nimble

Android Template v2.0.0

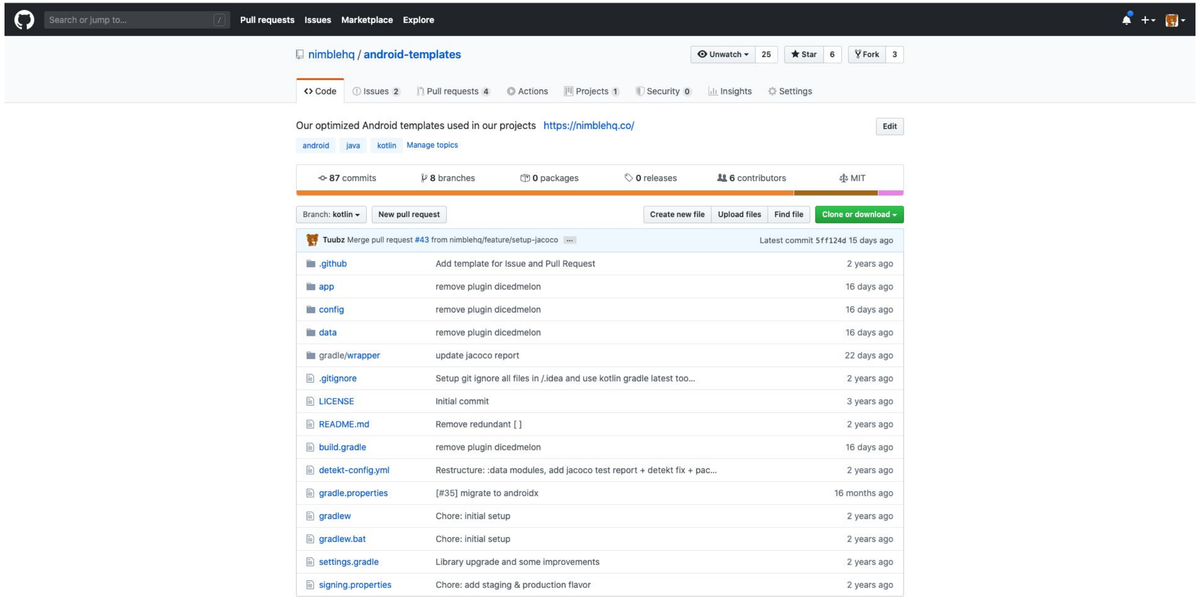
Toby & Pooh

Growth Session #28 - April 24 2020

Introduction - What is Android Template?

A code template that includes all the **base architecture** components:

- Easy & convenient whenever we want to start a new Android Project 🔥
- Align projects and developers by using the same coding environment 🚀



nimblehq / android-templates

Unwatch 25 | Star 6 | Fork 3

Code | Issues 2 | Pull requests 4 | Actions | Projects 1 | Security 0 | Insights | Settings

Our optimized Android templates used in our projects https://nimblehq.co/

android java kotlin Manage topics

87 commits | 8 branches | 0 packages | 0 releases | 6 contributors | MIT

Branch: kotlin | New pull request | Create new file | Upload files | Find file | Clone or download

Tuubz Merge pull request #43 from nimblehq/feature/setup-jacoco ... Latest commit 5ff124d 15 days ago

.github	Add template for Issue and Pull Request	2 years ago
app	remove plugin dicedmelon	16 days ago
config	remove plugin dicedmelon	16 days ago
data	remove plugin dicedmelon	16 days ago
gradle/wrapper	update jacoco report	22 days ago
.gitignore	Setup git ignore all files in /.idea and use kotlin gradle latest too...	2 years ago
LICENSE	Initial commit	3 years ago
README.md	Remove redundant [ ]	2 years ago
build.gradle	remove plugin dicedmelon	16 days ago
detekt-config.yml	Restructure: :data modules, add jacoco test report + detekt fix + pac...	2 years ago
gradle.properties	[#35] migrate to androidx	16 months ago
gradlew	Chore: initial setup	2 years ago
gradlew.bat	Chore: initial setup	2 years ago
settings.gradle	Library upgrade and some improvements	2 years ago
signing.properties	Chore: add staging & production flavor	2 years ago

Introduction - What's planned for Android Template v2.0.0?

