Espresso Patronum:
The Magic of the Robot Pattern

Adam McNeilly: Android Engineer - OkCupid

What is Espresso?

Use Espresso to write concise, beautiful, and reliable Android UI tests[1].

[1] https://developer.android.com/training/testing/espresso/index.html

Three Classes To Know

1. ViewMatchers
2. ViewActions
3. ViewAssertions

ViewMatchers

- `withId(...)`
- `withText(...)`
- `isFocusable()`
- `isChecked()`

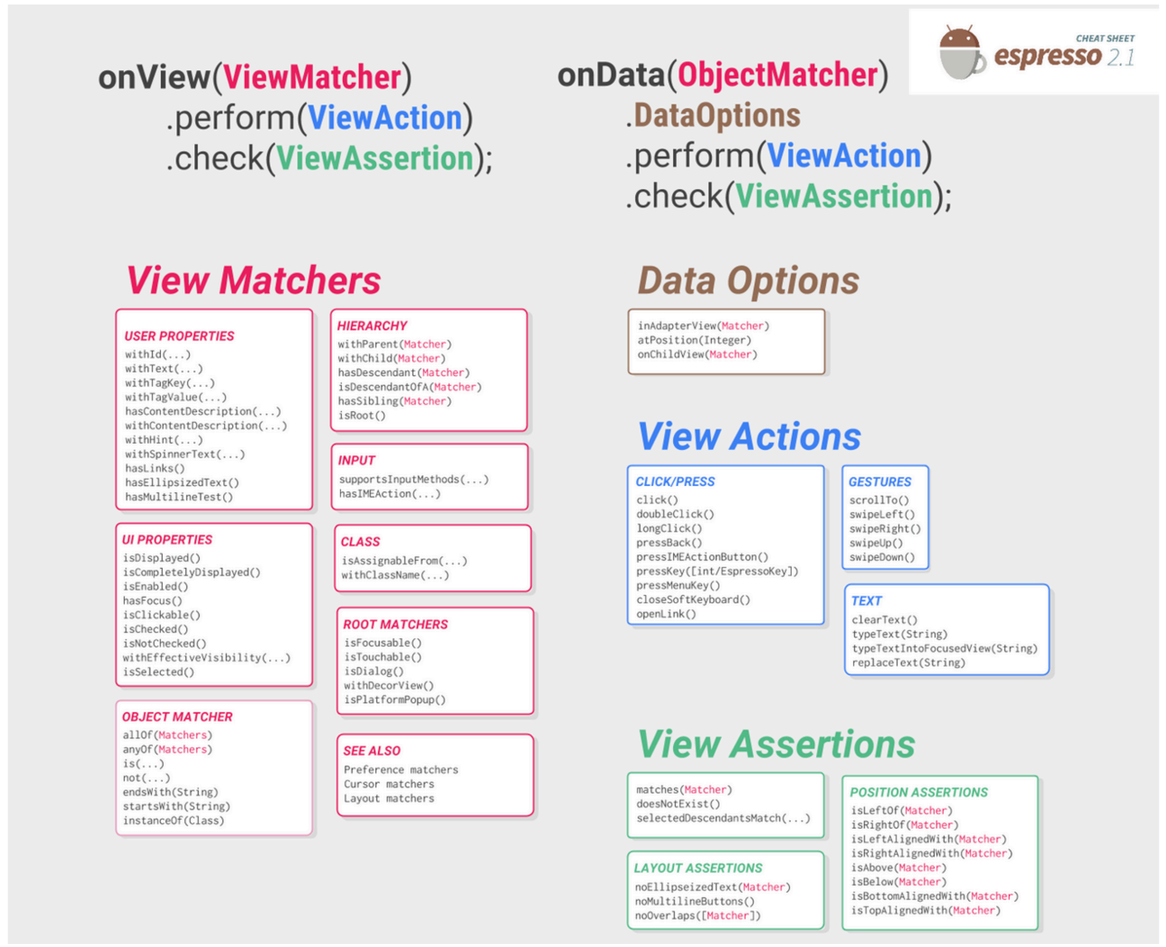
ViewActions

- `typeText(...)`
- `scrollTo()`
- `swipeLeft()`
- `click()`

ViewAssertions

- `matches(Matcher)`
- `isLeftOf(Matcher)`
- `doesNotExist()`

Espresso Cheatsheet[2]



CHEAT SHEET espresso 2.1

```
onView(ViewMatcher)
    .perform(ViewAction)
    .check(ViewAssertion);
```

```
onData(ObjectMatcher)
    .DataOptions
    .perform(ViewAction)
    .check(ViewAssertion);
```

View Matchers

USER PROPERTIES

```
withId(...)
withText(...)
withTagKey(...)
withTagValue(...)
hasContentDescription(...)
withContentDescription(...)
withHint(...)
withSpinnerText(...)
hasLinks()
hasEllipsizedText()
hasMultilineTest()
```

HIERARCHY

```
withParent(Matcher)
withChild(Matcher)
hasDescendant(Matcher)
isDescendantOfA(Matcher)
hasSibling(Matcher)
isRoot()
```

INPUT

```
supportsInputMethods(...)
hasIMEAction(...)
```

UI PROPERTIES

```
isDisplayed()
isCompletelyDisplayed()
isEnabled()
hasFocus()
isClickable()
isChecked()
isNotChecked()
withEffectiveVisibility(...)
isSelected()
```

CLASS

```
isAssignableFrom(...)
withClassName(...)
```

ROOT MATCHERS

```
isFocusable()
isTouchable()
isDialog()
withDecorView()
isPlatformPopup()
```

OBJECT MATCHER

```
allOf(Matchers)
anyOf(Matchers)
is(...)
not(...)
endsWith(String)
startsWith(String)
instanceOf(Class)
```

SEE ALSO

```
Preference matchers
Cursor matchers
Layout matchers
```

Data Options

```
inAdapterView(Matcher)
atPosition(Integer)
onChildView(Matcher)
```

View Actions

CLICK/PRESS

```
click()
doubleClick()
longClick()
pressBack()
pressIMEActionButton()
pressKey([int/EspressoKey])
pressMenuKey()
closeSoftKeyboard()
openLink()
```

GESTURES

```
scrollTo()
swipeLeft()
swipeRight()
swipeUp()
swipeDown()
```

TEXT

```
clearText()
typeText(String)
typeTextIntoFocusedView(String)
replaceText(String)
```

View Assertions

```
matches(Matcher)
doesNotExist()
selectedDescendantsMatch(...)
```

