Unit Testing an ASP.NET Core 6 MVC Web Application

Introduction to Unit Testing

Kevin Dockx
Architect

@KevinDockx https://www.kevindockx.com

Version Check

This version was created by using:

- ASP.NET Core 6.0
- xUnit 2.4.1
- Moq 4.17.2
- .NET 6.0
- Visual Studio 2022

Version Check

This course is 100% applicable to:

- ASP.NET Core 6.x
- xUnit 2.x
- Moq 4.x
- .NET 6.x

Relevant Notes

New course versions are regularly released:

- https://app.pluralsight.com/profile/author/kevin-dockx

Coming Up

Positioning this course

Prerequisites, frameworks and tooling

Introducing the demo scenario

The what, why and what not of unit testing

- Different types of tests

Coming Up

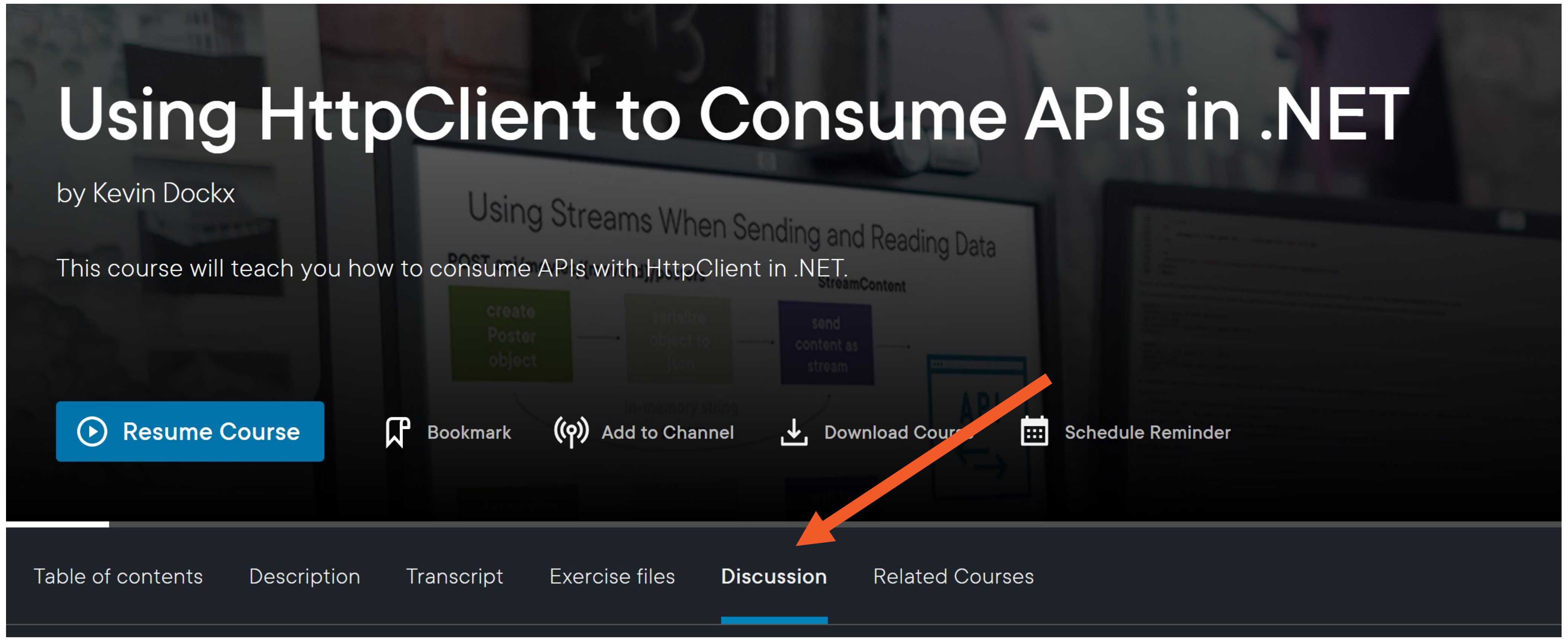
Using xUnit to write your first unit test