The idea is to **improve** our existing base template, as it was quite outdated:

Setup the right **code coverage tool**, by upgrading **Jacoco** ✅:
→ Shows which parts of the code have not been or have been tested already, to increase test coverage

Setup an **Android bootstrap** functionality ✅:
→ Align all our developers on the same code style, plugins, ...

Update our **module structure** 🚫:
→ Separate components in to multiple modules, to improve building time

Update our MVVM Architecture with **UseCase** 🚫:
→ Enhance a cleaner code architecture and improve testability

Update our **Dependency Injection** according to a single activity architecture 🚫:
→ Keep us up to date with the newest technologies

Setup basic fastlane components with **Firebase app distribution** 🚫:
→ Make all our lives a lot more easy, as the app distribution part will all be automated

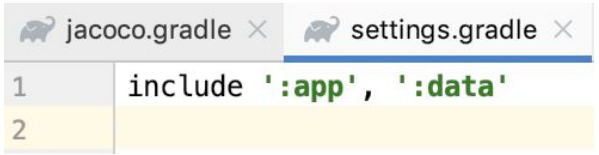
Upgrade Jacoco

Why we need to upgrade Jacoco? 🚀

To support a **full coverage** report from multiple modules we need to update the classes and paths accordingly. In our template we currently have two, called “app” & “data”

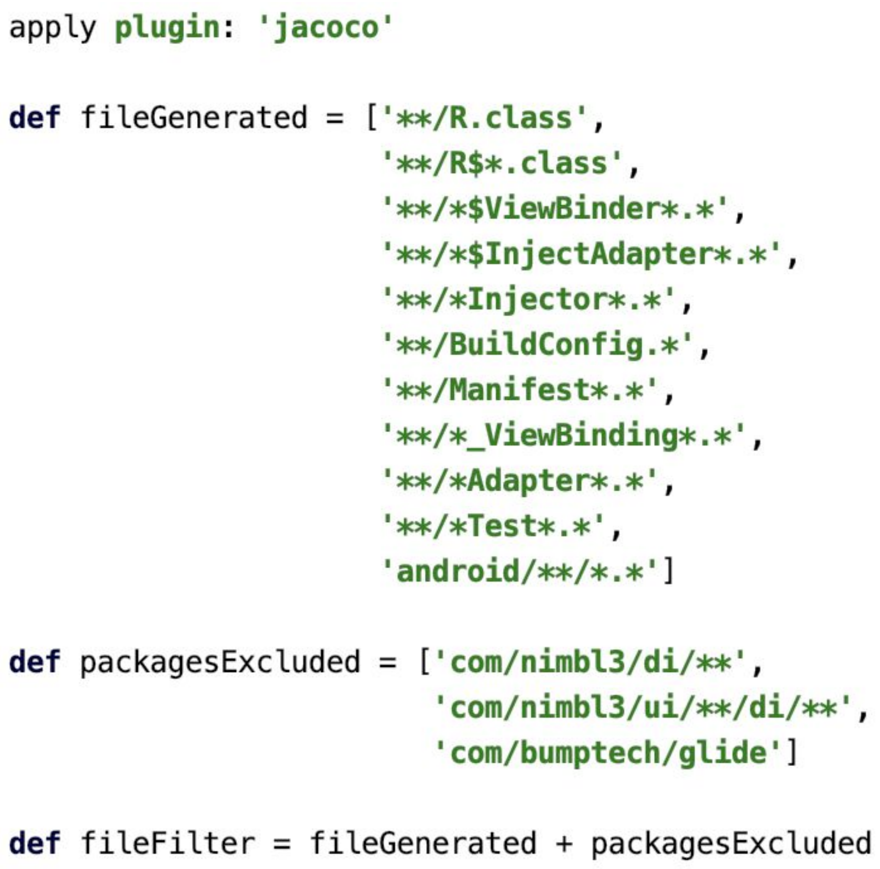
Expectation ✅

The report should have detailed information about which classes / paths are covered by tests correctly and which ones are not.



```
include ':app', ':data'
```

Upgrade Jacoco



```
apply plugin: 'jacoco'

def fileGenerated = ['**/R.class',
                     '**/R$*.class',
                     '**/*$ViewBinder*.*',
                     '**/*$InjectAdapter*.*',
                     '**/*Injector*.*',
                     '**/BuildConfig.*',
                     '**/Manifest*.*',
                     '**/*_ViewBinding*.*',
                     '**/*Adapter*.*',
                     '**/*Test*.*',
                     'android/**/*.*']

def packagesExcluded = ['com/nimbl3/di/**',
                        'com/nimbl3/ui/**/di/**',
                        'com/bumptech/glide']

def fileFilter = fileGenerated + packagesExcluded
```