LAYOUT ASSERTIONS

```
noEllipseizedText(Matcher)
noMultilineButtons()
noOverlaps([Matcher])
```

POSITION ASSERTIONS

```
isLeftOf(Matcher)
isRightOf(Matcher)
isLeftAlignedWith(Matcher)
isRightAlignedWith(Matcher)
isAbove(Matcher)
isBelow(Matcher)
isBottomAlignedWith(Matcher)
isTopAlignedWith(Matcher)
```

---

[2] https://developer.android.com/training/testing/espresso/cheat-sheet.html

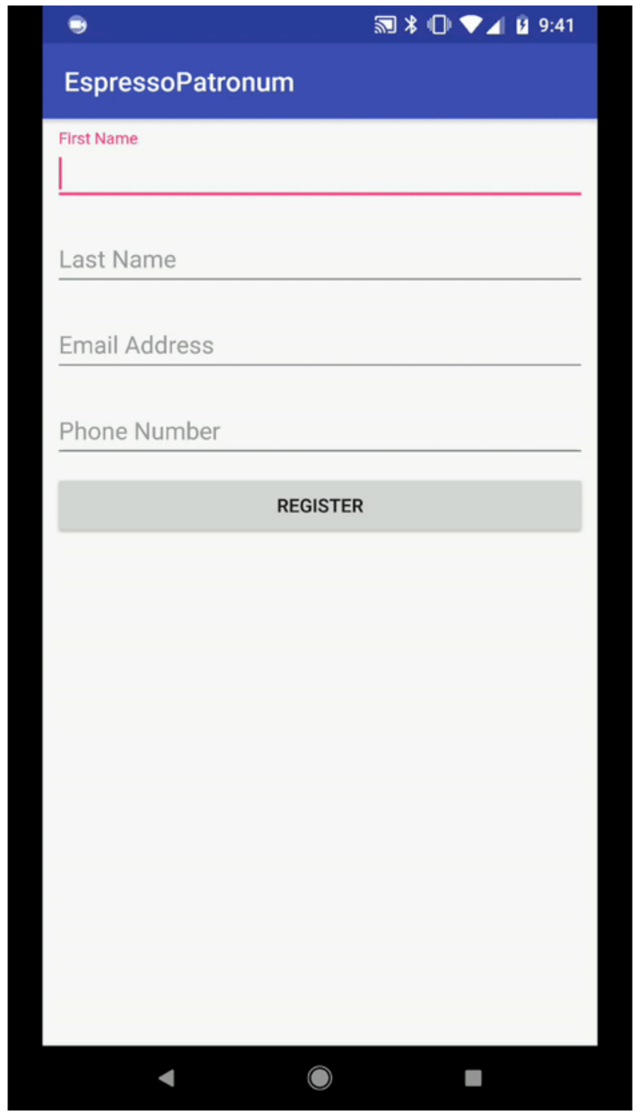
Espresso Example

```
// onView gives us a ViewInteraction where we can perform an action
// or check an assertion.
onView(ViewMatcher)
    .perform(ViewAction)
    .check(ViewAssertion)
```

Espresso Example

```
// Type into an EditText, verify it appears in a TextView
onView(withId(R.id.etInput)).perform(typeText("Adam"))
onView(withId(R.id.tvOutput)).check(matches(withText("Adam")))
```

Sample Project



The Problem

Before we introduce robots, let's take a look at the problem it solves.

Test Successful Registration

```
@Test
fun testSuccessfulRegistration() {
    onView(withId(R.id.etFirstName)).perform(typeText("Adam"))
    onView(withId(R.id.etLastName)).perform(typeText("McNeilly"))
    onView(withId(R.id.etEmail)).perform(typeText("amcneilly@okcupid.com"))
    onView(withId(R.id.etPhone)).perform(typeText("1234567890"))
    onView(withId(R.id.registerButton)).perform(click())

    onView(withId(R.id.tvFullName)).check(matches(withText("Adam McNeilly")))
    onView(withId(R.id.tvEmailAddress)).check(matches(withText("amcneilly@okcupid.com")))
    onView(withId(R.id.tvPhoneNumber)).check(matches(withText("(123)-456-7890")))
}
```

Test A Missing Field

```
@Test
fun testMissingEmailError() {
    onView(withId(R.id.etFirstName)).perform(typeText("Adam"))
    onView(withId(R.id.etLastName)).perform(typeText("McNeilly"))
    onView(withId(R.id.etPhone)).perform(typeText("1234567890"))
    onView(withId(R.id.registerButton)).perform(click())

    onView(withId(R.id.etEmail)).check(matches(hasErrorText("Must enter an email address.")))
}
```

One More Negative Test

```
@Test
fun testInvalidEmailError() {
    onView(withId(R.id.etFirstName)).perform(typeText("Adam"))
    onView(withId(R.id.etLastName)).perform(typeText("McNeilly"))
    onView(withId(R.id.etEmail)).perform(typeText("blahblah"))
    onView(withId(R.id.etPhone)).perform(typeText("1234567890"))
    onView(withId(R.id.registerButton)).perform(click())

    onView(withId(R.id.etEmail)).check(matches(hasErrorText("Must enter a valid email address.")))
}
```

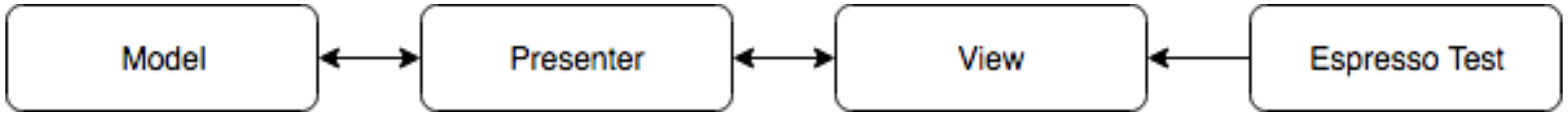
All Together

```
@Test
fun testSuccessfulRegistration() {
    onView(withId(R.id.etFirstName)).perform(typeText("Adam"))
    onView(withId(R.id.etLastName)).perform(typeText("McNeilly"))
    onView(withId(R.id.etEmail)).perform(typeText("amcneilly@okcupid.com"))
    onView(withId(R.id.etPhone)).perform(typeText("1234567890"))
    onView(withId(R.id.registerButton)).perform(click())

    onView(withId(R.id.tvFullName)).check(matches(withText("Adam McNeilly")))
    onView(withId(R.id.tvEmailAddress)).check(matches(withText("amcneilly@okcupid.com")))
    onView(withId(R.id.tvPhoneNumber)).check(matches(withText("(123)-456-7890")))
}

@Test
fun testMissingEmailError() {
    onView(withId(R.id.etFirstName)).perform(typeText("Adam"))
    onView(withId(R.id.etLastName)).perform(typeText("McNeilly"))
    onView(withId(R.id.etPhone)).perform(typeText("1234567890"))
    onView(withId(R.id.registerButton)).perform(click())

    onView(withId(R.id.etEmail)).check(matches(hasErrorText("Must enter an email address.")))
}

@Test
fun testInvalidEmailError() {
    onView(withId(R.id.etFirstName)).perform(typeText("Adam"))
    onView(withId(R.id.etLastName)).perform(typeText("McNeilly"))
    onView(withId(R.id.etEmail)).perform(typeText("blahblah"))
    onView(withId(R.id.etPhone)).perform(typeText("1234567890"))
    onView(withId(R.id.registerButton)).perform(click())

    onView(withId(R.id.etEmail)).check(matches(hasErrorText("Must enter a valid email address.")))
}
```

