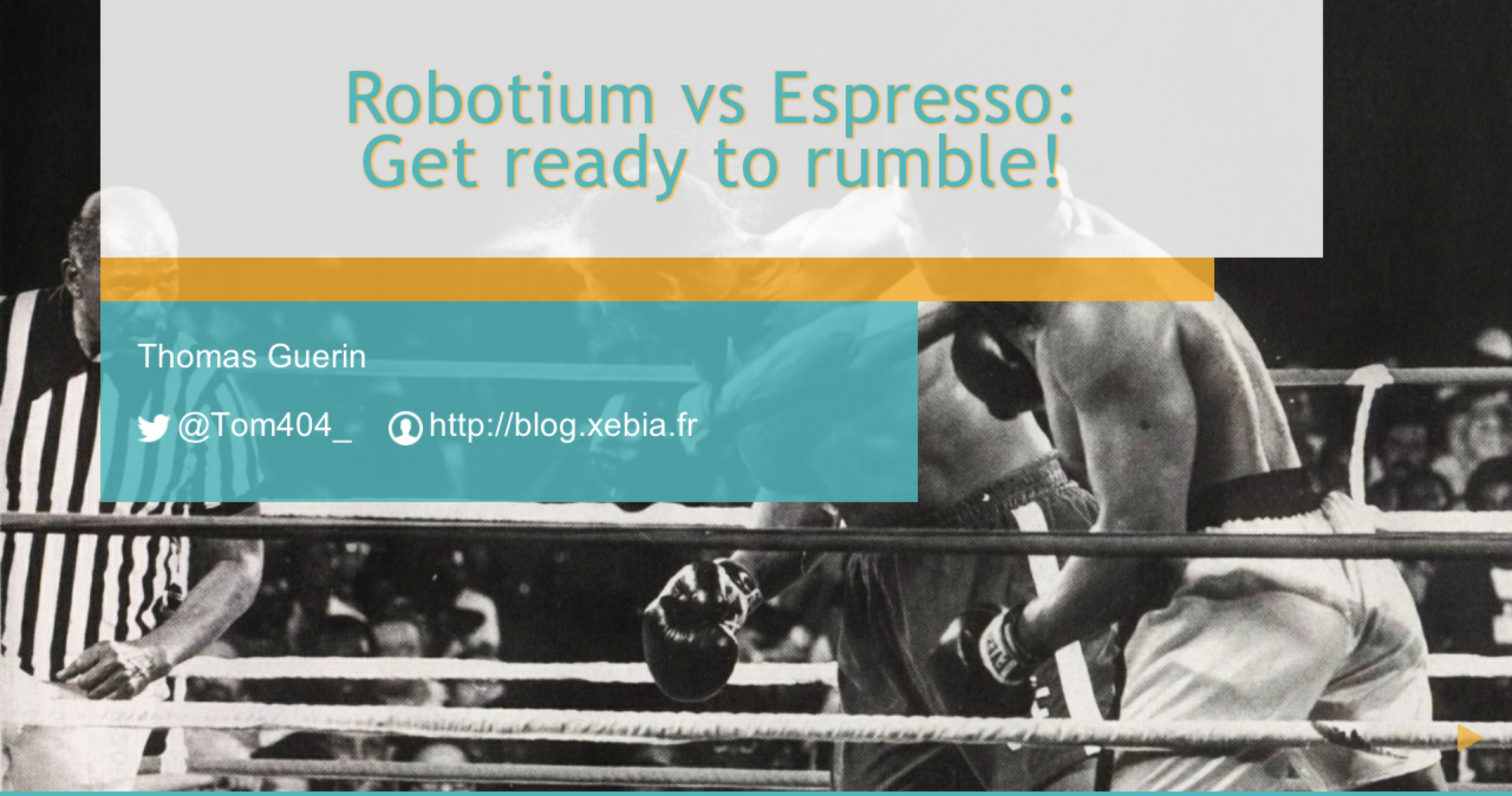
Robotium vs Espresso: Get ready to rumble!



