Mocking in .NET Core Unit Tests with Moq: Getting Started

GETTING STARTED WITH MOCKING AND MOQ

Jason Roberts
.NET MVP

@robertsjason dontcodetired.com

Overview

An overview of mocking

Why mock?

What is a unit?

Fakes, stubs, mocks, and test doubles

Demo code overview

Add a new unit test project

Write initial tests

Introduce a new dependency

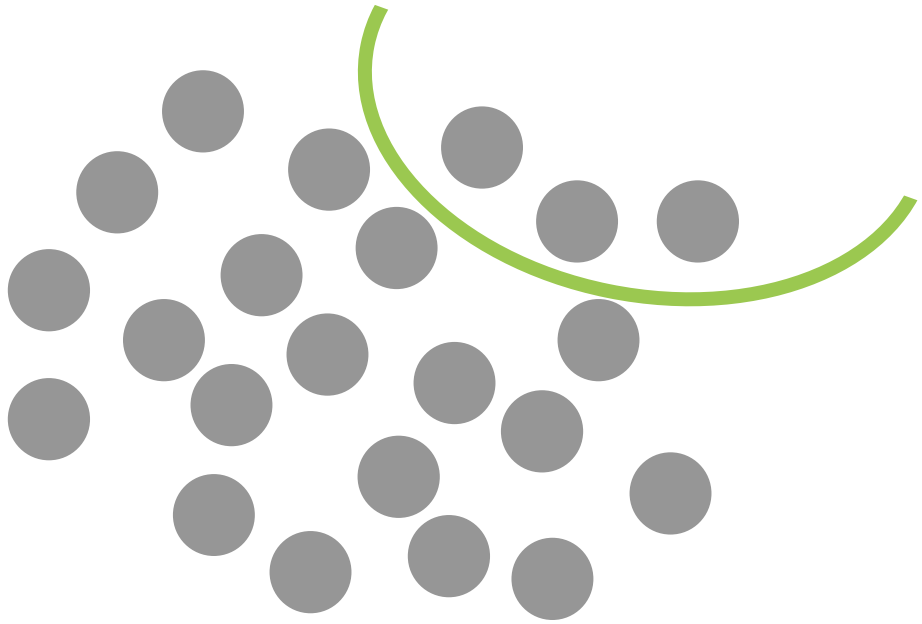
Breaks existing tests

