EBook Gratis

APRENDIZAJE android-espresso

Free unaffiliated eBook created from
Stack Overflow contributors.

#android-espresso

Tabla de contenido

Acerca de

You can share this PDF with anyone you feel could benefit from it, downloaded the latest version from: android-espresso

It is an unofficial and free android-espresso ebook created for educational purposes. All the content is extracted from Stack Overflow Documentation, which is written by many hardworking individuals at Stack Overflow. It is neither affiliated with Stack Overflow nor official android-espresso.

Use the content presented in this book at your own risk; it is not guaranteed to be correct nor accurate, please send your feedback and corrections to info@zzzprojects.com

Capítulo 1: Empezando con Android-espresso

Observaciones

Esta sección proporciona una descripción general de qué es Android-espresso y por qué un desarrollador puede querer usarlo.

También debe mencionar cualquier tema importante dentro de Android-espresso, y vincular a los temas relacionados. Dado que la Documentación para android-espresso es nueva, es posible que deba crear versiones iniciales de esos temas relacionados.

Examples

Instrucciones de configuración de espresso

- Configure su entorno de prueba
- Descargar Espresso
- Establecer el corredor de instrumentación.
- Ejemplo de archivo build.gradle
- Analítica
- Agrega la primera prueba
- Ejecución de pruebas Esta guía cubre la instalación de Espresso con el Administrador de SDK y su construcción con Gradle. Se recomienda Android Studio.

Configure su entorno de prueba

Para evitar la descamación, le recomendamos que desactive las animaciones del sistema en los dispositivos virtuales o físicos utilizados para las pruebas.

En su dispositivo, en Configuración-> Opciones de desarrollador, desactive las siguientes 3 configuraciones:

- Escala de animación de ventana
- Escala de animación de transición
- Escala de duración del animador

Descargar Espresso

- Asegúrese de haber instalado el último repositorio de soporte de Android en Extras (consulte las instrucciones).

- Abra el archivo build.gradle de su aplicación. Por lo general, este no es el archivo build.gradle de nivel superior, sino app / build.gradle.

- Agregue las siguientes líneas dentro de las dependencias:

  androidTestCompile 'com.android.support.test.espresso: espresso-core: 2.2.2'
  androidTestCompile 'com.android.support.test: runner: 0.5'

- Consulte la sección de descargas para ver más artefactos (espresso-contrib, espresso-web, etc.)

- Establecer el corredor de instrumentación.

Agregue al mismo archivo build.gradle la siguiente línea en android.defaultConfig: testInstrumentationRunner `"android.support.test.runner.AndroidJUnitRunner"` Ejemplo de archivo build.gradle

```
apply plugin: 'com.android.application'

    android {
        compileSdkVersion 22
        buildToolsVersion "22"

        defaultConfig {
            applicationId "com.my.awesome.app"
            minSdkVersion 10
            targetSdkVersion 22.0.1
            versionCode 1
            versionName "1.0"

            testInstrumentationRunner "android.support.test.runner.AndroidJUnitRunner"
        }
    }

dependencies {
    // App's dependencies, including test
    compile 'com.android.support:support-annotations:22.2.0'

    // Testing-only dependencies
    androidTestCompile 'com.android.support.test:runner:0.5'
    androidTestCompile 'com.android.support.test.espresso:espresso-core:2.2.2'
}
```

Analítica

Para asegurarnos de que estamos en el camino correcto con cada nueva versión, el corredor de pruebas recopila los análisis. Más específicamente, carga un hash del nombre del paquete de la aplicación bajo prueba para cada invocación. Esto nos permite medir tanto el recuento de paquetes únicos que utilizan Espresso como el volumen de uso.

Si no desea cargar estos datos, puede optar por no participar pasando el siguiente argumento al corredor de prueba: disableAnalytics "true" (vea cómo pasar los argumentos personalizados).