Thomas Guerin

@Tom404_ http://blog.xebia.fr

Les combattants

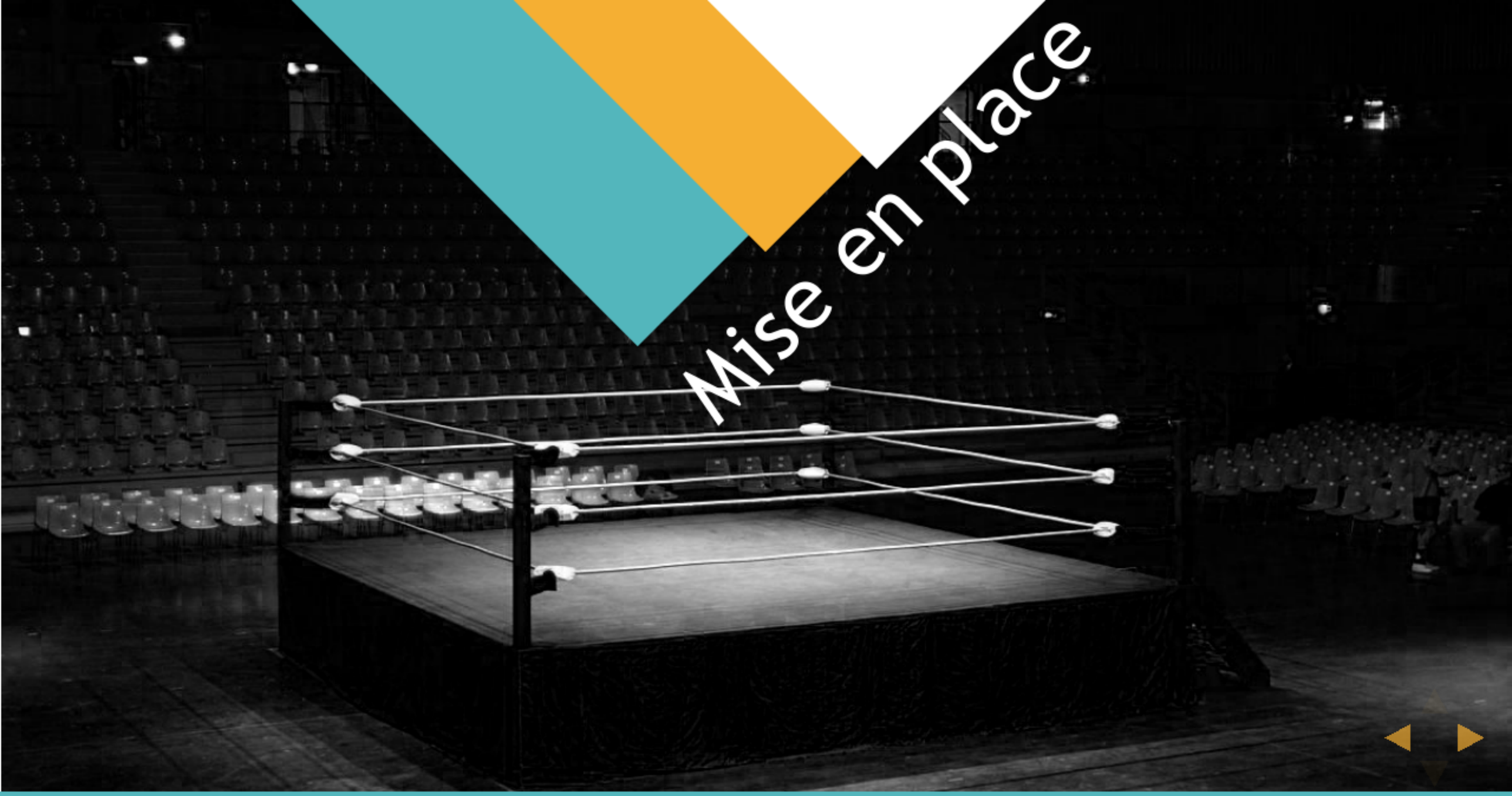
Robotium

Version: 5.1

Espresso

Version: 1.1

Mise en place



Dépendances Gradle

Robotium

```
dependencies {
    androidTestCompile('com.jayway.android.robotium:robotium-solo:5.1')
    androidTestCompile('com.squareup:fest-android:1.0.7')
}
```