Hard to use dependency

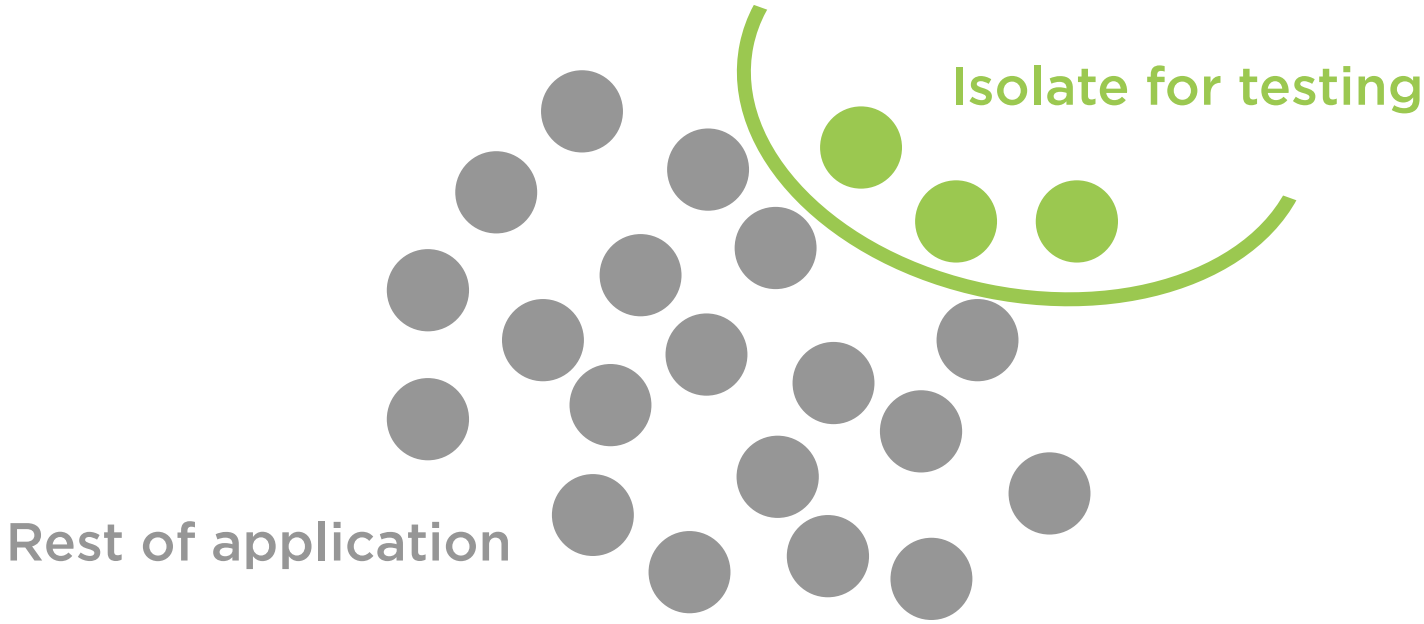
Install Moq

Course Outline

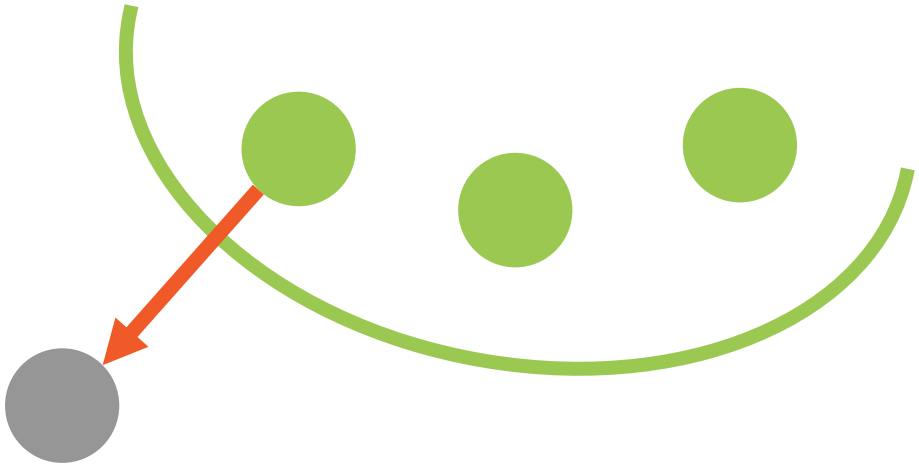
An Overview of Mocking



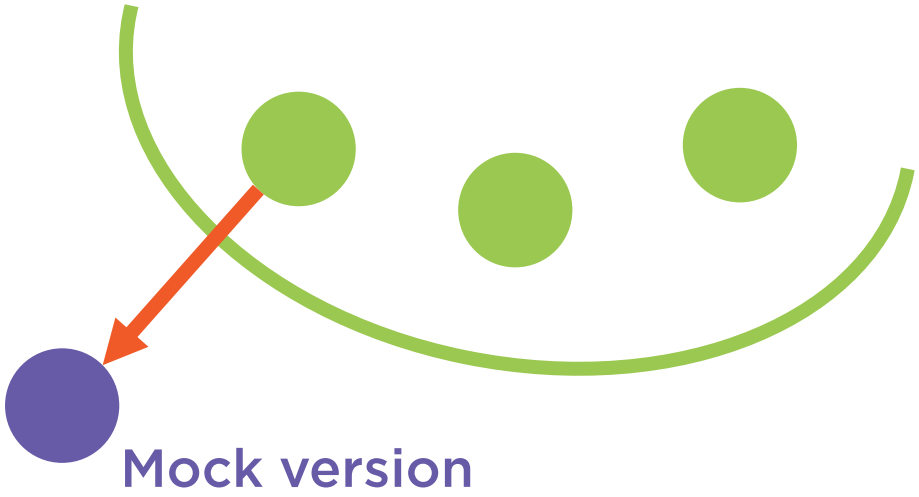
An Overview of Mocking



An Overview of Mocking



An Overview of Mocking



Replacing the actual dependency that would be used at production time, with a test-time-only version that enables easier isolation of the code we want to test.

