CODEPATH

Search

Getting Started with Gradle

Edit Page Page History

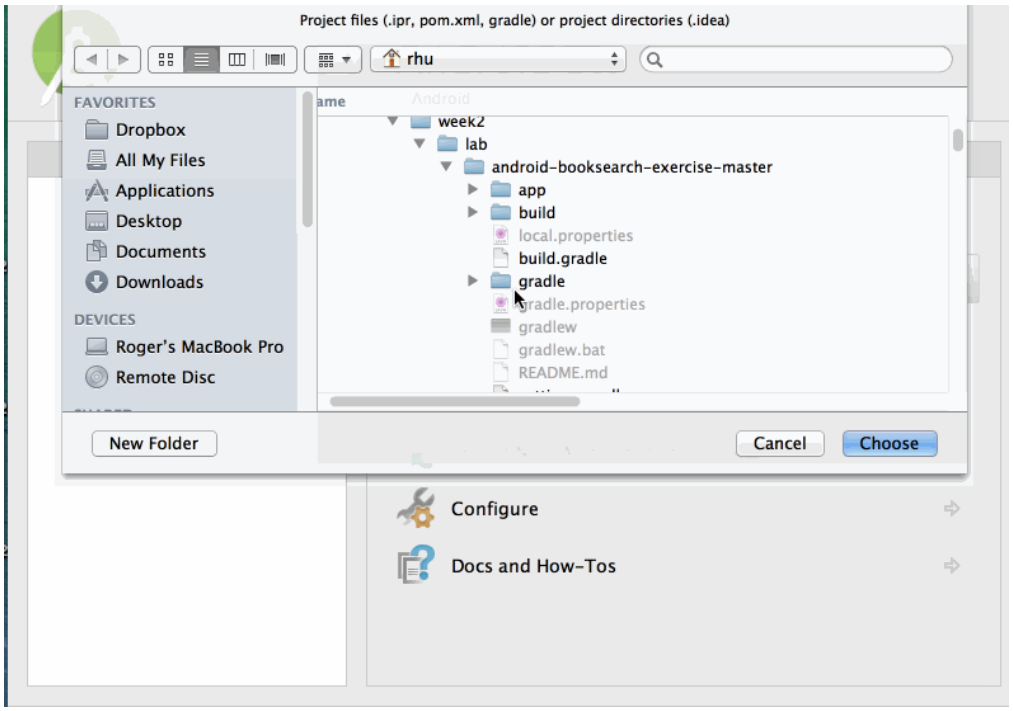
Intro to Gradle

Gradle is a powerful new dependency management system for Java. It provides the ability to provide more custom build logic than predecessors such as Maven or Ant.

Android Studio uses an Android Gradle plugin that handles most of the integration with Gradle. This plugin allows you to build different versions of your app (i.e. paid and unpaid, custom builds for different Android devices) while still leveraging the same code base. It also enables signing your APK and code obfuscation support too, as well as many other features.

Importing existing Android Studio projects

When importing existing Android Studio projects, you should make sure to select the `build.gradle` in the top-level directory. If you choose this option, you should see the `Use default gradle wrapper` option exposed and will be able to import the project properly. If you select on a folder or even the `app/build.gradle` file, you are likely to import a blank project and will need to redo the import.



If you are interested in what is the Gradle wrapper and why it's helpful to have, read the section below.

Intro to the Gradle wrapper

When you setup a project in Android Studio, it automatically generates several files for you that helps to allow people to build your code without needing to install Gradle ahead of time, which is useful especially when needing to do continuous integration testing.

By checking in the gradlew and gradlew.bat files into your source code repository, other people on Unix and Windows platforms do not need to go through the process of manually downloading Gradle or installing Android Studio.

The gradle-wrapper.properties file, which is created when you first create an Android Studio project, determines what version of Gradle to use:

```
#Wed Apr 10 15:27:10 PDT 2013
```

```
distributionBase=GRADLE_USER_HOME
distributionPath=wrapper/dists
zipStoreBase=GRADLE_USER_HOME
zipStorePath=wrapper/dists
distributionUrl=http\://services.gradle.org/distributions/gradle-2.21-all.zip
```

Gradle will use this configuration to see if the version has already been installed. If not, it will be downloaded and stored in a separate directory. (For Unix machines, the various Gradle downloads will live in ~/.gradle/wrapper.)

