ANDROID CODE COVERAGE

Michal Jeníček, Android Engineer at STRV

STRV

“Code coverage is not your enemy. It’s a tool to help you with development.“

JACOCO JAVA CODE COVERAGE

- Based on byte code, thus is for Kotlin too

MISSED INSTRUCTIONS & BRANCHES

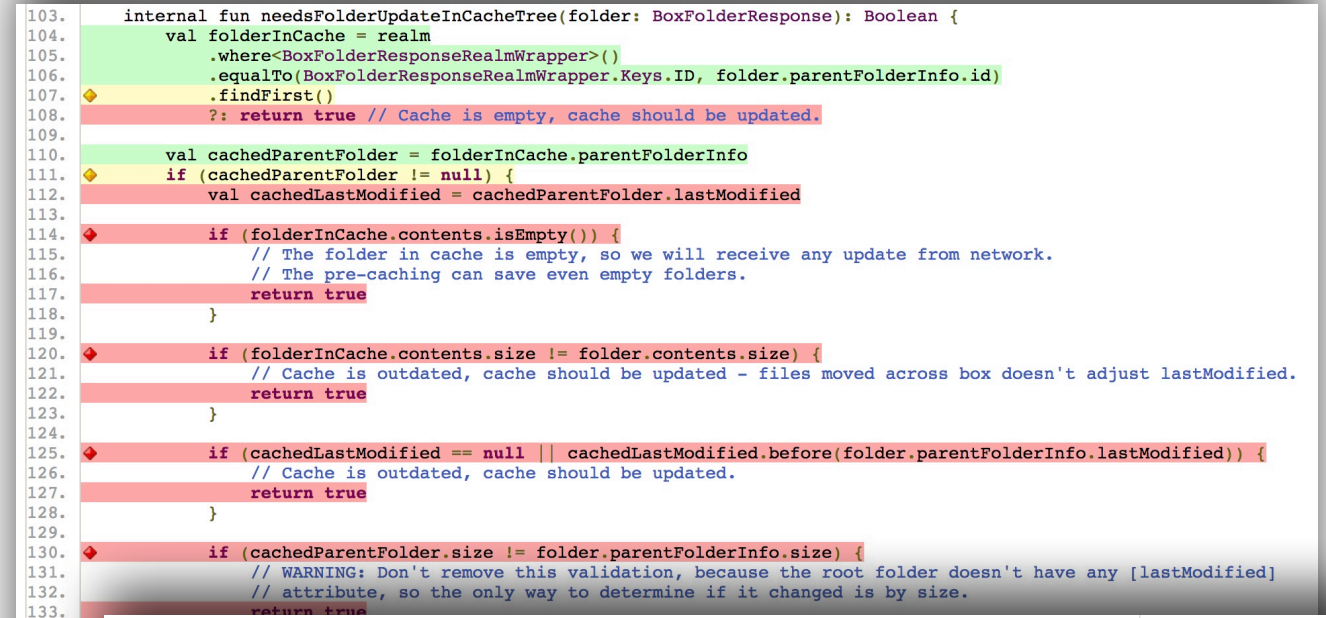
- JaCoCo offers two main metrics

	Missed Instructions	Cov.	Missed Branches	Cov.	Missed	Cxty	Missed	Lines
		38%		31%	121	199	143	257
ox		30%		12%	203	266	60	125
rer		3%		0%	133	137	144	152
		58%		39%	122	205	101	252
rer.actions.folder.createedit		0%		0%	89	89	98	98
g		18%		0%	58	69	57	62
rer.actions.transfer		0%		0%	55	55	70	70
r		32%		18%	29	38	29	46
rer.actions.folder		0%		0%	22	22	17	17
rer.actions.file		0%		0%	7	7	14	14
e.realm		0%		n/a	5	5	2	2
		0%		n/a	2	2	1	1
		100%		n/a	0	2	0	1
	7,420 of 10,451	29%	602 of 726	17%	846	1,096	736	1,097

CODE COVERAGE SAMPLE

```kotlin
internal fun needsFolderUpdateInCache(folder: BoxFolderResponse): Boolean {
    val folderInCache = realm
        .where<BoxFolderResponseRealmWrapper>()
        .equalTo(BoxFolderResponseRealmWrapper.Keys.ID, folder.parentFolderInfo.id)
        .findFirst()
        ?: return true // Cache is empty, cache should be updated.

    val cachedParentFolder = folderInCache.parentFolderInfo
    if (cachedParentFolder != null) {
        val cachedLastModified = cachedParentFolder.lastModified

        if (folderInCache.contents.isEmpty()) {
            // The folder in cache is empty, so we will receive any update from network.
            // The pre-caching can save even empty folders.
            return true
        }

        if (folderInCache.contents.size != folder.contents.size) {
            // Cache is outdated, cache should be updated - files moved across box doesn't adjust lastModified.
            return true
        }

        if (cachedLastModified == null || cachedLastModified.before(folder.parentFolderInfo.lastModified)) {
            // Cache is outdated, cache should be updated.
            return true
        }

        if (cachedParentFolder.size != folder.parentFolderInfo.size) {
            // WARNING: Don't remove this validation, because the root folder doesn't have any [lastModified]
            // attribute, so the only way to determine if it changed is by size.
            return true
        }

        refreshFolderCacheTimestamp(folderInCache)
        return false
    }
    // Cached folder does not contain enough information to compare folders, cache should be updated.
    return true
}
```

```
    internal fun needsFolderUpdateInCacheTree(folder: BoxFolderResponse): Boolean {
        val folderInCache = realm
            .where<BoxFolderResponseRealmWrapper>()
            .equalTo(BoxFolderResponseRealmWrapper.Keys.ID, folder.parentFolderInfo.id)
            .findFirst()
            ?: return true // Cache is empty, cache should be updated.

        val cachedParentFolder = folderInCache.parentFolderInfo
        if (cachedParentFolder != null) {
            val cachedLastModified = cachedParentFolder.lastModified

            if (folderInCache.contents.isEmpty()) {
                // The folder in cache is empty, so we will receive any update from network.
                // The pre-caching can save even empty folders.
                return true
            }

            if (folderInCache.contents.size != folder.contents.size) {
                // Cache is outdated, cache should be updated - files moved across box doesn't adjust lastModified.
                return true
            }

            if (cachedLastModified == null || cachedLastModified.before(folder.parentFolderInfo.lastModified)) {
                // Cache is outdated, cache should be updated.
                return true
            }

            if (cachedParentFolder.size != folder.parentFolderInfo.size) {
                // WARNING: Don't remove this validation, because the root folder doesn't have any [lastModified]
                // attribute, so the only way to determine if it changed is by size.
                return true
            }

            refreshFolderCacheTimestamp(folderInCache)
            return false
        }
        // Cached folder does not contain enough information to compare folders, cache should be updated.
        return true
    }
```

