INVERS

Let the compiler work for you –

how to automatically generate mock instances using Swift macros

Manuel Weiel

INVERS

Mocking

Mock Framework

- Simple DSL
- Easy to write tests
- Lightweight boilerplate code
- Similar to mockk for Android (https://mockk.io)

➡ We found SwiftMock (https://github.com/mflint/SwiftMock)
  - Currently limited: no async, throwing or fuzzy matching support
  - We extended it!

Mock Framework

```
protocol DemoRepository {
    func getPerson() async throws → Person

    func postAge(personId: Int, age: Int)
}
```

Mock Framework

```
let demoRepositoryMock = DemoRepositoryMock.create()

let person = Person(id: 1, name: "Manuel")
await demoRepositoryMock.expectAsync { mock in
  try await mock.getPerson()
}
.returning(person) // .throwing(MyError.someError)

let result = try await demoRepositoryMock.getPerson()
XCTAssertEqual(person, result)

await demoRepositoryMock.verifyAsync()
```

Mock Framework

```
class DemoRepositoryMock: Mock<DemoRepository>, DemoRepository {
    func getPerson() async throws → Person {
        try await acceptAsync()
    }

    func postAge(personId: Int, age: Int) {
        accept(args: [personId, age])
    }
}
```

Lot‘s of boilerplate code!

```
import Mock

public class ConfigRepositoryMock: Mock<ConfigRepository>, ConfigRepository {
    public func saveExpertToken(_ expertToken: String) async throws {
        try await acceptAsync(args: [expertToken])
    }

    public func loadExpertToken() -> AnyPublisher<String?, Never> {
        accept()
    }

    public func getLatestPresets(count: Int) async throws -> [AggregatedPreset] {
        try await acceptAsync(args: [count])
    }

    public func getAggregatedTestResultData(qnr: String) async throws -> AggregatedTestResultData? {
        try await acceptAsync(args: [qnr])
    }

    public func getAllAggregatedTestResultDatas(with status: TestStatus)
        -> AnyPublisher<[AggregatedTestResultData], Error> {
        accept(args: [status])
    }

    public func getIncompleteSharedConfigurationDatas() async throws -> [SearchSharedConfigurationData] {
        try await acceptAsync()
    }

    public func deleteTestResult(for sharedConfigurationDataId: Int64) async throws {
        try await acceptAsync(args: [sharedConfigurationDataId])
    }

    public func getAggregatedTestResult(for sharedConfigurationDataId: Int64)
        -> AnyPublisher<AggregatedTestResultData?, Error> {
        accept(args: [sharedConfigurationDataId])
    }

    public func saveSharedConfigurationData(
        _ sharedConfigurationData: SharedConfigurationData,
        isCompleted: Bool
```

Swift Macros to the rescue!

07.09.2023 Manuel Weiel

Swift Macros

```
10  @Mock
11  protocol DemoRepository {
12      func getPerson() async throws → Person
13
14      func postAge(personId: Int, age: Int)
15  }
16
```

Swift Macros

```
ClassDeclSyntax("""
  class \(raw: protocolName)Mock: Mock<\(raw: protocolName)>, \(raw: protocolName) {
    \(MemberBlockItemListSyntax(functionDeclarations + properties))
  }
""")
```

Swift Macros

```
ClassDeclSyntax("""
  class \(raw: protocolName)Mock: Mock<\(raw: protocolName)>, \(raw: protocolName) {
    \(MemberBlockItemListSyntax(functionDeclarations + properties))
  }
""")
```

```
class DemoRepositoryMock: Mock<DemoRepository>, DemoRepository {
  …
}
```

Swift Macros

```
ClassDeclSyntax("""
  class DemoRepositoryMock: Mock<DemoRepository>, DemoRepository {
    \(MemberBlockItemListSyntax(functionDeclarations + properties))
  }
""")
```

Swift Macros

```
ClassDeclSyntax("""
  class \(raw: protocolName)Mock: Mock<\(raw: protocolName)>, \(raw: protocolName) {
    \(MemberBlockItemListSyntax(functionDeclarations + properties))
  }
""")
let functionDeclarations = protocolDefinition.memberBlock.members
  .compactMap { $0.decl.as(FunctionDeclSyntax.self) }
  .map {
    let body = CodeBlockSyntax(statements: CodeBlockItemListSyntax([createMethodBody($0)]))
    return $0.trimmed.with(\.body, body)
  } …
```

Swift Macros

```
ClassDeclSyntax("""
  class \(raw: protocolName)Mock: Mock<\(raw: protocolName)>, \(raw: protocolName) {
    \(MemberBlockItemListSyntax(functionDeclarations + properties))
  }
""")

let functionDeclarations = protocolDefinition.memberBlock.members
  .compactMap { $0.decl.as(FunctionDeclSyntax.self) }
  .map {
    let body = CodeBlockSyntax(statements: CodeBlockItemListSyntax([createMethodBody($0)]))
    return $0.trimmed.with(\.body, body)
  } …
```

Swift Macros

```
ClassDeclSyntax("""
  class \(raw: protocolName)Mock: Mock<\(raw: protocolName)>, \(raw: protocolName) {
    \(MemberBlockItemListSyntax(functionDeclarations + properties))
  }
""")
let functionDeclarations = protocolDefinition.memberBlock.members
  .compactMap { $0.decl.as(FunctionDeclSyntax.self) }
  .map {
    let body = CodeBlockSyntax(statements: CodeBlockItemListSyntax([createMethodBody($0)]))
    return $0.trimmed.with(\.body, body)
  } …
```

```
try await acceptAsync(args: [x, y, z])
```

Swift Macros

```
ClassDeclSyntax("""
  class \(raw: protocolName)Mock: Mock<\(raw: protocolName)>, \(raw: protocolName) {
    \(MemberBlockItemListSyntax(functionDeclarations + properties))
  }
""")
let functionDeclarations = protocolDefinition.memberBlock.members
  .compactMap { $0.decl.as(FunctionDeclSyntax.self) }
  .map {
    let body = CodeBlockSyntax(statements: CodeBlockItemListSyntax([createMethodBody($0)]))
    return $0.trimmed.with(\.body, body)
  } …
```

INVERS

Conclusion

- Entry into code generation greatly reduced
- Already really stable in the current beta
  (to be released in Swift 5.9)
- Great potential for the future
- If you write macros, use https://swift-ast-explorer.com/

INVERS

Thank You for Your Attention!

Manuel Weiel
@xmanu@mastodon.social