Why Mock?

Improved test execution speed

- Slow algorithms
- External resources: DB, Web service, etc.

Support parallel development streams

- Real object not yet developed
- Another team
- External contractor

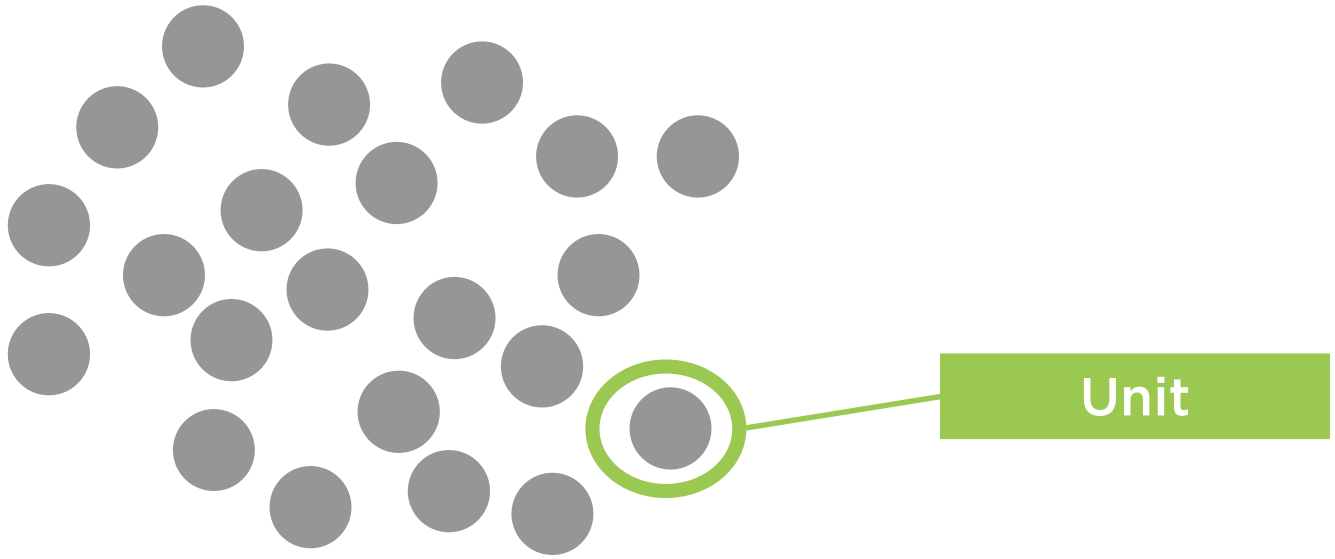
Improve test reliability

Reduce development/testing costs

- External company bills per usage
- Interfacing with mainframe
- Developer effort (complexity)

Test when non-deterministic dependency

What Is a Unit?