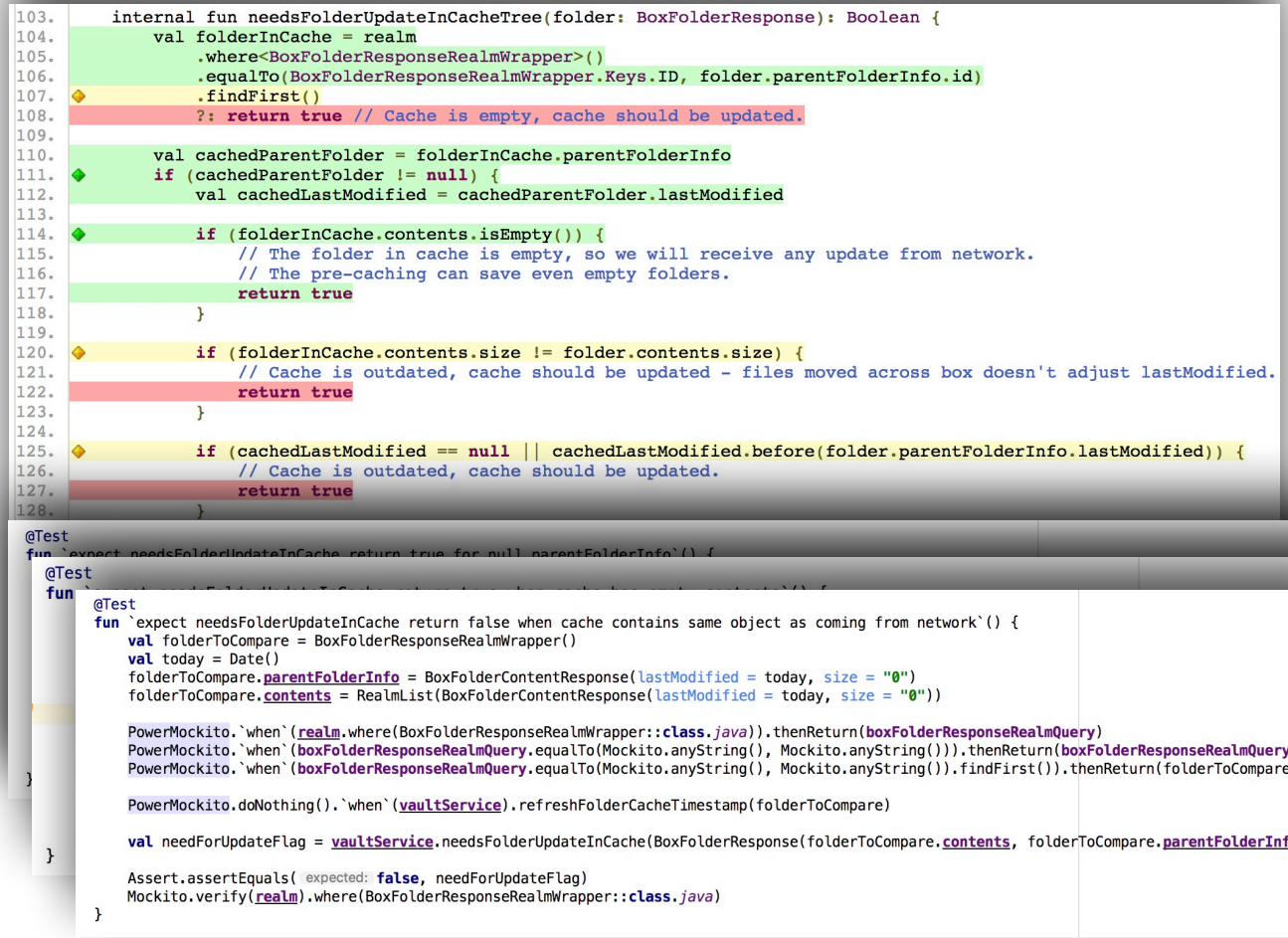


```
internal fun needsFolderUpdateInCacheTree(folder: BoxFolderResponse): Boolean {
    val folderInCache = realm
        .where<BoxFolderResponseRealmWrapper>()
        .equalTo(BoxFolderResponseRealmWrapper.Keys.ID, folder.parentFolderInfo.id)
        .findFirst()
        ?: return true // Cache is empty, cache should be updated.

    val cachedParentFolder = folderInCache.parentFolderInfo
    if (cachedParentFolder != null) {
        val cachedLastModified = cachedParentFolder.lastModified

        if (folderInCache.contents.isEmpty()) {
            // The folder in cache is empty, so we will receive any update from network.
            // The pre-caching can save even empty folders.
            return true
        }

        if (folderInCache.contents.size != folder.contents.size) {
            // Cache is outdated, cache should be updated - files moved across box doesn't adjust lastModified.
            return true
        }

        if (cachedLastModified == null || cachedLastModified.before(folder.parentFolderInfo.lastModified)) {
            // Cache is outdated, cache should be updated.
            return true
        }

        if (cachedParentFolder.size != folder.parentFolderInfo.size) {
            // WARNING: Don't remove this validation, because the root folder doesn't have any [lastModified]
            // attribute, so the only way to determine if it changed is by size.
            return true
```

```
@Test
fun `expect needsFolderUpdateInCache return true for null parentFolderInfo`() {
    val folderWithNullParentFolderInfo = BoxFolderResponseRealmWrapper()
    folderWithNullParentFolderInfo.parentFolderInfo = null

    PowerMockito.`when`(realm.where(BoxFolderResponseRealmWrapper::class.java)).thenReturn(boxFolderResponseRealmQuery)
    PowerMockito.`when`(boxFolderResponseRealmQuery.equalTo(Mockito.anyString(), Mockito.anyString())).thenReturn(boxFolderResponseRealmQuery)
    PowerMockito.`when`(boxFolderResponseRealmQuery.equalTo(Mockito.anyString(), Mockito.anyString()).findFirst()).thenReturn(folderWithNullParentFolderInfo)

    val needForUpdateFlag = vaultService.needsFolderUpdateInCache(BoxFolderResponse(listOf(BoxFolderContentResponse()), BoxFolderContentResponse()))

    Assert.assertEquals( expected: true, needForUpdateFlag)
    Mockito.verify(realm).where(BoxFolderResponseRealmWrapper::class.java)
}
```

```
103.    internal fun needsFolderUpdateInCacheTree(folder: BoxFolderResponse): Boolean {
104.        val folderInCache = realm
105.            .where<BoxFolderResponseRealmWrapper>()
106.            .equalTo(BoxFolderResponseRealmWrapper.Keys.ID, folder.parentFolderInfo.id)
107.            .findFirst()
108.            ?: return true // Cache is empty, cache should be updated.
109.
110.        val cachedParentFolder = folderInCache.parentFolderInfo
111.        if (cachedParentFolder != null) {
112.            val cachedLastModified = cachedParentFolder.lastModified
113.
114.            if (folderInCache.contents.isEmpty()) {
115.                // The folder in cache is empty, so we will receive any update from network.
116.                // The pre-caching can save even empty folders.
117.                return true
118.            }
119.
120.            if (folderInCache.contents.size != folder.contents.size) {
121.                // Cache is outdated, cache should be updated - files moved across box doesn't adjust lastModified.
122.                return true
123.            }
124.
125.            if (cachedLastModified == null || cachedLastModified.before(folder.parentFolderInfo.lastModified)) {
126.                // Cache is outdated, cache should be updated.
127.                return true
128.            }
129.
130.            if (cachedParentFolder.size != folder.parentFolderInfo.size) {
```

```
@Test
@Test
fun `expect needsFolderUpdateInCache return true when cache has empty contents`() {
    val folderToCompare = BoxFolderResponseRealmWrapper()
    folderToCompare.parentFolderInfo = BoxFolderContentResponse(lastModified = Date(), size = "0")
    folderToCompare.contents = RealmList() // Empty contents

    PowerMockito.`when`(realm.where(BoxFolderResponseRealmWrapper::class.java)).thenReturn(boxFolderResponseRealmQuery)
    PowerMockito.`when`(boxFolderResponseRealmQuery.equalTo(Mockito.anyString(), Mockito.anyString())).thenReturn(boxFolderResponseRealmQuery)
    PowerMockito.`when`(boxFolderResponseRealmQuery.equalTo(Mockito.anyString(), Mockito.anyString()).findFirst()).thenReturn(folderToCompare)

    val needForUpdateFlag = vaultService.needsFolderUpdateInCache(BoxFolderResponse(folderToCompare.contents, folderToCompare.parentFolderInfo!!))

    Assert.assertEquals( expected: true, needForUpdateFlag)
    Mockito.verify(realm).where(BoxFolderResponseRealmWrapper::class.java)
}
```