The Arrange, Act, Assert (AAA) pattern

Comparing xUnit, nUnit and MSTest

Discussion tab on the course page

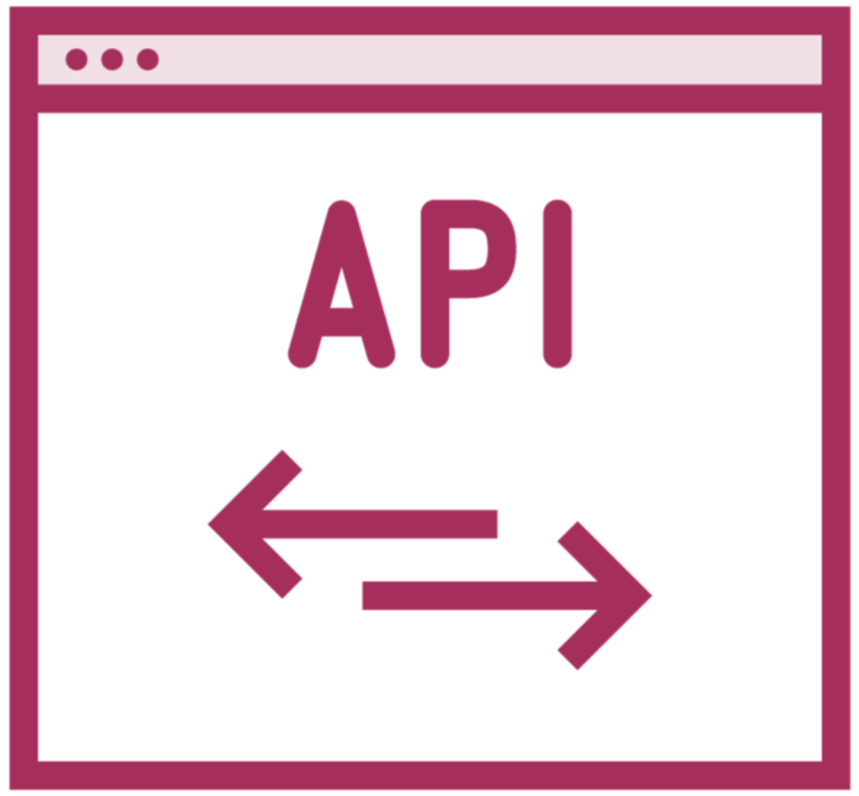
Twitter: @KevinDockx



(course shown is one of my other courses, not this one)

Positioning This Course

Unit Testing an ASP.NET Core 6 MVC Web Application



Unit Testing an ASP.NET Core 6 Web API

Positioning This Course

Unit Testing an ASP.NET Core 6 MVC Web Application

Unit Testing an ASP.NET Core 6 Web API

Asserting, setting up (data-driven) tests and test isolation (different demo scenarios)