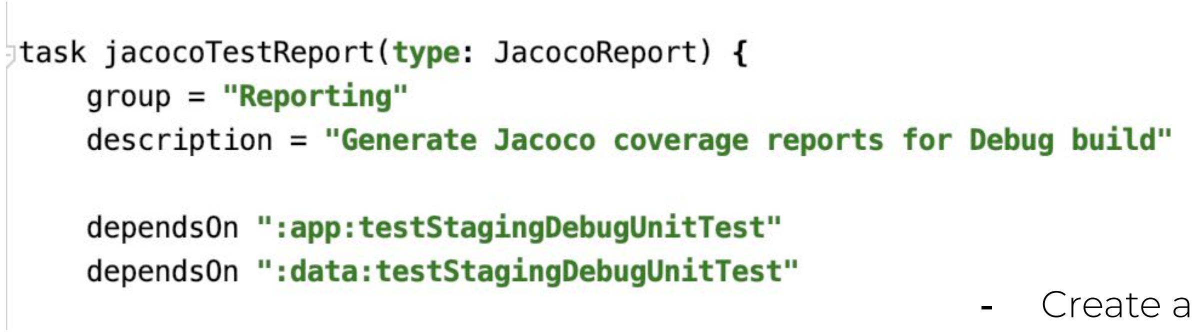
Let's start with project setup 💎

First step

- Apply plugin jacoco.
- Prepare file filters by specifying the files and/or packages that you don't want to be shown in the final report.

Upgrade Jacoco

Next step 💎



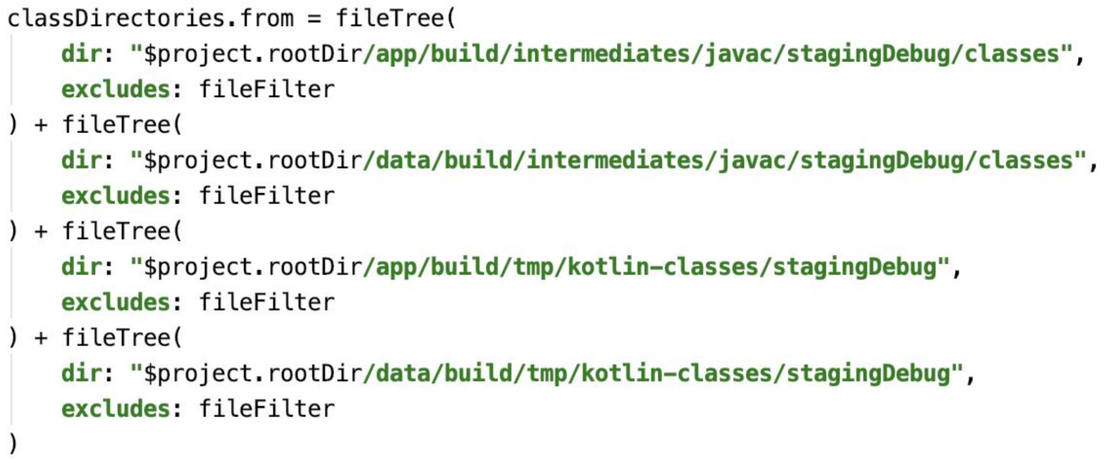
```
task jacocoTestReport(type: JacocoReport) {
    group = "Reporting"
    description = "Generate Jacoco coverage reports for Debug build"

    dependsOn ":app:testStagingDebugUnitTest"
    dependsOn ":data:testStagingDebugUnitTest"
```

- Create a **task** for jacoco report.
- Create a name group and add description.
- Define the modules we want to include in the generated report.