Dépendances Gradle

Espresso

```
android.packagingOptions {
    exclude 'META-INF/LICENSE.txt'
    exclude 'LICENSE.txt'
    exclude 'META-INF/LICENSE'
    exclude 'META-INF/NOTICE'
}
```

```
dependencies {
    /* Jars récupérés depuis https://code.google.com/p/android-test-kit */
    androidTestCompile files('libs/espresso-1.1.jar', 'libs/testrunner-1.1.
        'libs/testrunner-runtime-1.1.jar')
    androidTestCompile 'com.google.guava:guava:14.0.1',
        'com.squareup.dagger:dagger:1.1.0',
        'org.hamcrest:hamcrest-core:1.1',
        'org.hamcrest:hamcrest-integration:1.1',
        'org.hamcrest:hamcrest-library:1.1'
}
```

Dépendances Gradle

Double Espresso

https://github.com/JakeWharton/double-espresso

```
android.packagingOptions {
    exclude 'META-INF/LICENSE.txt'
    exclude 'LICENSE.txt'
    exclude 'META-INF/LICENSE'
    exclude 'META-INF/NOTICE'
}
```

```
androidTestCompile 'com.jakewharton.espresso:espresso:1.1-r2'
androidTestCompile 'com.jakewharton.espresso:espresso-support-v4:1.1-r2'
```

Test runner

Configuration

- Robotium utilise le test runner de la plateforme
- Espresso nécessite un test runner spécifique

```
defaultConfig {
testInstrumentationRunner
"com.google.android.apps.common.testing.testrunner.GoogleInstrumentationTestRunn

}
```

Test runner

GoogleInstrumentationTestRunner

- onCreate de l'application finalisé avant le début des tests
- Quand l'instrumentation est terminée, les activités sont finies
- Monitoring des activités plus fiable
- Peut être utilisé par d'autres frameworks

Initialisation

```
public class SimpleActionsTest
            extends ActivityInstrumentationTestCase2<MainActivity> {

    private Solo solo;

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

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

Initialisation

Espresso

```
public class SimpleActionsTest
        extends ActivityInstrumentationTestCase2<MainActivity> {

    @Override
    protected void setUp() throws Exception {
        super.setUp();
        // Espresso ne va pas lancer l'activité pour nous
        getActivity();
    }
}
```


Expressions

Robotium API

- Une classe centrale : Solo
- Des méthodes utilitaires :
  - getView(int viewId)
  - clickOnButton(int index)
  - clickOnButton(String text)
  - searchText(String text)
  - et bien d'autres encore
- Assertions avec FEST Android

Robotium API

Exemple

```
// Récupération de la vue
TextView sectionLabel = (TextView) solo.getView(R.id.section_label);

// Assertions
ANDROID.assertThat(sectionLabel).hasText("Section 1").isVisible();
```

Espresso API

- Un unique point d'entrée : Espresso
- 2 types d'interactions possibles (onView et onData)
- 3 composants essentiels :
  - ViewMatchers
  - ViewAssertions
  - ViewActions
- Utilise activement Hamcrest

Espresso API

ViewMatchers

- Collection d'objets pour localiser une vue dans la hiérarchie
  - ViewMatchers.withId(int viewId)
  - ViewMatchers.withText(String text)
  - ViewMatchers.hasSibling(Matcher<View> siblingMatcher)
- Tous implémentent l'interface Matcher <? super View>
- Manque de restrictions = AmbiguousViewMatcherException

Espresso API

Exemples ViewMatchers