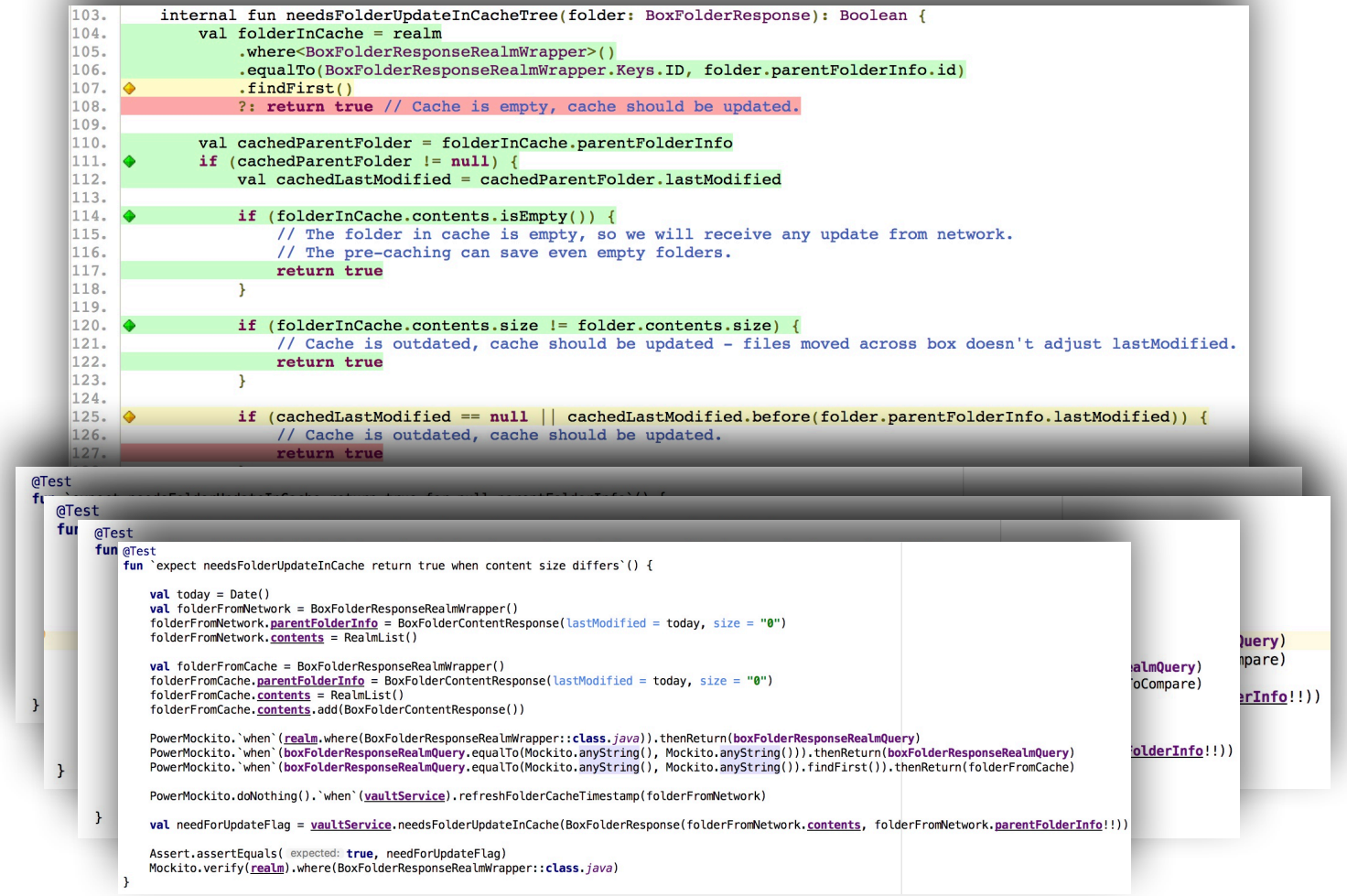


```
internal fun needsFolderUpdateInCacheTree(folder: BoxFolderResponse): Boolean {
    val folderInCache = realm
        .where<BoxFolderResponseRealmWrapper>()
        .equalTo(BoxFolderResponseRealmWrapper.Keys.ID, folder.parentFolderInfo.id)
        .findFirst()
        ?: return true // Cache is empty, cache should be updated.

    val cachedParentFolder = folderInCache.parentFolderInfo
    if (cachedParentFolder != null) {
        val cachedLastModified = cachedParentFolder.lastModified

        if (folderInCache.contents.isEmpty()) {
            // The folder in cache is empty, so we will receive any update from network.
            // The pre-caching can save even empty folders.
            return true
        }

        if (folderInCache.contents.size != folder.contents.size) {
            // Cache is outdated, cache should be updated - files moved across box doesn't adjust lastModified.
            return true
        }

        if (cachedLastModified == null || cachedLastModified.before(folder.parentFolderInfo.lastModified)) {
            // Cache is outdated, cache should be updated.
            return true
        }
```

```
@Test
fun `expect needsFolderUpdateInCache return true for null parentFolderInfo`() {
```

```
@Test
fun
```

```
@Test
fun `expect needsFolderUpdateInCache return false when cache contains same object as coming from network`() {
    val folderToCompare = BoxFolderResponseRealmWrapper()
    val today = Date()
    folderToCompare.parentFolderInfo = BoxFolderContentResponse(lastModified = today, size = "0")
    folderToCompare.contents = RealmList(BoxFolderContentResponse(lastModified = today, size = "0"))

    PowerMockito.`when`(realm.where(BoxFolderResponseRealmWrapper::class.java)).thenReturn(boxFolderResponseRealmQuery)
    PowerMockito.`when`(boxFolderResponseRealmQuery.equalTo(Mockito.anyString(), Mockito.anyString())).thenReturn(boxFolderResponseRealmQuery)
    PowerMockito.`when`(boxFolderResponseRealmQuery.equalTo(Mockito.anyString(), Mockito.anyString()).findFirst()).thenReturn(folderToCompare)

    PowerMockito.doNothing().`when`(vaultService).refreshFolderCacheTimestamp(folderToCompare)

    val needForUpdateFlag = vaultService.needsFolderUpdateInCache(BoxFolderResponse(folderToCompare.contents, folderToCompare.parentFolderInfo!!))

    Assert.assertEquals( expected: false, needForUpdateFlag)
    Mockito.verify(realm).where(BoxFolderResponseRealmWrapper::class.java)
}
```



```
internal fun needsFolderUpdateInCacheTree(folder: BoxFolderResponse): Boolean {
    val folderInCache = realm
        .where<BoxFolderResponseRealmWrapper>()
        .equalTo(BoxFolderResponseRealmWrapper.Keys.ID, folder.parentFolderInfo.id)
        .findFirst()
        ?: return true // Cache is empty, cache should be updated.

    val cachedParentFolder = folderInCache.parentFolderInfo
    if (cachedParentFolder != null) {
        val cachedLastModified = cachedParentFolder.lastModified

        if (folderInCache.contents.isEmpty()) {
            // The folder in cache is empty, so we will receive any update from network.
            // The pre-caching can save even empty folders.
            return true
        }

        if (folderInCache.contents.size != folder.contents.size) {
            // Cache is outdated, cache should be updated - files moved across box doesn't adjust lastModified.
            return true
        }

        if (cachedLastModified == null || cachedLastModified.before(folder.parentFolderInfo.lastModified)) {
            // Cache is outdated, cache should be updated.
            return true
```

```
@Test
fun `expect needsFolderUpdateInCache return true when content size differs`() {

    val today = Date()
    val folderFromNetwork = BoxFolderResponseRealmWrapper()
    folderFromNetwork.parentFolderInfo = BoxFolderContentResponse(lastModified = today, size = "0")
    folderFromNetwork.contents = RealmList()

    val folderFromCache = BoxFolderResponseRealmWrapper()
    folderFromCache.parentFolderInfo = BoxFolderContentResponse(lastModified = today, size = "0")
    folderFromCache.contents = RealmList()
    folderFromCache.contents.add(BoxFolderContentResponse())

    PowerMockito.`when`(realm.where(BoxFolderResponseRealmWrapper::class.java)).thenReturn(boxFolderResponseRealmQuery)
    PowerMockito.`when`(boxFolderResponseRealmQuery.equalTo(Mockito.anyString(), Mockito.anyString())).thenReturn(boxFolderResponseRealmQuery)
    PowerMockito.`when`(boxFolderResponseRealmQuery.equalTo(Mockito.anyString(), Mockito.anyString()).findFirst()).thenReturn(folderFromCache)

    PowerMockito.doNothing().`when`(vaultService).refreshFolderCacheTimestamp(folderFromNetwork)

    val needForUpdateFlag = vaultService.needsFolderUpdateInCache(BoxFolderResponse(folderFromNetwork.contents, folderFromNetwork.parentFolderInfo!!))

    Assert.assertEquals( expected: true, needForUpdateFlag)
    Mockito.verify(realm).where(BoxFolderResponseRealmWrapper::class.java)
}
```

CODE COVERAGE - DOES IT WORTH?

Code coverage analysis will

- Help to find missing branches in unit testing
- Help to keep in mind when there will be decreasing coverage with new code

Code coverage analysis will NOT

- Ensure, you have been writing tests right

CODE COVERAGE - WHEN TO USE

Whenever tests are written

- It’s small effort to get helpful tool useful for checking our tests

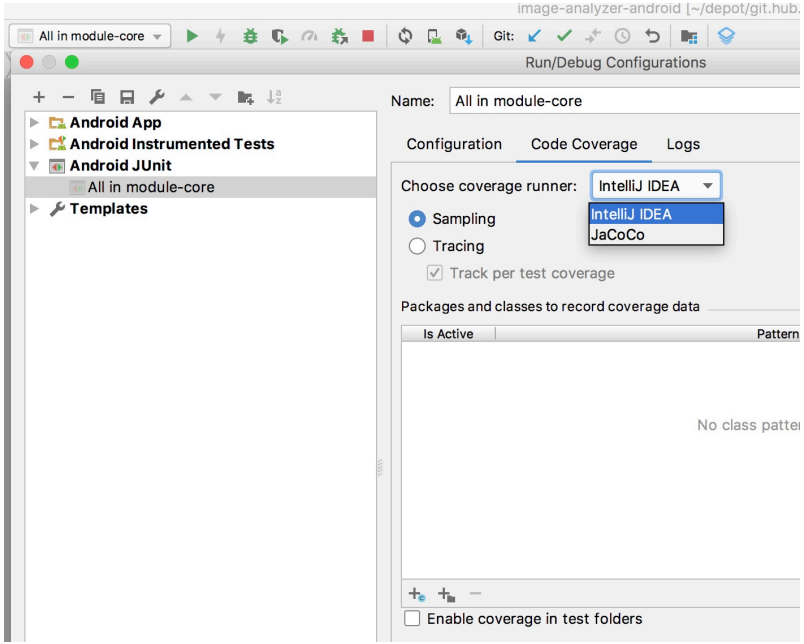
Whenever tests are executed on CI

- It’s small effort to connect code coverage to continuous integration

RUN CODE COVERAGE

Intellij IDEA

- Analyse simple project locally (no standard outputs)



GRADLE PLUGIN DEFAULT TASK

- Analyse simple project for CI (instrumentation only)

```
android {
    ...
    buildTypes {
        debug {
            testCoverageEnabled true
        }
        ...
    }
}
```