Scenarios specific to unit testing web applications

Scenarios specific to unit testing api applications

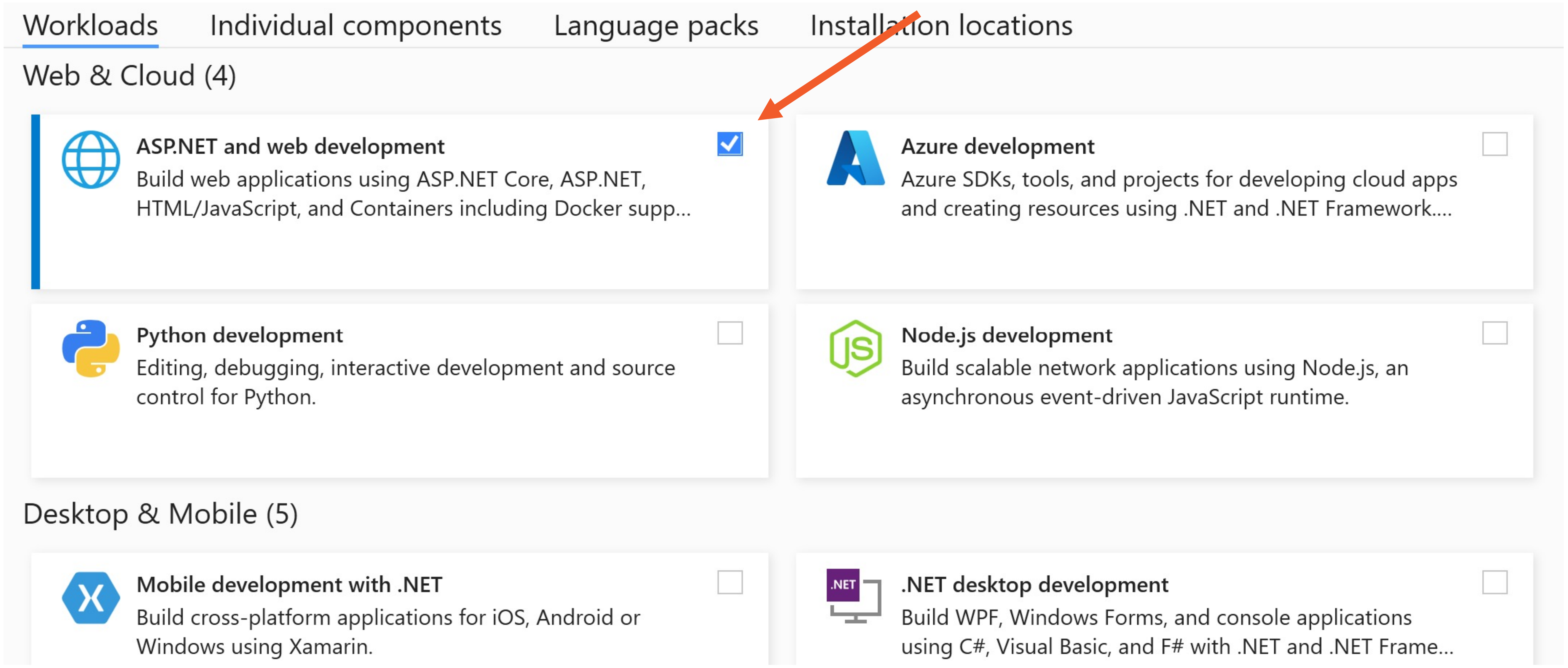
Integrating unit tests in your CI/CD pipeline

Course Prerequisites and Frameworks

C# 10

ASP.NET Core 6 MVC

Installing Visual Studio



Workloads Individual components Language packs Installation locations

Web & Cloud (4)

[✓] **ASP.NET and web development**
Build web applications using ASP.NET Core, ASP.NET, HTML/JavaScript, and Containers including Docker supp...

[ ] **Azure development**
Azure SDKs, tools, and projects for developing cloud apps and creating resources using .NET and .NET Framework....

[ ] **Python development**
Editing, debugging, interactive development and source control for Python.

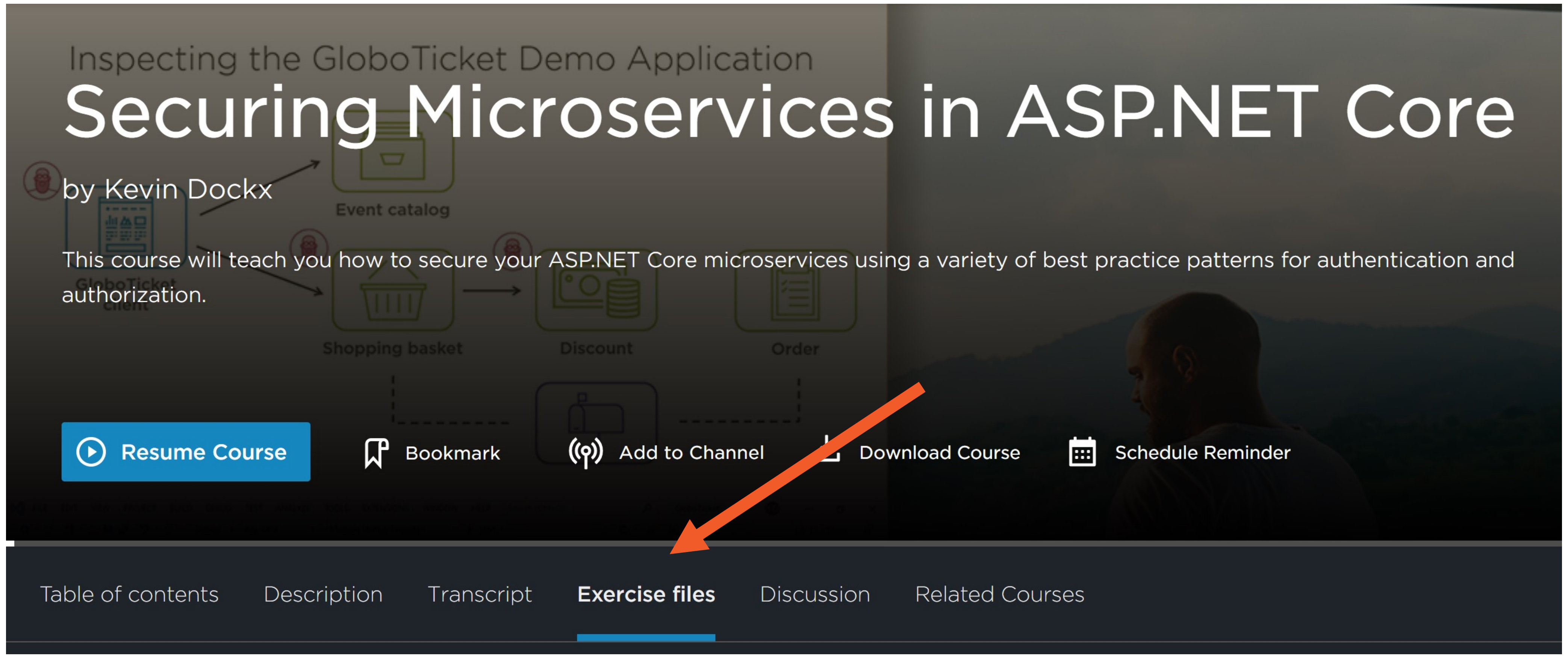
[ ] **Node.js development**
Build scalable network applications using Node.js, an asynchronous event-driven JavaScript runtime.

