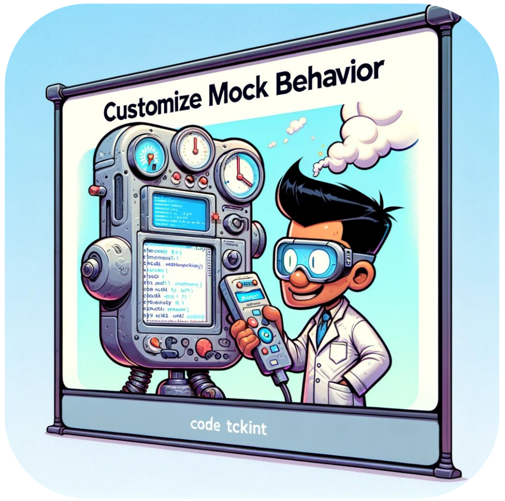
Customize Mock Behavior with MockSettings



Understanding MockSettings

Creating a mock with default settings

```java
import org.mockito.Mockito;
import org.mockito.MockSettings;

// Creating a mock with default MockSettings
MockSettings mockSettings = Mockito.withSettings();
MyClass myMock = Mockito.mock(MyClass.class, mockSettings);
```

Using serializable() in MockSettings

```java
import org.mockito.Mockito;
import org.mockito.MockSettings;

// Making a mock serializable using MockSettings
MockSettings serializableSettings = Mockito.withSettings().serializable();
MyClass serializableMock = Mockito.mock(MyClass.class, serializableSettings);
```

Setting defaultAnswer()

```java
import org.mockito.Mockito;
import org.mockito.stubbing.Answer;
import org.mockito.MockSettings;

// Setting a default answer for unstubbed calls
Answer<Object> defaultAnswer = invocation -> "Default Response";
MockSettings defaultAnswerSettings =
Mockito.withSettings().defaultAnswer(defaultAnswer);
MyClass defaultAnswerMock = Mockito.mock(MyClass.class, defaultAnswerSettings);
```

Adding invocationListeners()

```java
import org.mockito.Mockito;
import org.mockito.listeners.InvocationListener;
import org.mockito.listeners.MethodInvocationReport;
import org.mockito.MockSettings;

// Adding an invocation listener to MockSettings
InvocationListener listener = new InvocationListener() {
    @Override
    public void reportInvocation(MethodInvocationReport methodInvocationReport) {
        System.out.println("Method called: "
                           +
                         methodInvocationReport.getInvocation().getMethod().getName());
    }
};

MockSettings listenerSettings = Mockito.withSettings().invocationListeners(listener);
MyClass listenerMock = Mockito.mock(MyClass.class, listenerSettings);
```

Advanced Configuration with MockSettings

Mock setup with consecutive calls

```java
// Creating a mock with consecutive behavior for method calls
Foo mockFoo = mock(Foo.class, withSettings().name("ConsecutiveBehaviorMock"));

when(mockFoo.bar(anyString()))
    .thenReturn("Hello")    // First call returns "Hello"
    .thenThrow(new RuntimeException("Unexpected call")) // Second call throws an exception
    .thenReturn("World");   // Third call returns "World"
```

Customizing Verification with Verification Modes

```java
// A mock for a class that we want to verify with custom settings
Foo mockFoo = mock(Foo.class);

// ...some interactions with the mock...

// Custom verification with specific timeout and call count
verify(mockFoo, timeout(100).times(1)).bar(eq("expectedParam"));
```

Extra Interfaces and Mocking Details

```
// Mock creation with extra interfaces
Foo mockFoo = mock(Foo.class, withSettings()
    .extraInterfaces(AnotherInterface.class));

// Casting to use the extra interface
AnotherInterface anotherBehavior = (AnotherInterface) mockFoo;

MockingDetails details = mockingDetails(mockFoo);
if (details.isMock()) {
    // Detailed inspection here
}
```

MockSettings: Best Practices and Pitfalls

1. Use Sparingly
2. Readable Tests
3. Default Answers

1. Overcomplicating Mocks
2. Forgetting Maintenance
3. Ignoring Performance