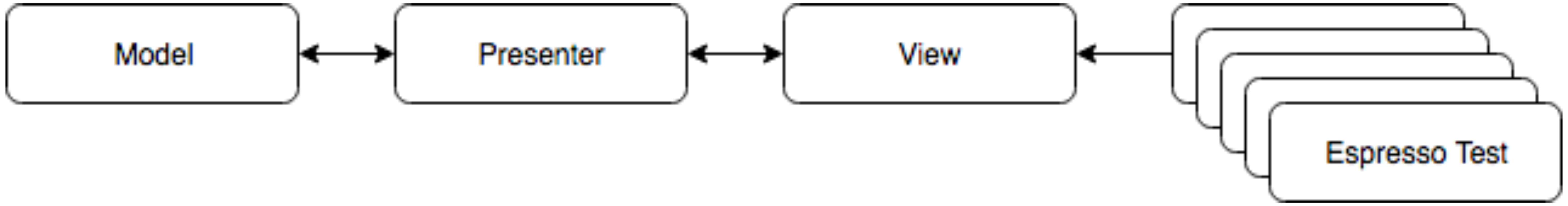
The Problem

1. Extremely Verbose & Unreadable
2. Not Easy To Maintain

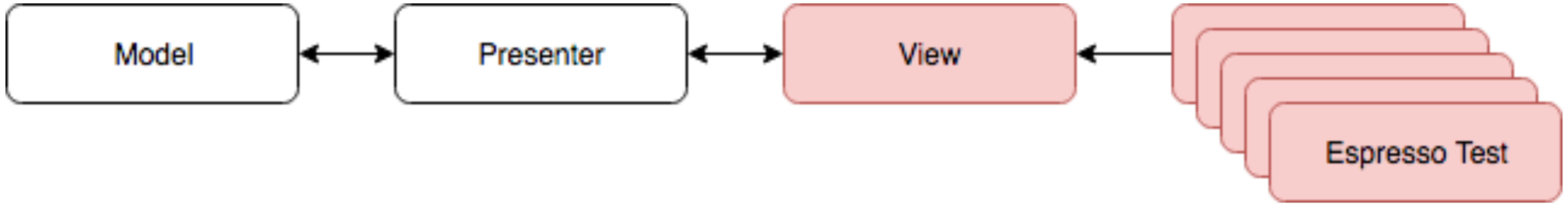
No Separation Of Concerns



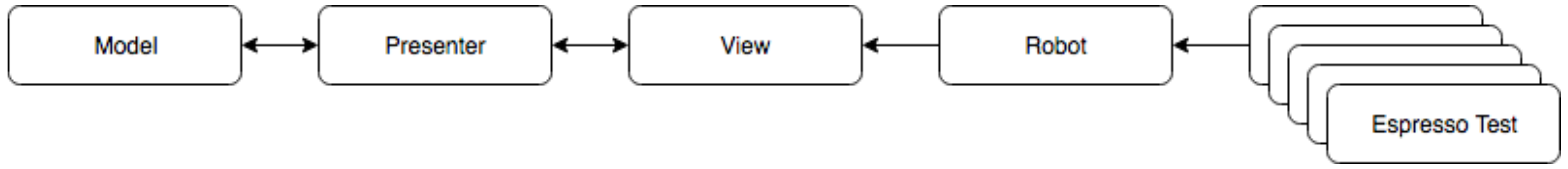
No Separation Of Concerns



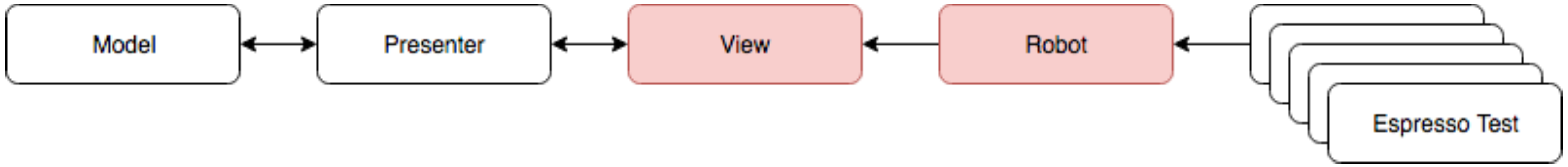
No Separation Of Concerns



Introducing Robots



Separation Of Concerns



Write your automated tests as if you're telling a Quality Assurance Engineer what to do.

Usage

```
@Test
fun testSuccessfulRegistration() {
    RegistrationRobot()
            .firstName("Adam")
            .lastName("McNeilly")
            .email("amcneilly@okcupid.com")
            .phone("1234567890")
            .register()
            .assertFullNameDisplay("Adam McNeilly")
            .assertEmailDisplay("amcneilly@okcupid.com")
            .assertPhoneDisplay("(123)-456-7890")
}
```

Define ViewMatchers

```kotlin
class RegistrationRobot {

    companion object {
        private val FIRST_NAME_INPUT_MATCHER = withId(R.id.etFirstName)
        private val LAST_NAME_INPUT_MATCHER = withId(R.id.etLastName)
        private val EMAIL_INPUT_MATCHER = withId(R.id.etEmail)
        private val PHONE_INPUT_MATCHER = withId(R.id.etPhone)
        private val REGISTER_INPUT_MATCHER = withId(R.id.registerButton)

        private val FULL_NAME_DISPLAY_MATCHER = withId(R.id.tvFullName)
        private val EMAIL_DISPLAY_MATCHER = withId(R.id.tvEmailAddress)
        private val PHONE_DISPLAY_MATCHER = withId(R.id.tvPhoneNumber)
    }
}
```

