Unit Testing

Mockito vs MockK

<https://android.jlelse.eu/mockk-io-our-new-mocking-framework-d82e3d104cb7>

With Mockito this is how you mocked an object:

import static org.mockito.Mockito.mock;MyClass mockedClass = mock(MyClass.class);

With Mockk it looks like this:

import io.mockk.mockk val mockedClass = mockk<MyClass>()

<https://medium.com/@prashantspol/mockk-better-to-way-to-mock-in-kotlin-than-mockito-1b659c5232ec>

<https://github.com/powermock/powermock/wiki/MockPrivate>

<https://github.com/powermock/powermock>

private val sharedPrefMock = *mockkClass*(SharedPreferencesManager::class, relaxed = true)

just Runs should be used in case return value is Unit.

**every { mock1.callReturningUnit(5) } just Runs**

**Annotations:**

**Spy**

**Spies allow to mix mocks and real objects.**

verify {} 🡪 checks to see if this particular was called on the mock object.

every {} 🡪 mock response function

private method UT:

@Test  
fun `test render pdf`() {  
 launchFragment()  
 fragment.currentPageNumber = 2  
 val mockObj = *spyk*<ContentPresentationFragment>(recordPrivateCalls = true)  
 *every* **{** mockObj["renderPdf"]() **}** returns "javascript:queueRenderPage(2)"  
 fragment.currentPageNumber = 0  
 *every* **{** mockObj["renderPdf"]() **}** returns "javascript:addPdf(${fragment.contentId},${fragment.filePresentatationUrl},${fragment.currentPageNumber})"  
}

@Test  
 fun `test handleCamera`() {  
 launchFragment()  
 fragment.isCameraShared = false  
 val mockObj = *spyk*<ContentPresentationFragment>(recordPrivateCalls = true)  
 *every* **{** mockObj["handleCamera"]() **}** returns "javascript:stopLocalStream();"  
// assertEquals(fragment.webView.url, "javascript:stopLocalStream();")  
 Handler().postDelayed(**{** assertEquals(fragment.webView.*url*, "javascript:stopLocalStream();")  
 **}**,2000)  
  
 fragment.isCameraShared = true  
 *every* **{** mockObj["handleCamera"] () **}** returns "javascript:addCameraStream(true);"  
  
 Handler().postDelayed(**{** assertEquals(fragment.webView.*url*, "javascript:addCameraStream(true);")  
 **}**,2000)  
 }

Test for activity as BaseActivity

val spy = *spyk*(thankYouFragment!!)  
 val activity = BaseActivity()  
 *every* **{** spy.*activity* **}** returns activity  
 spy.moveToLastFragment()

// example

@Test  
fun `test onRecentMeetingClick for SearchResult`() {  
  
 val mockContext: Context = *mockkClass*(Context::class)  
 val oldDao = ApplicationDao.get(mockContext)  
 oldDao.invalidate()  
 ApplicationDao.setInstance(null)  
  
 val context: Context = ApplicationProvider.getApplicationContext()  
 val applicationDao = ApplicationDao.get(context)  
  
 val searchResult = SearchResult()  
 searchResult.*hubConfId* = 12345  
 searchResult.*conferenceId* = 1234  
 searchResult.*useHtml5* = true  
 searchResult.*furl* = "https://pgi.globalmeet.com/gauravsingh"  
  
 startKoinApp()  
 scenario = launchFragment()  
 val spy = *spyk*(fragment!!)  
 val activity = BaseActivity()  
 *every* **{** spy.*activity* **}** returns activity  
 spy.onRecentMeetingClick(searchResult)  
}