Desktop & Mobile (5)

[ ] **Mobile development with .NET**
Build cross-platform applications for iOS, Android or Windows using Xamarin.

[ ] **.NET desktop development**
Build WPF, Windows Forms, and console applications using C#, Visual Basic, and F# with .NET and .NET Frame...

Exercise files tab on the course page


Inspecting the GloboTicket Demo Application

Securing Microservices in ASP.NET Core

by Kevin Dockx

This course will teach you how to secure your ASP.NET Core microservices using a variety of best practice patterns for authentication and authorization.

(course shown is one of my other courses, not this one)

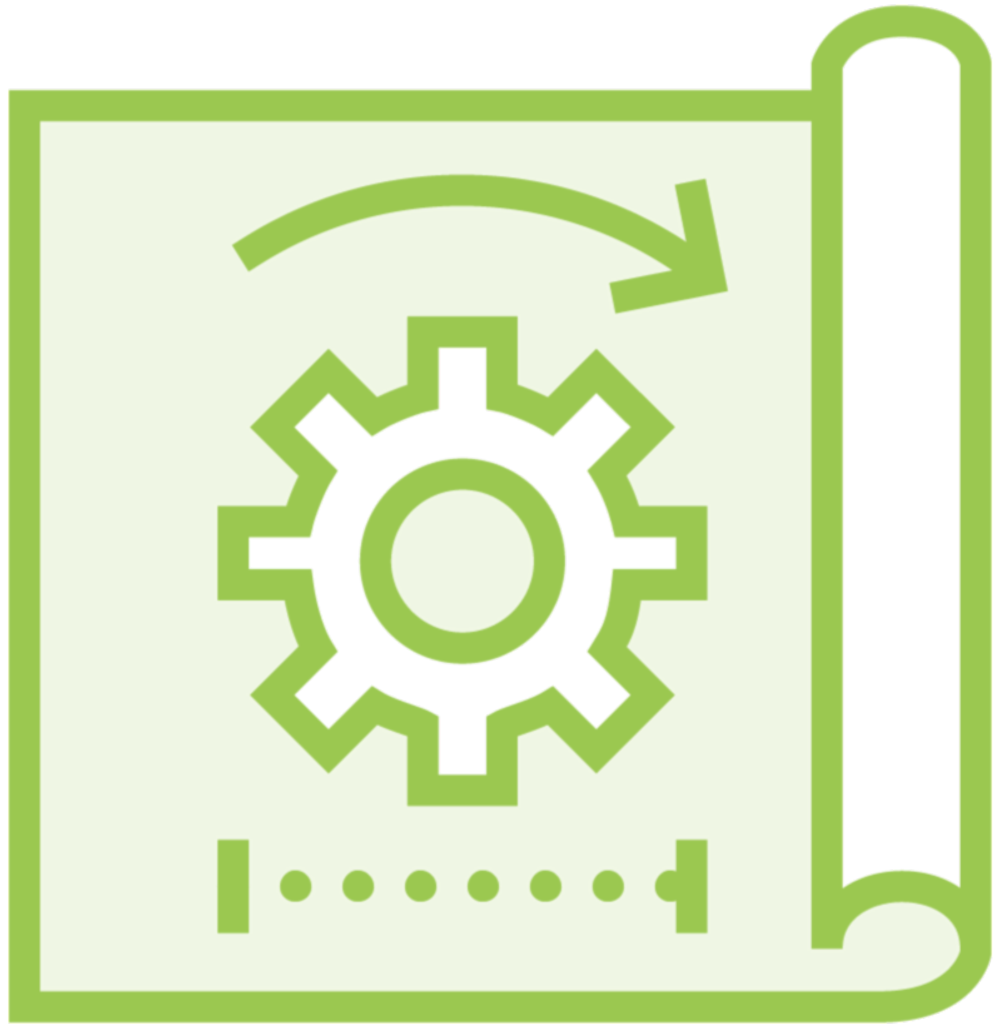
Demo

Introducing the demo scenario

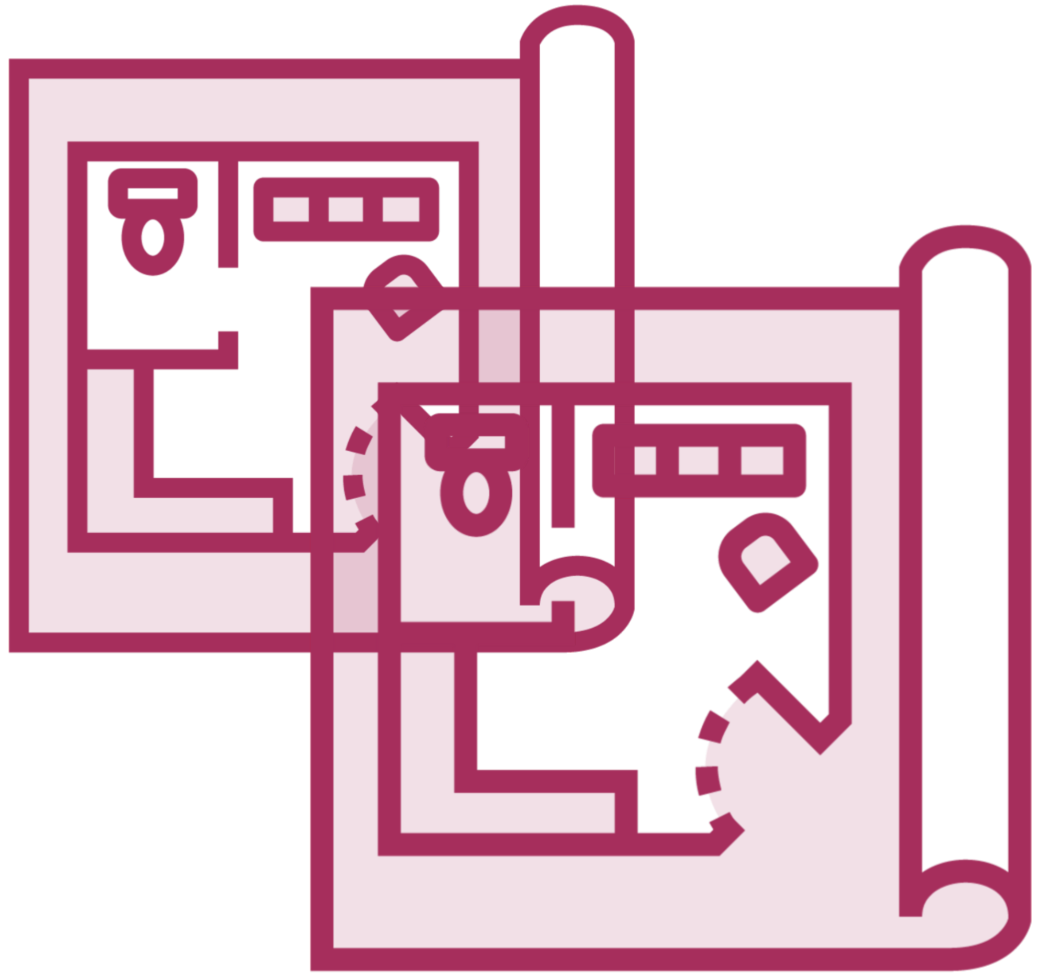
Unit test

A unit test is an automated test that tests a small piece of behavior

The What, Why and What Not of Unit Testing



Often just (part of) a method of a class



Potentially functionally related behavior across classes

The What, Why and What Not of Unit Testing

Unit tests should have low complexity

Unit tests should be fast

Unit tests should be well encapsulated
Helps with ensuring it’s “the thing we’re testing” that fails/passes

