**Основы тестирования Android**

## **Организуйте свой код для тестирования**

### Итеративно создавать и тестировать код

### Просмотрите свое приложение как серию модулей

## **Настройте свою тестовую среду**

### Организуйте каталоги тестов в зависимости от среды выполнения

androidTest Каталог

test Каталог

…….

### Рассмотрите компромиссы при запуске тестов на разных типах устройств.

Настоящее устройство

Виртуальное устройство (например, [эмулятор](https://developer.android.com/studio/run/emulator) в Android Studio)

Моделируемое устройство (например, Робоэлектрик)

### Подумайте, стоит ли использовать тестовые двойники

НЕТ

Объект - это объект данных.

Объект не может функционировать, если он не взаимодействует с реальной версией объекта зависимости. Хороший пример - обработчик обратного вызова события.

Трудно воспроизвести взаимодействие объекта с зависимостью. Хорошим примером является обработчик базы данных SQL, где база данных в памяти предоставляет более надежные тесты, чем подделки результатов базы данных.

ДА

Длительные операции, например обработка большого файла.

Негерметичные действия, такие как подключение к произвольному открытому порту.

Сложные в создании конфигурации.

## **Напишите свои тесты**

### Уровни пирамиды тестирования

* [Небольшие тесты](https://developer.android.com/training/testing/fundamentals#small-tests) - 70%- это модульные тесты, которые проверяют поведение вашего приложения по одному классу за раз.
* [Средние тесты](https://developer.android.com/training/testing/fundamentals#medium-tests) – 20%- это интеграционные тесты, которые проверяют либо взаимодействия между уровнями стека внутри модуля, либо взаимодействия между связанными модулями.
* [Большие тесты](https://developer.android.com/training/testing/fundamentals#large-tests) – 10%- это сквозные тесты, которые проверяют пути пользователя, охватывающие несколько модулей вашего приложения.

### Напишите небольшие тесты

#### Локальные модульные тесты

API-интерфейсы AndroidX Test

использовать [Robolectric](http://robolectric.org/):

* Жизненные циклы компонентов
* Циклы событий
* Все ресурсы

#### Инструментальные модульные тесты

AndroidX Test использует потоки

### Напишите средние тесты

1. Взаимодействия между представлением и моделью представления, такие как тестирование [Fragment](https://developer.android.com/reference/androidx/fragment/app/Fragment) объекта, проверка XML макета или оценка логики привязки данных [ViewModel](https://developer.android.com/reference/androidx/lifecycle/ViewModel) объекта.
2. Тесты на уровне репозитория вашего приложения, которые проверяют, что ваши различные источники данных и объекты доступа к данным (DAO) взаимодействуют должным образом.
3. Вертикальные фрагменты вашего приложения, тестирование взаимодействия на определенном экране. Такой тест проверяет взаимодействия на всех уровнях стека вашего приложения.
4. Мультифрагментные тесты, оценивающие определенную область вашего приложения. В отличие от других типов средних тестов, упомянутых в этом списке, этот тип теста обычно требует реального устройства, поскольку тестируемое взаимодействие включает несколько элементов пользовательского интерфейса.

Чтобы провести эти тесты, сделайте следующее:

1. Используйте методы из библиотеки [Espresso Intents](https://developer.android.com/training/testing/espresso/intents) . Чтобы упростить информацию, которую вы передаете в эти тесты, используйте подделки и заглушки.
2. Комбинируйте ваше использование [IntentSubject](https://developer.android.com/reference/androidx/test/ext/truth/content/IntentSubject) утверждений, основанных на истине, для проверки зафиксированных намерений.

#### Используйте эспрессо при проведении инструментальных средних тестов

* Выполнение действий над [View](https://developer.android.com/reference/android/view/View) объектами.
* Оценка того, как пользователи с потребностями в специальных возможностях могут использовать ваше приложение.
* Поиск и активация предметов внутри [RecyclerView](https://developer.android.com/reference/androidx/recyclerview/widget/RecyclerView)и [AdapterView](https://developer.android.com/reference/android/widget/AdapterView) объектов.
* Проверка состояния исходящих намерений.
* Проверка структуры DOM внутри [WebView](https://developer.android.com/reference/android/webkit/WebView) объектов.

