TESTOWANIE FUNKCJONALNE METODĄ CZARNEJ SKRZYNKI Z WYKORZYSTANIEM ROBOTIUM

by
Ryszard Sułkowski

SPIS TREŚCI

MINI SŁOWNICZEK TERMINÓW

- **APK** – Archiwum w jakim są dystrybuowane aplikacje Androidowe (format ZIP)
- AndroidManifest.xml – Plik podstawowych ustawień aplikacji
- Android – Mobilny system operacyjny na bazie Linuxa
- Instrumentacja – Mechanizm pozwalający na wspólne działania testu i aplikacji testowanej w tym samym procesie (def. własna: byt pasożytniczo-symbiotyczny)
- Aktywność (Activity) – mniej więcej: Ekran prezentowany użytkownikowi
- Intencja (Intent) – mniej więcej: Chęć do zrobienia czegoś np. Możemy chcieć wyświetlić użytkownikowi nową aktywność lub aplikację lub zadzwonić lub wysłać sms ...
- Inne...

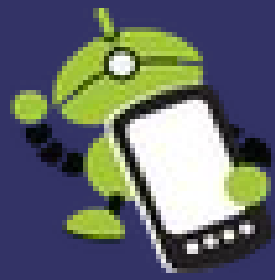
KILKA SŁÓW O TESTACH CZARNEJ SKRZYNKI I AUTOMATYZACJI TESTÓW FUNKCJONALNYCH

Czego potrzebujemy?

→ Aplikację w formie APKa
→ Scenariusz testu
→ Podstawowe informacje o testowanej aplikacji (nazwa: pakietu, aktywności)
→ Pomysł na architekturę testów
→ Przynajmniej podstawową* umiejętność programowania w języku Java (lub podobnym)
→ Przepływu informacji ze strony deweloperów aplikacji

Czego nie mamy (ale możemy mieć) ?

→ Kodu źródłowego testowanej aplikacji



CZYM JEST ROBOTIUM I CO POWINNIŚMY WIEDZIEĆ ZANIM NAPISZEMY PIERWSZY TEST?

- ROBOTIUM:
  - Framework stworzony z myślą o łatwym tworzeniu potężnych i solidnych automatycznych testów black-boxowych dla aplikacji Androidowych
  - Pozwala na tworzenie testów automatycznych: systemowych, funkcjonalnych i akceptacyjnych poprzez wiele aktywności
  - Posiada wsparcie dla większości elementów Androida
  - Testy działają w tym samym procesie co aplikacja

KONFIGURACJA ŚRODOWISKA

CZEGO POTRZEBUJEMY?

- Android SDK
- Java JDK
- Eclipse IDE*
- ADT Plugin for Eclipse IDE
- Emulator / Device
- Biblioteka: robotium-solo.jar

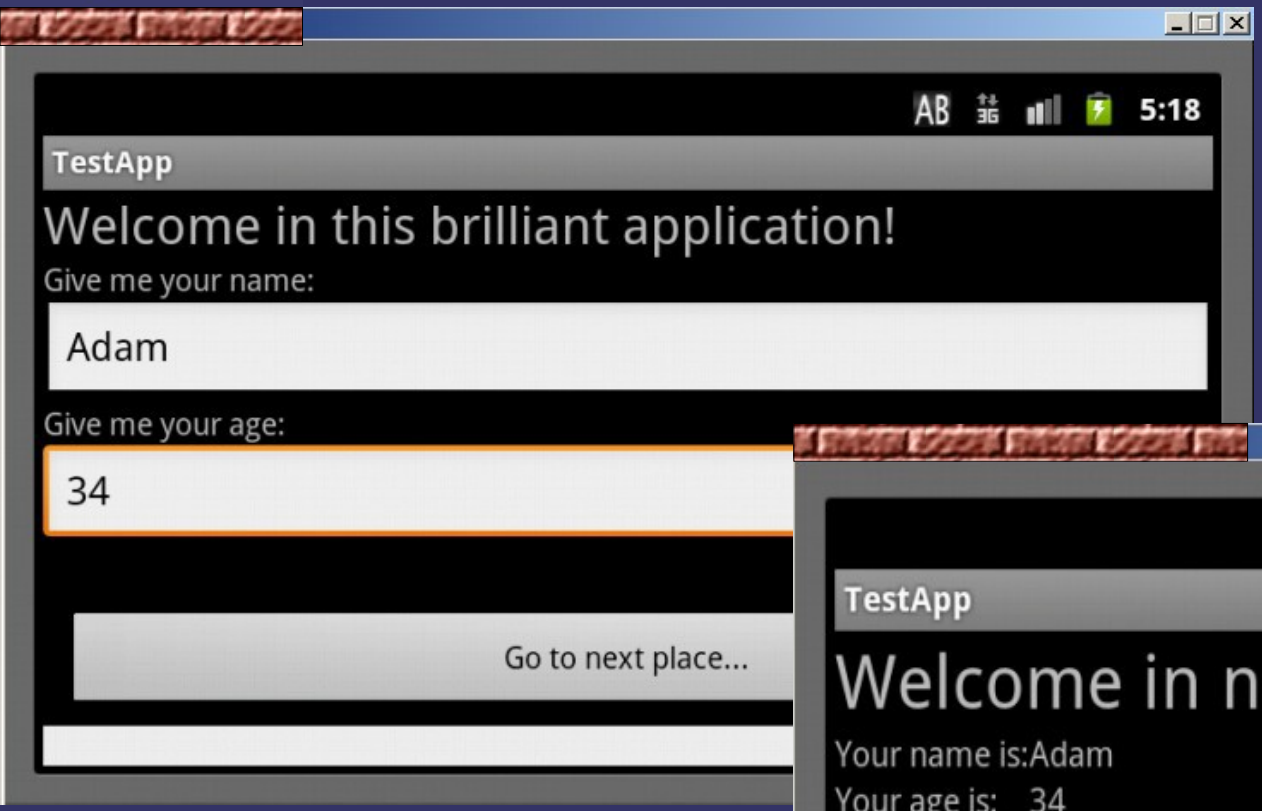
CO POWINNIŚMY USTAWIĆ?

