Composing With Confidence

Adam McNeilly - @AdamMc331

With New Tools Comes New Responsibilities

Getting Started With Compose Testing[1]

[1] https://goo.gle/compose-testing

Two Options For Compose Testing

Two Options For Compose Testing

- Individual components

Two Options For Compose Testing

- Individual components
- Activities

Compose Rule Setup

```
class PrimaryButtonTest {

    /**
     * Use createComposeRule to test individual composable functions.
     */
    @get:Rule
    val composeTestRule = createComposeRule()
}
```

Compose Rule Setup

```
class PrimaryButtonTest {

    /**
     * Use createAndroidComposeRule to start up a specific activity.
     */
    @get:Rule
    val composeTestRule = createAndroidComposeRule<MainActivity>()
}
```

Rendering Content

```
@Test
fun renderEnabledButton() {
    composeTestRule.setContent {
        PrimaryButton(...)
    }
}
```

Test Recipe

```
// Find component
composeTestRule.onNodeWithText("Test Button")

// Make assertion
composeTestRule.onNode(...).assertIsEnabled()

// Perform action
composeTestRule.onNode(...).performClick()
```

Test Recipe

```
// Find component
composeTestRule.onNodeWithText("Test Button")

// Make assertion
composeTestRule.onNode(...).assertIsEnabled()

// Perform action
composeTestRule.onNode(...).performClick()
```

Test Recipe

```
// Find component
composeTestRule.onNodeWithText("Test Button")

// Make assertion
composeTestRule.onNode(...).assertIsEnabled()

// Perform action
composeTestRule.onNode(...).performClick()
```

Finding Components

```
composeTestRule.onNode(matcher)

composeTestRule.onNode(hasProgressBarRangeInfo(...))
composeTestRule.onNode(isDialog())
```

```
composeTestRule.onNode(matcher)

composeTestRule.onNode(hasProgressBarRangeInfo(...))
composeTestRule.onNode(isDialog())

// Helpers
composeTestRule.onNodeWithText("")
composeTestRule.onNodeWithTag("")
composeTestRule.onNodeWithContentDescription("")
```

```
composeTestRule.onAllNodes(matcher)

composeTestRule.onAllNodesWithText("")
```

Making Assertions

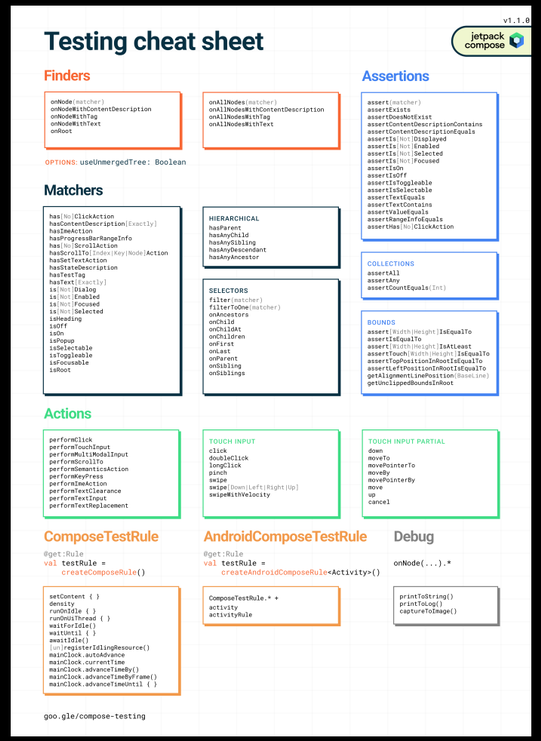
```
composeTestRule.onNode(...)
    .assert(matcher)

composeTestRule.onNode(...)
    .assert(hasText("Test Button"))

composeTestRule.onNode(...)
    .assert(isEnabled())
```

```
composeTestRule.onNode(...)
    .assert(hasText("Test Button"))

// Helpers
composeTestRule.onNode(...)
    .assertTextEquals("Test Button")
```

Performing Actions

```
composeTestRule.onNode(...)
    .performClick()

composeTestRule.onNode(...)
    .performTextInput(...)
```

Cheat Sheet[2]



[2] https://developer.android.com/static/images/jetpack/compose/compose-testing-cheatsheet.png

Test Tags

```
// In app
PrimaryButton(
    modifier = Modifier.testTag("login_button")
)

// In test
composeTestRule.onNodeWithTag("login_button")
```