### Напишите большие тесты *Поддержка синхронизации в Espresso*

## **Выполните другие задачи тестирования с помощью AndroidX Test**

### Создавайте более читаемые утверждения, используя Истину [Truth](https://google.github.io/truth/) . JUnit или Hamcrest

### Написать тесты пользовательского интерфейса Espresso

### Запустить UI-тесты JUnit 4

### Взаимодействовать с видимыми элементами [Espresso-Intent](https://developer.android.com/training/testing/espresso/intents) UI Automator

### Добавьте проверки доступности для проверки удобства использования

#### Робоэлектрик

#### Эспрессо

### Активность диска и жизненные циклы фрагментов ActivityScenario и FragmentScenario

### Управляйте жизненными циклами службы

### Оценить все варианты поведения, которые различаются версией SDK

*// Android Studio*

*IDE lint – проверка кода*

*Desinger – как будет выглядеть*

*LogCat – Timber отладочные сообщ*

*Debug – останов - значения*

*Profile – кто что грузит и тормозит*

*Layout Inspector – проверка дизайна*

*DataBase Inspector – просмотр базы данных*

*Analiser Code – общий проверщик*

*Gradle – что-то усть + debug*

* Junit4 5 (Java)
* [AndroidX Test Library](https://developer.android.com/training/testing/set-up-project)
* [AndroidX Architecture Components Core Test Library](https://developer.android.com/reference/android/arch/core/executor/testing/package-summary)
* [Hamcrest](http://hamcrest.org/)
* Mockito
* PowerMock
* Espresso
* UIAutomator
* Robotium, Appium, Calabash, Robolectric
* AndroidX Architecture Components Core Test Library
* UI/Application Exerciser Monkey (AS)

implementation "com.jakewharton.timber:timber:$timberVersion"

debugImplementation 'com.facebook.stetho:stetho-okhttp3:1.5.1'

*// Once https://issuetracker.google.com/127986458 is fixed this can be testImplementation*debugImplementation "androidx.fragment:fragment-testing:$fragmentVersion"  
implementation "androidx.test:core:$androidXTestCoreVersion"  
implementation "androidx.test.espresso:espresso-idling-resource:$espressoVersion"  
  
*// Dependencies for local unit tests*testImplementation "junit:junit:$junitVersion"  
testImplementation "org.mockito:mockito-core:$mockitoVersion"  
testImplementation "org.hamcrest:hamcrest-all:$hamcrestVersion"  
testImplementation "androidx.arch.core:core-testing:$archTestingVersion"  
testImplementation "org.jetbrains.kotlinx:kotlinx-coroutines-android:$coroutinesVersion"  
testImplementation "org.jetbrains.kotlinx:kotlinx-coroutines-test:$coroutinesVersion"  
testImplementation "org.robolectric:robolectric:$robolectricVersion"  
testImplementation "androidx.test.espresso:espresso-core:$espressoVersion"  
testImplementation "androidx.test.espresso:espresso-contrib:$espressoVersion"  
testImplementation "androidx.test.espresso:espresso-intents:$espressoVersion"  
testImplementation "com.google.truth:truth:$truthVersion"  
  
*// AndroidX Test - JVM testing*testImplementation "androidx.test:core-ktx:$androidXTestCoreVersion"  
testImplementation "androidx.test.ext:junit-ktx:$androidXTestExtKotlinRunnerVersion"  
testImplementation "androidx.test:rules:$androidXTestRulesVersion"  
  
*// Dependencies for Android unit tests*androidTestImplementation "junit:junit:$junitVersion"  
androidTestImplementation "org.mockito:mockito-core:$mockitoVersion"  
androidTestImplementation "com.linkedin.dexmaker:dexmaker-mockito:$dexMakerVersion"  
androidTestImplementation "org.jetbrains.kotlinx:kotlinx-coroutines-test:1.3.5"  
  