Upgrade Jacoco

Next step 💎

Specify compiled classes and add the **file filters** we specified in the previous step for each module to control files that we want to exclude from our final report.



```
classDirectories.from = fileTree(
        dir: "$project.rootDir/app/build/intermediates/javac/stagingDebug/classes",
        excludes: fileFilter
) + fileTree(
        dir: "$project.rootDir/data/build/intermediates/javac/stagingDebug/classes",
        excludes: fileFilter
) + fileTree(
        dir: "$project.rootDir/app/build/tmp/kotlin-classes/stagingDebug",
        excludes: fileFilter
) + fileTree(
        dir: "$project.rootDir/data/build/tmp/kotlin-classes/stagingDebug",
        excludes: fileFilter
)
```

Upgrade Jacoco

```
sourceDirectories.from = files([
    "$project.rootDir/app/src/main/java",
    "$project.rootDir/data/src/main/java"
])

executionData.from = fileTree(dir: project.rootDir, includes: [
    "app/build/jacoco/testStagingDebugUnitTest.exec",
    "data/build/jacoco/testStagingDebugUnitTest.exec"
])
```

Next step 💎

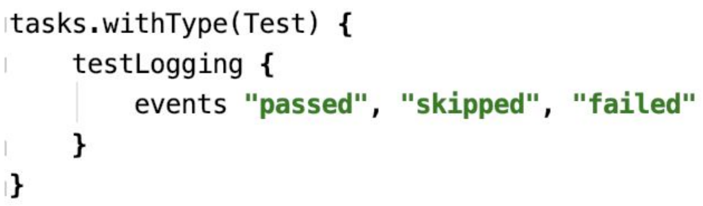
- Define the **source code** paths in sourceDirectories.
- For executionData, define the path to test run reports needed for Jacoco to generate a report stored in module's build folder.