Let's Test A Component

```
@Composable
fun PrimaryButton(
    text: String,
    onClick: () -> Unit,
    enabled: Boolean = true,
)
```

```
@Composable
fun PrimaryButton(
    text: String,
    onClick: () -> Unit,
    enabled: Boolean = true,
)
```

Setup

```
@RunWith(AndroidJUnit4::class)
class PrimaryButtonTest {

    @get:Rule
    val composeTestRule = createComposeRule()

    @Test
    fun handleClickWhenEnabled() {
        // ...
    }
}
```

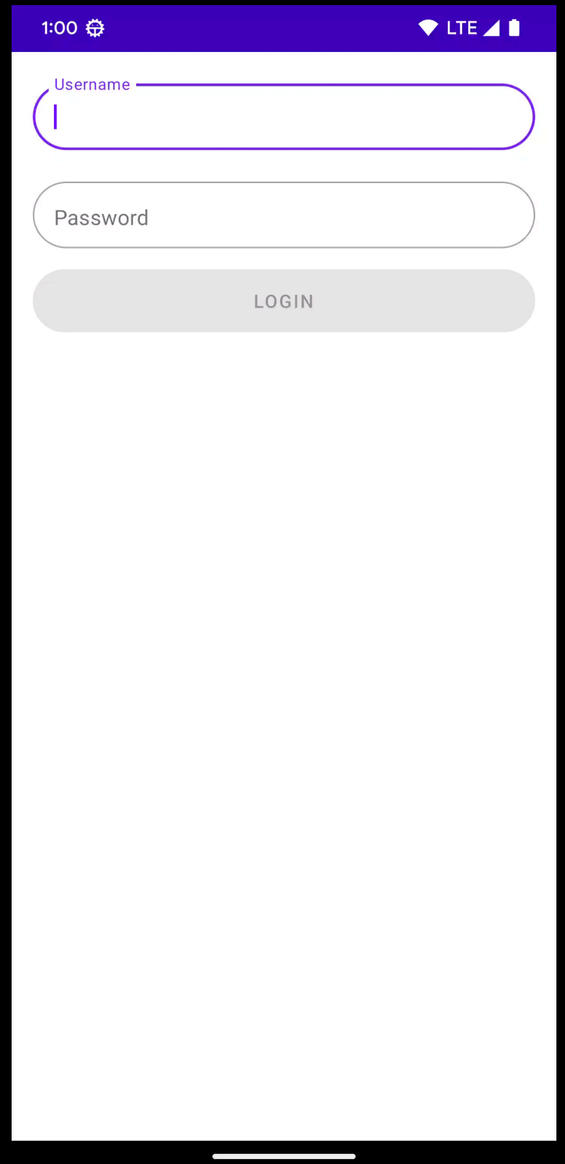
Render Content

```
var wasClicked = false

composeTestRule.setContent {
    PrimaryButton(
        text = "Test Button",
        onClick = {
            wasClicked = true
        },
        enabled = true,
    )
}
```

Verify Behavior

```
composeTestRule.onNodeWithText("Test Button").performClick()

assertTrue(wasClicked)
```

Let's Write A Bigger Test


1:00
LTE
Username
Password
LOGIN

Setup

```
@RunWith(AndroidJUnit4::class)
class MainActivityTest {

    @get:Rule
    val composeTestRule = createAndroidComposeRule<MainActivity>()

    @Test
    fun successfulLogin() {
        // ...
    }
}
```

Verify Login Button Disabled

```
composeTestRule
    .onNodeWithTag("login_button")
    .assertIsNotEnabled()
```

Type Username

```
composeTestRule
    .onNodeWithTag("username_text_field")
    .performTextInput("adammc331")
```

Type Password

```
composeTestRule
    .onNodeWithTag("password_text_field")
    .performTextInput("Hunter2")
```

Verify Login Button Enabled

```
composeTestRule
    .onNodeWithTag("login_button")
    .assertIsEnabled()
```

Click Login Button

```
composeTestRule
    .onNodeWithTag("login_button")
    .performClick()
```

Verify Home Screen Displayed

```
composeTestRule
    .onNodeWithTag("home_screen_label")
    .assertIsDisplayed()
```

```
@Test
fun successfulLogin() {
    composeTestRule
        .onNodeWithTag("login_button")
        .assertIsNotEnabled()

    composeTestRule
        .onNodeWithTag("username_text_field")
        .performTextInput("adammc331")

    composeTestRule
        .onNodeWithTag("login_button")
        .assertIsNotEnabled()

    composeTestRule
        .onNodeWithTag("password_text_field")
        .performTextInput("Hunter2")

    composeTestRule
        .onNodeWithTag("login_button")
        .assertIsEnabled()

    composeTestRule
        .onNodeWithTag("login_button")
        .performClick()

    composeTestRule
        .onNodeWithTag("home_screen_label")
        .assertIsDisplayed()
}
```