Reasons for Test Automation

Improved reliability at a relatively low cost

Write once, use without additional cost

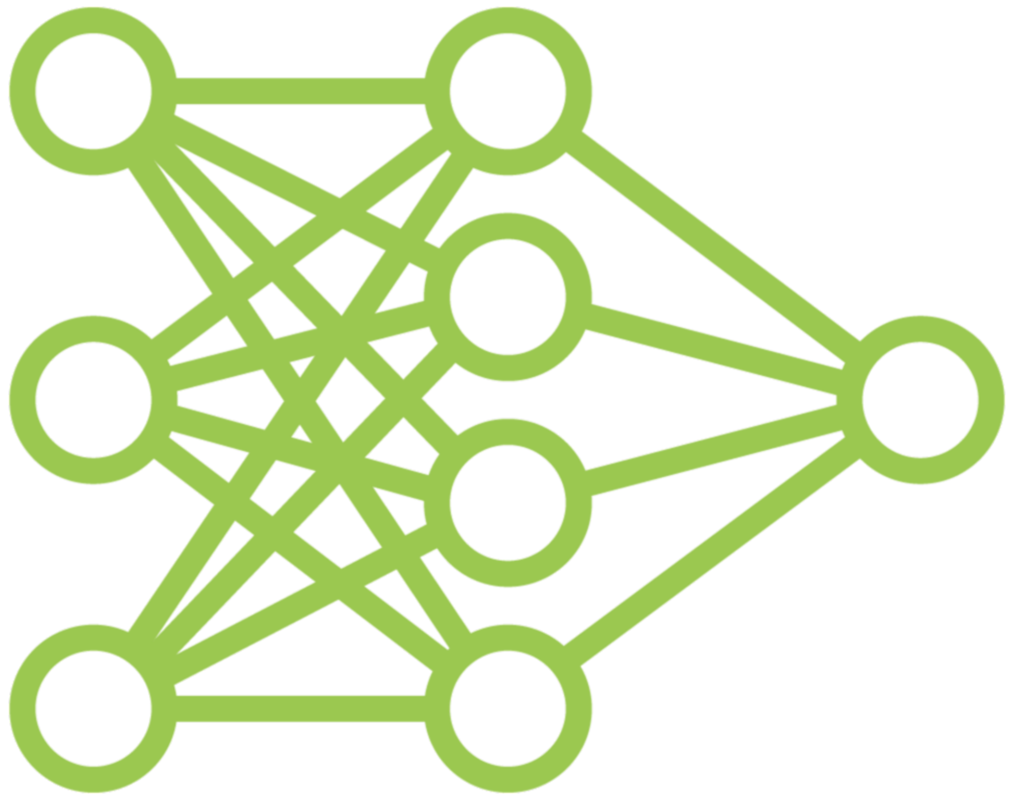
Enables testing often and multiple times