Upgrade Jacoco

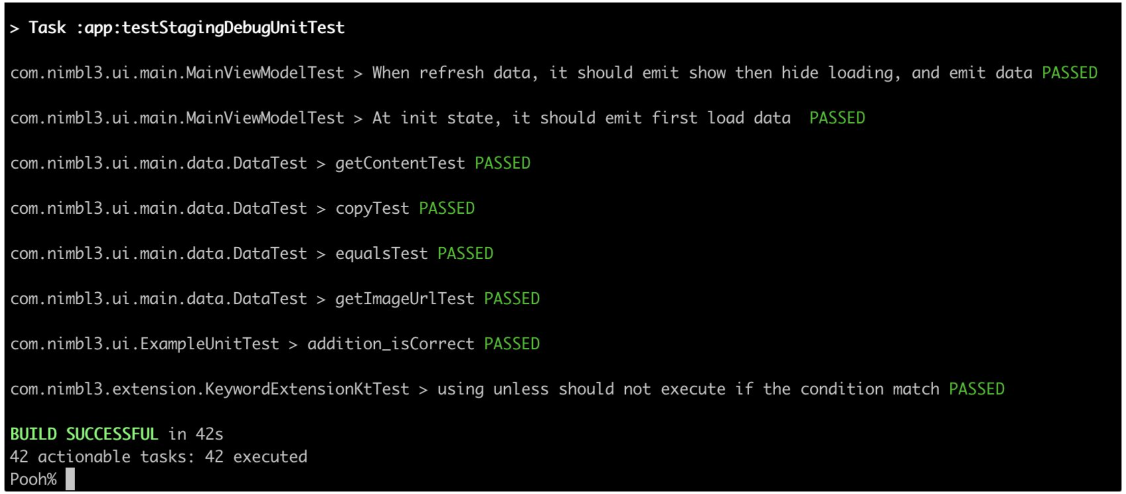
Last step 💎

Create a task to **log the results** when running/executing tests &

Run ./gradlew jacocoTestReport testStagingDebugUnitTest



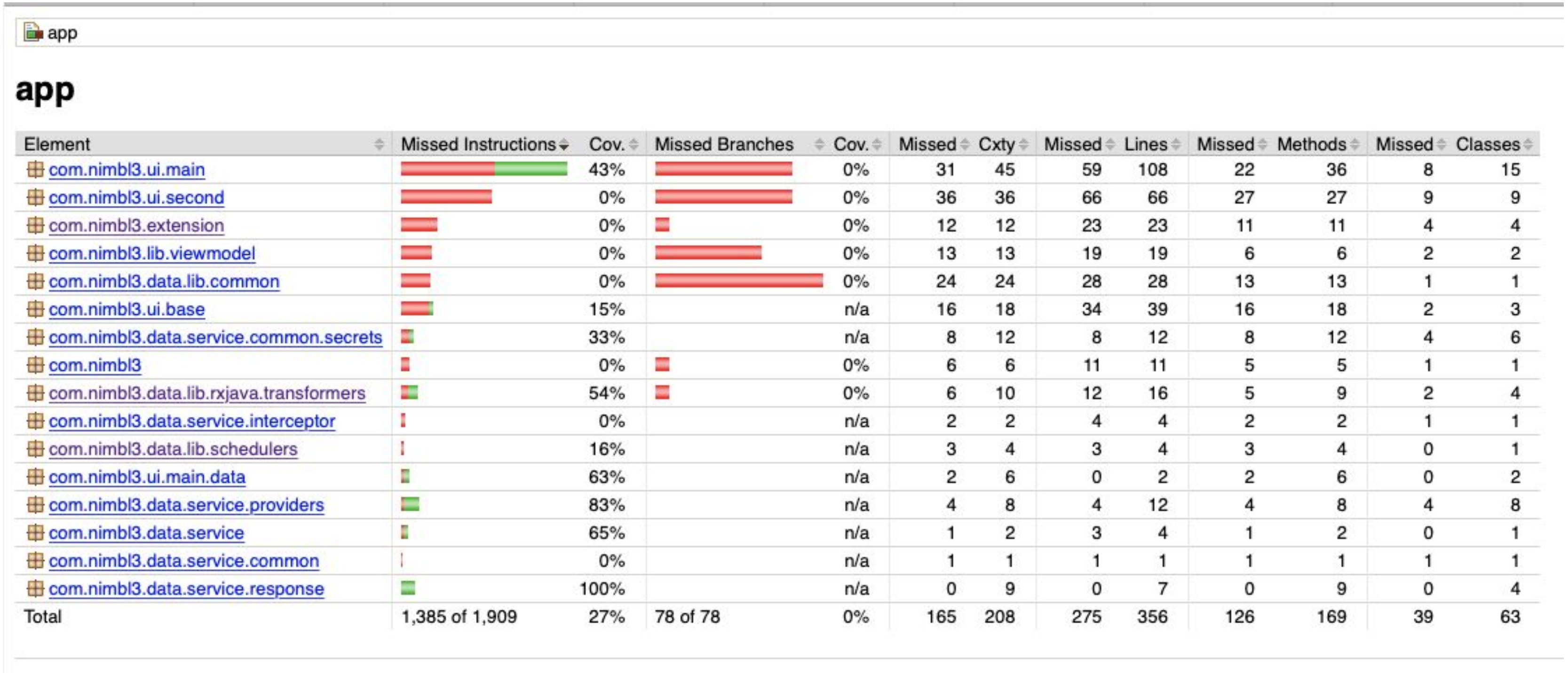
```
tasks.withType(Test) {
    testLogging {
        events "passed", "skipped", "failed"
    }
}
```



```
> Task :app:testStagingDebugUnitTest

com.nimbl3.ui.main.MainViewModelTest > When refresh data, it should emit show then hide loading, and emit data PASSED

com.nimbl3.ui.main.MainViewModelTest > At init state, it should emit first load data  PASSED

com.nimbl3.ui.main.data.DataTest > getContentTest PASSED

com.nimbl3.ui.main.data.DataTest > copyTest PASSED

com.nimbl3.ui.main.data.DataTest > equalsTest PASSED

com.nimbl3.ui.main.data.DataTest > getImageUrlTest PASSED

com.nimbl3.ui.ExampleUnitTest > addition_isCorrect PASSED

com.nimbl3.extension.KeywordExtensionKtTest > using unless should not execute if the condition match PASSED

BUILD SUCCESSFUL in 42s
42 actionable tasks: 42 executed
Pooh%
```

Upgrade Jacoco

The final report will include test coverage for both modules like this.