GRADLE PLUGIN CUSTOM TASK

- Analyse complex project for CI

UNIT TEST ANALYSIS

```
task jacocoTestReport(type: JacocoReport,
        dependsOn: "test${jacocoEnv.capitalize()}${jacocoBuildType.capitalize()}UnitTest") {

    group = "Reporting"
    description = "Generate Jacoco coverage reports for Debug build"

    reports {
        xml.enabled = true
        html.enabled = true
    }

    // generated classes
    classDirectories = fileTree(
            dir: "$buildDir/intermediates/classes/${jacocoEnv}/${jacocoBuildType}",
            excludes: project.excludeFromAnalysis
    ) + fileTree(
            dir: "$buildDir/tmp/kotlin-classes/${jacocoEnv}${jacocoBuildType.capitalize()}",
            excludes: project.excludeFromAnalysis
    )

    // sources
    sourceDirectories = files([
            android.sourceSets.main.java.srcDirs,
            "src/main/java", "src/${jacocoBuildType}/java", "src/server/java"
    ])
    println(">>>$buildDir/../../jacoco/jacocoAllModules.exec")
    executionData = files("$buildDir/../../jacoco/jacocoAllModules.exec")
}
```

```
task jacocoTestReport(type: JacocoReport,
        dependsOn: "test${jacocoEnv.capitalize()}${jacocoBuildType.capitalize()}UnitTest") {

    group = "Reporting"
    description = "Generate Jacoco coverage reports for Debug build"

    reports {
        xml.enabled = true
        html.enabled = true
    }

    // generated classes
    classDirectories = fileTree(
            dir: "$buildDir/intermediates/classes/${jacocoEnv}/${jacocoBuildType}",
            excludes: project.excludeFromAnalysis
    ) + fileTree(
            dir: "$buildDir/tmp/kotlin-classes/${jacocoEnv}${jacocoBuildType.capitalize()}",
            excludes: project.excludeFromAnalysis
    )

    // sources
    sourceDirectories = files([
            android.sourceSets.main.java.srcDirs,
            "src/main/java", "src/${jacocoBuildType}/java", "src/server/java"
    ])
    println(">>>$buildDir/../../jacoco/jacocoAllModules.exec")
    executionData = files("$buildDir/../../jacoco/jacocoAllModules.exec")
}
```

```
task jacocoTestReport(type: JacocoReport,
        dependsOn: "test${jacocoEnv.capitalize()}${jacocoBuildType.capitalize()}UnitTest") {

    group = "Reporting"
    description = "Generate Jacoco coverage reports for Debug build"

    reports {
        xml.enabled = true
        html.enabled = true
    }

    // generated classes
    classDirectories = fileTree(
            dir: "$buildDir/intermediates/classes/${jacocoEnv}/${jacocoBuildType}",
            excludes: project.excludeFromAnalysis
    ) + fileTree(
            dir: "$buildDir/tmp/kotlin-classes/${jacocoEnv}${jacocoBuildType.capitalize()}",
            excludes: project.excludeFromAnalysis
    )

    // sources
    sourceDirectories = files([
            android.sourceSets.main.java.srcDirs,
            "src/main/java", "src/${jacocoBuildType}/java", "src/server/java"
    ])
    println(">>>$buildDir/../../jacoco/jacocoAllModules.exec")
    executionData = files("$buildDir/../../jacoco/jacocoAllModules.exec")
}
```

```
task jacocoTestReport(type: JacocoReport,
        dependsOn: "test${jacocoEnv.capitalize()}${jacocoBuildType.capitalize()}UnitTest") {

    group = "Reporting"
    description = "Generate Jacoco coverage reports for Debug build"

    reports {
        xml.enabled = true
        html.enabled = true
    }

    // generated classes
    classDirectories = fileTree(
            dir: "$buildDir/intermediates/classes/${jacocoEnv}/${jacocoBuildType}",
            excludes: project.excludeFromAnalysis
    ) + fileTree(
            dir: "$buildDir/tmp/kotlin-classes/${jacocoEnv}${jacocoBuildType.capitalize()}",
            excludes: project.excludeFromAnalysis
    )

    // sources
    sourceDirectories = files([
            android.sourceSets.main.java.srcDirs,
            "src/main/java", "src/${jacocoBuildType}/java", "src/server/java"
    ])
    println(">>>$buildDir/../../jacoco/jacocoAllModules.exec")
    executionData = files("$buildDir/../../jacoco/jacocoAllModules.exec")
}
```

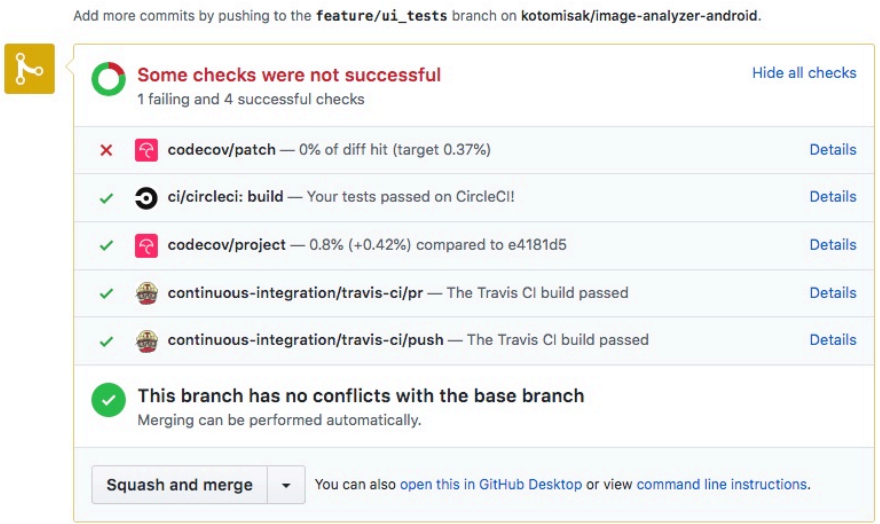
```
task jacocoTestReport(type: JacocoReport,
        dependsOn: "test${jacocoEnv.capitalize()}${jacocoBuildType.capitalize()}UnitTest") {

    group = "Reporting"
    description = "Generate Jacoco coverage reports for Debug build"

    reports {
        xml.enabled = true
        html.enabled = true
    }

    // generated classes
    classDirectories = fileTree(
            dir: "$buildDir/intermediates/classes/${jacocoEnv}/${jacocoBuildType}",
            excludes: project.excludeFromAnalysis
    ) + fileTree(
            dir: "$buildDir/tmp/kotlin-classes/${jacocoEnv}${jacocoBuildType.capitalize()}",
            excludes: project.excludeFromAnalysis
    )

    // sources
    sourceDirectories = files([
            android.sourceSets.main.java.srcDirs,
            "src/main/java", "src/${jacocoBuildType}/java", "src/server/java"
    ])
    println(">>>$buildDir/../../jacoco/jacocoAllModules.exec")
    executionData = files("$buildDir/../../jacoco/jacocoAllModules.exec")
}
```

```
ext {
    excludeFromAnalysis = [
            'android/**/*.*',
            'com/android/**/*',
            'com/bumptech/**/*',
            'io/realm/**/*',
            '**/R.class',
            '**/BR.class',
            '**/R$*.class',
            '**/*$ViewInjector*.*',
            '**/BuildConfig.*',
            '**/Manifest*.*',
            '**/*Test*.*',
            '**/*$Lambda$*.*', // Jacoco can not handle several "$" in class name.
            '**/*Module.*', // Modules for Dagger
            '**/*Dagger*.*', // Dagger auto-generated code.
            '**/*MembersInjector*.*', // Dagger auto-generated code.
            '**/*_Provide*Factory*.*',
            '**/*_Factory.*', //Dagger auto-generated code
            '**/*Fragment.*',
            '**/*Activity.*',
            'com/example/android/analyzer/**/databinding', //we don't test databinding classes
            'com/example/android/analyzer/**/di/**', //nothing to test on di
            'com/example/android/analyzer/**/glide', //exclude hard to test glide customization

            //Test ktools code after all app's business logic will be covered.
            'com/example/android/analyzer/core/ktools/**'
    ]
```

CI GIT INTEGRATION

CI BRANCH CHECKS

- Run checks with every push to branch automatically
- Merge pull request / branch based on available checks



CIRCLE CI 2.0

```
- run:
    name: CleanUp
    command: ./gradlew clean
- run:
    name: Test module-core
    command: ./gradlew module-core:jacocoTestReport -PjacocoBuildType=debug -PjacocoEnv=dev
- run:
    name: Test module-face-detection
    command: ./gradlew module-face-detection:jacocoTestReport -PjacocoBuildType=release -PjacocoEnv=dev
- run:
    name: Test module-text-recognition
    command: ./gradlew module-text-recognition:jacocoTestReport -PjacocoBuildType=release -PjacocoEnv=dev
- run:
    name: Test mobile
    command: ./gradlew mobile:jacocoTestReport -PjacocoBuildType=release -PjacocoEnv=dev

- run:
    name: Trigger codecov publishing
    command: |
      curl https://codecov.io/bash -o codecov.sh
      chmod +x codecov.sh
      ./codecov.sh -X gcov -X coveragepy -X xcode \
          -f module-core/build/reports/jacocoTestReport/jacocoTestReport.xml \
          -f module-face-detection/build/reports/jacocoTestReport/jacocoTestReport.xml \
          -f module-text-recognition/build/reports/jacocoTestReport/jacocoTestReport.xml \
          -f mobile/build/reports/jacocoTestReport/jacocoTestReport.xml

- store_artifacts:
    path: module-core/build/reports
    destination: reports

- store_artifacts:
    path: module-face-detection/build/reports
    destination: reports
```