In addition, you will need to setup the Android Gradle plugin by setting your build.gradle to have a dependency:

```
buildscript {
    repositories {
        jcenter()
    }
    dependencies {
        classpath 'com.android.tools.build:gradle:1.2.3'
    }
}
```

Google maintains this Android plugin and you can find the latest updates at http://tools.android.com/tech-docs/new-build-system. Both Gradle and the Android Studio plugin are constantly evolving, so you check the site to see what versions of Gradle are supported for which plugin.

Also, keep in mind that the Android Gradle plugin finds your SDK by what is defined in the local.properties file. You should already find this local.properties file in your project directory:

```
sdk.dir=/Applications/Android Studio.app/sdk
```

If you don't want to set local.properties, you can also define the ANDROID_HOME environment variable which points to your Android SDK.

```
// Unix
export ANDROID_HOME=~/android-sdks

// Windows
set ANDROID_HOME=C:\android-sdks
```

Testing Gradle

Finally, you can check your working installation by running:

```
./gradlew -v
```

If you are using Windows, you will be using gradlew.bat instead.

```
./gradlew.bat -v
```

Using Gradle

To build the APK, run this command in the root directory of your project:

```
./gradlew assemble
```

If a build occurs and you see the output:

```
BUILD SUCCESSFUL

Total time: 5.738 secs
```

Then Gradle is successfully set up for your project. If you get an error, try googling the error. Usually the issue is that you need to install the build tools.

If you have integration tests you want to run, make sure you have an open emulator or connected device and run this command in the root directory:

```
./gradlew build
```

This will build the apk, then automatically compile and run your integration tests.

If you want to delete old APKs before you re-run either `./gradlew` (build without tests) or `./gradlew build` (build with tests), run:

```
./gradlew clean
```

Declaring Dependencies

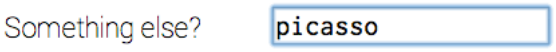
To add dependencies to your project, you need to modify the `build.gradle` file and add extra lines configuring the packages you require. For example, for certain Google or Android, dependencies look like:

```
android {
  ...
  dependencies {
      // Google Play Services
      compile 'com.google.android.gms:play-services:6.5.+'

      // Support Libraries
      compile 'com.android.support:support-v4:22.2.0'
      compile 'com.android.support:appcompat-v7:22.2.+'
      compile 'com.android.support:gridlayout-v7:22.2.+'
      compile 'com.android.support:support-v7-mediarouter:22.2.0+'
      compile 'com.android.support:support-v13:22.2.0+'

      // Note: these libraries require the "Google Repository" and "Android Repository"
      //       to be installed via the SDK manager.
  }
}
```

You can also add dependencies based on the Maven Central Repository. The best tool for finding packages is actually the Gradle Please utility that takes care of helping you locate the correct package and version to add to your gradle file for any library:




```
dependencies {
    compile 'com.squareup.picasso:picasso:2.4.0'
}
```

How to setup the Gradle wrapper

If you are planning to setup your own project without Android Studio (i.e. via Eclipse), the current recommendation is to setup the Gradle wrapper (http://www.gradle.org/docs/current/userguide/gradle_wrapper.html). The reason is that other people trying to use your project do not need to install Gradle themselves once you've generated the files needed to bootstrap the process.

To generate this initial set of files (Android Studio will automatically handle this work for you), you need to add these lines to your build.gradle file:

```
task wrapper(type: Wrapper) {
    gradleVersion = '2.2.1'
}
```

Then run:

```
gradle wrapper
```

The files below will be generated. Similarly, Android Studio will create the same directory structure when the "wrapper" option is used :

```
gradlew
gradlew.bat
gradle/wrapper/
  gradle-wrapper.jar
  gradle-wrapper.properties
```

Customizing Android Builds

Gradle provides enormous flexibility in enabling you to customize your builds for paid/unpaid versions, debug/production releases, and use different signing keys. Take a look at this guide for more information.

Resources

Check out the following links for more details:

- Automating Android Builds with Gradle
- Building with Gradle
- Gradle User Guide
- Gradle Example Project
- Building Android Projects with Gradle This resource covers creating a Gradle wrapper for use with Android Studio.
- Gradle wrapper More background information of the Gradle wrapper.