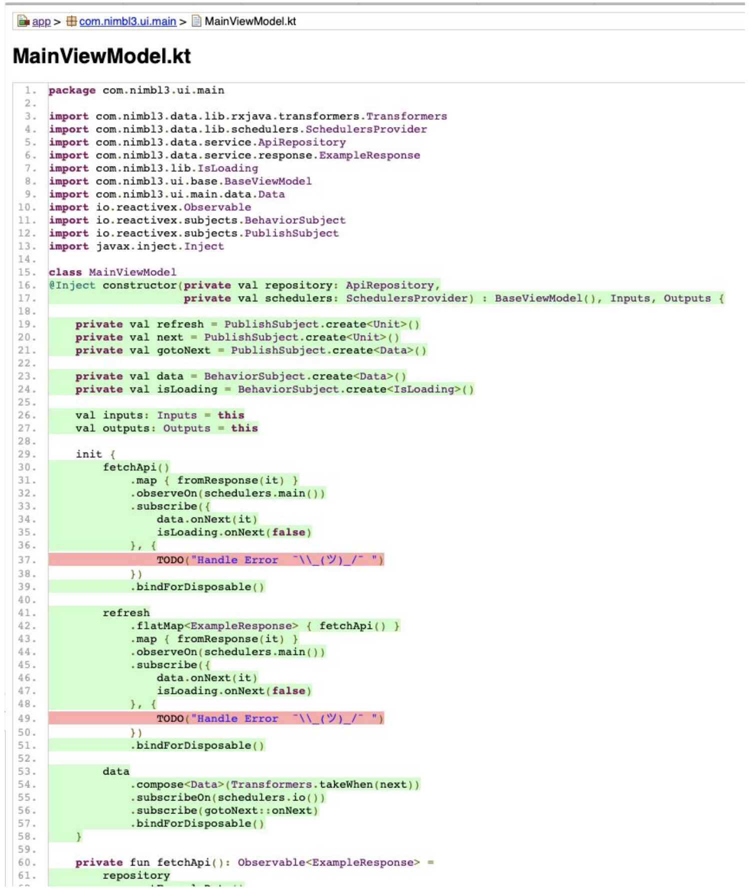


app

app

Element	Missed Instructions	Cov.	Missed Branches	Cov.	Missed	Cxty	Missed	Lines	Missed	Methods	Missed	Classes
com.nimbl3.ui.main		43%		0%	31	45	59	108	22	36	8	15
com.nimbl3.ui.second		0%		0%	36	36	66	66	27	27	9	9
com.nimbl3.extension		0%		0%	12	12	23	23	11	11	4	4
com.nimbl3.lib.viewmodel		0%		0%	13	13	19	19	6	6	2	2
com.nimbl3.data.lib.common		0%		0%	24	24	28	28	13	13	1	1
com.nimbl3.ui.base		15%		n/a	16	18	34	39	16	18	2	3
com.nimbl3.data.service.common.secrets		33%		n/a	8	12	8	12	8	12	4	6
com.nimbl3		0%		0%	6	6	11	11	5	5	1	1
com.nimbl3.data.lib.rxjava.transformers		54%		0%	6	10	12	16	5	9	2	4
com.nimbl3.data.service.interceptor		0%		n/a	2	2	4	4	2	2	1	1
com.nimbl3.data.lib.schedulers		16%		n/a	3	4	3	4	3	4	0	1
com.nimbl3.ui.main.data		63%		n/a	2	6	0	2	2	6	0	2
com.nimbl3.data.service.providers		83%		n/a	4	8	4	12	4	8	4	8
com.nimbl3.data.service		65%		n/a	1	2	3	4	1	2	0	1
com.nimbl3.data.service.common		0%		n/a	1	1	1	1	1	1	1	1
com.nimbl3.data.service.response		100%		n/a	0	9	0	7	0	9	0	4
Total	1,385 of 1,909	27%	78 of 78	0%	165	208	275	356	126	169	39	63

Upgrade Jacoco



app > com.nimbl3.ui.main > MainViewModel.kt

MainViewModel.kt

```kotlin
package com.nimbl3.ui.main

import com.nimbl3.data.lib.rxjava.transformers.Transformers
import com.nimbl3.data.lib.schedulers.SchedulersProvider
import com.nimbl3.data.service.ApiRepository
import com.nimbl3.data.service.response.ExampleResponse
import com.nimbl3.lib.IsLoading
import com.nimbl3.ui.base.BaseViewModel
import com.nimbl3.ui.main.data.Data
import io.reactivex.Observable
import io.reactivex.subjects.BehaviorSubject
import io.reactivex.subjects.PublishSubject
import javax.inject.Inject

class MainViewModel
@Inject constructor(private val repository: ApiRepository,
                    private val schedulers: SchedulersProvider) : BaseViewModel(), Inputs, Outputs {

    private val refresh = PublishSubject.create<Unit>()
    private val next = PublishSubject.create<Unit>()
    private val gotoNext = PublishSubject.create<Data>()

    private val data = BehaviorSubject.create<Data>()
    private val isLoading = BehaviorSubject.create<IsLoading>()

    val inputs: Inputs = this
    val outputs: Outputs = this

    init {
        fetchApi()
            .map { fromResponse(it) }
            .observeOn(schedulers.main())
            .subscribe({
                data.onNext(it)
                isLoading.onNext(false)
            }, {
                TODO("Handle Error  ¯\\_(ツ)_/¯ ")
            })
            .bindForDisposable()

        refresh
            .flatMap<ExampleResponse> { fetchApi() }
            .map { fromResponse(it) }
            .observeOn(schedulers.main())
            .subscribe({
                data.onNext(it)
                isLoading.onNext(false)
            }, {
                TODO("Handle Error  ¯\\_(ツ)_/¯ ")
            })
            .bindForDisposable()

        data
            .compose<Data>(Transformers.takeWhen(next))
            .subscribeOn(schedulers.io())
            .subscribe(gotoNext::onNext)
            .bindForDisposable()
    }

    private fun fetchApi(): Observable<ExampleResponse> =
        repository
```