Unit Tests

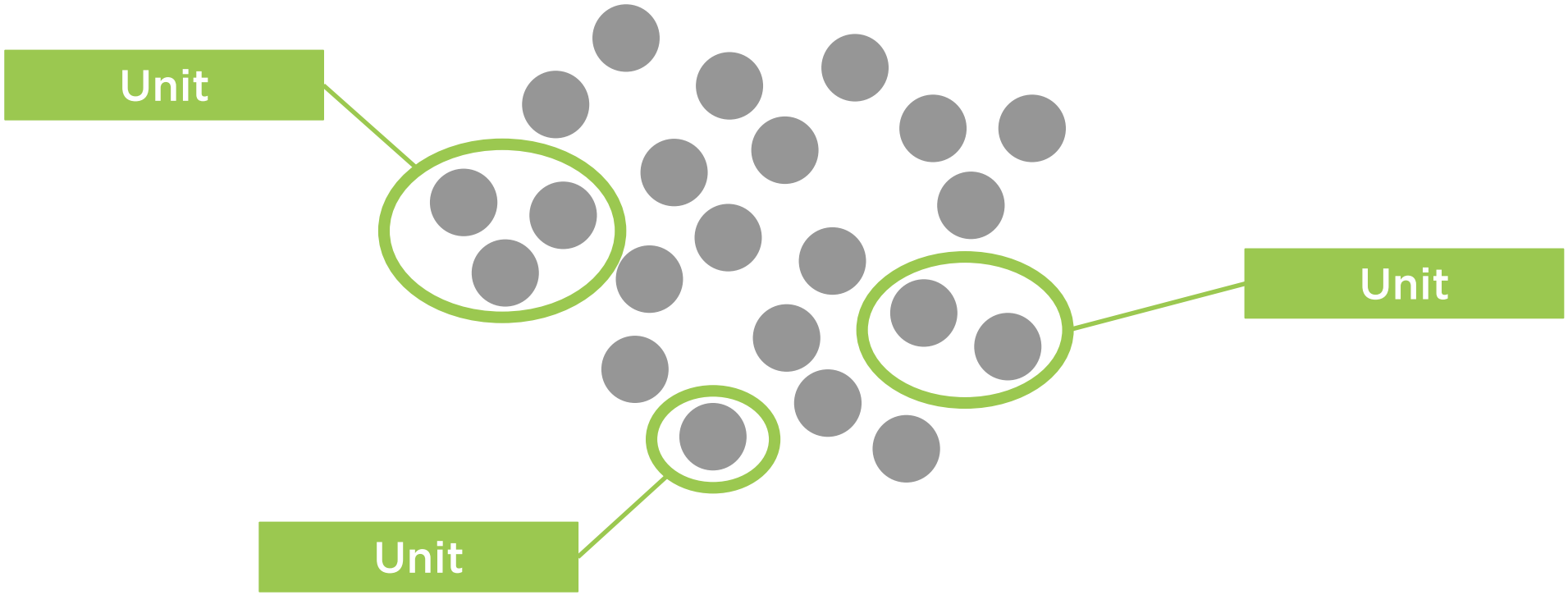
Low level

Highly focused

Quick to execute

Easier to test all parts / paths of code

Unit Tests

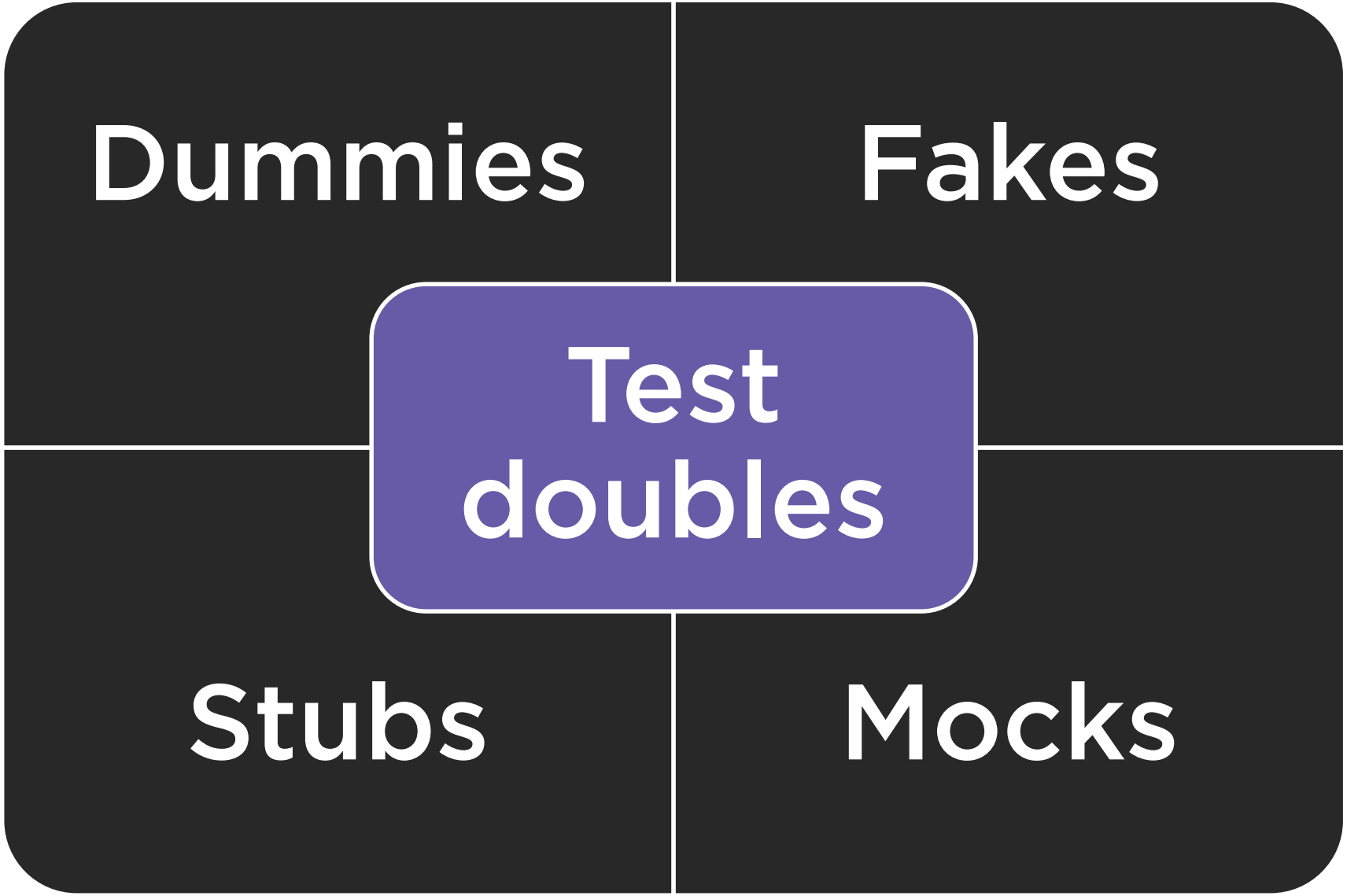


“...it's a situational thing - the team decides what makes sense to be a unit for the purposes of their understanding of the system and its testing”

Martin Fowler

https://martinfowler.com/bliki/UnitTest.html

Units of behavior over units of implementation


Dummies
Fakes
Test doubles
Stubs
Mocks

“Test Double is a generic term for any case where you replace a production object for testing purposes.”

Martin Fowler

https://martinfowler.com/bliki/TestDouble.html

Fakes, Dummies, Stubs, and Mocks

Fakes

Working implementation

Not suitable for production

EF Core in-memory provider

Dummies

Passed around

Never used / accessed

Satisfy parameters

Stubs

Provide answers to calls

Property gets

Method return values

Mocks

Expect/verify calls

Properties

Methods

Moq

In this course we'll use the generic term "mock" object.

An Overview of Moq

“Mock-you” (“Mock”)

Open source project

https://github.com/moq

17.9 million NuGet downloads

Design goals:

- Simple
- Practical
- Straight-forward to use

Summary

- Isolation using test-time-only dependencies
- Why mock?
- What is a unit?
- Fakes, stubs, mocks, and test doubles
- Added a new xUnit.net test project
- Wrote initial tests
- IFrequentFlyerNumberValidator
- Broke existing tests
- Installed Moq

Next:

Mocking Method Calls