**ESPRESSO TESTING TOOLS**

****

DISUSUN OLEH :

KELOMPOK 9

MUHAMMAD FAISAL AMIR (6706160014)

BRYAN RAFSANZANI (6706160065)

MUHAMAD IKHSAN RAMADHAN (6706162062)

D3IF-40-02

**D3 TEKNIK INFORMATIKA**

**FAKULTAS ILMU TERAPAN**

**TELKOM UNIVERSITY**

**BANDUNG**

**2018**

1. **Pengertian Espresso**

Espresso adalah framework pengujian untuk Android yang memudahkan penguji dalam penulisan pengujian antarmuka pengguna (UI) yang l untuk aplikasi. Framework, yang merupakan bagian dari *Android Support Repository*, menyediakan API untuk menulis hasil pengujian UI untuk menyimulasikan interaksi pengguna di dalam aplikasi, mulai dari mengklik tombol dan menavigasi tampilan sampai memilih pilihan menu.

1. **Tujuan Espresso Testing**

Aplikasi yang baik adalah aplikasi yang memiliki sedikit bug dan eror, adapun cara untuk meminimalisir hal tersebut dilakukanlah testing. Espresso menghadirkan kemudahan dalam melakukan testing mengenai user interface di banyak perangkat, berikut tujuan utama espresso testing

* Otomatis dapat mendeteksi error dan bugs selama proses pengembangan aplikasi
* Memastikan kualitas dari produk yang membantu untuk meraih kepercayaan dari pelanggan
* Melakukan testing di berbagai perangkat dengan mudah tanpa melakukan testing dengan cara coba coba
* Memastikan ketepatan dari keinginan pelanggan saat mengembangakan aplikasi
* Effisiensi testing sebuah aplikasi andorid dengan menghemat tenaga manusia

1. **Hal – hal yang perlu di ketahui untuk menggunakan Espresso**

* Membuat dan menjalankan aplikasi dalam Android Studio.
* Membuat dan mengedit elemen UI menggunakan Layout Editor, memasukkan kode XML secara langsung, dan mengakses elemen UI dari kode Java .
* Menambahkan fungsionalitas onClick ke tombol.
* Membangun dua aplikasi TwoActivities dari pelajaran sebelumnya.
* Membangun aplikasi PhoneNumberSpinner dari pelajaran sebelumnya.
* Membangun aplikasi RecyclerView dari pelajaran sebelumnya.

1. **Petunjuk Pengaturan**

Saat melakukan pengujian dengan Espresso, terdapat beberapa hal yang harus diatur terlebih dahulu. Termasuk pengaturan pada *Environment* perangkat pengujian. Guna menghindari beberapa kesalahan seperti tidak terlihatnya dengan jelas hasil testing, sangat dianjurkan untuk mematikan 3 hal pada perangkat testing.

Pada perangkat pengujian pilih menu *Settings > Developer Options*, matikan atau *disable* 3 hal berikut :

* Window Animation Scale
* Transition Animation Scale
* Animator Duration Scale

Kemudian Tambahkan *Dependencies* pada projek dengan cara sebagai berikut :

* Buka projek aplikasi, pada file *build.gradle* atau app/build.gradle
* Tambahkan dependencies berikut :

androidTestImplementation 'com.android.support.test.espresso:espresso-core:3.0.2'

androidTestImplementation 'com.android.support.test:runner:1.0.2'

androidTestImplementation 'com.android.support.test:rules:1.0.2'

1. **Installasi Espresso**
   1. Untuk menggunakan espresso ini kita perlu menambahkan beberapa konfigurasi pada app/build.gradle. Berikut ini konfigurasi yang perlu kamu tambahkan.
   2. Konfigurasi pada app/build.gradle.

androidTestCompile('com.android.support.test.espresso:espresso-core:2.2.2', {  
 exclude group: 'com.android.support', module: 'support-annotations'  
})