Agrega la primera prueba

Android Studio crea pruebas de forma predeterminada en src / androidTest / java / com.example.package /

Ejemplo de prueba JUnit4 usando Reglas:

```
@RunWith(AndroidJUnit4.class)
@LargeTest
public class HelloWorldEspressoTest {

    @Rule
    public ActivityTestRule<MainActivity> mActivityRule = new
ActivityTestRule(MainActivity.class);

    @Test
    public void listGoesOverTheFold() {
        onView(withText("Hello world!")).check(matches(isDisplayed()));
    }
}
```

Pruebas de carrera

En Android Studio

Crear una configuración de prueba

En Android Studio:

- Abrir el menú Ejecutar -> Editar configuraciones
- Agregar una nueva configuración de pruebas de Android
- Elige un módulo
- Agregue un corredor de instrumentación específico:

  android.support.test.runner.AndroidJUnitRunner

Ejecute la configuración recién creada.

Desde la línea de comandos a través de Gradle

Ejecutar

```
./gradlew connectedAndroidTest
```

El espresso tiene básicamente tres componentes:

1. ViewMatchers: permite buscar vistas en la jerarquía de vistas actual
2. ViewActions - permite realizar acciones en las vistas
3. ViewAssertions - permite afirmar el estado de una vista

Prueba de Base Espresso

```
onView(ViewMatcher)       -- 1
```

```
 .perform(ViewAction)     -- 2
   .check(ViewAssertion); -- 3
```

1. Encuentra la vista
2. Realiza una acción en la vista.
3. Valida una afirmación

Verificando elementos de un Menú de Opciones (usando Spoon para hacer capturas de pantalla)

```
/**
 * @author piotrek1543
 *
 * This example provides a specific UI testing problem and how it is already solved
 * with Google's Espresso. Notice that I used also Spoon framework, as Espresso
 * lacks of taking screenshots functionality.
 */

@RunWith(AndroidJUnit4.class)
public class MainActivityAndroidTest {
    @Rule
    public ActivityTestRule<MainActivity> mRule = new ActivityTestRule<>(MainActivity.class);

    @Test
    public void checkIfSettingsMenuItemsAreVisible() throws InterruptedException {
        //open OptionsMenu to see available items
        openActionBarOverflowOrOptionsMenu(mRule.getActivity());
        //create a screenshot with 'options_menu' TAG
        Spoon.screenshot(mRule.getActivity(), "options_menu");
        //check if Settings item is Visible
        onView(withText(R.string.action_settings)).check(matches(isDisplayed()));
        //check if `Sort` item is Visible
        onView(withText(R.string.action_sort)).check(matches(isDisplayed()));
        //perform click on `Sort` OptionsMenu item
        onView(withText(R.string.action_sort)).perform(click());
        //create a screenshot with 'options_menu_sort' TAG
        Spoon.screenshot(mRule.getActivity(), "options_menu_sort");
        //check if `Sort -> By Value id` item is Visible
        onView(withText(R.string.menu_sort_length)).check(matches(isDisplayed()));
        //check if `Sort -> By Joke length` item is Visible
        onView(withText(R.string.menu_sort_a_z)).check(matches(isDisplayed()));
    }
}
```

Ver prueba

```
onView(withId(R.id.greet_button)) // withId(R.id.my_view) is a ViewMatcher
.perform(click())                 // click() is a ViewAction
.check(matches(not(isEnabled()))); // matches(isEnabled()) is a ViewAssertion
```

Encuentra alguna vista por ID

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

Buscar vista por el texto

```
onView(withText("Pay"))
onView(withText(R.string.pay))
```

Hola Mundo Espresso Ejemplo

Este es un tutorial para crear un ejemplo de hello world: utilizado para este ejemplo: Android Studio 2.3;

Para comenzar a usar Android Studio para crear un nuevo proyecto con una actividad vacía. Luego agregamos algunas funciones simples a la aplicación que podemos probar: agregamos un botón que cuando se hace clic en "Hello World" en una vista de texto.