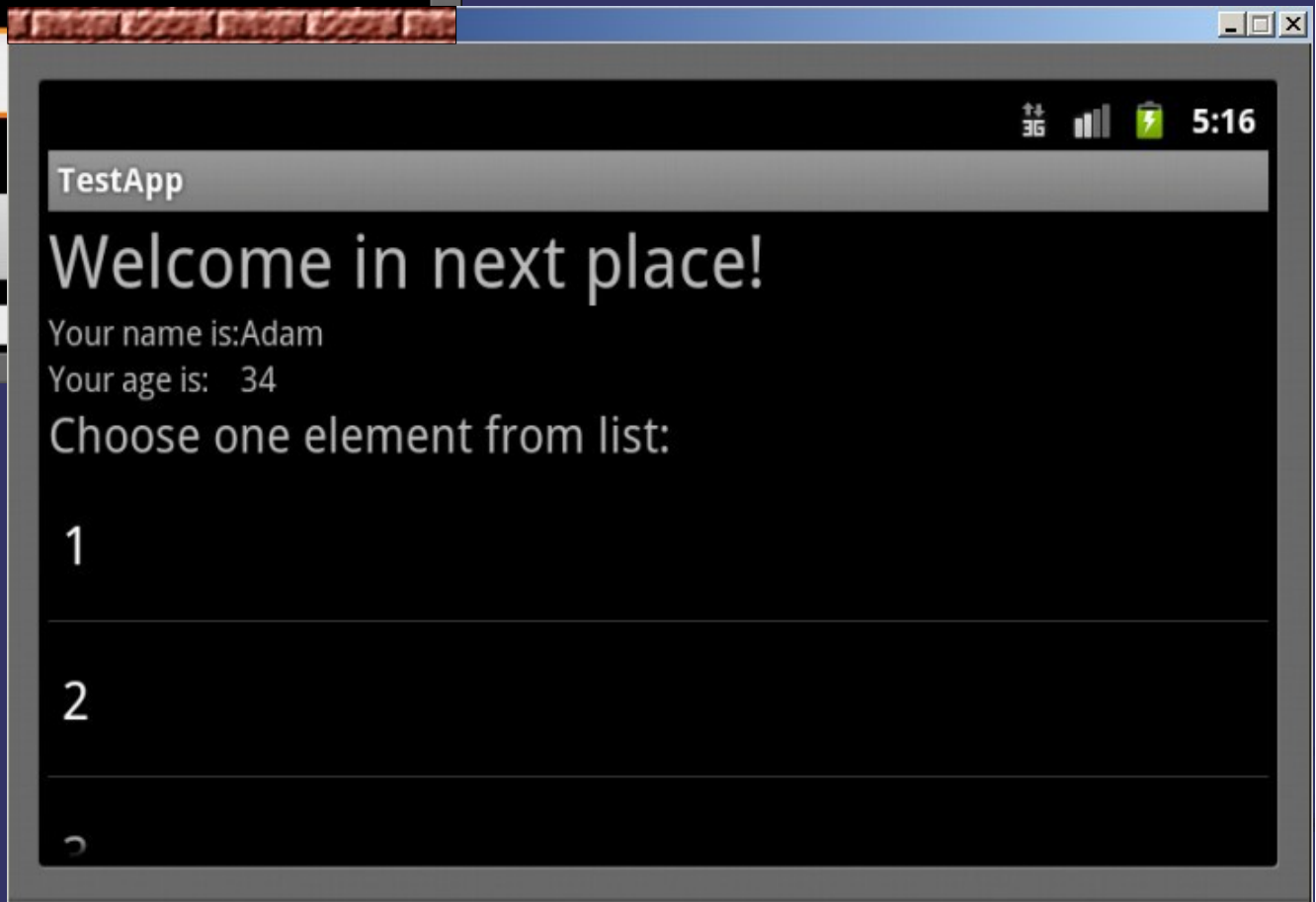
- Zmienną środowiskową: ANDROID_HOME na ścieżkę do SDK
- Dodać: %ANDROID_HOME%\tools;%ANDROID_HOME %\platform-tools; do PATHa
- Ścieżkę do SDK w Eclipse IDE:

Window → Preferences → Android → SDK Location

- Dodać: robotium-solo.jar do CLASSPATHa

PIERWSZY TEST: Aplikacja przykładowa





PIERWSZY TEST: Czym dysponujemy?

- `Solo solo = new Solo(getInstrumentation(), getActivity());`

- `assertCurrentActivity(String message, String activity)`
- `sleep(int milliseconds)`
- `clickOnView(View view)`
- `enterText(EditText edit, String text)`
- `clickOnButton(String name)`
- `waitForActivity(String actName, int milliseconds)`
- `clickOnText(String text)`

PIERWSZY TEST

- Piszemy.. zacznijmy od klasy bazowej..

```
@SuppressWarnings("rawtypes")
public abstract class BaseRobotiumTestCase extends ActivityInstrumentationTestCase2 {

	protected static final String TARGET_PACKAGE_ID = "com.rysu_test.app";
	protected static final String LAUNCHER_ACTIVITY_FULL_CLASSNAME = "com.rysu_test.app.TestAppActivity";
	protected Solo solo;

	private static Class<?> launcherActivityClass;
	static {
		try {
			launcherActivityClass = Class
					.forName(LAUNCHER_ACTIVITY_FULL_CLASSNAME);
		} catch (ClassNotFoundException e) {
			throw new RuntimeException(e);
		}
	}

	@SuppressWarnings("unchecked")
	public BaseRobotiumTestCase() throws ClassNotFoundException {
		super(TARGET_PACKAGE_ID, launcherActivityClass);
	}

	@Override
	protected void setUp() throws Exception {
		super.setUp();
		solo = new Solo(getInstrumentation(), getActivity());
	}

	@Override
	public void tearDown() throws Exception {
		solo.finishOpenedActivities();
		super.tearDown();
	}
}
```

URUCHAMIANIE TESTU W ECLIPSE IDE

- Utwórz AVD lub zainstaluj urządzenie
- Stwórz Android Test Project
- Dodaj: robotium-solo.jar do CLASSPATH
- Dodaj: instrumentation package do AndroidManifest.xml
- Prawym klikiem na Android Test Project i „Run As: Android JUnit Test”

URUCHAMIANIE TESTU Z LINII KOMEND

adb shell am instrument

adb shell am instrument -w
com.android.foo/android.test.InstrumentationTestRunner

adb shell am instrument -w -e package com.android.foo.subpkg
com.android.foo/android.test.InstrumentationTestRunner

adb shell am instrument -w -e class com.android.foo.FooTest
com.android.foo/android.test.InstrumentationTestRunner

adb shell am instrument -w -e class
com.android.foo.FooTest#testFoo
com.android.foo/android.test.InstrumentationTestRunner

→ other

CZEGO SIĘ SPODZIEWAĆ JAKO WYNIKU DZIAŁANIA TESTU?

Przy uruchomieniu z linii komend:

com.mobica.rysu_test.app.test.HomeScreenMobicaRobotiumTestCase:....
Test results for InstrumentationTestRunner=....
Time: 41.284
OK (4 tests)