* 1. Tambahkan instrument tes runner di tes config

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

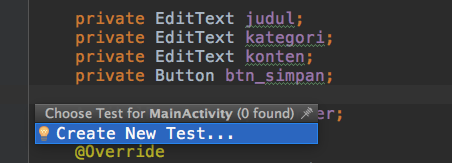
* 1. Berikut kode lengkapnya :

|  |  |
| --- | --- |
|  | apply plugin: 'com.android.application' |
|  |  |
|  | android { |
|  | compileSdkVersion 24 |
|  | buildToolsVersion "24.0.1" |
|  |  |
|  | defaultConfig { |
|  | applicationId "com.ahmadrosid.belajartdd" |
|  | minSdkVersion 15 |
|  | targetSdkVersion 24 |
|  | versionCode 1 |
|  | versionName "1.0" |
|  | testInstrumentationRunner "android.support.test.runner.AndroidJUnitRunner" |
|  | } |
|  | buildTypes { |
|  | release { |
|  | minifyEnabled false |
|  | proguardFiles getDefaultProguardFile('proguard-android.txt'), 'proguard-rules.pro' |
|  | } |
|  | } |
|  | } |
|  |  |
|  | dependencies { |
|  | compile fileTree(dir: 'libs', include: ['\*.jar']) |
|  |  |
|  | testCompile 'junit:junit:4.12' |
|  | testCompile 'org.mockito:mockito-core:1.10.19' |
|  | androidTestCompile('com.android.support.test.espresso:espresso-core:2.2.2', { |
|  | exclude group: 'com.android.support', module: 'support-annotations' |
|  | }) |
|  |  |
|  | compile 'com.android.support:appcompat-v7:24.1.1' |
|  | compile 'com.android.support:design:24.1.1' |
|  | } |

1. **Testing With Espresso**

Sudah menambahkan testing dependencies sekarang waktunya kita untuk melakukan testing. Pada skema test kali ini kita akan melakukan test untuk memastikan beberapa view telah di display di tampilan user.

Yang pertama kita lakukan adalah membuat file testnya. Untuk membuat file testnya bisa dilakukan dengan manual atau otomatis. Nah disini cara yang paling mudah untuk membuat file testnya adalah dengan menekan tombol CTRL + SHIFT + T.



Create new test dan tekan enter untuk membuat file testnya dan jangan lupa untuk memilih di package androidTest nya.

Selajuntnya tambahkan annotation test runnernya :

@RunWith(AndroidJUnit4.class)  
public class MainActivityTest {  
  
}

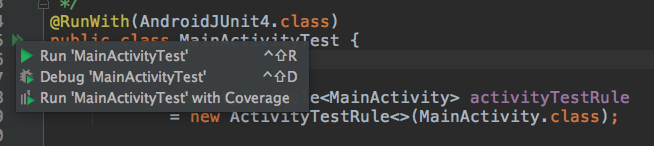
Setelah itu yang selanjutnya perlu kita lakukan adalah menambahkan test rule object activity yang akan kita test.

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

Selanjutnya kita jalankan skema test untuk memastikan bahwa view kita sudah di tampilkan di user interface atau belum. Berikut ini codenya

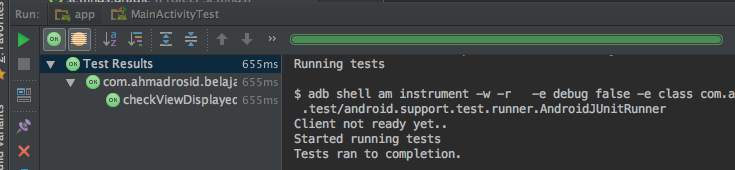
@Test  
public void checkViewDisplayed(){  
 activityTestRule.launchActivity(new Intent());  
 *onView*(*withId*(R.id.*judul*)).check(*matches*(*isDisplayed*()));  
 *onView*(*withId*(R.id.*kategori*)).check(*matches*(*isDisplayed*()));  
 *onView*(*withId*(R.id.*konten*)).check(*matches*(*isDisplayed*()));  
}

Untuk menjalankan klik icon run di samping kiri editor android studio seperti ini.



Untuk test ini kita perlu menggunakan emulator atau device karna yang dilakukan oleh espresso adalah dengan benar — benar memastikan aplikasi terinstall di devices dan di jalankan.