El código de actividad se ve así:

```
package com.example.testing.helloworld;

import android.os.Bundle;
import android.support.v7.app.AppCompatActivity;
import android.view.View;
import android.widget.TextView;

public class MainActivity extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        final TextView textView = (TextView) findViewById(R.id.textView);

        findViewById(R.id.button).setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                textView.setText("Hello World!");
            }
        });
    }
}
```

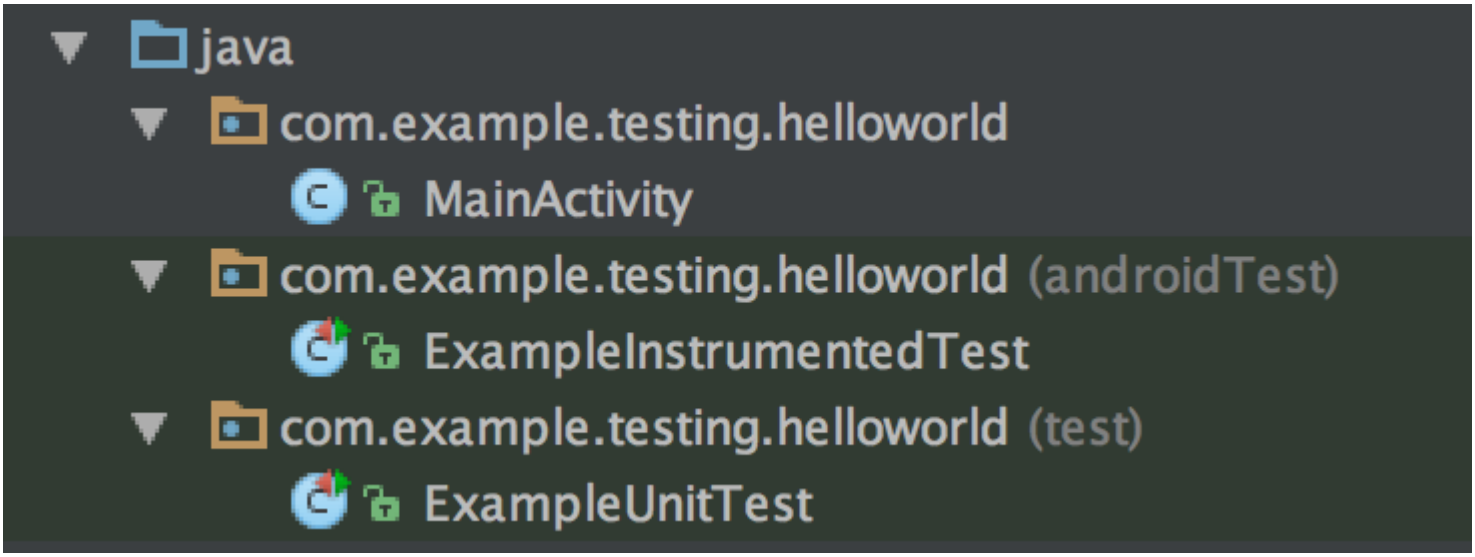
Y el diseño `activity_main` para esta actividad se ve así:

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical">

    <TextView
        android:id="@+id/textView"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="" />
```

```
    <Button
        android:id="@+id/button"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Say Hello" />
</LinearLayout>
```

Ahora queremos probar el comportamiento de esta nueva aplicación creada utilizando espresso. En general, el código de su aplicación está dentro del paquete `main` , las Pruebas unitarias están dentro de la `test` y las pruebas de instrumentación expreso están dentro del paquete `androidTest` . Si creas un nuevo proyecto de actividad vacío con Android Studio, ya debería haber creado esos paquetes y clases y debería tener este aspecto:



Para comenzar con espresso, debemos asegurarnos de que la dependencia de `espresso-core` esté incluida en el archivo `build.gradle` (tenga en cuenta que no está anotado con la palabra clave `compile` sino con `androidTestCompile`). Las dependencias en el archivo `build.gradle` creado por Android studio deberían tener este aspecto:

```
dependencies {
    compile fileTree(dir: 'libs', include: ['*.jar'])
    androidTestCompile('com.android.support.test.espresso:espresso-core:2.2.2', {
        exclude group: 'com.android.support', module: 'support-annotations'
    })
    compile 'com.android.support:appcompat-v7:25.2.0'
    compile 'com.android.support.constraint:constraint-layout:1.0.2'
    testCompile 'junit:junit:4.12'
}
```

Ahora que todo está configurado, podemos comenzar con la prueba real: abra el archivo `ExampleInstrumentationTest` y verá que ya hay una prueba de `useAppContext` generada en `useAppContext` interior. Cambiaremos esta clase de prueba y crearemos una prueba para verificar el comportamiento de nuestra aplicación:

```
@RunWith(AndroidJUnit4.class)
public class ExampleInstrumentedTest {

    @Rule
```

```
    public ActivityTestRule<MainActivity> mActivityRule = new ActivityTestRule<>(
        MainActivity.class, false, true);

    @Test
    public void checkHelloWorld() throws Exception {
        onView(withId(R.id.textView)).check(matches(withText("")));
        onView(withId(R.id.button)).perform(click());
        onView(withId(R.id.textView)).check(matches(withText("Hello World!")));
    }
}
```

Comience la prueba ejecutando la clase `ExampleInstrumentedTest` . Esta prueba entonces hace tres cosas:

1. Comprueba si la vista de texto contiene una cadena vacía ("")
2. Hace clic en el botón en nuestro diseño.
3. Vuelve a revisar el texto de la vista de texto si contiene "¡Hola mundo!"

La ActivityTestRule en la parte superior define qué actividad se prueba y la inicia al comienzo de la prueba. (También puede desactivar el inicio automático de una actividad y en su lugar iniciarlo manualmente dentro de cada prueba)

Las reglas de prueba son bastante simples:

- `onView(withId(R.id.textView))` busca una vista dentro de la pantalla actual por el ID de la vista dentro de nuestro archivo de diseño `activity_main` .
- `.check(matches(withText("")));` luego realiza un caso de prueba en esa vista.
- `.perform(click())` realiza una acción en una vista: `.perform(click())` acciones pueden ser clics, clics largos o swipes o algo más.

Este fue un tutorial para comenzar con las pruebas de instrumentación de espresso de android, ¡espero que te haya dado algunas ideas!

Lea Empezando con Android-espresso en línea: https://riptutorial.com/es/android-espresso/topic/3651/empezando-con-android-espresso

Capítulo 2: ¿Cómo crear Matchers personalizados?

Examples

Ejemplo de emparejador personalizado para probar el mensaje de error de TextView

1. Cree un nombre de clase `ErrorMatcher` dentro de su paquete de prueba con el siguiente código:

```
public class ErrorMatcher {

    @NonNull
    public static Matcher<View> withError(final String expectedErrorText) {
        Checks.checkNotNull(expectedErrorText);
        return new BoundedMatcher<View, TextView>(TextView.class) {
            @Override
            public void describeTo(final Description description) {
                description.appendText("error text: ");
                stringMatcher.describeTo(description);
            }

            @Override
            public boolean matchesSafely(final TextView textView) {
                return expectedErrorText.equals(textView.getError().toString());
            }
        };
    }
}
```

La lógica de coincidencia es encontrar el elemento `TextView` , cuyo texto de mensaje de error es igual al valor del texto de error esperado, pasando por el subconjunto de campos de `TextView` presentes en la jerarquía de diseño. `describeTo` método `describeTo` se utiliza para la salida de depuración.

2. Luego puede usar su emparejador personalizado en el caso de prueba como se muestra a continuación:

```
@Test
public void verifiesSignInErrorIsShown() {
    onView(withId(R.id.email_sign_in_button)).perform(click());
    onView(ErrorMatcher.withError("Your error text")).check(matches(isDisplayed()));
}
```

Lea ¿Cómo crear Matchers personalizados? en línea: https://riptutorial.com/es/android-espresso/topic/6748/-como-crear-matchers-personalizados-

Creditos

S. No	Capítulos	Contributors
1	Empezando con Android-espresso	Ahmad Aghazadeh, anuja jain, Community, piotrek1543, stamanuel
2	¿Cómo crear Matchers personalizados?	anuja jain, denys, piotrek1543