```
onView( withId(R.id.section_label) )
```

```
onView( allOf(withId(R.id.section_label), withText("Section 1")))
```

```
onView( withIdAndText(R.id.section_label, "Section 1") )
```

Espresso API

ViewAssertions

- Collection d'objets pour vérifier l'état d'une vue
  - ViewAssertions.doesNotExist()
  - ViewAssertions.matches(matcher)
  - ViewAssertions.selectedDescendantsMatch(selector, matcher)
- Utilisation de matchers
- Passées en paramètres de ViewInteraction.check

Espresso API

Exemples ViewAssertions

```
onView(withId(R.id.section_label)).check(doesNotExist())
```

```
onView(withId(R.id.section_label)).check(matches(isDisplayed()))
```

Espresso API

ViewActions

- Collection d'objets pour intéragir avec une vue
  - ViewActions.click()
  - ViewActions.typeText(text)
- Passées en paramètres de ViewInteraction.perform

```
onView(withId(R.id.my_button)).perform(click())
```

ListView

Robotium

```
// Clic sur item
solo.clickOnText("textToFind");

// Si jamais il y'en a plusieurs -> utilisation d'un index
solo.clickOnText("textToFind", 3);

// Ou choisir directement la ligne
solo.clickInList(2);
```

ListView

Espresso

- ~~Espresso.onView~~ Espresso.onData

```
onData(allOf(is(instanceOf(String.class)), is("textToFind"))).perform(click(

// Ou directement à une position
onData(is(instanceOf(String.class))).atPosition(0).perform(click());

// Possibilité de préciser une listview, utile pour le view pager
onData(allOf(is(instanceOf(String.class)), is("textToFind")))
        .inAdapterView(withId(R.id.my_list)).perform(click());

// Interaction avec une vue enfant de la ligne
onData(allOf(is(instanceOf(String.class)), is("textToFind")))
        .onChildView(withId(R.id.my_child))
        .perform(click());
```

ListView

Espresso (cas plus complexe) 1/2

```
// Type renvoyé par la méthode getItem de l'adapteur
public class Item {
    public String name;
    public String itemContent;
}
```

```
onData(allOf(is(instanceOf(Item.clazz)),
    hasProperty("name", equalTo("nameToFind")))).perform(click());
```

ListView

Espresso (cas plus complexe) 2/2

```
public static Matcher<Object> withItemName(final Matcher<String> itemTextMat
    return new BoundedMatcher<Object, Item>(Item.class) {
        @Override
        public boolean matchesSafely(Item item) {
            return itemTextMatcher.matches(item.name);
        }

        @Override
        public void describeTo(Description description) {
            description.appendText("with item name: ");
            itemTextMatcher.describeTo(description);
        }
    };
}
```

