

Using Arguments and Matchers



Dror Helper

@dhelper <http://helpercode.com>



Module Overview



Setting behavior on specific arguments

Using matchers

- Overload based expectations
- GMock built-in matchers

Creating custom matchers

More GTest assertions using matchers



```
ON_CALL(myMock, SomeMethod(_)).WillByDefault(Return(42));
```

```
EXPECT_CALL(myMock, SomeMethod(42))  
    .Times(AtLeast(1))  
    .WillRepeatedly(Throw(meaningException));
```

Recap: Controlling Mock Behavior

Use **ON_CALL** to set default behavior

Use **EXPECT_CALL** to set behavior and expectation in test

Both can use **matchers**



Using Wildcards

```
EXPECT_CALL(fake, MyMethod(_))
```

```
EXPECT_CALL(fake, MyMethod(A<int>()))
```

```
EXPECT_CALL(fake, MyMethod(An<int>()))
```



Generic Comparisons

```
EXPECT_CALL(fake, Count(Eq(100)) // arg == 100
```

```
EXPECT_CALL(fake, Count(Ne(100)) // arg != 100
```

```
EXPECT_CALL(fake, Count(Gt(100)) // arg > 100
```

```
EXPECT_CALL(fake, Count(Lt(100)) // arg < 100
```

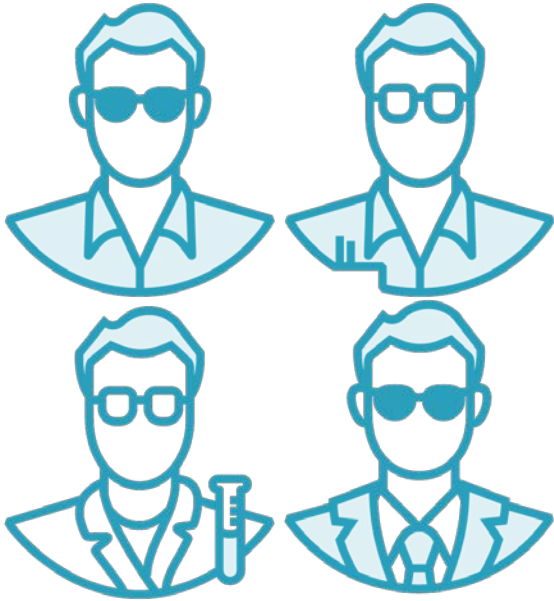
```
EXPECT_CALL(fake, Print(IsNull()) // arg == NULL/nullptr
```

```
EXPECT_CALL(fake, Print(NotNull()) // arg != NULL/nullptr
```

```
EXPECT_CALL(fake, Print(Ref(str))
```



Why Use Matchers?



Different
behaviors for
the same
method



Create complex
workflows



Improve GTest
Assertions



Verify the exact
expectation

Type Based Matchers

```
EXPECT_CALL(fake, MyMethod(An<int>()))
```

```
EXPECT_CALL(fake, MyMethod(TypedEq<int>(50)))
```

```
EXPECT_CALL(fake, MyMethod(Matcher<int>(Gt(50))))
```



Using Matchers to Perform Different Actions

```
EXPECT_CALL(fake, MyMethod(_)).WillRepeatedly(Return(1));
```

```
EXPECT_CALL(fake, MyMethod(Gt(10))).WillRepeatedly(Return(5));
```

```
EXPECT_CALL(fake, MyMethod(Gt(20))).WillRepeatedly(Return(10));
```

```
EXPECT_CALL(fake, MyMethod(A<char>())).WillRepeatedly(Return(200));
```

MyMethod('a') → 200

MyMethod(25) → 10

MyMethod(10) → 1



String Matchers



Combining Matchers

`AllOf(m1, m2, ...)`

`AnyOf(m1, m2, ...)`

`Not(m)`

`EXPECT_CALL(fake, Func(AllOf(NotNull()), Not(StrEq(""))), 5))`

`MatcherCast<T>(m)`

`MatcherSafeCast<T>(m)`



```
Field(&class::field, m)
```

```
Property(&class::property, m)
```

```
Key(v/m) // EXPECT_CALL(myMap, Contains(Key(42)))
```

```
Pair(m1, m2)
```

Member Matchers

Used to check fields, methods of arguments passed to fake method





Do not try to re-create the system
under test using mocks and matchers



Assertions with Matchers

```
ASSERT_THAT(result, AllOf(NotNull(), StrNe("")));
```

```
EXPECT_THAT(result, AnyOf(Gt(100), Le(-100)));
```



Container Matchers

Whole matchers

ContainerEq(other)

IsEmpty()

Sizes(m)

Contains(e)

Each(e)

Individual items matchers

ElementsAre(e0, e1, ...)

ElementsAreArray({})

Pointwise(m, container)

UnorderedElementsAre(...)

WhenSorted(m)

WhenSortedBy(comparator, m)



```
EXPECT_CALL(fake, Method(a, b)).With(Eq())
```

```
EXPECT_CALL(fake, Method(a, b, c)).With(AllArgs(Eq()))
```

```
EXPECT_CALL(fake, Method(a, b, c)).With(Args<1, 3>(Eq()))
```

Multiargument Matchers

Defined using *With*

Matches a tuple (x,y) using *Eq*, *Ge*, *Gt*, *Le*, *Lt*, *Ne*

Can select all arguments (default) or select a subset



Additional Matchers

Floating point

- DoubleEq, FloatEq
- DoubleNear, FloatNear
- NanSensitive

Pointer

- Pointee(m)
- WhenDynamicCastTo<T>(m)

Result of a function

- ResultOf(f, m)

Creating New Matchers

Truly(predicate)

MATCHER macro

MatcherInterface



Defining New Matchers

```
MATCHER(name, description){. . .}
```

```
MATCHER(IsEven, ""){return arg % 2 == 0;}
```

```
MATCHER_P(name, param_name, description){. . .}
```

```
MATCHER_P(IsDividable, value, ""){return arg % value == 0;}
```

```
MATCHER_P2(InCloseRange, low, hi, ""){  
    return low <= arg && arg <= high;  
}
```



Writing New Monomorphic Matchers

```
template <typename T>
class MatcherInterface {
public:
    virtual ~MatcherInterface();

    virtual bool MatchAndExplain(T x, MatchResultListener* listener) const = 0;

    virtual void DescribeTo(::std::ostream* os) const = 0;

    virtual void DescribeNegationTo(::std::ostream* os) const;
};
```



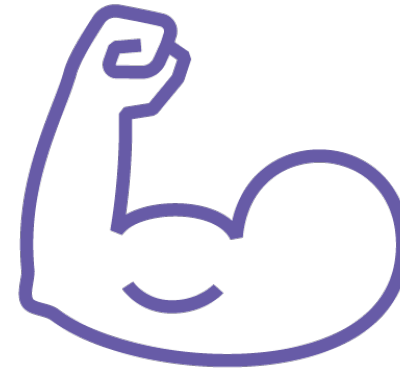
Matchers Best Practices



When in doubt –
do not use



When verifying
calls, use
matchers for
better check



Use to improve
assertions



Keep it simple

Summary



Using Matchers

- ON_CALL
- EXPECT_CALL
- ASSERT_THAT/EXPECT_THAT

Built-in matchers

Creating new matchers

- Truly
- MATCHER macros
- MatcherInterface