Testing

Example project

Testing

- bit će live coding session kasnije
- checkout repo: git@github.com:ikust/mvp_isa.git
- projekt se bazira na primjeru s prošlog predavanja

Robolectric

Unit testing library

Robolectric

Android Unit testing library

- mock UI related classes
- run unit tests with UI code from Android Sudio!
- baziran na JUnit-u
- http://robolectric.org/
- https://github.com/robolectric/robolectric

Test dependencies

Robolectric

- build.gradle na razini projekta
- build.gradle na razini aplikacije
- pogledati example projekt!

Test dependencies

Robolectric

- build.gradle na razini projekta
- build.gradle na razini modula aplikacije
- pogledati example projekt!

Postavljanje testova

Robolectric

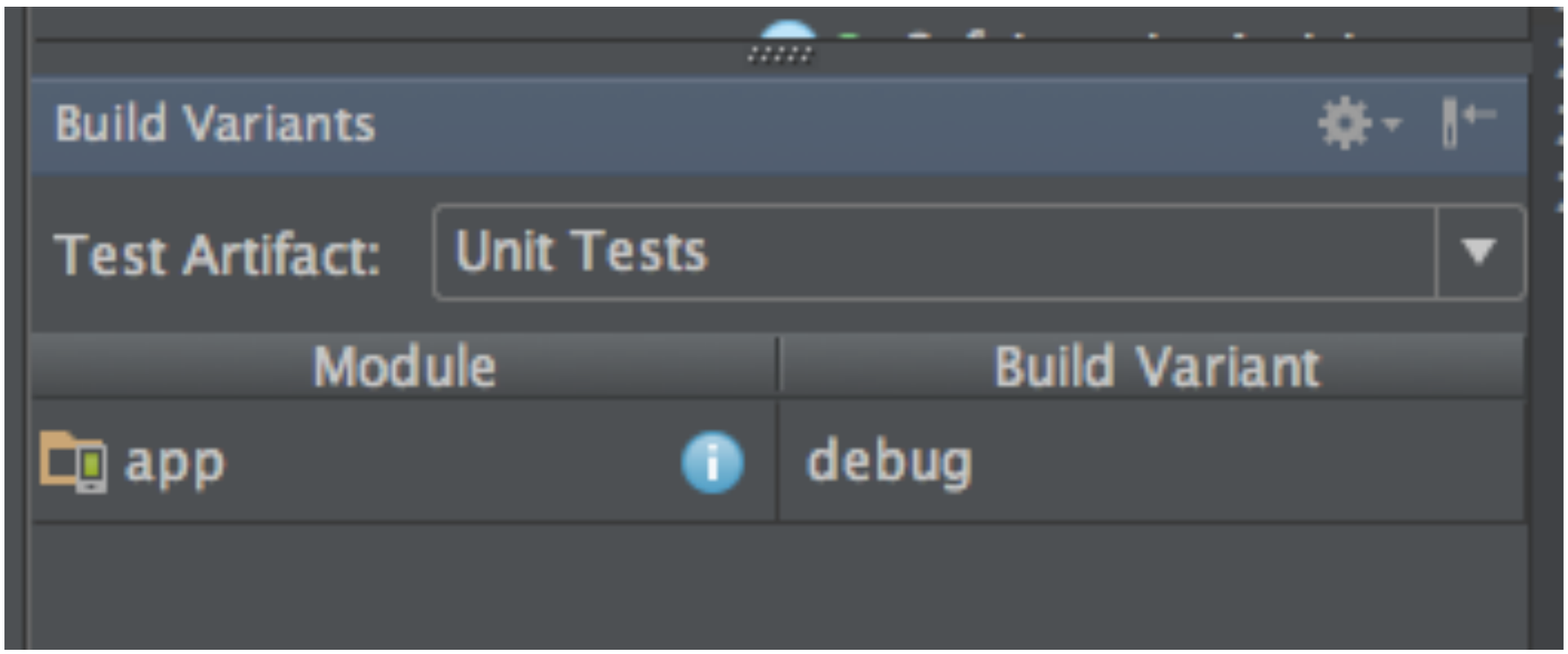
- testApplicationId - package name .apk s testovima
- mora biti različit od applicationId

```
android {
    …

    defaultConfig {
        applicationId "co.infinum.boatit"
        minSdkVersion 14
        targetSdkVersion 22
        versionCode 1
        versionName "1.0"
        testApplicationId "co.infinum.hpb.instrument.test"
    }
}
```