- The **highlighted** code provided by jacoco, will show us which parts of the code have not been covered yet.
- The **red lines** indicate that we should write more test functions for these parts.

Introduction - Android Bootstrap

An automated **shell script** that will set-up **our** standardized coding environment for you:

- Simplify the **onboarding** process, whenever new developers join the company
- Simplify the **project kick-off** process, whenever new projects are started
- Keeping all our developers **aligned** on the same coding environment

What’s (currently) included in this bootstrap script?

1. Set up **IDE related** configurations:

   → These configurations apply for **all projects**

   - Ensure line feed at file end on Save
   - Useful plugins

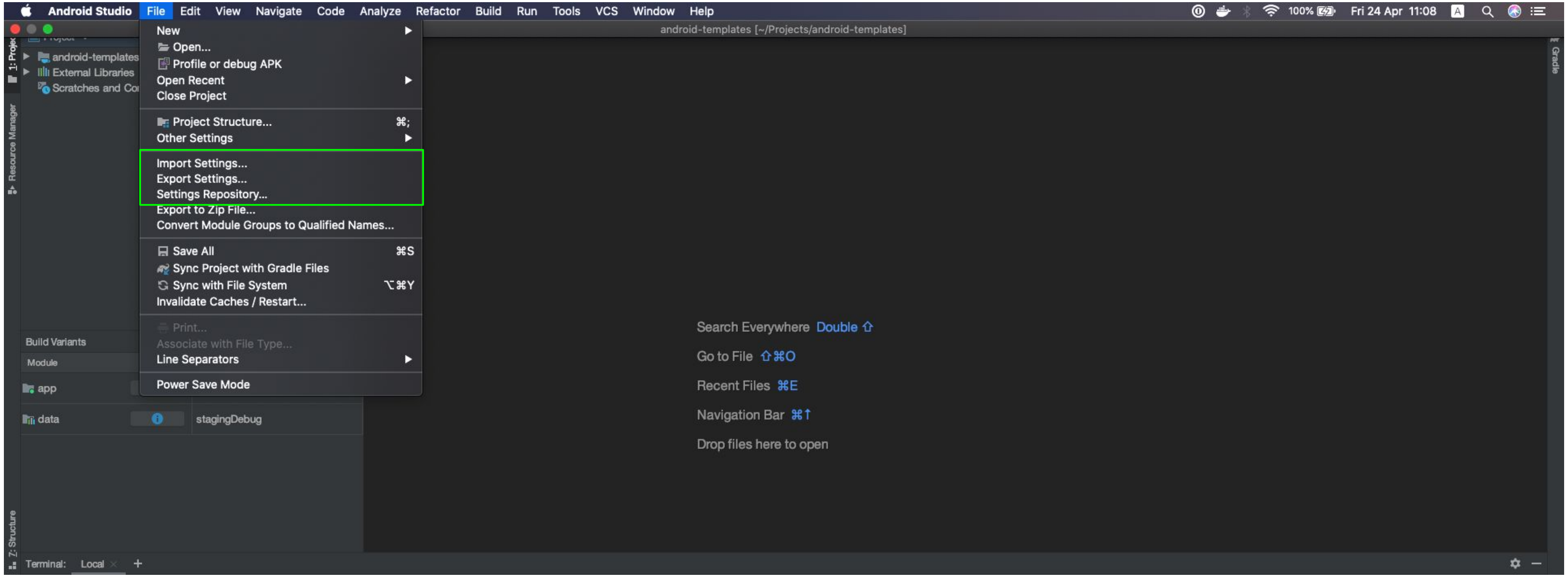
2. Set up **project specific** configurations:

   → These configurations only apply for that **specific project** you’re working on

   - Java, Kotlin, Groovy & XML code schemes
   - Add unambiguous imports on the fly (Java & Kotlin)
   - Optimize imports on the fly (Java & Kotlin)

Why don’t we make use of the Setting features of IntelliJ instead?