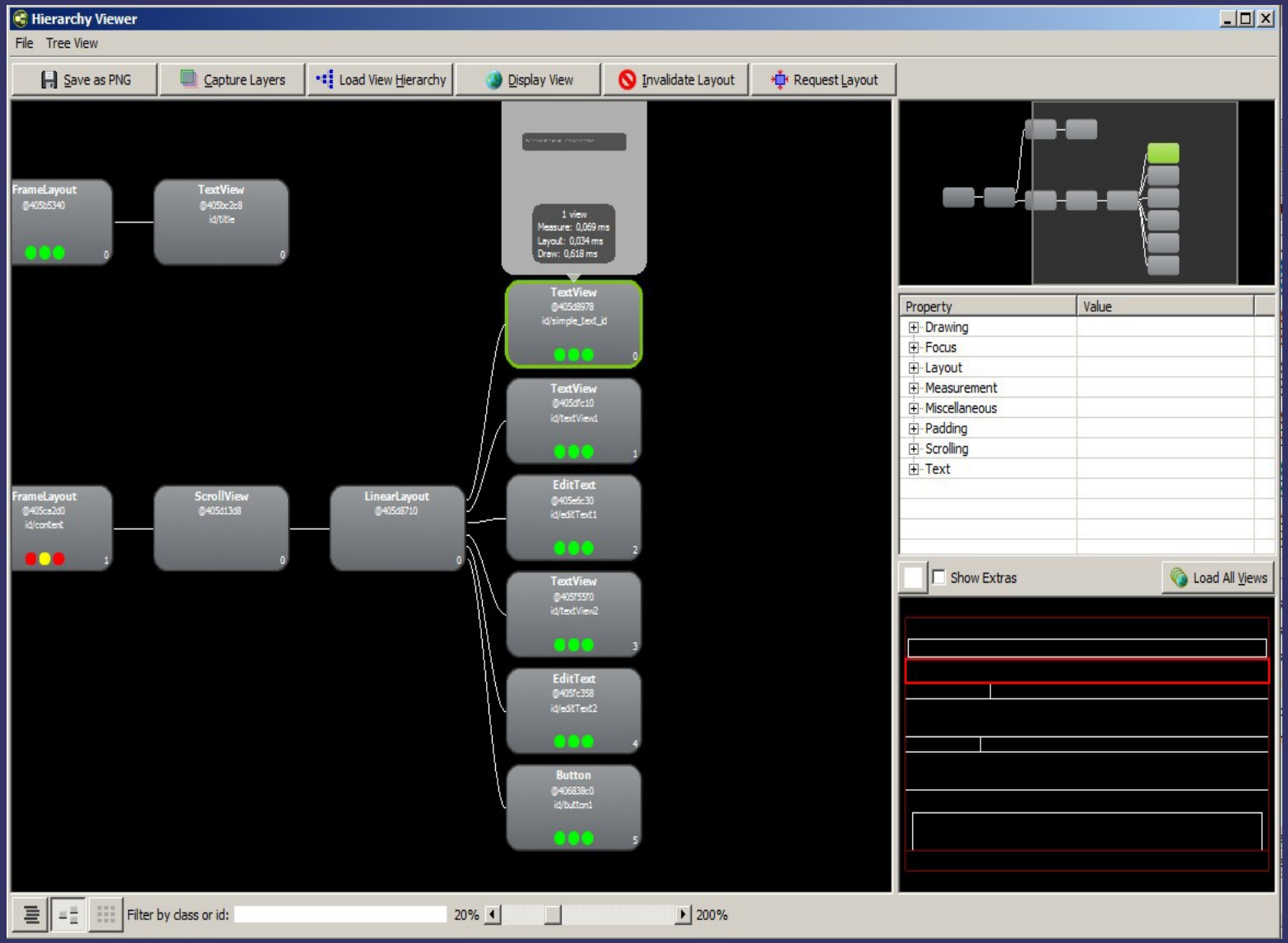
Przy uruchomieniu z Eclipse IDE:

Dużo zielonego koloru :)

NARZĘDZIA O KTÓRYCH ISTNIENIU POWINNIŚMY WIEDZIEĆ

- Hierarchyviewer
- DDMS
- Inne

Hierarchyviewer



DYGRESJA NA TEMAT ARCHITEKTURY TESTÓW

- Co wydarzy się w przyszłości prawie na pewno ...
- Zmiany w UI – Jak się przygotować ?
- Re-factoring – Jak nie stracić mnóstwa czasu ?
- Własna biblioteka klas narzędziowych? Wzorzec projektowy ?

DRUGI, TRZECI I KOLEJNE TESTY

Zajrzyjmy do kodu...

DRUGI, TRZECI I KOLEJNE TESTY: BŁĘDY

- “INSTRUMENTATION_STATUS_CODE: -1”

- Eeloo.. czegoś tu nie ma...

- “INSTRUMENTATION_STATUS_CODE: 0” ***

- Ej.. Tym razem wszystko jest ok więc o co kaman?

- „Zwiecha”

Sprawdź czy masz poprawną metodę tearDown()

- Inne

PRZYDATNE LINKI

- http://developer.android.com/guide/topics/testing/index.html
- http://code.google.com/p/robotium/
- www.google.com

KONIEC

DZIĘKUJĘ ZA UWAGĘ