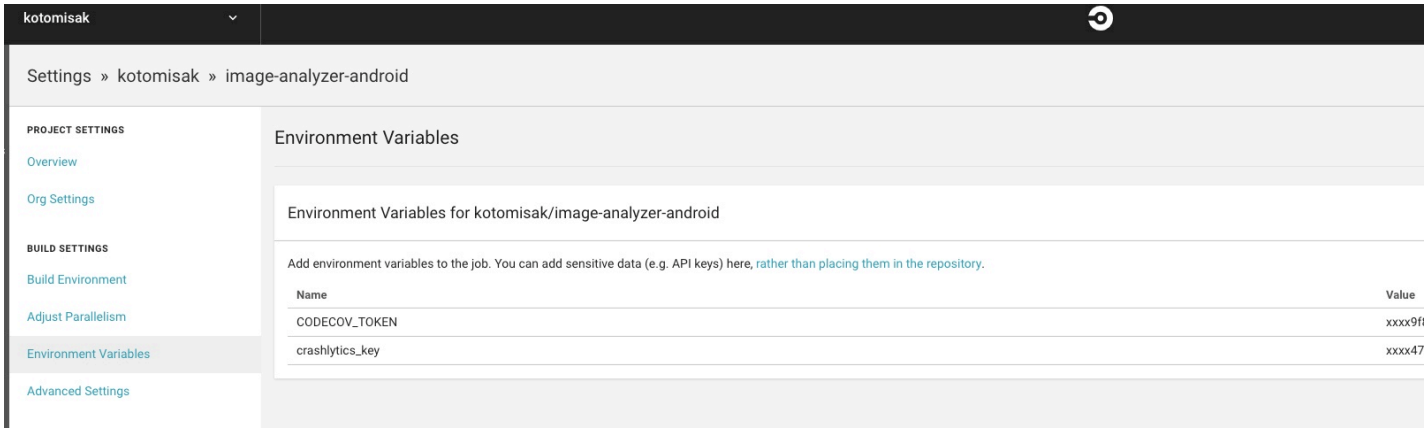
CIRCLE CI 2.0

```
- run:
    name: CleanUp
    command: ./gradlew clean
- run:
    name: Test module-core
    command: ./gradlew module-core:jacocoTestReport -PjacocoBuildType=debug -PjacocoEnv=dev
- run:
    name: Test module-face-detection
    command: ./gradlew module-face-detection:jacocoTestReport -PjacocoBuildType=release -PjacocoEnv=dev
- run:
    name: Test module-text-recognition
    command: ./gradlew module-text-recognition:jacocoTestReport -PjacocoBuildType=release -PjacocoEnv=dev
- run:
    name: Test mobile
    command: ./gradlew mobile:jacocoTestReport -PjacocoBuildType=release -PjacocoEnv=dev

- run:
    name: Trigger codecov publishing
    command: |
      curl https://codecov.io/bash -o codecov.sh
      chmod +x codecov.sh
      ./codecov.sh -X gcov -X coveragepy -X xcode \
          -f module-core/build/reports/jacocoTestReport/jacocoTestReport.xml \
          -f module-face-detection/build/reports/jacocoTestReport/jacocoTestReport.xml \
          -f module-text-recognition/build/reports/jacocoTestReport/jacocoTestReport.xml \
          -f mobile/build/reports/jacocoTestReport/jacocoTestReport.xml

- store_artifacts:
    path: module-core/build/reports
    destination: reports

- store_artifacts:
    path: module-face-detection/build/reports
    destination: reports
```

CIRCLE CI 2.0

```
- run:
    name: CleanUp
    command: ./gradlew clean
- run:
    name: Test module-core
    command: ./gradlew module-core:jacocoTestReport -PjacocoBuildType=debug -PjacocoEnv=dev
- run:
    name: Test module-face-detection
    command: ./gradlew module-face-detection:jacocoTestReport -PjacocoBuildType=release -PjacocoEnv=dev
- run:
    name: Test module-text-recognition
    command: ./gradlew module-text-recognition:jacocoTestReport -PjacocoBuildType=release -PjacocoEnv=dev
- run:
    name: Test mobile
    command: ./gradlew mobile:jacocoTestReport -PjacocoBuildType=release -PjacocoEnv=dev

- run:
    name: Trigger codecov publishing
    command: |
      curl https://codecov.io/bash -o codecov.sh
      chmod +x codecov.sh
      ./codecov.sh -X gcov -X coveragepy -X xcode \
          -f module-core/build/reports/jacocoTestReport/jacocoTestReport.xml \
          -f module-face-detection/build/reports/jacocoTestReport/jacocoTestReport.xml \
          -f module-text-recognition/build/reports/jacocoTestReport/jacocoTestReport.xml \
          -f mobile/build/reports/jacocoTestReport/jacocoTestReport.xml

- store_artifacts:
    path: module-core/build/reports
    destination: reports

- store_artifacts:
    path: module-face-detection/build/reports
    destination: reports
```

CIRCLE CI 2.0


kotomisak
Settings » kotomisak » image-analyzer-android
PROJECT SETTINGS
Overview
Org Settings
BUILD SETTINGS
Build Environment
Adjust Parallelism
Environment Variables
Advanced Settings
Environment Variables
Environment Variables for kotomisak/image-analyzer-android
Add environment variables to the job. You can add sensitive data (e.g. API keys) here, rather than placing them in the repository.
Name
Value
CODECOV_TOKEN
xxxx9f
crashlytics_key
xxxx47

TRAVIS CI

```
android:
  components:
  # Uncomment the lines below if you want to
  # use the latest revision of Android SDK Tools
  - tools
  - platform-tools
  # The BuildTools version used by your project
  - build-tools-28.0.2
  # The SDK version used to compile your project
  - android-28


  - extra-android-m2repository
  - sys-img-${ANDROID_ABI}-${ANDROID_TARGET}

  licenses:
  - 'android-sdk-preview-license-52d11cd2'
  - 'android-sdk-license-.+'
  - 'google-gdk-license-.+'

script:
- ./gradlew clean
- ./gradlew module-core:jacocoTestReport -PjacocoBuildType=debug -PjacocoEnv=dev
- ./gradlew module-face-detection:jacocoTestReport -PjacocoBuildType=release -PjacocoEnv=dev
- ./gradlew module-text-recognition:jacocoTestReport -PjacocoBuildType=release -PjacocoEnv=dev
- ./gradlew mobile:jacocoTestReport -PjacocoBuildType=release -PjacocoEnv=dev

after_success:
- bash <(curl -s https://codecov.io/bash)
```

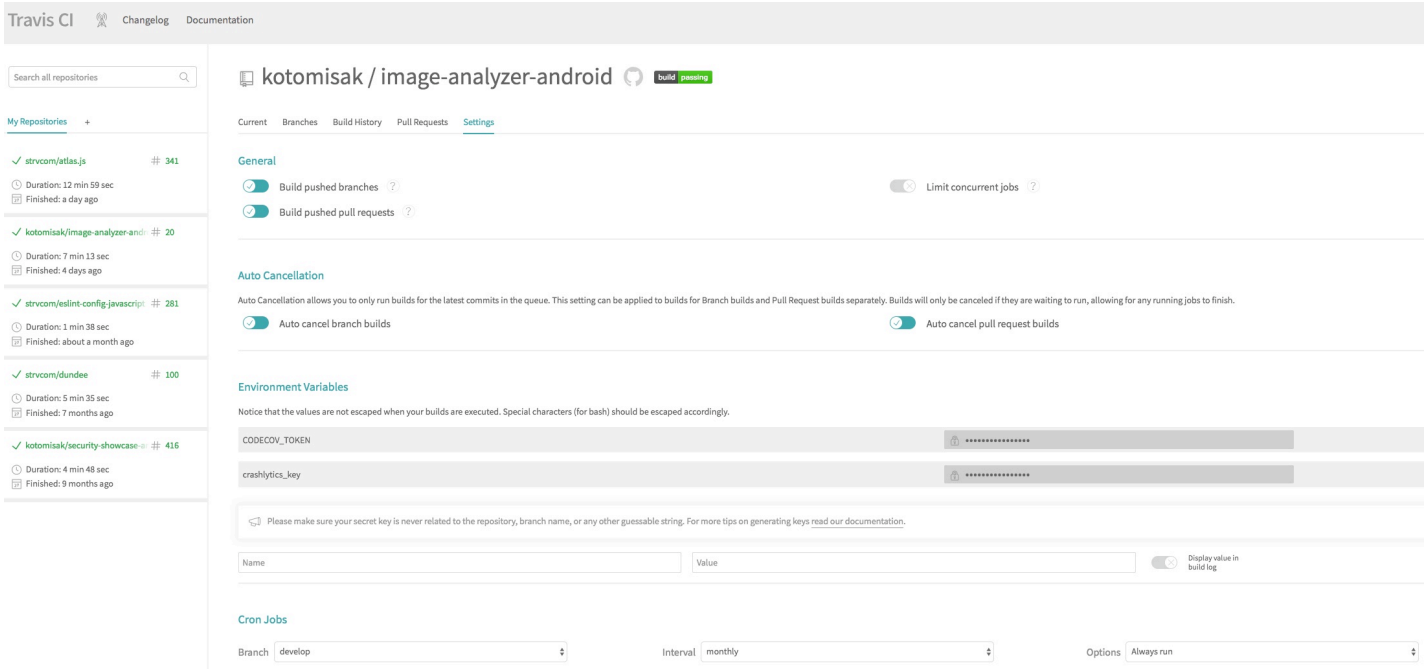
TRAVIS CI

```
android:
  components:
  # Uncomment the lines below if you want to
  # use the latest revision of Android SDK Tools
  - tools
  - platform-tools
  # The BuildTools version used by your project
  - build-tools-28.0.2
  # The SDK version used to compile your project
  - android-28


  - extra-android-m2repository
  - sys-img-${ANDROID_ABI}-${ANDROID_TARGET}

  licenses:
  - 'android-sdk-preview-license-52d11cd2'
  - 'android-sdk-license-.+'
  - 'google-gdk-license-.+'

script:
- ./gradlew clean
- ./gradlew module-core:jacocoTestReport -PjacocoBuildType=debug -PjacocoEnv=dev
- ./gradlew module-face-detection:jacocoTestReport -PjacocoBuildType=release -PjacocoEnv=dev
- ./gradlew module-text-recognition:jacocoTestReport -PjacocoBuildType=release -PjacocoEnv=dev
- ./gradlew mobile:jacocoTestReport -PjacocoBuildType=release -PjacocoEnv=dev

after_success:
- bash <(curl -s https://codecov.io/bash)
```