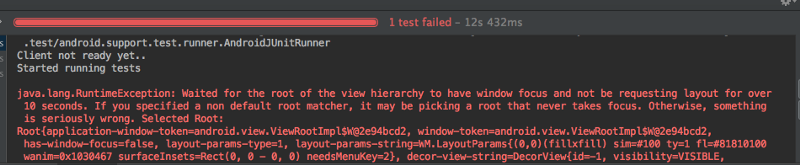
Jika test sudah dijalankan dan case yang kita cek benar — benar berjalan maka kita akan mendapatkan result seperti ini.



Sekarang coba kita jalankan skema jika salah satu view visibilitynya kita hilangkan.

<EditText  
 android:visibility="gone"  
 android:id="@+id/konten"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:layout\_margin="10dp"  
 android:hint="Konten"/>

Dan sekarang coba jalankan lagi testnya.

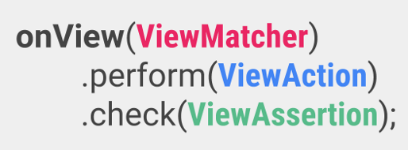


Dengan begitu kita mendapatkan errornya.

1. **Konsep Espresso**

Espresso cheat sheet yaitu kumpulan sintaks kode untuk menjalankan fungsi espresso. Berikut cheat espresso :

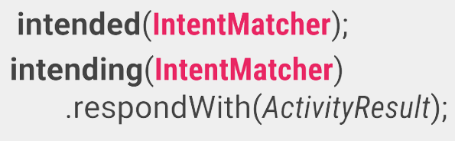
1. View Matcher

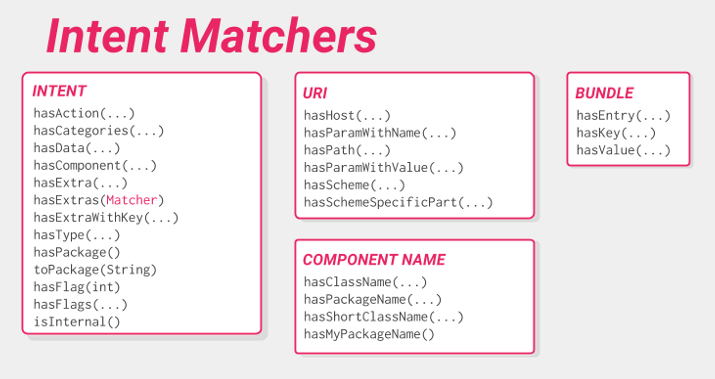




View Matcher ini digunakan untuk untuk mendaparkan object view jika di activity atau fragment biasa kita lakukan dengan *findViewById.*

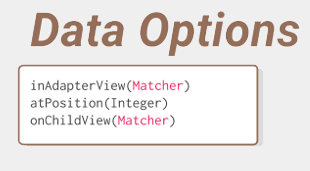
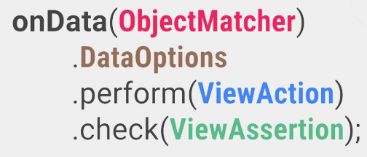
1. Intend Matcher





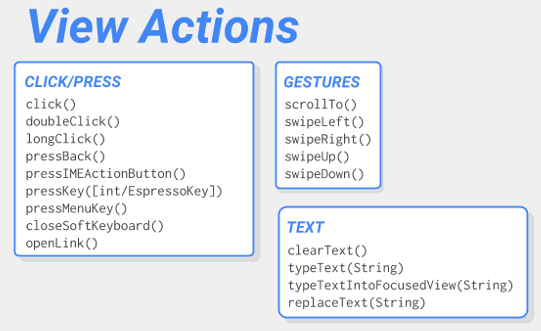
Intend matcher ini digunakan untuk menguji hasil pindahan activity apakah activity dapat berpindah dengan benar atau salah