Postavljanje testova

Robolectric

- odabir Test Artifact-a:



Robolectric

Android Unit testing library

- unit testovi idu u “src/test” direktorij
- kod testiranja generira se dodani .apk s testovima koji instalira na telefon i pokreće
- ako nije odabran Unit tests artifact, testovi neće raditi

Resursi za testove

Robolectric

- testovi ne mogu sadržavat resurse (koji inače idu u “res” folder)
- dodat u build.gradle od aplikacije:

```
android.applicationVariants.all { variant ->
    flavorName = variant.name
    variant.mergeAssets.dependsOn generateBuildConfig
}
```

- dodat resurse u “src/test/resources”

SomeClass.class.getClassLoader().getResourceAsStream(“file.json”);

Threading

Robolectric

- testovi se izvršavaju u JEDNOM threadu (UI i networking)
- potrebno Robolectricu postaviti executore koji će izvršavati requestove u istom threadu

Test Application klasa

Robolectric

- extendati application klasu i dodati prefix Test npr.

  BoatitApplication -> TestBoatitApplication

- kod pokretanja testova koristiti će se application klasa s Test prefiksom
- inicijalizacija na razini aplikacije vezana uz testove (mock podaci itd.)

Pisanje testova

Robolectric

- dodati novu klasu, anotirati sa:

@Config(sdk = 21, application = TestHpbApplication.class)
@RunWith(RobolectricTestRunner.class)

- metoda anotirana sa @Before se izvodi prije svakog testa
- metoda anotirana sa @After se izvodi nakon svakog testa
- metode anotirane sa @Test predstavljaju testove

Pokreatanje testova

Robolectric

- iz komandne linije (iz root direktorija projekta)
- ./gradlew test
    - izvršava testove za sve build variante
- ./gradlew test{buildVariant}
- gradle test
    - u slučaju da se ne koristi gradle wrapper

MockWebServer

Part of OkHttp library

MockWebServer

Robolectric

- web server koji omogućava pisanje mock API response-ova
- dio OkHttp library-ja
- https://github.com/square/okhttp/tree/master/mockwebserver

Princip

Mock Web Server

- pokreće se lokalni HTTP server na mašini na kojoj se izvršavaju testovi
- inicijalizira se Retrofit -> URL mock servera
- skripanje mock responseova
- izvršavanje testova
- provjera requestova i promjena na UI-u

Robolectric

Coding session

Espresso

Instrumentation testing library

Espresso

Android instrumentation testing library

- testing UI
- https://developer.android.com/training/testing/ui-testing/espresso-testing.html
- https://code.google.com/p/android-test-kit/wiki/Espresso

Test dependencies

Espresso

- build.gradle na razini projekta
- pogledati example projekt!