Test Robots

LoginScreenRobot

```
class LoginScreenRobot(
    composeTestRule: ComposeTestRule,
) {
    private val usernameInput = composeTestRule.onNodeWithTag("username_text_field")
    private val passwordInput = composeTestRule.onNodeWithTag("password_text_field")
    private val loginButton = composeTestRule.onNodeWithTag("login_button")
}
```

LoginScreenRobot

```
class LoginScreenRobot {

    fun enterUsername(username: String) {
        usernameInput.performTextInput(username)
    }

    fun enterPassword(password: String) {
        passwordInput.performTextInput(password)
    }
}
```

Kotlin Magic

```
fun loginScreenRobot(
    composeTestRule: ComposeTestRule,
    block: LoginScreenRobot.() -> Unit,
) {
    LoginScreenRobot(composeTestRule).apply(block)
}
```

```
loginScreenRobot(composeTestRule) {
    verifyLoginButtonDisabled()
    enterUsername("adammc331")
    enterPassword("Hunter2")
    verifyLoginButtonEnabled()
    clickLoginButton()
}
```

```
@Test
fun successfulLogin() {
    loginScreenRobot(composeTestRule) {
        verifyLoginButtonDisabled()
        enterUsername("adammc331")
        enterPassword("Hunter2")
        verifyLoginButtonEnabled()
        clickLoginButton()
    }

    homeScreenRobot(composeTestRule) {
        verifyLabelDisplayed()
    }
}
```