Importing/Exporting/Syncing settings, **only** includes **IDE** settings, **no project specific** settings!



How does android bootstrap script work?

All configurations are stored in form of an **XML** file:

1. Where are **IDE** related configurations stored?

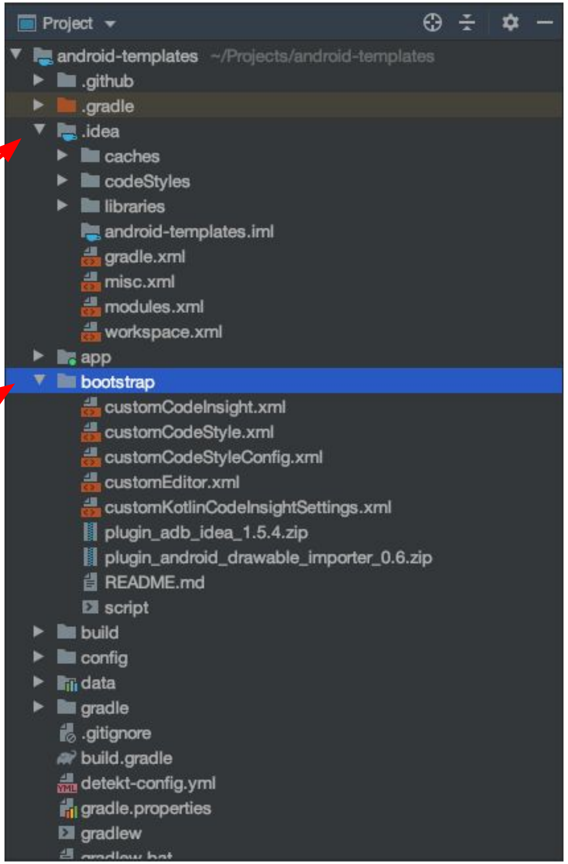
```
cd Users/toby/Library/Preferences/AndroidStudio3.6
cd Users/toby/Library/Application Support/AndroidStudio3.6
```

2. Where are **project specific** configurations stored?

```
cd $PROJECT_ROOT/.idea
```

3. Where are the **bootstrap components** stored?

```
cd $PROJECT_ROOT/bootstrap
```



```
Project
▼ android-templates ~/Projects/android-templates
  ▶ .github
  ▶ .gradle
  ▼ .idea
    ▶ caches
    ▶ codeStyles
    ▶ libraries
      android-templates.iml
      gradle.xml
      misc.xml
      modules.xml
      workspace.xml
  ▶ app
  ▼ bootstrap
      customCodeInsight.xml
      customCodeStyle.xml
      customCodeStyleConfig.xml
      customEditor.xml
      customKotlinCodeInsightSettings.xml
      plugin_adb_idea_1.5.4.zip
      plugin_android_drawable_importer_0.6.zip
      README.md
      script
  ▶ build
  ▶ config
  ▶ data
  ▶ gradle
    .gitignore
    build.gradle
    detekt-config.yml
    gradle.properties
    gradlew
```

What’s next for this android bootstrap script?

1. Proper **testing** on multiple devices, as every environment is different
2. Handle **error cases**, by double checking everything before finalizing
3. Download the plugins **dynamically**
4. Proper **documentation**, so it can easily be maintained
5. Gradually update the code styling
6. Adding **live**

   → IntelliJ feature to auto-complete code structures

Get more value with less effort

Thanks!

Contact Nimble

nimblehq.co

hello@nimblehq.co

Bangkok

399 Interchange 21 Sukhumvit Road, Unit #2402-03, Klong Toei, Wattana, Bangkok 10110, Thailand

Singapore

28C Stanley St, Singapore 068737

Hong Kong

20th Floor, Central Tower
28 Queen's Road, Central, Hong Kong

nimble