Each Action As A Method

```
class RegistrationRobot {

    fun firstName(firstName: String): RegistrationRobot {
        onView(FIRST_NAME_MATCHER).perform(clearText(), typeText(firstName), closeSoftKeyboard())
        return this
    }

    fun register(): RegistrationRobot {
        onView(REGISTER_INPUT_MATCHER).perform(click())
        return this
    }

    fun assertFullNameDisplay(fullName: String): RegistrationRobot {
        onView(FULL_NAME_DISPLAY_MATCHER).check(matches(withText(fullName)))
        return this
    }

    ...
}
```

One Robot Per Screen

```
@Test
fun testSuccessfulRegistration() {
    RegistrationRobot()
            .firstName("Adam")
            .lastName("McNeilly")
            .email("amcneilly@okcupid.com")
            .phone("1234567890")
            .register()

    UserProfileRobot()
            .assertFullNameDisplay("Adam McNeilly")
            .assertEmailDisplay("amcneilly@okcupid.com")
            .assertPhoneDisplay("(123)-456-7890")
}
```

Negative Test

```
@Test
fun testMissingEmailError() {
    RegistrationRobot()
            .firstName("Adam")
            .lastName("McNeilly")
            .phone("1234567890")
            .register()
            .assertEmailError("Must enter an email address.")
}
```

Benefits

1. Readibility
2. Maintainability & Separation Of Concerns
3. Tests Become Easier To Write

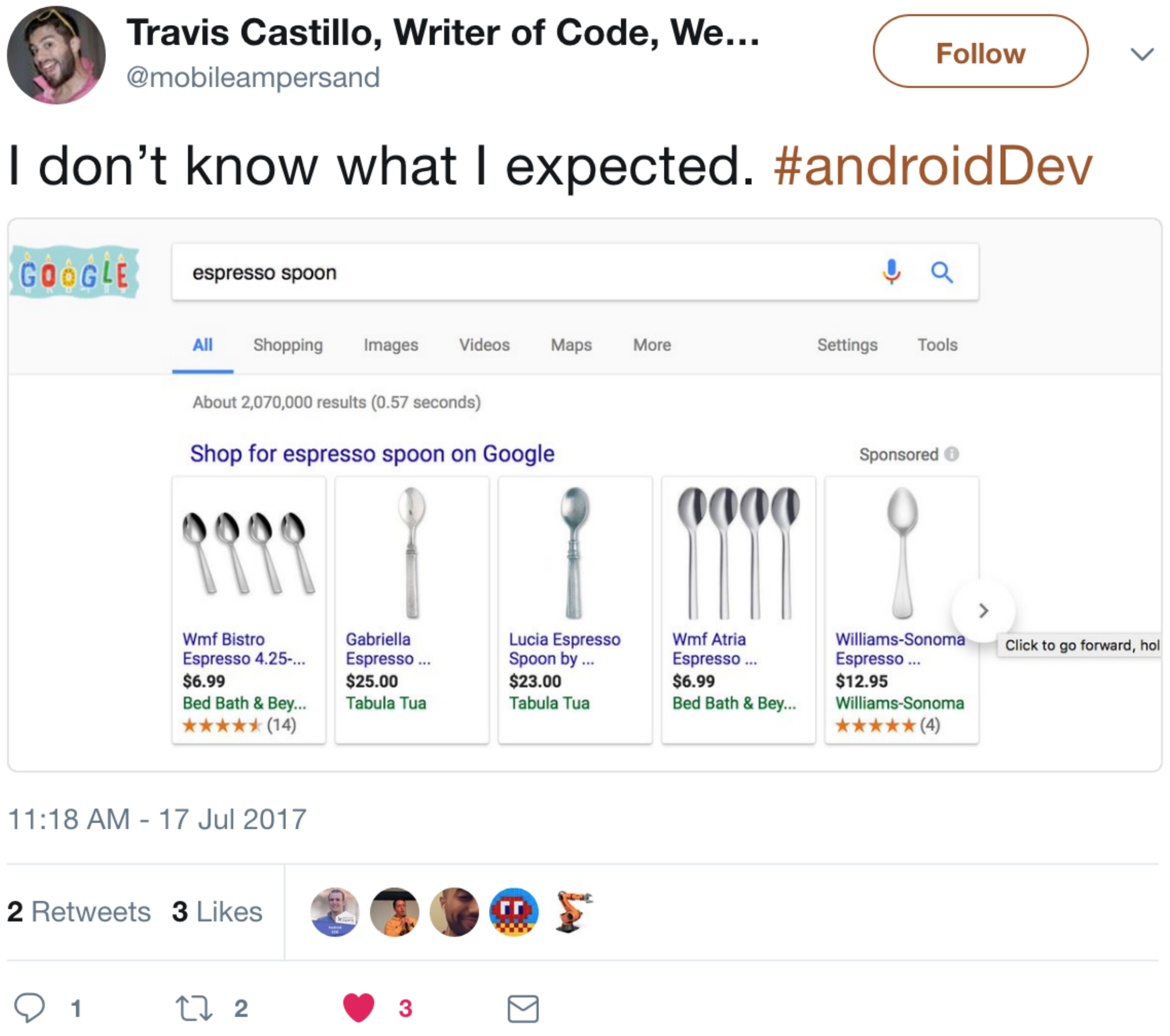
What Else?

Better Test Reporting Using Spoon & Falcon

- Spoon[3] will run all of our instrumentation tests and build us a static HTML report at the end.
- Falcon[4] takes better screenshots, and has a `SpoonCompat` library for the best of both worlds.

