Android/Kotlin – Testes Unitários (Incluindo Dependências Simuladas)

Por padrão, o Plug-in do Android para Gradle executa seus testes de unidade locais em uma versão modificada da biblioteca `android.jar`, que não contém nenhum código real. Em vez disso, as chamadas de método para classes do Android feitas no teste de unidade geram uma exceção. Isso garante que você teste apenas seu código e não dependa de nenhum comportamento específico da plataforma Android (que você não tenha criado ou simulado explicitamente).

Simular dependências do Android

Se você tem dependências mínimas do Android e precisa testar interações específicas entre um componente e a dependência dele no app, use um framework de simulação para criar stubs de dependências externas no código. Dessa forma, você terá facilidade para testar se o componente interage com a dependência da maneira esperada. Substituindo dependências do Android por objetos simulados, você pode isolar seu teste de unidade do restante do sistema Android enquanto verifica se métodos corretos são chamados nessas dependências.

Para adicionar um objeto simulado ao teste unitário usando esse framework, siga este modelo de programação:

1.Inclua a dependência da biblioteca Mockito no arquivo **build.gradle**.

2.No início da definição da classe de teste de unidade, adicione a anotação **@RunWith(MockitoJUnitRunner.class)**. Essa anotação pede que o executor de testes do Mockito verifique se o uso do framework está correto e simplifica a inicialização dos objetos simulados.

3.Para criar um objeto simulado para uma dependência do Android, adicione a anotação **@Mock** antes da declaração do campo.

4.Para criar stubs do comportamento da dependência, especifique uma condição e retorne um valor quando a condição for atendida por meio dos métodos **when()** e **thenReturn()**.

O exemplo a seguir mostra como criar um teste unitário que usa um objeto **Context**_ simulado.

import android.content.Context

```
import com.google.common.truth.Truth.assertThat
import org.junit.Test
import org.junit.runner.RunWith
import org.mockito.Mock
import org.mockito.Mockito.`when`
```

```
import org.mockito.junit.MockitoJUnitRunner
private const val FAKE_STRING = "HELLO WORLD"
@RunWith(MockitoJUnitRunner::class)
class UnitTestSample {
    @Mock
    private lateinit var mockContext: Context
    @Test
    fun readStringFromContext_LocalizedString() {
        // Given a mocked Context injected into the object under test...
        `when`(mockContext.getString(R.string.hello_word))
                .thenReturn(FAKE_STRING)
        val myObjectUnderTest = ClassUnderTest(mockContext)
        // ...when the string is returned from the object under test...
        val result: String = myObjectUnderTest.getHelloWorldString()
        // ...then the result should be the expected one.
        assertThat(result, `is`(FAKE_STRING))
    }
}
```