TRAVIS CI

```
android:
  components:
  # Uncomment the lines below if you want to
  # use the latest revision of Android SDK Tools
  - tools
  - platform-tools
  # The BuildTools version used by your project
  - build-tools-28.0.2
  # The SDK version used to compile your project
  - android-28


  - extra-android-m2repository
  - sys-img-${ANDROID_ABI}-${ANDROID_TARGET}

  licenses:
  - 'android-sdk-preview-license-52d11cd2'
  - 'android-sdk-license-.+'
  - 'google-gdk-license-.+'

script:
- ./gradlew clean
- ./gradlew module-core:jacocoTestReport -PjacocoBuildType=debug -PjacocoEnv=dev
- ./gradlew module-face-detection:jacocoTestReport -PjacocoBuildType=release -PjacocoEnv=dev
- ./gradlew module-text-recognition:jacocoTestReport -PjacocoBuildType=release -PjacocoEnv=dev
- ./gradlew mobile:jacocoTestReport -PjacocoBuildType=release -PjacocoEnv=dev

after_success:
- bash <(curl -s https://codecov.io/bash)
```

TRAVIS CI


Travis CI
Changelog
Documentation
Search all repositories
My Repositories
strvcom/atlas.js
341
Duration: 12 min 59 sec
Finished: a day ago
kotomisak/image-analyzer-and
20
Duration: 7 min 13 sec
Finished: 4 days ago
strvcom/eslint-config-javascript
281
Duration: 1 min 38 sec
Finished: about a month ago
strvcom/dundee
100
Duration: 5 min 35 sec
Finished: 7 months ago
kotomisak/security-showcase
416
Duration: 4 min 48 sec
Finished: 9 months ago
kotomisak / image-analyzer-android
build passing
Current
Branches
Build History
Pull Requests
Settings
General
Build pushed branches
Build pushed pull requests
Limit concurrent jobs
Auto Cancellation
Auto Cancellation allows you to only run builds for the latest commits in the queue. This setting can be applied to builds for Branch builds and Pull Request builds separately. Builds will only be canceled if they are waiting to run, allowing for any running jobs to finish.
Auto cancel branch builds
Auto cancel pull request builds
Environment Variables
Notice that the values are not escaped when your builds are executed. Special characters (for bash) should be escaped accordingly.
CODECOV_TOKEN
crashlytics_key
Please make sure your secret key is never related to the repository, branch name, or any other guessable string. For more tips on generating keys read our documentation.
Name
Value
Display value in build log
Cron Jobs
Branch
develop
Interval
monthly
Options
Always run

JENKINS PIPELINE

```
stage("UnitTestingWithCoverage") {
    when {
        expression {build_allowed}
    }
    steps {
        echo "Code coverage cleanup ${WORKSPACE}..."
        dir('jacoco') {
            deleteDir()
        }
        echo "Code coverage computation for Env=${params.environment} and BuildType=${params.build_type} ..."
        sh "./gradlew module-core:jacocoTestReport -PjacocoEnv=${params.environment} -PjacocoBuildType=${params.build_type}"
        sh "./gradlew module-face-detection:jacocoTestReport -PjacocoEnv=${params.environment} -PjacocoBuildType=${params.build_type}"
        sh "./gradlew module-text-recognition:jacocoTestReport -PjacocoEnv=${params.environment} -PjacocoBuildType=${params.build_type}"
        sh "./gradlew mobile:jacocoTestReport -PjacocoEnv=${params.environment} -PjacocoBuildType=${params.build_type}"

        //It's required to repeat all patterns which are defined in jacoco.gradle for analyzer. Plugin shows incorrect results otherwise.
        jacoco(execPattern: '**/jacocoAllModules.exec',
                sourcePattern: "**/src/main/java/**/*,**/src/server/java/**/*,**/src/${params.build_type}/java/**/*",
                classPattern: '**/tmp/kotlin-classes/**/*,**/intermediates/classes/**/*',
                exclusionPattern: '**/com/example/**/*$*,**/com/example/**/databinding/*,**/com/example/**/di/*,**/com/example/**/rxfinge
                inclusionPattern: '**/com/example/**/*'
        )
    }

    post {
        always {
            echo 'Publishing test results'
            archive "**/test-results/**/*"
            junit '**/test-results/**/*.xml'
            step([$class: 'Publisher'])
        }
    }
}
```

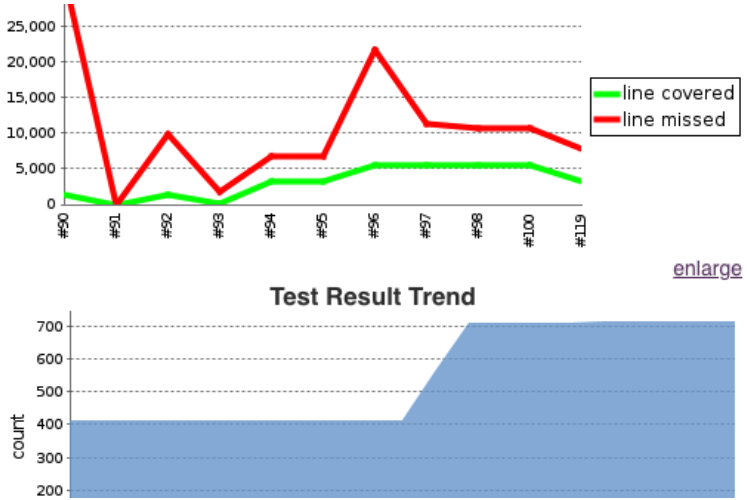
JENKINS PIPELINE

```
stage("UnitTestingWithCoverage") {
    when {
        expression {build_allowed}
    }
    steps {
        echo "Code coverage cleanup ${WORKSPACE}..."
        dir('jacoco') {
            deleteDir()
        }
        echo "Code coverage computation for Env=${params.environment} and BuildType=${params.build_type} ..."
        sh "./gradlew module-core:jacocoTestReport -PjacocoEnv=${params.environment} -PjacocoBuildType=${params.build_type}"
        sh "./gradlew module-face-detection:jacocoTestReport -PjacocoEnv=${params.environment} -PjacocoBuildType=${params.build_type}"
        sh "./gradlew module-text-recognition:jacocoTestReport -PjacocoEnv=${params.environment} -PjacocoBuildType=${params.build_type}"
        sh "./gradlew mobile:jacocoTestReport -PjacocoEnv=${params.environment} -PjacocoBuildType=${params.build_type}"

        //It's required to repeat all patterns which are defined in jacoco.gradle for analyzer. Plugin shows incorrect results otherwise.
        jacoco(execPattern: '**/jacocoAllModules.exec',
                sourcePattern: "**/src/main/java/**/*,**/src/server/java/**/*,**/src/${params.build_type}/java/**/*",
                classPattern: '**/tmp/kotlin-classes/**/*,**/intermediates/classes/**/*',
                exclusionPattern: '**/com/example/**/*$*,**/com/example/**/databinding/*,**/com/example/**/di/*,**/com/example/**/rxfinge
                inclusionPattern: '**/com/example/**/*'
        )
    }

    post {
        always {
            echo 'Publishing test results'
            archive "**/test-results/**/*"
            junit '**/test-results/**/*.xml'
            step([$class: 'Publisher'])
        }
    }
}
```

