Settings Fake Object's Behavior

Dror Helper

@dhelper http://helpercode.com

Module Overview

Fake object's default behavior

- And how to change it

Setting fake behavior during tests

Best practices and pitfalls to avoid

Recap: Mocking Frameworks

Create fake objects

Set fake's behavior

Verify calls were made

GMock Default Return Values

Void method → do nothing

Return "default value"

- bool → false
- Numeric → 0
- ptr → NULL

C++ 11 – return instance if have default c'tor

Can be changed according to need

Why We Care About Default Return Value

Reduce test code

Increase readability

Future-proof tests

Setting the Default Return Value

```
DefaultValue<T>::Set(value);
DefaultValue<T>::SetFactory(&makeT);
DefaultValue<T>::Clear();


ON_CALL(mock_object, method).WillByDefault(…);


ON_CALL(fakeFoo, MyMethod(_)).WillByDefault(Return(-1));
ON_CALL(fakeFoo, MyMethod(0)).WillByDefault(Return(0));
```

Setting Test Behavior

```
EXPECT_CALL(fakeFoo, MyMethod(“abc”)).WillOnce(...);
EXPECT_CALL(fakeFoo, MyMethod(_)).WillOnce(...);


EXPECT_CALL(const(fakeFoo), MyMethod(“abc”)).WillOnce(...);


EXPECT_CALL(fakeFoo, MyMethod(“abc”)).WillRepeatedly(...);
```

Returning a Value

```
using namespace testing;


EXPECT_CALL(fakeFoo, MyMethod()).WillOnce(Return(-1));


EXPECT_CALL(fakeFoo, MyMethodReturningRef()).WillOnce(ReturnRef(bar1));


// Values are evaluated only once
int n = 0;
EXPECT_CALL(fakeFoo, MyMethod()).WillRepeatedly(Return(n++));
EXPECT_CALL(myFake, SomeMethod(true, _))
    .WillOnce(SetArgPointee<1>(10))


EXPECT_CALL(myFake, SomeMethodReturningBool(true, _))
    .WillOnce(DoAll(SetArgPointee<1>(10), Return(true))
```

Side Effects

Some methods use parameters to return values

In case you need to specify return value use *DoAll*

Use SetArrayArgument<> to set array parameter

```
EXPECT_CALL(myFake, SomeMethod())
             .WillOnce(Throw(exception);
```

Throwing Exceptions

Used to mimic errors

Useful for testing corner cases

Must be a copyable value

Invoking a Function

```
EXPECT_CALL(myFake, SomeMethod())
             .WillOnce(InvokeWithoutArgs(OtherMethod));
```

```
EXPECT_CALL(myFake, SomeMethod())
             .WillOnce(InvokeWithoutArgs(IgnoreResult(OtherMethod)));
```

```
EXPECT_CALL(myFake, SomeMethod())
             .WillOnce(WithArgs<0, 2, 3>(OtherMethod));
```

```
EXPECT_CALL(myFake, SomeMethod()).WillOnce(InvokeArgument<1>(5));
```

Composite Actions

DoAll(a1, a2, ..., an)

IgnoreResult(*action*)

WithArg<N>(*action*)

WithArgs(N1, N2, ..., Nk(*action*)

WithoutArgs(*action*)

```
ACTION(Sum){ return arg0 + arg1; }

ACTION_P(Plus, n){ return arg0 + n }

ACTION_PK(MyAction, p1, …, pk){ … }
```

Defining Actions

A quick way to create action for *Invoke*

Defined outside of methods/tests

Can use *arg0..argn*

Selecting Between Behaviors

```
EXPECT_CALL(fake, MyMethod(100)).WillOnce(Return(true));
EXPECT_CALL(fake, MyMethod(200)).WillOnce(Return(false));
EXPECT_CALL(fake, MyMethod(300)).WillOnce(Throw(exception));

EXPECT_CALL(fake, MyMethod(_)).WillRepeatedly(Return(true));
EXPECT_CALL(fake, MyMethod(100)).WillRepeatedly(Return(false));
```

Faking Behavior Pitfalls

Mocked Test

Copy of existing system

Implementation-bound

Confusing (multiple) behaviors

Best Practices

Keep fake behavior as simple as possible

Test structure: Arrange Act Assert

Avoid mocking fine-grained / chatty interfaces

Don’t mock everything

Summary

GMock default behavior

- How to change it

Setting behavior on fakes/mocks

- WillOnce/WillRepeatedly
- Return Value
- Throw
- Invoke

Multiple behaviors

Pitfalls and best practices