```
onData(withItemName(equalTo("nameToFind"))).perform(click());
```

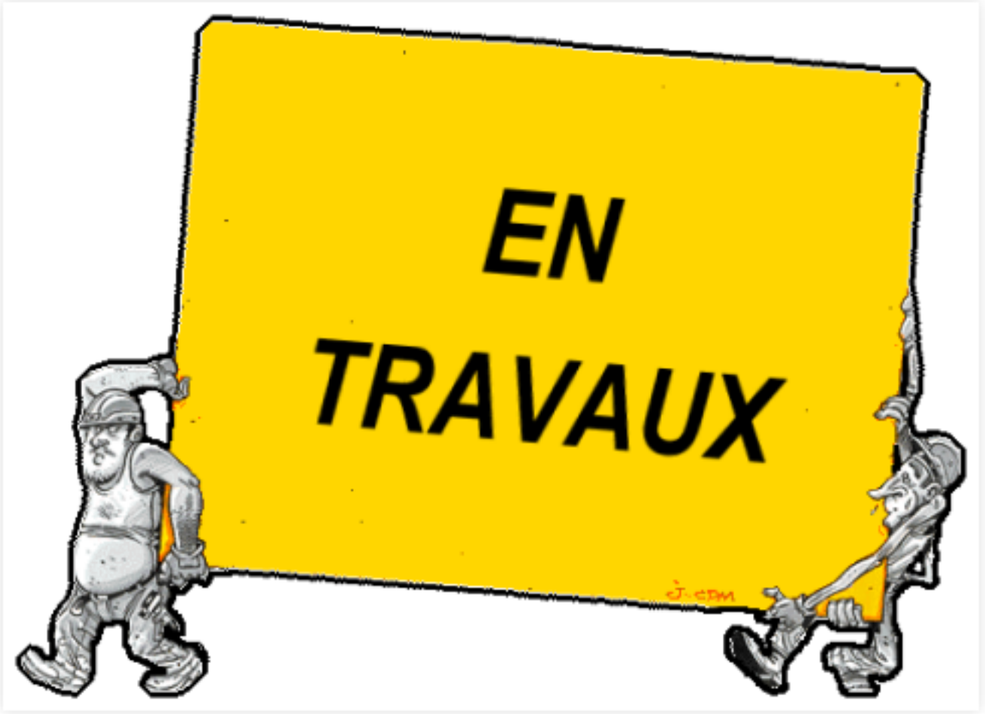
Webview

Robotium

- Possibilité d'interagir avec les webviews
  - solo.getWebElements(by)
  - solo.clickOnWebElement(by)
- Plusieurs types de recherches disponibles
  - By.id(String id)
  - By.className(String className)
  - By.textContent(String textContent)
  - By.name(String name)
  - By.cssSelector(String selectors)

Webview

Espresso



Drawer layout

Robotium

```
// Attention l'id du frame layout du menu doit absolument être "left_drawer
solo.setNavigationDrawer(Solo.OPENED);

solo.setNavigationDrawer(Solo.CLOSED);
```

Espresso

```
DrawerActions.openDrawer(R.id.drawer_layout);

DrawerActions.closeDrawer(R.id.drawer_layout);
```

Action bar

Robotium

```
solo.clickOnActionBarItem(R.id.action_example);

// Cas de l'action bar overflow
solo.sendKey(KeyEvent.KEYCODE_MENU);
solo.clickOnText("action");
```

Espresso

```
onView(withId(R.id_action_example)).perform(click());

// Cas de l'action bar overflow
openActionBarOverflowOrOptionsMenu(getInstrumentation().getTargetContext());
onView(withText("action")).perform(click());
```

Back button et keyboard

Robotium

```
solo.goBack();
// solo.goBack() est aussi utilisé pour fermer le clavier
```

Espresso

```
Espresso.pressBack();
Espresso.closeSoftKeyboard();
```

Enchaînements

Robotium

- Repose sur le principe du sleep and see
  - solo.waitForActivity(Class class, int timeout)
  - solo.waitForCondition(Condition condition, int timeout)
  - solo.waitForDialogToOpen()

```
solo.clickOnView(R.id.start_activity_button);

// On attend que la vue apparaisse avant d'agir
solo.waitForView(R.id.view_in_new_activity);

solo.clickOnView(R.id.view_in_new_activity);
```

Espresso

- Entrée en jeu du GoogleInstrumentationTestRunner
- Monitoring plus fin des activités et des ressources
- Analyse du ThreadUI pour savoir quand agir
- Pas de wait

```
onView(withId(R.id.start_activity_button)).perform(click())
onView(withId(R.id.view_in_new_activity)).perform(click());
```

Espresso

- Possibilité d'enregistrer des idlingResources

```
class EspressoThreadPool extends ThreadPoolExecutor implements IdlingResourc
    private int threadCount = 0;
    ...
    @Override
    public synchronized void execute(Runnable r) {
        threadCount++;
        super.execute(r);
    }
    @Override
    public synchronized boolean isIdleNow() {
        return threadCount == 0;
    }
    @Override
    protected synchronized void afterExecute(Runnable r, Throwable t) {
        super.afterExecute(r, t);
        threadCount--;
    }
}
```

Espresso


```
Espresso.registerIdlingResources(espressoThreadPool);
```

Conclusion