[3] https://github.com/square/spoon

[4] https://github.com/jraska/Falcon/



Travis Castillo, Writer of Code, We...
@mobileampersand

Follow

I don't know what I expected. #androidDev

espresso spoon

All Shopping Images Videos Maps More Settings Tools

About 2,070,000 results (0.57 seconds)

Shop for espresso spoon on Google

Sponsored

Wmf Bistro Espresso 4.25-...	Gabriella Espresso ...	Lucia Espresso Spoon by ...	Wmf Atria Espresso ...	Williams-Sonoma Espresso ...
$6.99	$25.00	$23.00	$6.99	$12.95
Bed Bath & Bey...	Tabula Tua	Tabula Tua	Bed Bath & Bey...	Williams-Sonoma
★★★★★ (14)				★★★★★ (4)

Click to go forward, hol

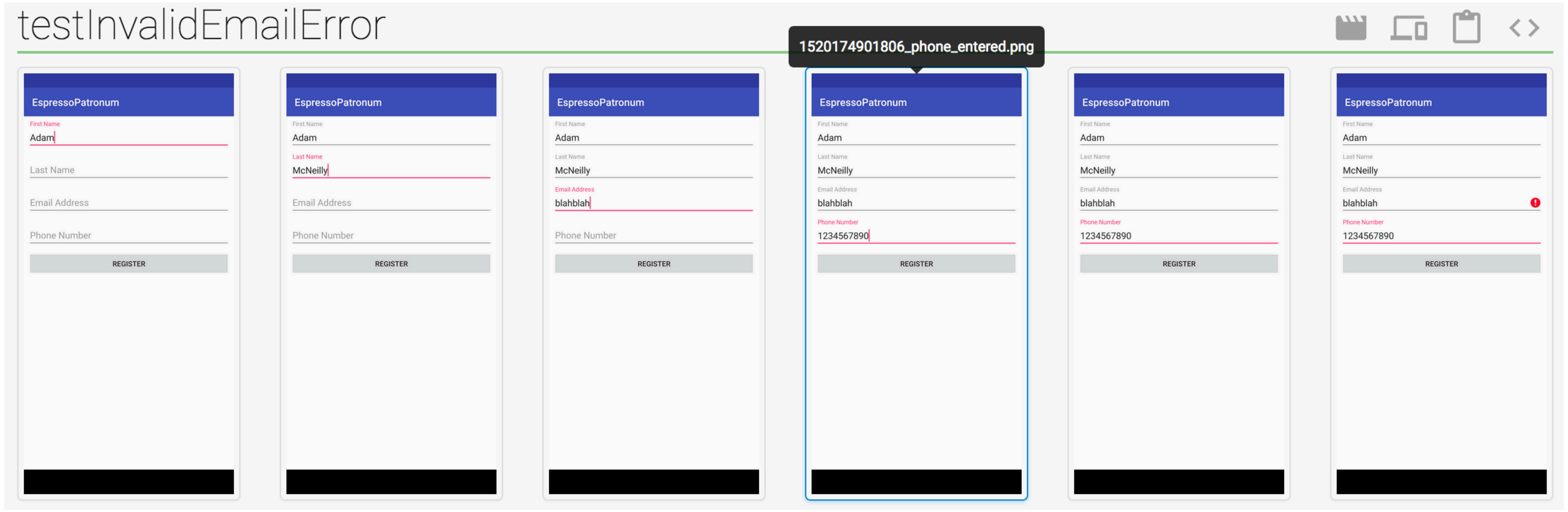
11:18 AM - 17 Jul 2017

2 Retweets 3 Likes

1 2 3

@AdamMc331
#ChicagoRoboto

Example Spoon Report



testInvalidEmailError

1520174901806_phone_entered.png

EspressoPatronum
First Name
Adam
Last Name
Email Address
Phone Number
REGISTER

EspressoPatronum
First Name
Adam
Last Name
McNeilly
Email Address
Phone Number
REGISTER

EspressoPatronum
First Name
Adam
Last Name
McNeilly
Email Address
blahblah
Phone Number
REGISTER

EspressoPatronum
First Name
Adam
Last Name
McNeilly
Email Address
blahblah
Phone Number
1234567890
REGISTER

EspressoPatronum
First Name
Adam
Last Name
McNeilly
Email Address
blahblah
Phone Number
1234567890
REGISTER

EspressoPatronum
First Name
Adam
Last Name
McNeilly
Email Address
blahblah
Phone Number
1234567890
REGISTER

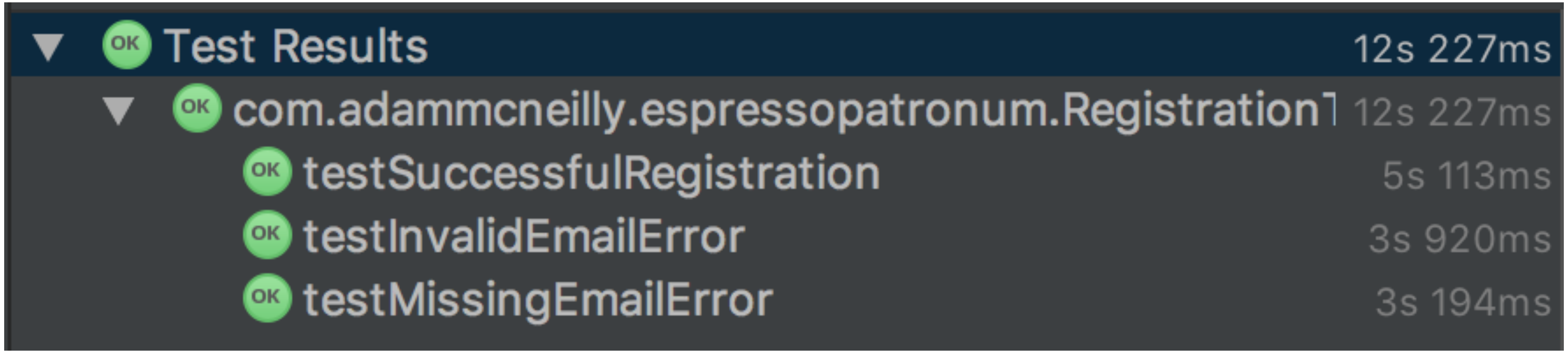
When To Take Screenshots

- After assertions
- After actions - unless that action leads to another screen
- On failure

Why screenshots?