Postavljanje testova

Espresso

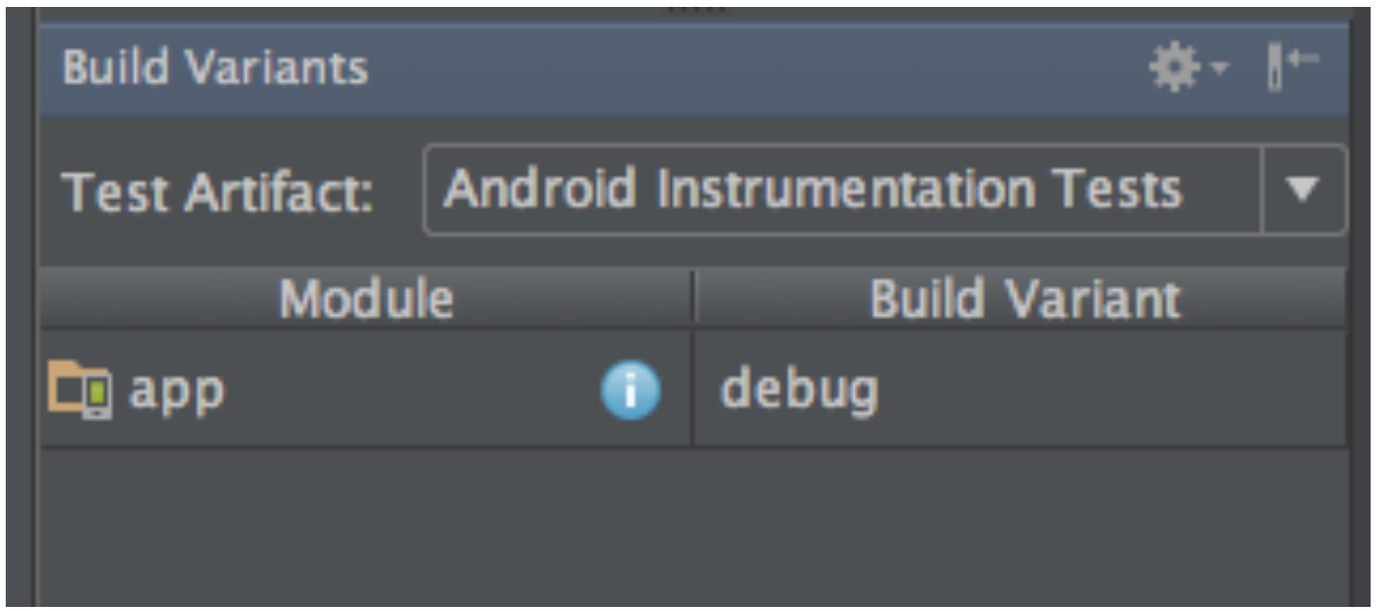
- dodat testInstrumentationRunner u build.gradle od app modula
- packagingOptions exclude

```
android {
    …
    defaultConfig {
        …
        testApplicationId "co.infinum.hpb.instrument.test"
        testInstrumentationRunner "android.support.test.runner.AndroidJUnitRunner"

    }


    packagingOptions {
        exclude 'LICENSE.txt'
    }
}
```

Postavljanje testova

Espresso

- odabir Test Artifact-a:



Postavljanje testova

Espresso

- isključiti animacije
- u Developer options postaviti na off:
    - Window animation scale
    - Transition animation scale
    - Animator duration scale

Robolectric

Android Unit testing library

- intsrument testovi idu u “src/androidTest” direktorij
- ako nije odabran Android instrumentation tests artifact, testovi neće raditi

Threading

Espresso

- testovi se izvršavaju na uređaju
- networking pozive je potrebno raditi u background threadu (kao i u aplikaciji)

Resursi za testove

Epresso

- testovi ne mogu sadržavat resurse (koji inače idu u “res” folder)
- isto kao i kod Robolectrica
- dodat resurse u “src/androidTest/resources”

```
SomeClass.class.getClassLoader().getResourceAsStream(“file.json”);
```

Pisanje testova

Robolectric

- dodati novu klasu koja extenda ActivityInstrumentationTestCase2
- metoda setUp() se izvršava prije svakog testa
- metoda tearDown() se izvršava nakon svakog testa
- metode anotirane sa @Test predstavljaju testove

Pisanje testova

Robolectric

- u setUp() metodi inicijalizirati instrumentation i activity:

```
@Override
protected void setUp() throws Exception {
    super.setUp();
    injectInstrumentation(InstrumentationRegistry.getInstrumentation());
    activity = getActivity();
}
```

Pokreatanje testova

Espresso

- iz komandne linije (iz root direktorija projekta)
- ./gradlew connectedCheck
- ./gradlew cC
- ./gradlew connectedAndroidTest
    - izvršava testove za sve build variante
- ./gradlew connectedAndroidTest{buildVariant}

Espresso

Coding session

Zadaća

Testing

Nadograditi aplikaciju iz prošle zadaće:
 - dodati upboat / downboat opcije

Napisati barem jedan unit test (robolectric) i jedan instrumentation test (espresso).