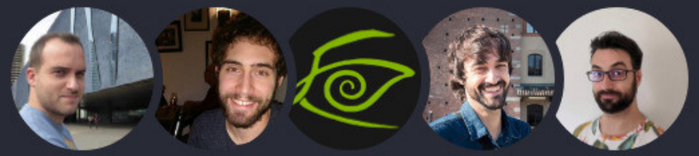
Other Testing Options

Shot

pedrovgs / Shot

1002

Screenshot testing library for Android



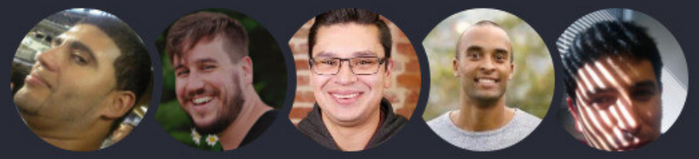
32 Contributors

Paparazzi

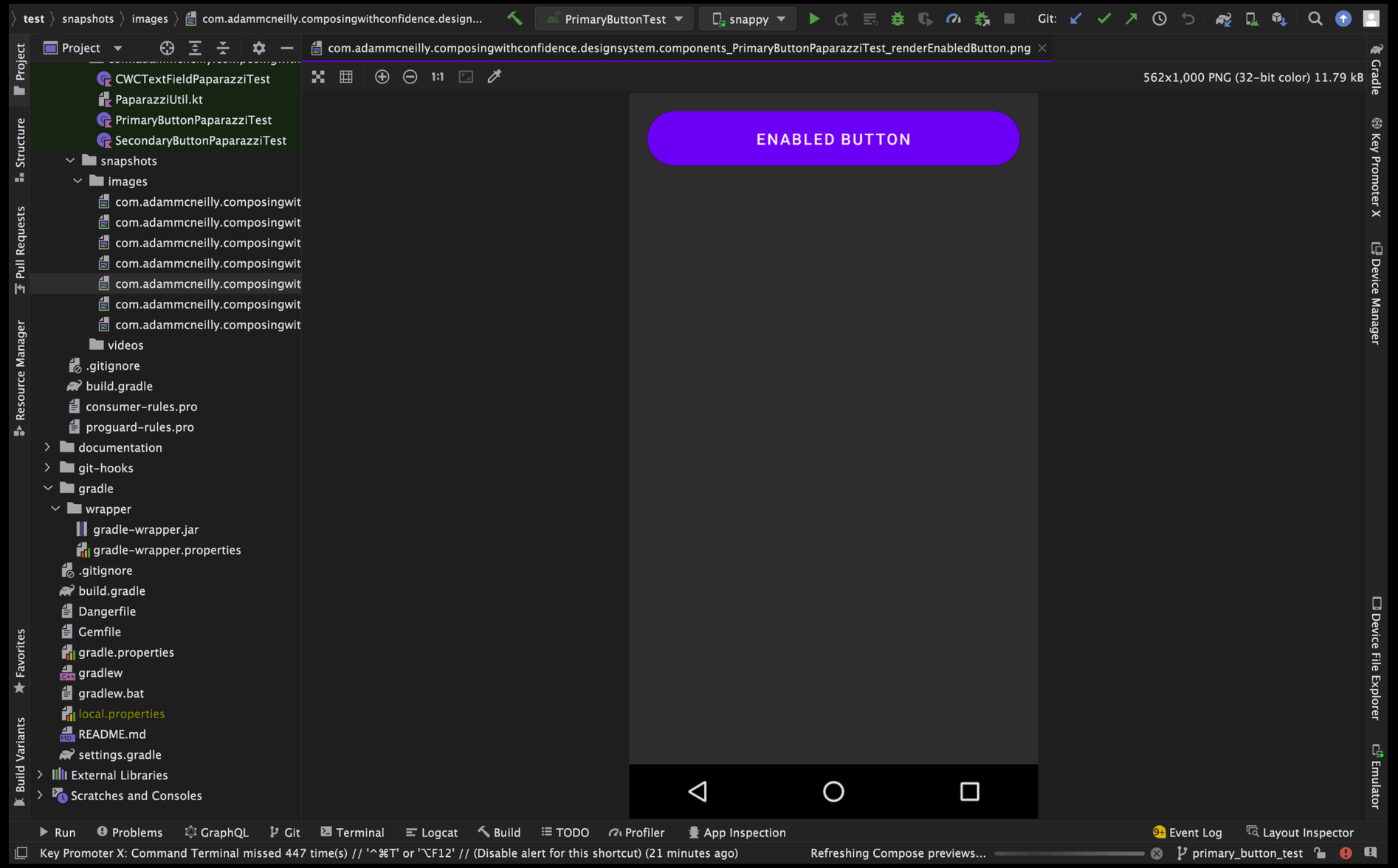
cashapp / paparazzi

1372 ★

Render your Android screens without a physical device or emulator



37 Contributors

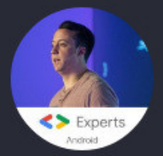

test
snapshots
images
com.adammcneilly.composingwithconfidence.design...
PrimaryButtonTest
snappy
Git:
Project
com.adammcneilly.composingwithconfidence.designsystem.components_PrimaryButtonPaparazziTest_renderEnabledButton.png
562x1,000 PNG (32-bit color) 11.79 kB
CWCTextFieldPaparazziTest
PaparazziUtil.kt
PrimaryButtonPaparazziTest
SecondaryButtonPaparazziTest
snapshots
images
videos
.gitignore
build.gradle
consumer-rules.pro
proguard-rules.pro
documentation
git-hooks
gradle
wrapper
gradle-wrapper.jar
gradle-wrapper.properties
.gitignore
build.gradle
Dangerfile
Gemfile
gradle.properties
gradlew
gradlew.bat
local.properties
README.md
settings.gradle
External Libraries
Scratches and Consoles
ENABLED BUTTON
Run
Problems
GraphQL
Git
Terminal
Logcat
Build
TODO
Profiler
App Inspection
Event Log
Layout Inspector
Key Promoter X: Command Terminal missed 447 time(s) // '^⌘T' or '⌥F12' // (Disable alert for this shortcut) (21 minutes ago)
Refreshing Compose previews...
primary_button_test

One More Repo...

Composing With Confidence Sample Project

AdamMc331 / ComposingWithConfidence 0

Sample repo that demonstrates various options for testing Jetpack Compose applications.



AdamMc331

README.md

Composing With Confidence

This is the sample repository for the Composing With Confidence presentation from Droidcon NYC in 2022. If you run the sample app, you won't see much other than a basic login form that requires filling out a username and password before navigating into a basic home screen.

The main benefit of this repo is to run and explore different testing options in Jetpack Compose. The main points of interest are the following:

1. The PrimaryButtonTest file shows how we can test an individual component by itself.
2. The PrimaryButtonPaparazziTest shows how we can use the Paparazzi library to screenshot test on the JVM.
3. The MainActivityTest shows how we can write a full integration test to verify a screen's behavior including navigation.

If you'd like to view the slides from this presentation, you can find them in the presentation folder.

Thank You!