# Day 4 Unit Testing in Android

This session will focus on teaching attendees the basics of TDD in Android through simple example tests.

# Session Objectives / Key Learning Points

# By the end of the session students should –

# Understand how to setup your android project in IntelliJ along with Robolectric as the unit-testing framework.

# Understand how Robolectric works.

# Be comfortable writing simple tests using Robolectric.

# Understand testing techniques used for Android.

# Understand basic difference between JVM and DVM.

# 

# Session Overview

Assumption: students already have the necessary libraries to setup Robolectric.

|  |  |  |
| --- | --- | --- |
| Activity | Time | Elapsed Time |
| Clone the project repository and setup Robolectric to work with the project in IntelliJ | 00:10 | 00:10 |
| Overview of testing in Android development | 00:05 | 00:15 |
| Explain JVM and DVM | 00:05 | 00:20 |
| Testing techniques in Android development | 00:10 | 00:30 |
| Explain why Robolectric is better solution | 00:05 | 00:35 |
| Assignment 1 (Hands on). Write tests in the cloned project. | 00:45 | 01:20 |
| Questions and Answers regarding the tests implemented as part of the assignment | 00:15 | 01:35 |
| Explain working of Robolectric | 00.15 | 01:50 |
| Explain What to test and what not to. | 00:03 | 01:53 |
| Understand good practices while TDD-ing in Android. | 00:02 | 01:55 |
| Question and Answers | 00:05 | 02:00 |

Session Notes

Why is TDD ignored in Android development?

* Unavailability of Robust framework.
* No direct way of running the tests on JVM.

Significance and difference between JVM and DVM (in context of Android).

* JVM executes on .class files while DVM uses .dex files.
* Dexing converts .class files to .dex and hence a time consuming process.

Approaches followed for testing in Android (generally).

* Try to separate UI from the business logic. Business logic becomes your library of tested POJOs.
* The UI part has lot of navigation logic and view validations. Android Instrumentation framework provides a way to test this along with the business logic.
* Instrumentation framework runs the tests on DVM. Eats up a lot of time in Dexing and running the app on emulator every time you run the tests.

A better solution – Robolectric.

* Runs on JVM. No Dexing is required.
* Power to test both POJOs as well as UI.
* More test coverage at unit testing level.
* Helps you to follow a quick iterative cycle of TDD (Red, Green, Refactor).

Robolectric working.

* View and Resource loading:

It parses all the resources and views in trees and maintains the state in sync with your running application.

* Shadow Objects.

These are proxies to the Android classes. So if you do button.getText(), the shadow button object will be proxied and it will have its own behavior and state. This way you can assert on things which are private to the Android classes or whose behavior cant be extended or mocked easily.

For some detailed explanation, refer Robolectric website or <http://dasherize.blogspot.in/p/robolectric.html>. For the version of Robolectric checked-in in the Github repository, blog’s explanation holds true. In their recent release, they have almost removed all shadow classes.

Dos and Don’ts.

* Test functionality, business logic and basic view validations.
* Do not test styling details and much of the xml attributes as these are prone to change during the project development.

Assignment 1

Problem:

The app is in broken structure. Some tests are already being provided in the project. Some tests are empty. A student has to complete those incomplete tests and simultaneously write code in the app in order to make the tests green.

The app calculates addition or factorial of input(s) based on the selection of operation. The result has to be displayed in the new activity. If addition is selected, user should see two operands, while in case of factorial operation user should see only one operand. If the user clicks the result button without providing any inputs, user should be shown an alert dialog with appropriate error message.

Solution:

The incomplete tests have to be implemented, based on the name of the test methods. The failing tests have to fixed.

For testing the result on the new Activity, passing of extras and testing the Alert Dialog, Shadow classes have to be used.