Reasons for Test Automation

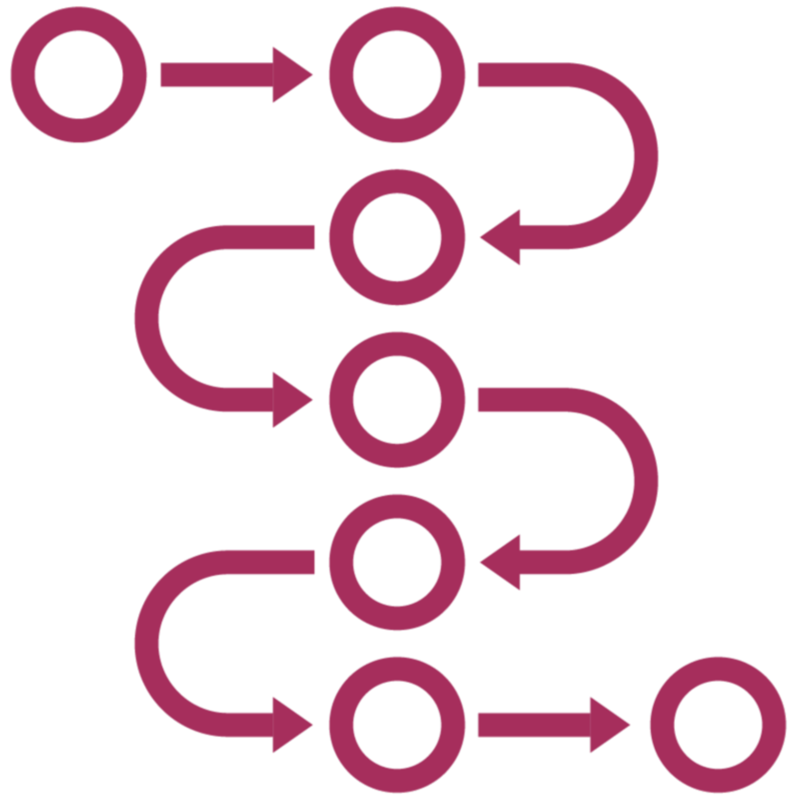
Bugs are found faster and easier

- Makes them cheaper to fix

The What, Why and What Not of Unit Testing



A unit test does not test the whole system



A unit test does not test how parts of a system that are related to each other interact

Comparing Unit Tests, Integration Tests and Functional Tests

Most applications should be tested with a combination of automated tests

- Unit tests
- Integration tests
- Functional (end-to-end) tests

Unit Test Characteristics

Unit tests should have low complexity

Unit tests should be fast

Unit tests should be well encapsulated

Integration test

A integration test is an automated test that tests whether or not two or more components work together correctly

Comparing Unit Tests, Integration Tests and Functional Tests

Can test a full request/response cycle, but doesn't have to

Can be created with the same frameworks as unit tests

- Optionally combined with Microsoft TestHost and TestServer

Integration Test Characteristics

Integration tests have medium complexity

Integration tests are relatively slow

Integration tests are not well encapsulated

Comparing Unit Tests, Integration Tests and Functional Tests

# unit tests > # integration tests

Functional test

A unit test is an automated test that tests the full request/response cycle of an application

Comparing Unit Tests, Integration Tests and Functional Tests

Can be automated with

- Selenium (web applications)
- Postman (APIs)
- Microsoft TestHost and TestServer

