Mocking interfaces in Go with GoMock

Author
Gurleen Sethi on 18 June 2022

Getting Started

In this article we are going to learn how to use GoMock to generate mock implementations of `interface`s in Go.

Installing GoMock

GoMock works as a command line application to generate mocked source code, but you also have to install it in your application to use the generated mock code.

Installing the cli tool:

```
go install github.com/golang/mock/mockgen@v1.6.0
```

Installing gomock in your go project:

```
go get github.com/golang/mock/gomock
```

Generating mock code

Let's say we have the following code in your `main.go` file.

```go
package main

// 👇 an interface acting as API Client
type ApiClient interface {
    GetData() string
}

// 👇 a function using the ApiClient interface
func Process(client ApiClient) int {
    data := client.GetData()
    return len(data)
}

func main() {

}
```

In the above piece of code we have an interface `ApiClient` that is being used by the `Process` function. Lets say we want to test the `Process` function and we require a mock implementation of `ApiClient` when doing so.

To generate mock implementation of `ApiClient` run the following code in the project root:

```
mockgen -source=main.go -destination=mocks/main.go
```

This will generate a mock implementation of the interfaace `ApiClient` and write the code into `mocks/main.go` file.

We don't really care about the generated code but here is the generated code to satisfy your curiosity (don't bother to understand it, just glance over it).

```go
// Code generated by MockGen. DO NOT EDIT.
// Source: main.go

// Package mock_main is a generated GoMock package.
package mock_main

import (
    reflect "reflect"

    gomock "github.com/golang/mock/gomock"
)

// MockApiClient is a mock of ApiClient interface.
type MockApiClient struct {
    ctrl     *gomock.Controller
    recorder *MockApiClientMockRecorder
}

// MockApiClientMockRecorder is the mock recorder for MockApiClient.
type MockApiClientMockRecorder struct {
    mock *MockApiClient
}

// NewMockApiClient creates a new mock instance.
func NewMockApiClient(ctrl *gomock.Controller) *MockApiClient {
    mock := &MockApiClient{ctrl: ctrl}
    mock.recorder = &MockApiClientMockRecorder{mock}
    return mock
}

// EXPECT returns an object that allows the caller to indicate expected use.
func (m *MockApiClient) EXPECT() *MockApiClientMockRecorder {
    return m.recorder
```

```
}

// GetData mocks base method.
func (m *MockApiClient) GetData() string {
    m.ctrl.T.Helper()
    ret := m.ctrl.Call(m, "GetData")
    ret0, _ := ret[0].(string)
    return ret0
}

// GetData indicates an expected call of GetData.
func (mr *MockApiClientMockRecorder) GetData() *gomock.Call {
    mr.mock.ctrl.T.Helper()
    return mr.mock.ctrl.RecordCallWithMethodType(mr.mock, "GetData", reflect.TypeOf((*MockApiC
}
```

Look closely we have an struct called `MockApiClient` that has all the functions of `ApiClient` (`GetData` in this case). Additionally we have a function called `NewMockApiClient` that we can use to get an instance of `MockApiClient` in our tests.

Using the mock code

Create a new file called `mock_test.go` and add the following code:

```
package main

import (
    "github.com/golang/mock/gomock"
    mock_main "github.com/gurleensethi/playground/mocks"
    "testing"
)

func TestProcess(t *testing.T) {
    ctrl := gomock.NewController(t)

    // 👇 create new mock client
    mockApiClient := mock_main.NewMockApiClient(ctrl)

    // 👇 configure our mock `GetData` function to return mock data
    mockApiClient.EXPECT().GetData().Return("Hello World")

    length := Process(mockApiClient)

    if length != 11 {
        t.Fatalf("want: %d, got: %d\n", 11, length)
    }
}
```

We use the `NewMockApiClient` function to get an instance that we can use wherever `ApiClient` interface is expected.

```
mockApiClient := mock_main.NewMockApiClient(ctrl)
```

Pay attention to how we configure the `GetData` function on the mock client to return the data that we want.

```
mockApiClient.EXPECT().GetData().Return("Hello World")
```

Check out GoMock documentation it has a lot more options to configure mocks, it has a reflection mode which allows you to generate mocks of interfaces from packages that you have not written yourself and much more.

Thank your for reading this article 🙏 hopefully you learned something new.