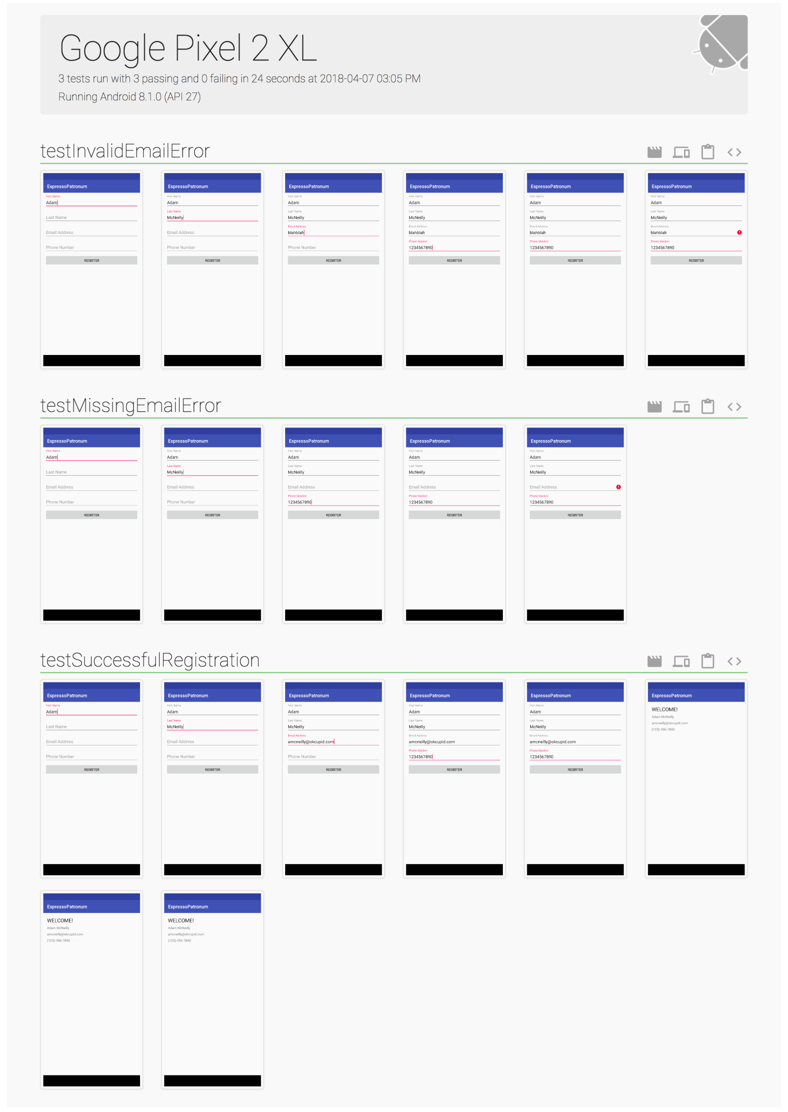
- Human readable output
- See exactly how things were tested
- Diagnose failures quicker

Console Output



```
▼ OK Test Results                                              12s 227ms
  ▼ OK com.adammcneilly.espressopatronum.Registration          12s 227ms
       OK testSuccessfulRegistration                            5s 113ms
       OK testInvalidEmailError                                 3s 920ms
       OK testMissingEmailError                                 3s 194ms
```

Spoon Output



Google Pixel 2 XL

3 tests run with 3 passing and 0 failing in 24 seconds at 2018-04-07 03:05 PM

Running Android 8.1.0 (API 27)