*// AndroidX Test - Instrumented testing*androidTestImplementation "androidx.test:core-ktx:$androidXTestCoreVersion"  
androidTestImplementation "androidx.test.ext:junit-ktx:$androidXTestExtKotlinRunnerVersion"  
androidTestImplementation "androidx.test:rules:$androidXTestRulesVersion"  
androidTestImplementation "androidx.arch.core:core-testing:$archTestingVersion"  
androidTestImplementation "androidx.test.espresso:espresso-core:$espressoVersion"  
androidTestImplementation "androidx.test.espresso:espresso-contrib:$espressoVersion"  
androidTestImplementation "androidx.test.espresso:espresso-intents:$espressoVersion"  
androidTestImplementation "androidx.test.espresso.idling:idling-concurrent:$espressoVersion"  
androidTestImplementation "org.robolectric:annotations:$robolectricVersion"  
implementation "androidx.test.espresso:espresso-idling-resource:$espressoVersion"  
  
*// Resolve conflicts between main and test APK:*androidTestImplementation "androidx.annotation:annotation:$androidXAnnotations"  
androidTestImplementation "androidx.legacy:legacy-support-v4:$androidXLegacySupport"  
androidTestImplementation "androidx.recyclerview:recyclerview:$recyclerViewVersion"  
androidTestImplementation "androidx.appcompat:appcompat:$appCompatVersion"  
androidTestImplementation "com.google.android.material:material:$materialVersion"

androidTestImplementation "androidx.work:work-testing:$rootProject.workVersion"

androidTestImplementation "androidx.room:room-testing:$roomVersion"

В AndroidX Test доступны следующие зависимости на основе Gradle:

dependencies {  
  // Core library  
  androidTestImplementation 'androidx.test:core:1.0.0'  
  
  // AndroidJUnitRunner and JUnit Rules  
  androidTestImplementation 'androidx.test:runner:1.1.0'  
  androidTestImplementation 'androidx.test:rules:1.1.0'  
  
  // Assertions  
  androidTestImplementation 'androidx.test.ext:junit:1.0.0'  
  androidTestImplementation 'androidx.test.ext:truth:1.0.0'  
  androidTestImplementation 'com.google.truth:truth:0.42'  
  
  // Espresso dependencies  
  androidTestImplementation 'androidx.test.espresso:espresso-core:3.1.0'  
  androidTestImplementation 'androidx.test.espresso:espresso-contrib:3.1.0'  
  androidTestImplementation 'androidx.test.espresso:espresso-intents:3.1.0'  
  androidTestImplementation 'androidx.test.espresso:espresso-accessibility:3.1.0'  
  androidTestImplementation 'androidx.test.espresso:espresso-web:3.1.0'  
  androidTestImplementation 'androidx.test.espresso.idling:idling-concurrent:3.1.0'  
  
  // The following Espresso dependency can be either "implementation"  
  // or "androidTestImplementation", depending on whether you want the  
  // dependency to appear on your APK's compile classpath or the test APK  
  // classpath.  
  androidTestImplementation 'androidx.test.espresso:espresso-idling-resource:3.1.0'  
}

Вот несколько видов тестовых двойников:

**Fake** Подделка

Двойник теста, имеющий "рабочую" реализацию класса, но реализованный таким образом, что он удобен для тестов, но не подходит для производства.

**Mock** Насмешка

Двойник теста отслеживает, какие из его методов были вызваны. Затем он проходит или не проходит тест в зависимости от того, правильно ли были вызваны его методы.

**Stub** Заглушка

Тестовый двойник, который не включает логики и возвращает только то, что вы его запрограммировали. Например, A StubTaskRepositoryможет быть запрограммирован на возврат определенных комбинаций задач getTasks.

**Dummy** Манекен

Тестовый двойник, который передается, но не используется, например, если вам просто нужно указать его в качестве параметра. Если вы имели NoOpTaskRepository, было бы просто реализовать TaskRepositoryс не кода в любом из методов.

**Spy** Шпион

Тестовый дублер, который также отслеживает некоторую дополнительную информацию; например, если вы создали SpyTaskRepository, он может отслеживать количество вызовов addTaskметода.

Для получения дополнительной информации о тестовых двойниках ознакомьтесь с разделом «Тестирование в туалете: знай свои тестовые пары» .