JENKINS PIPELINE

```
stage("UnitTestingWithCoverage") {
    when {
        expression {build_allowed}
    }
    steps {
        echo "Code coverage cleanup ${WORKSPACE}..."
        dir('jacoco') {
            deleteDir()
        }
        echo "Code coverage computation for Env=${params.environment} and BuildType=${params.build_type} ..."
        sh "./gradlew module-core:jacocoTestReport -PjacocoEnv=${params.environment} -PjacocoBuildType=${params.build_type}"
        sh "./gradlew module-face-detection:jacocoTestReport -PjacocoEnv=${params.environment} -PjacocoBuildType=${params.build_type}"
        sh "./gradlew module-text-recognition:jacocoTestReport -PjacocoEnv=${params.environment} -PjacocoBuildType=${params.build_type}"
        sh "./gradlew mobile:jacocoTestReport -PjacocoEnv=${params.environment} -PjacocoBuildType=${params.build_type}"

        //It's required to repeat all patterns which are defined in jacoco.gradle for analyzer. Plugin shows incorrect results otherwise.
        jacoco(execPattern: '**/jacocoAllModules.exec',
                sourcePattern: "**/src/main/java/**/*,**/src/server/java/**/*,**/src/${params.build_type}/java/**/*",
                classPattern: '**/tmp/kotlin-classes/**/*,**/intermediates/classes/**/*',
                exclusionPattern: '**/com/example/**/*$*,**/com/example/**/databinding/*,**/com/example/**/di/*,**/com/example/**/rxfinge
                inclusionPattern: '**/com/example/**/*'
        )
    }

    post {
        always {
            echo 'Publishing test results'
            archive "**/test-results/**/*"
            junit '**/test-results/**/*.xml'
            step([$class: 'Publisher'])
        }
    }
}
```

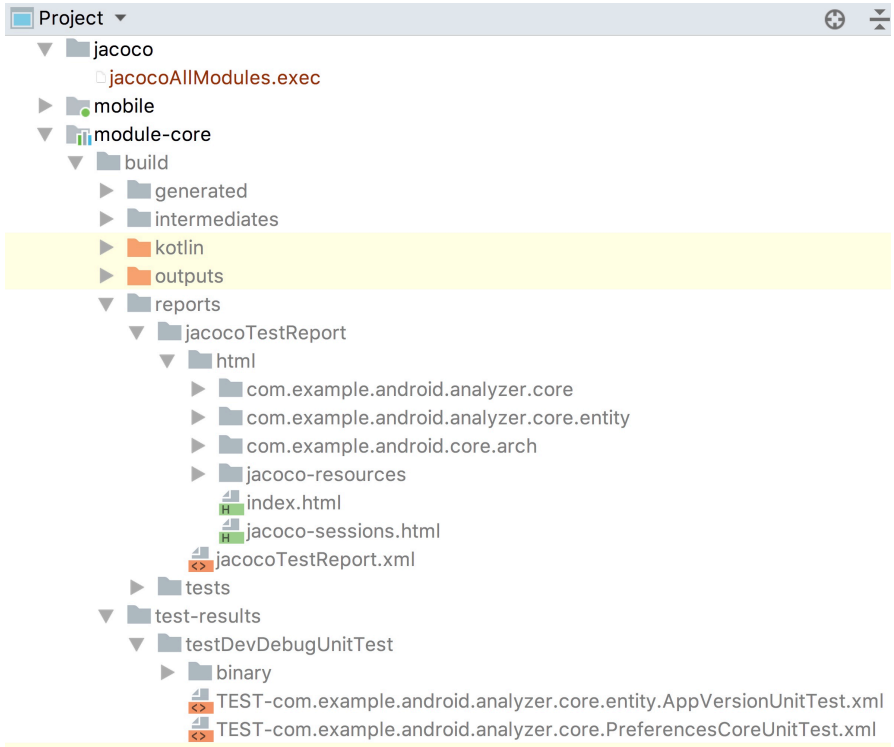
JENKINS UI



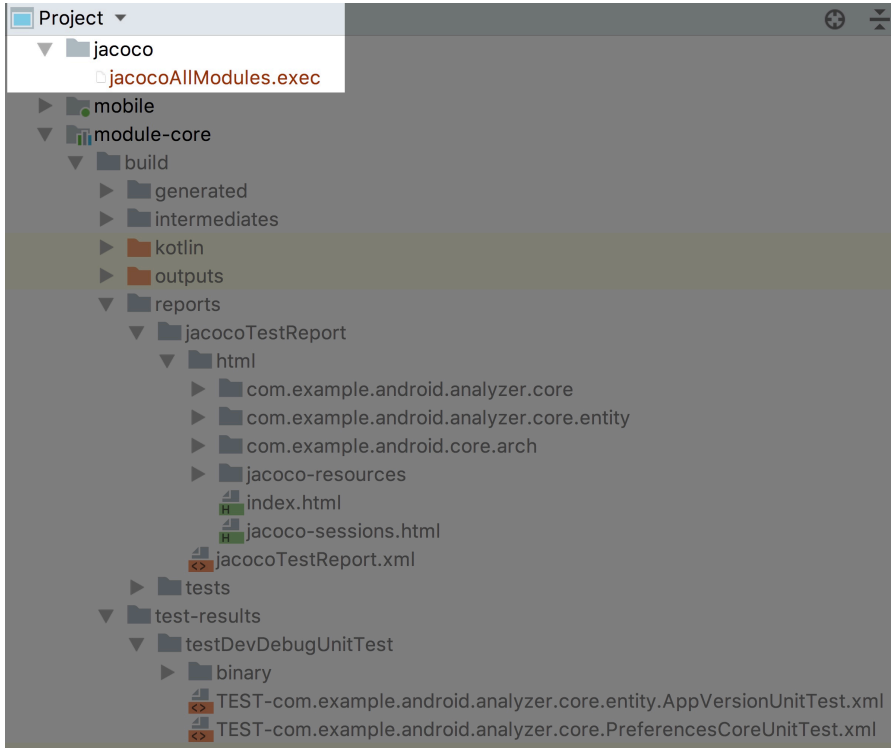
	Declarative: Tool Install	SCM	ReleaseNotes	Cleaning	UnitTestingWithCoverage	AssembleAPK	Notify
Average stage times: (Average full run time: ~42min 26s)	193ms	32s	44s	11s	11min 45s	1min 43s	816ms
#171 Sep 25 12:56 1 commits	127ms	35s	119ms	70ms	71ms	76ms	71ms
#170 Sep 25 10:56 5 commits	155ms	36s	1min 57s	16s	40s failed	73ms failed	88ms failed
#169 Sep 21 16:28 2 commits	119ms	38s	133ms	59ms	47ms	49ms	49ms
#168 Sep 21 14:50 1 commits	182ms	36s	1min 31s	19s	30min 13s	5min 54s	5s
#167 Sep 21 14:08 47 commits	243ms	32s	50s	9s	35min 46s	4min 2s	1s
#166 Sep 19 20:44 1 commits	348ms	22s	238ms	98ms	92ms	62ms	82ms
#165 Sep 19 19:58 1 commits	149ms	33s	120ms	66ms	62ms	68ms	65ms
#164 Sep 19 18:08 2 commits	135ms	30s	1min 11s	20s	28min 15s	4min 22s	973ms
#163 Sep 19 16:51 1 commits	190ms	26s	166ms	76ms	73ms	76ms	68ms