1. Object Matcher



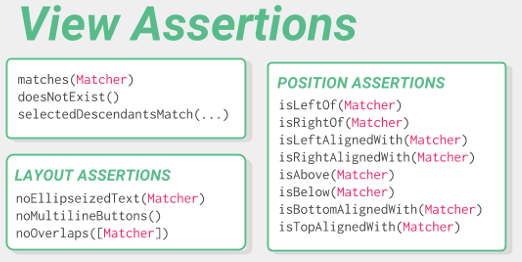
Object Matcher digunakan untuk menguji apakah data terambil maupun berubah saat di beri kode program

1. View Actions



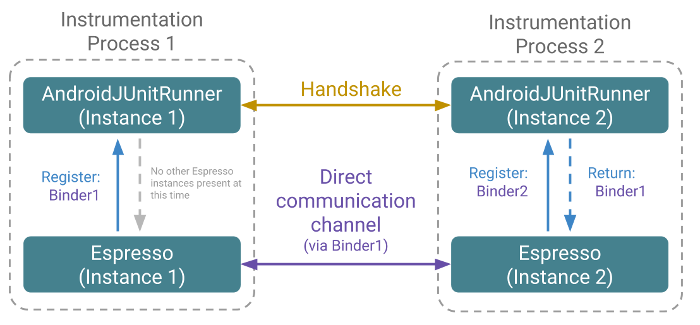
Dengan view action ini yang memungkinkan kita untuk melakukan perintah diantaranya click, scroll, input text dan action view yang lainya.

1. View Assertion



Dengan View Assertion ini kita akan mendapatkan hasil dari test yang telah kita lakukan.

1. **Arsitektur Espresso**



* 1. Kerangka Android membuat dan memulai proses baru untuk mengikuti struktur navigasi aplikasi . Setiap proses Instrumentasi termasuk contoh baru dari AndroidJUnitRunner. Pada tahap ini, 2 proses instrumentasi tidak dapat berkomunikasi satu sama lain.
  2. Setiap instance AndroidJUnitRunner mendaftarkan Espresso sebagai kerangka pengujiannya. 2 instance AndroidJUnitRunner melakukan jabat tangan untuk membuat koneksi antara satu sama lain. Pada saat yang sama, setiap instance AndroidJUnitRunner menghubungkan semua klien terdaftar seperti Espresso dengan rekan-rekan masing-masing dalam proses lain sehingga klien ini dapat membentuk saluran komunikasi langsung di antara mereka sendiri.
  3. Setiap instance AndroidJUnitRunner terus mencari contoh instansasi yang baru ditambahkan dan menguji klien kerangka kerja, membuat saluran komunikasi tambahan sesuai kebutuhan.

1. **Pemeriksaan Aksesibilitas**

Kelas AccessibilityCheck memungkinkan menggunakan kode pengujian yang ada untuk menguji masalah aksesibilitas. Saat berinteraksi dengan View selama pengujian, kerangka pengujian aksesibilitas menjalankan pemeriksaan secara otomatis sebelum melanjutkan. Cukup impor kelas dan tambahkan kode berikut ke metode pengaturan yang dianotasikan dengan @Before:

import android.support.test.espresso.contrib.AccessibilityChecks;  
  
@RunWith(AndroidJUnit4.class)  
@LargeTest  
public class AccessibilityChecksIntegrationTest {  
    @BeforeClass  
    public static void enableAccessibilityChecks() {  
        **AccessibilityChecks.enable();**  
    }  
}

Ini akan menyebabkan pemeriksaan aksesibilitas berjalan pada tampilan yang diberikan setiap kali menggunakan ViewAction dari kelas ViewActions. Untuk menjalankan pemeriksaan ini pada semua tampilan dalam hierarki, gunakan logika berikut:

AccessibilityChecks.enable()  
        .setRunChecksFromRootView(true);

Saat pertama kali mengaktifkan pemeriksaan, mungkin mengalami sejumlah masalah yang mungkin tidak dapat tangani dengan segera. dapat menekan kesalahan ini dengan menetapkan matcher untuk hasil yang ingin tekan. Pencocokan untuk AksesibilitasCheckResult muncul di AccessibilityCheckResultUtils dalam kerangka pengujian aksesibilitas. Misalnya, untuk menekan semua kesalahan untuk tampilan dengan ID R.id.example\_view:

AccessibilityChecks.enable()  
        .setSuppressingResultMatcher(matchingViews(withId(R.id.example\_view)));

1. **Espresso Test Recorder**

Salah satu jenis uji dari Espresso adalah menggunakan Test Recorder. Alat Espresso Test Recorder memungkinkan membuat pengujian UI untuk aplikasi tanpa menulis kode pengujian apa pun. Dengan merekam skenario pengujian, kita bisa merekam interaksi dengan sebuah perangkat dan menambahkan pernyataan untuk memverifikasi elemen UI dalam snapshot aplikasi tertentu. Espresso Test Recorder kemudian mengambil rekaman yang sudah disimpan dan secara otomatis menghasilkan pengujian UI yang sesuai dan bisa dijalankan untuk menguji aplikasi.

Espresso Test Recorder menulis pengujian berdasarkan [*framework Pengujian Espresso*](https://google.github.io/android-testing-support-library/docs/espresso/), sebuah API dalam [*Android Testing Support Library*](https://developer.android.com/topic/libraries/testing-support-library/index.html?hl=id). Espresso API mendorong untuk membuat pengujian UI yang ringkas dan dapat dilkan berdasarkan tindakan pengguna. Dengan menyatakan harapan, interaksi, dan pernyataan tanpa secara langsung mengakses aktivitas dan tampilan aplikasi yang mendasarinya, struktur ini mencegah gagalnya pengujian dan mengoptimalkan kecepatan pengujian.

Merekam Pengujian Espresso

Pengujian Espresso terdiri dari dua komponen utama: Interaksi UI dan pernyataan pada elemen View. Interaksi UI mencakup tindakan tap dan ketik yang mungkin digunakan pengguna untuk berinteraksi dengan aplikasi. Pernyataan memverifikasi keberadaan atau konten elemen visual pada layar. Misalnya, uji Espresso untuk [aplikasi pengujian Catatan](https://github.com/googlecodelabs/android-testing) mungkin mencakup interaksi UI untuk mengklik tombol dan menulis catatan baru tetapi menggunakan pernyataan untuk memverifikasi adanya tombol dan konten catatan.

Untuk menjalankan pengujian Espresso, gunakan jendela Project https://developer.android.com/studio/images/studio-icon.png?hl=id di sisi kiri Android Studio IDE:

1. Buka folder modul aplikasi yang diinginkan dan buka pengujian yang ingin jalankan. Lokasi pengujian tergantung pada lokasi akar [pengujian instrumentasi](https://developer.android.com/training/testing/unit-testing/instrumented-unit-tests.html?hl=id) dan nama paket aktivitas yang diluncurkan. Contoh berikut menunjukkan tempat penyimpanan pengujian untuk [aplikasi pengujian Catatan](https://github.com/googlecodelabs/android-testing):
   * Bila menggunakan [tampilan *Android*](https://developer.android.com/studio/projects/index.html?hl=id#ProjectFiles) di dalam jendela, buka *java > com.example.username.appname (androidTest)*.
   * Bila menggunakan tampilan *Project* di dalam jendela, buka *src > androidTest > java > com.example.username.appname* dalam folder modul.
2. Klik kanan pada pengujian dan klik Run ‘testName.’ Atau, bisa membuka file pengujian dan mengklik kanan metode atau kelas pengujian yang dihasilkan.
3. Pada jendela *Select Deployment Target*, pilih perangkat yang ingin jalankan pengujiannya. Bila perlu, [buat](https://developer.android.com/studio/run/managing-avds.html?hl=id) *[Android Virtual Device](https://developer.android.com/studio/run/managing-avds.html?hl=id)* [baru](https://developer.android.com/studio/run/managing-avds.html?hl=id). Klik OK.

Monitor kemajuan pengujian di jendela Run di bagian bawah IDE. Android Studio menjalankan versi lengkap project dan membuka tab dengan nama pengujian di jendela Run. Kita juga bisa memeriksa apakah pengujian berhasil atau gagal dalam tab ini, serta seberapa lama pengujian itu berjalan. Saat pengujian selesai, tab akan mencatat log "Tests ran to completion.