Functional Test Characteristics

Functional tests have high complexity

Functional tests are slow

Functional tests are badly encapsulated

Comparing Unit Tests, Integration Tests and Functional Tests

# unit tests > **# integration tests** > **# functional tests**

Demo

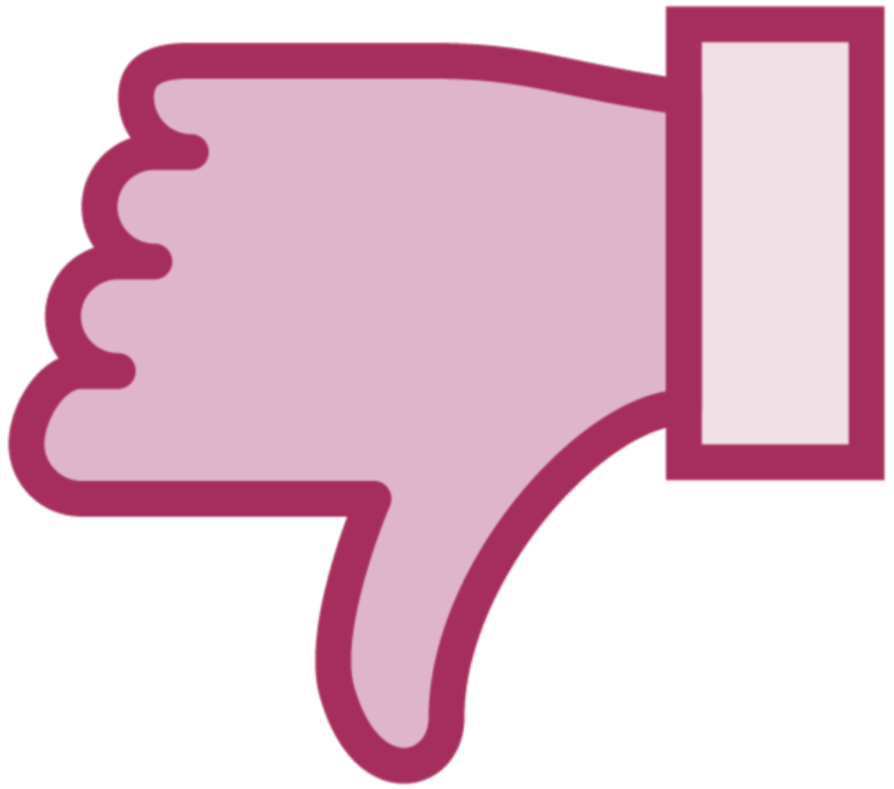
Adding a unit test project

Demo

Writing your first unit test

Good and Bad Candidates for a Unit Test

Good candidates
Algorithms, behavior, rules



Bad candidates
Data access, UI, system interactions

Naming Guidelines for Unit Tests

CreateEmployee_ConstructInternalEmployee_SalaryMustBe2500

Naming Guidelines for Unit Tests

CreateEmployee_ConstructInternalEmployee_SalaryMustBe2500

A name for the unit that's being tested

Naming Guidelines for Unit Tests

CreateEmployee_ConstructInternalEmployee_SalaryMustBe2500

The scenario under which the unit is being tested

Naming Guidelines for Unit Tests

CreateEmployee_ConstructInternalEmployee_SalaryMustBe2500

The expected behavior when the scenario is invoked

The Arrange, Act, Assert Pattern

Arrange
Setting up the test

Act
Executing the actual test

Assert
Verifying the executed action

```
[Fact]
public void CreateEmployee_ConstructInternalEmployee_SalaryMustBe2500() {

    var employeeFactory = new EmployeeFactory();

    var employee = (InternalEmployee)employeeFactory.CreateEmployee("Kevin", "Dockx");

    Assert.Equal(2500, employee.Salary);
}
```

The Arrange, Act, Assert Pattern

```
[Fact]
public void CreateEmployee_ConstructInternalEmployee_SalaryMustBe2500() {

    // Arrange
    var employeeFactory = new EmployeeFactory();


    var employee = (InternalEmployee)employeeFactory.CreateEmployee("Kevin", "Dockx");


    Assert.Equal(2500, employee.Salary);
}
```

The Arrange, Act, Assert Pattern

Arrange: setting up the test

```
[Fact]
public void CreateEmployee_ConstructInternalEmployee_SalaryMustBe2500() {

    // Arrange
    var employeeFactory = new EmployeeFactory();

    // Act
    var employee = (InternalEmployee)employeeFactory.CreateEmployee("Kevin", "Dockx");


    Assert.Equal(2500, employee.Salary);
}
```

The Arrange, Act, Assert Pattern

Act: executing the actual test

```
[Fact]
public void CreateEmployee_ConstructInternalEmployee_SalaryMustBe2500() {

    // Arrange
    var employeeFactory = new EmployeeFactory();

    // Act
    var employee = (InternalEmployee)employeeFactory.CreateEmployee("Kevin", "Dockx");

    // Assert
    Assert.Equal(2500, employee.Salary);
}
```

The Arrange, Act, Assert Pattern

Assert: verifying the executed action

Comparing xUnit, nUnit and MSTest

MSTest

Microsoft's built-in unit test framework

Support for .NET (Core) since v2.0

nUnit

A port of jUnit

Been around for a long time

MSTest and nUnit

Can be used to test .NET 6 code

- But they carry technical debt with them…

Designed nor coded with .NET Core or .NET 6 in mind

Comparing xUnit, nUnit and MSTest

MSTest

Microsoft's built-in unit test framework

Support for .NET (Core) since v2.0

nUnit

A port of jUnit

Been around for a long time

xUnit

Built with .NET (Core) and new .NET features in mind

xUnit

Successor of nUnit, built with .NET (Core) and new .NET features in mind

- Improves test isolation, and extensibility
- Encourages cleaner testing code

Summary

A unit test is an automated test that tests a small piece of behavior, often simply testing the methods of a class

- Improves application reliability at a much lower cost than manual testing

Summary

xUnit is the current de facto standard framework for unit testing in .NET

- `[Fact]` signifies a unit test method

Test explorer makes it easy to run tests and inspect test results

Up Next:
Tackling Basic Unit Testing Scenarios