Instituto Superior de Engenharia de Lisboa

Licenciatura em Engenharia Informática e de Computadores

2019

A biblioteca Mocky implementa geração dinâmica de objectos mock (<https://en.wikipedia.org/wiki/Mock_object>). Os objectos mock simulam o comportamento de objectos de tipos reais (classes ou interfaces) e são úteis na fase de testes quando ainda não existem as implementações destes tipos.

A biblioteca Mocky pode ser usada de acordo com o exemplo seguinte para uma interface ICalculator resultando no comportamento apresentado:

Mocker mock = new Mocker(typeof(ICalculator));

mock.When("Add").With(5, 7).Return(12);

mock.When("Add").With(3, 4).Return(7);

mock.When("Mul").With(3, 3).Return(9);

ICalculator calc = (ICalculator) mock.Create();

Assert.AreEqual(calc.Add(5, 7), 12);

Assert.AreEqual(calc.Add(3, 4), 7);

Assert.AreEqual(calc.Add(4, 1), 5); // FAIL !!!! returns 0 rather than 5

Assert.AreEqual(calc.Mul(3, 3), 9);

Assert.AreEqual(calc.Sub(2, 1), 1); // NotImplementedException

De notar que quando é chamado calc.Add(4, 1) é retornado 0 porque não foi especificado em mock qual o resultado a dar para aqueles argumentos. **Logo é retornado o valor default do tipo de retorno desse método**.

Por sua vez, para os métodos que não foi especificado qualquer comportamento (e.g. Sub e Div) é dada a excepção NotImplementedException.

As classes da biblioteca Mocky têm a seguinte organização:

[Graphical user interface, text

Description automatically generated](https://github.com/isel-leic-ave/wing/blob/master/assets/Mocky.jpg)

O método Create de Mocker para a interface ICalculator lança a excepção NotImplementedException na chamada a cada um dos seus métodos Add, Sub, Mul e Div. Exemplo:

Mocker mock = new Mocker(typeof(ICalculator));

ICalculator calc = (ICalculator)mock.Create();

Console.WriteLine(calc.ToString()); // > OK ToString() de Object

calc.Add(11, 8); // > throws NotImplementedException

O método Create de Mocker funciona correctamente para a interface IHttpRequest da biblioteca Request lançando a excepção NotImplementedException na chamada aos métodos GetBody e Dispose.

Note que GetMethods da API de reflexão para uma interface só retorna os métodos da própria interface. Neste caso percorre a hierarquia de interfaces para obter todos os métodos.

A classe MockMethod  verifica o seguinte caso de teste:

MockMethod add = new MockMethod(typeof(ICalculator), "Add");

add.With(5, 3).Return(8);

add.With(2, 7).Return(9);

Assert.AreEqual(add.Call(5, 3), 8);

Assert.AreEqual(add.Call(2, 7), 9);

Assert.AreEqual(add.Call(4, 8), 0);

Por exemplo o TestLoadSearchOportoOnMock substitui:

using (WeatherWebApi api = new WeatherWebApi())

por:

Mocker mocker = new Mocker(typeof(IWeatherWebApi));

mocker

.When("Search")

.With("oporto")

.Return(new LocationInfo[] {

null,

null,

null,

null,

null,

new LocationInfo("Cuba", "", 0, 0)});

using (IWeatherWebApi api = (IWeatherWebApi) mocker.Create()){ ...

O teste TestLoadSearchOportoOnRequestMock  usa o WeatherWebApi sobre um mock de IHttpRequest da seguinte forma:

Mocker mocker = new Mocker(typeof(IHttpRequest));

mocker.When(...)...;

IHttpRequest req = (IHttpRequest)mocker.Create();

using (IWeatherWebApi api = new WeatherWebApi(req)) {

...

}

A API da biblioteca Mocky oferece o método Then(...) que permite especificar o comportamento de um método através de um delegate conforme os exemplos seguintes:

Mocker mockCalc ...

...

mockCalc.When("Add").Then<int, int, int>((a,b) => a + b);

...

Mocker mockReq = ...

...

mockReq.When("GetBody").Then<string, string>(url => ...);

mockReq.When("Dispose").Then(() => {/\* do nothing \*/});

Note que o método Then pode ter várias sobrecargas consoante o tipo de delegates suportados.

A execução do método Then  verifica se o tipo de delegate adicionado é compatível com o descritor do método especificado. Em caso de incompatibilidade lança excepção. A listagem seguinte apresenta alguns exemplos de utilização incongruentes que dão excepção na execução do Then.

mockCalc.When("Add").Then<int, int, double>((a, b) => a + b);

...

mockReq.When("Dispose").Then<string>((arg) => {/\* do nothing \*/});

A biblioteca Mocky suporta overload de métodos. Ou seja, suporta a adição do mesmo método através do When diversas vezes quando este se refere a tipos de parâmetros diferentes.

Por exemplo, a interface ICalculator com um novo método Add(int, int,int): int:

m.When("Add").then((a, b) => a + b);

m.When("Add").then((a, b, c) => a + b + c);

A utilização através do método With também surpota overload.