Activities)
Iterative flow of android development, Gradle (build and package) Emulator deploy

Practical Requirements

Android SDK, Gradle, IDE Eclipse or Intellij

Practical

1. Android-> SDK Manager

    Select/Check the following boxes:
    - Android SDK Tools
    - Android SDK Platform-tools
    - Android 4.4 (API 19)
        - Doc, SDK, Samples, Intel x86, APIs
    - Android 4.3 (API 18)
        - Doc, SDK, Samples, Intel x86, APIs
    - Extras
        - Android Support Library
        - Google Play services
        - Google Play for Froyo
        - Google USB Driver(If Windows OS)
        - Intel x86 HAXM (If Windows OS)

2. Setup Virtual Devices/Android Emulator. AVD Manager

Use the following settings

AVD Name: Main
Device: Nexus S
Target: 4.3
CPU: Intel Atom (x86)(Windows) ARM(Linux)
Check Use Host GPU

Launch Virtual Device/Emulator to check functionality

4. Set up unit testing framework
   - Create a new folder src/test/java for the tests.
   - Edit build.gradle in your project folder.
   put android jar last in classpath
   src *.class files /home/ubu/eduonix-gradle/build/classes/debug
   test *.class files /home/ubu/eduonix-gradle/build/test-classes
5. build the source code

6. Run tests in our testing framework

7. Run the application in the emulator