CODE COVERAGE OUTPUTS

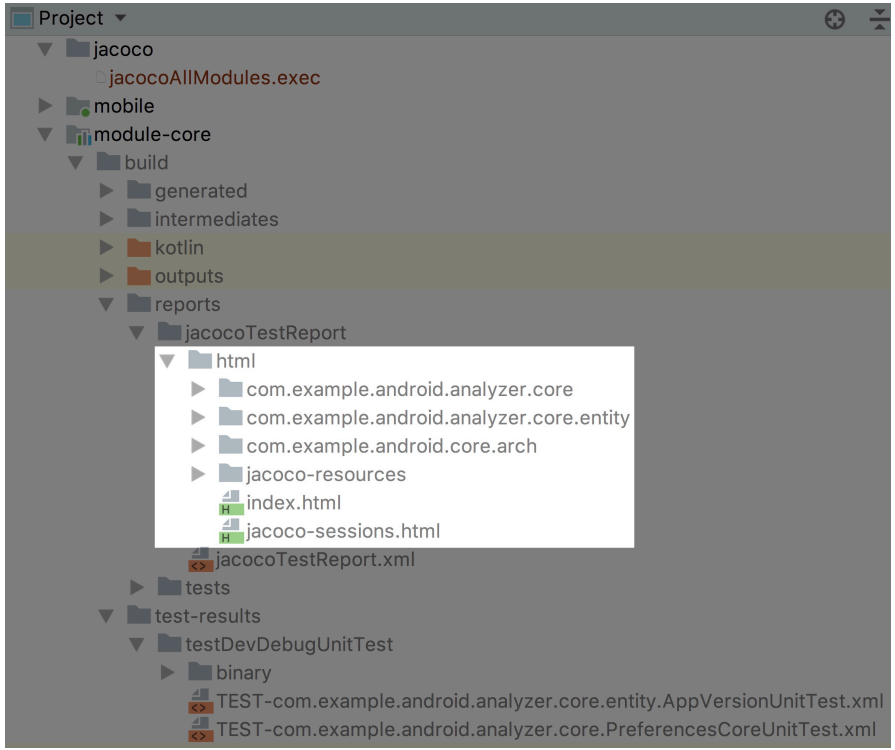
OUTPUTS STRUCTURE



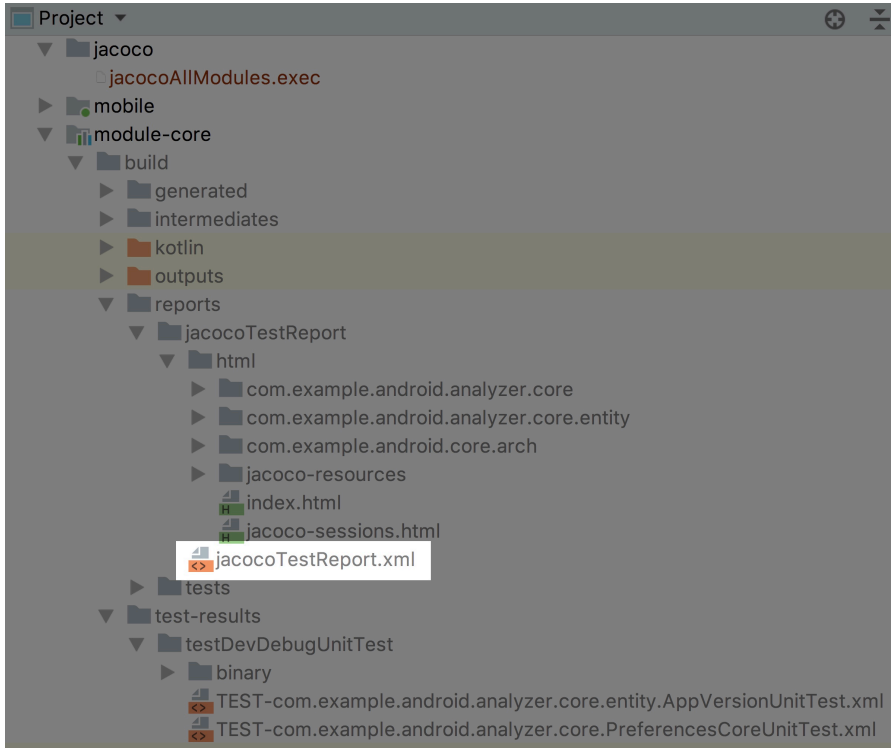
OUTPUTS STRUCTURE



OUTPUTS STRUCTURE



OUTPUTS STRUCTURE



JACOCO LOCAL

Element		Missed Instructions	Cov.	Missed Branches	Cov.	Missed	Cxty	Missed	Lines	Missed	Methods	Missed	Classes
	vault.service		38%		31%	121	199	143	257	88	156	26	53
	vault.entity.box		30%		12%	203	266	60	125	136	196	16	32
	vault.ui.explorer		3%		0%	133	137	144	152	72	76	22	24
	vault.ui		58%		39%	122	205	101	252	50	107	8	23
	vault.ui.explorer.actions.folder.createedit		0%		0%	89	89	98	98	53	53	18	18
	vault.ui.sorting		18%		0%	58	69	57	62	32	43	6	12
	vault.ui.explorer.actions.transfer		0%		0%	55	55	70	70	40	40	16	16
	vault.ui.picker		32%		18%	29	38	29	46	22	30	7	10
	vault.ui.explorer.actions.folder		0%		0%	22	22	17	17	13	13	3	3
	vault.ui.explorer.actions.file		0%		0%	7	7	14	14	6	6	1	1
	vault.database.realm		0%		n/a	5	5	2	2	5	5	1	1
	vault.tealium		0%		n/a	2	2	1	1	2	2	1	1
	vault.rest		100%		n/a	0	2	0	1	0	2	0	2
Total		7,420 of 10,451	29%	602 of 726	17%	846	1,096	736	1,097	519	729	125	196

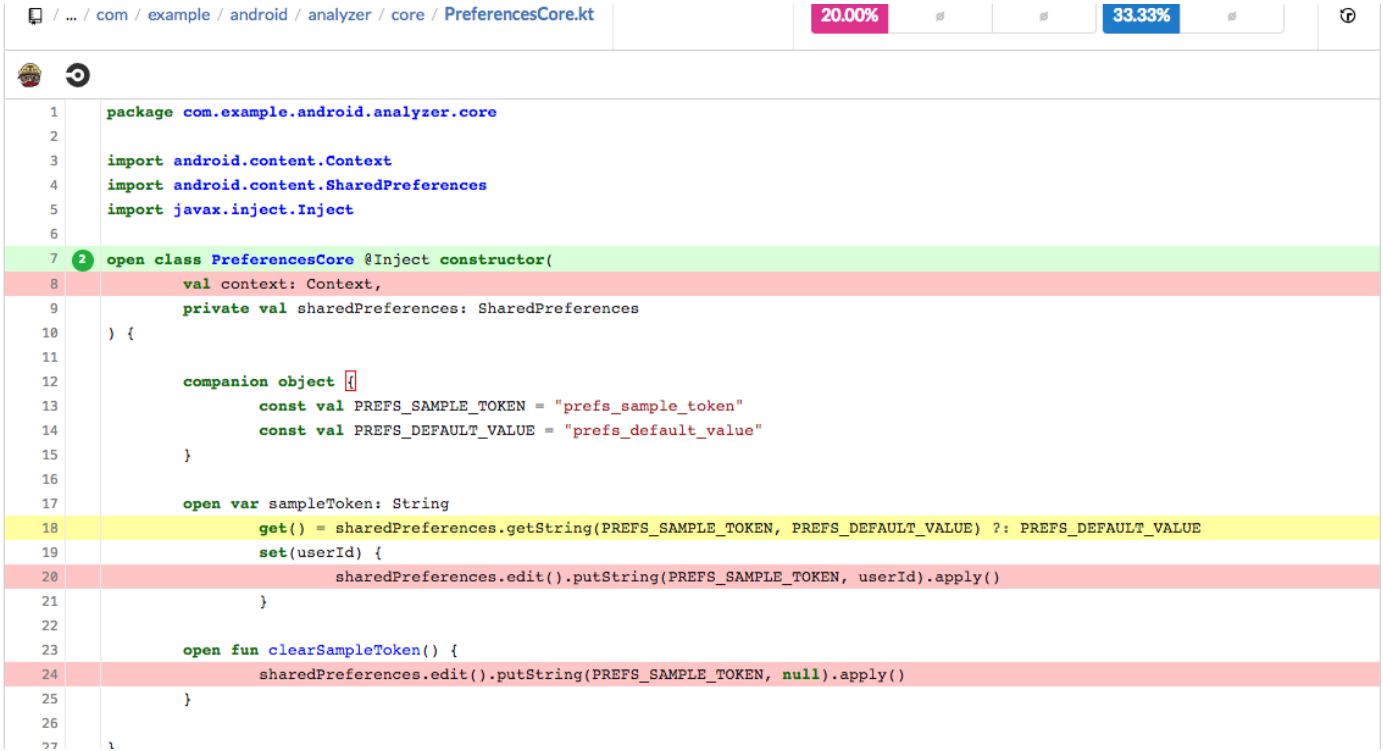
JACOCO LOCAL

module-core > com.example.android.analyzer.core > PreferencesCore.kt

PreferencesCore.kt

```
1.  package com.example.android.analyzer.core
2.
3.  import android.content.Context
4.  import android.content.SharedPreferences
5.  import javax.inject.Inject
6.
7.  open class PreferencesCore @Inject constructor(
8.      val context: Context,
9.      private val sharedPreferences: SharedPreferences
10. ) {
11.
12.     companion object {
13.         const val PREFS_SAMPLE_TOKEN = "prefs_sample_token"
14.         const val PREFS_DEFAULT_VALUE = "prefs_default_value"
15.     }
16.
17.     open var sampleToken: String
18.         get() = sharedPreferences.getString(PREFS_SAMPLE_TOKEN, PREFS_DEFAULT_VALUE) ?: PREFS_DEFAULT_VALUE
19.         set(userId) {
20.             sharedPreferences.edit().putString(PREFS_SAMPLE_TOKEN, userId).apply()
21.         }
22.
23.     open fun clearSampleToken() {
24.         sharedPreferences.edit().putString(PREFS_SAMPLE_TOKEN, null).apply()
25.     }
26.
27. }
```

CODECOV


/ ... / com / example / android / analyzer / core / PreferencesCore.kt
20.00%
33.33%
1 package com.example.android.analyzer.core
2
3 import android.content.Context
4 import android.content.SharedPreferences
5 import javax.inject.Inject
6
7 2 open class PreferencesCore @Inject constructor(
8 val context: Context,
9 private val sharedPreferences: SharedPreferences
10) {
11
12 companion object {
13 const val PREFS_SAMPLE_TOKEN = "prefs_sample_token"
14 const val PREFS_DEFAULT_VALUE = "prefs_default_value"
15 }
16
17 open var sampleToken: String
18 get() = sharedPreferences.getString(PREFS_SAMPLE_TOKEN, PREFS_DEFAULT_VALUE) ?: PREFS_DEFAULT_VALUE
19 set(userId) {
20 sharedPreferences.edit().putString(PREFS_SAMPLE_TOKEN, userId).apply()
21 }
22
23 open fun clearSampleToken() {
24 sharedPreferences.edit().putString(PREFS_SAMPLE_TOKEN, null).apply()
25 }
26
27 }

SAMPLE PROJECT

HTTPS://GITHUB.COM/KOTOMISAK/IMAGE-ANALYZER-ANDROID

I have prepared stub of sample application to apply all kinds of code coverage ability.

Sample application is built following way:

- Single activity multi-module application
- Navigation architecture component
- MVVM approach
- DI with Dagger
- Connected to Travis, Circle CI, Jenkins CI & Codecov

THAT'S IT

Michal Jeníček
michal.jenicek@strv.com

STRV

QUESTIONS

STRV