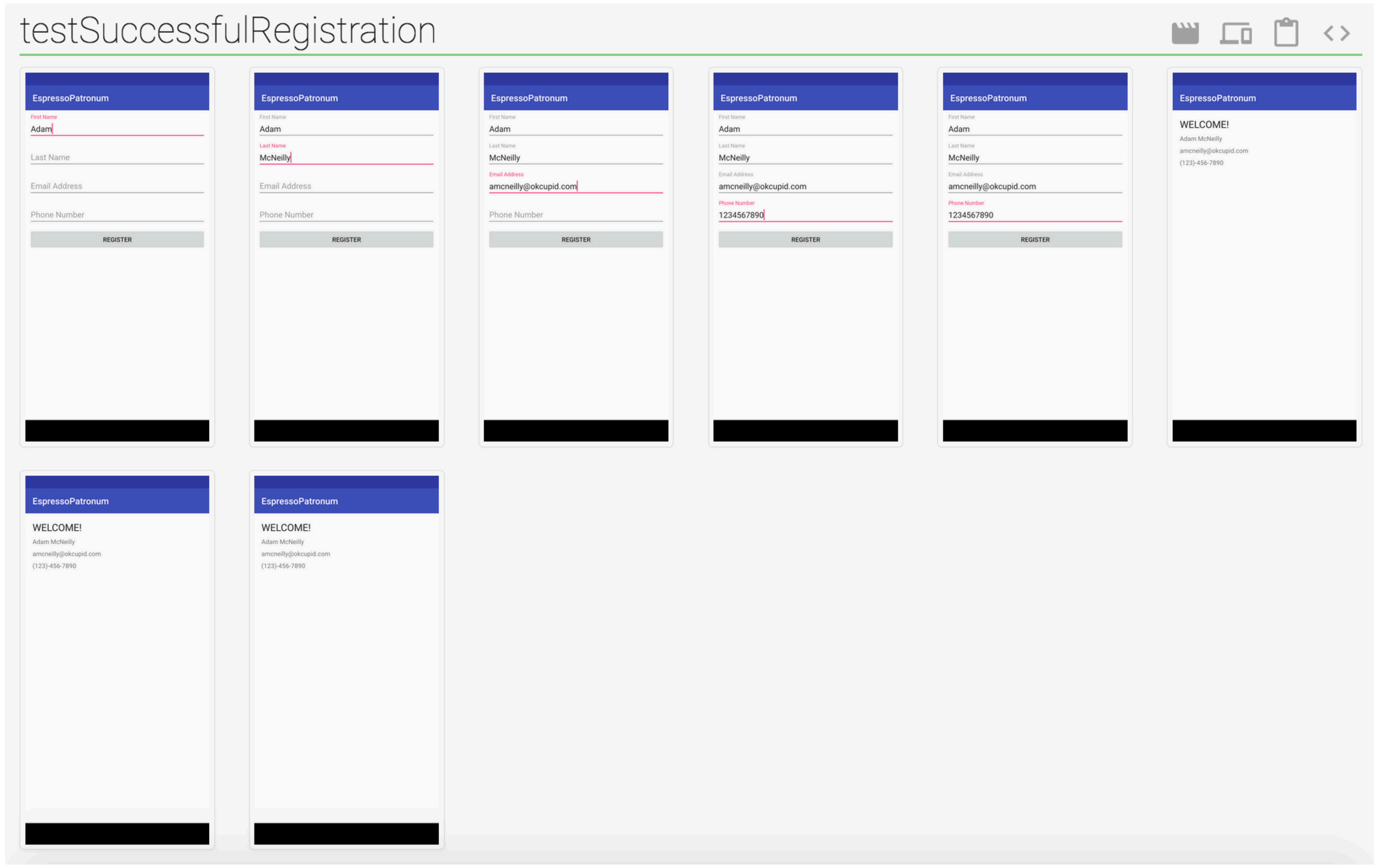
testInvalidEmailError

testMissingEmailError

testSuccessfulRegistration

@AdamMc331
#ChicagoRoboto

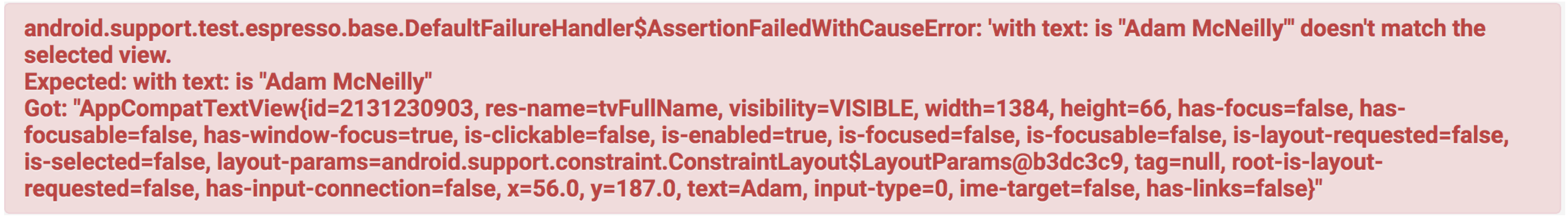
See Steps Taken



testSuccessfulRegistration

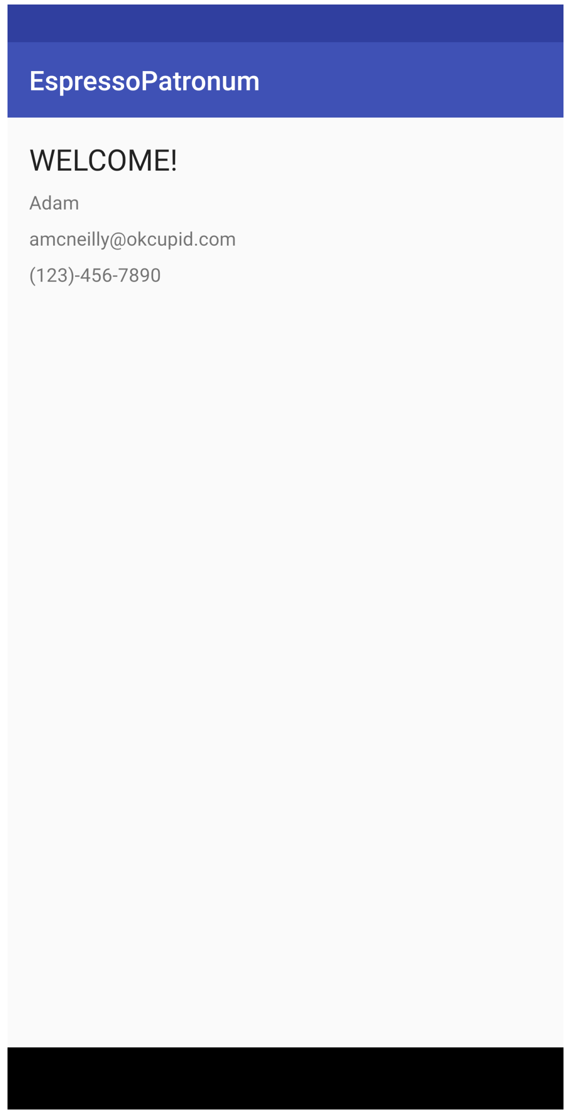
@AdamMc331
#ChicagoRoboto

Diagnose Failures - Stack Trace



```
android.support.test.espresso.base.DefaultFailureHandler$AssertionFailedWithCauseError: 'with text: is "Adam McNeilly"' doesn't match the
selected view.
Expected: with text: is "Adam McNeilly"
Got: "AppCompatTextView{id=2131230903, res-name=tvFullName, visibility=VISIBLE, width=1384, height=66, has-focus=false, has-
focusable=false, has-window-focus=true, is-clickable=false, is-enabled=true, is-focused=false, is-focusable=false, is-layout-requested=false,
is-selected=false, layout-params=android.support.constraint.ConstraintLayout$LayoutParams@b3dc3c9, tag=null, root-is-layout-
requested=false, has-input-connection=false, x=56.0, y=187.0, text=Adam, input-type=0, ime-target=false, has-links=false}"
```

Diagnose Failures - Clear Image



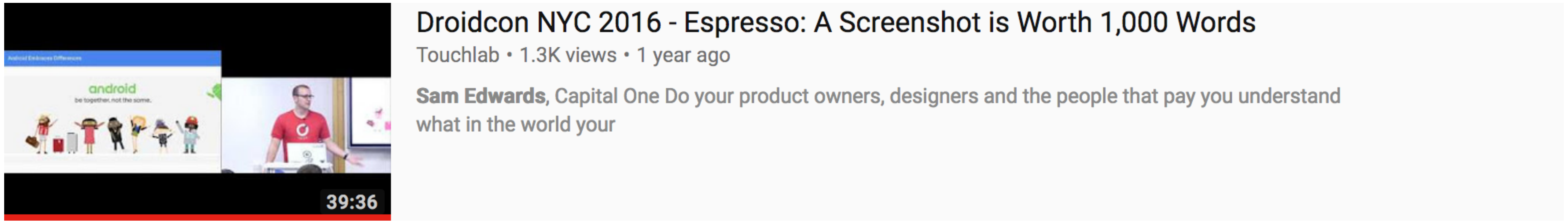
WELCOME!

Adam

amcneilly@okcupid.com

(123)-456-7890

Learn More[5]



Droidcon NYC 2016 - Espresso: A Screenshot is Worth 1,000 Words

Touchlab • 1.3K views • 1 year ago

Sam Edwards, Capital One Do your product owners, designers and the people that pay you understand what in the world your

[5] https://www.youtube.com/watch?v=fhx_Ji5s3p4

Adding Screenshots To Our Robot

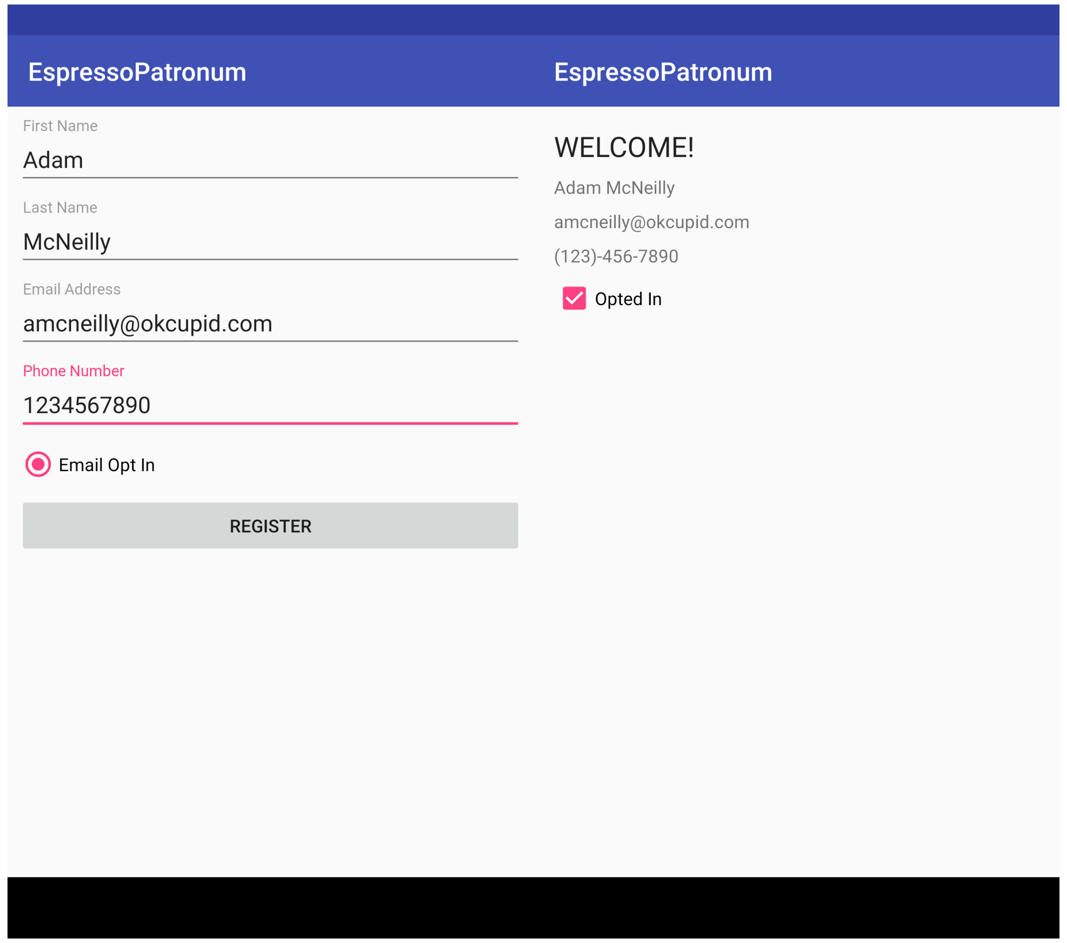
```
fun firstName(firstName: String): RegistrationRobot {
    onView(FIRST_NAME_INPUT_MATCHER).perform(clearText(), typeText(firstName), closeSoftKeyboard())
    takeScreenshot(spoon, "first_name_entered")
    return this
}

fun register(): RegistrationRobot {
    takeScreenshot(spoon, "register_clicked")
    onView(REGISTER_INPUT_MATCHER).perform(click())
    return this
}

fun setFailureHandler(spoon: SpoonRule, context: Context) {
    Espresso.setFailureHandler { error, viewMatcher ->
        takeScreenshot(spoon, "test_failed")
        DefaultFailureHandler(context).handle(error, viewMatcher)
    }
}
```

Why did we need a robot?

```
@Test
fun testSuccessfulRegistration() {
    onView(withId(R.id.etFirstName)).perform(typeText("Adam"))
    takeScreenshot(spoon, "first_name_entered")
    onView(withId(R.id.etLastName)).perform(typeText("McNeilly"))
    takeScreenshot(spoon, "last_name_entered")
    onView(withId(R.id.etEmail)).perform(typeText("amcneilly@okcupid.com"))
    takeScreenshot(spoon, "email_entered")
    onView(withId(R.id.etPhone)).perform(typeText("1234567890"))
    takeScreenshot(spoon, "phone_entered")
    takeScreenshot(spoon, "register_clicked")
    onView(withId(R.id.registerButton)).perform(click())

    onView(withId(R.id.tvFullName)).check(matches(withText("Adam McNeilly")))
    takeScreenshot(spoon, "full_name_displayed")
    onView(withId(R.id.tvEmailAddress)).check(matches(withText("amcneilly@okcupid.com")))
    takeScreenshot(spoon, "email_displayed")
    onView(withId(R.id.tvPhoneNumber)).check(matches(withText("(123)-456-7890")))
    takeScreenshot(spoon, "phone_displayed")
}
```

Let's Add To It

Your manager just came by and asked for an email opt in field.



EspressoPatronum

First Name
Adam

Last Name
McNeilly

Email Address
amcneilly@okcupid.com

Phone Number
1234567890

● Email Opt In

REGISTER

EspressoPatronum

WELCOME!

Adam McNeilly

amcneilly@okcupid.com

(123)-456-7890

✓ Opted In

RegistrationRobot

Only requires one new method on registration.

```
fun emailOptIn(): RegistrationRobot {
    onView(OPT_IN_MATCHER).perform(click())
    takeScreenshot(spoon, "opted_in")
    return this
}
```

UserProfileRobot

Only needs to consider each state.

```
fun assertOptedIn(): UserProfileRobot {
    onView(EMAIL_OPT_IN_DISPLAY_MATCHER).check(matches(isChecked()))
    takeScreenshot(spoon, "assert_email_opt_in")
    return this
}

fun assertOptedOut(): UserProfileRobot {
    onView(EMAIL_OPT_IN_DISPLAY_MATCHER).check(matches(isNotChecked()))
    takeScreenshot(spoon, "assert_email_opt_out")
    return this
}
```

Test

Test now only requires two really quick add ons to consider.

```
@Test
fun testSuccessfulRegistrationWithOptIn() {
    RegistrationRobot(spoon)
            .firstName("Adam")
            .lastName("McNeilly")
            .email("amcneilly@okcupid.com")
            .phone("1234567890")
            .emailOptIn()
            .register()

    UserProfileRobot(spoon)
            .assertFullNameDisplay("Adam McNeilly")
            .assertEmailDisplay("amcneilly@okcupid.com")
            .assertPhoneDisplay("(123)-456-7890")
            .assertOptedIn()
}
```

Recap

1. Use robots to solve separation of concerns problem
2. Makes your tests more readable and fun to write
3. Leverage this for better reporting
4. This is not specific to Espresso/Spoon/Falcon.

Contact

- Adam McNeilly - OkCupid (We're Hiring!)
- Twitter - @AdamMc331
- https://github.com/AdamMc331/EspressoPatronum