EIU
TRƯỜNG ĐẠI HỌC
QUỐC TẾ
MIỀN ĐÔNG
EASTERN
INTERNATIONAL
UNIVERSITY

Backend Development

Chapter 6: Testing

EIU

TABLE CONTENT

LESSON OBJECTIVES

- Understanding the Difference Between Unit Test & Integration Test
- Know how to write tests in ASP.NET Core with xUnit, Moq, TestServer
- Apply tests to Controllers, Services, and Repositories

WHY WE NEED TESTING?

- Make sure the logic is correct when the code is complex
- Refactor, CI/CD
- Early Fault Detection → Cost Savings

EIU

UNIT TEST

What is Unit Test?

- Test the smallest unit of the application (function, method, class)
- No Database, API, or Network Involvement

Example:

```
decimal CalculateDiscount(Order order)
```

- → Test the exact value returned for each input

Unit Test Tool for ASP.NET Core

- xUnit: Official Test Framework
- MOQ: Create a mock object to isolate the unit test
- FluentAssertions: Writing asserts is easier to understand

AAA (Arrange–Act–Assert) Model

- Arrange – Prepare data/mock
- Act – Calling the function to be tested
- Assert – Check the return results

```
// Arrange
var service = new OrderService();
var order = new Order(...);

// Act
var total = service.CalculateTotal(order);

// Assert
Assert.Equal(500, total);
```

Demo Unit Test with xUnit + MOQ

```
[Fact]
public void GetProductPrice_WithDiscount_ReturnsCorrectPrice()
{
    var mockRepo = new Mock<IProductRepository>();
    mockRepo.Setup(r => r.GetBasePrice(1)).Returns(100);

    var service = new ProductService(mockRepo.Object);

    var price = service.GetFinalPrice(1);

    Assert.Equal(90, price);
}
```

EIU

When should we write a Unit Test?

- ✅ Write when processing computational logic, validate
- ✅ When fixing bugs: write tests to control bugs
- ✅ Before refactor code
- ❌ Don't write Unit Tests for UI or call real DBs

Common mistakes

- ❌ No separation of logic from dependencies (non-DI)
- ❌ Write a real DB dependency test (→ Integration Test)
- ❌ Not asserting enough cases (e.g., null, exception)
- ❌ The test is too general → it is not clear where the error is

EIU

Advantages of Unit Test

- ⚡ Very fast → CI/CD compliant
- 🔁 Easy retesting when changing
- 🎯 Focus on the right core logic
- ✅ Easy to write, easy to maintain

EIU

INTEGRATION TESTS

What is Integration Test?

Test the coordination between layers in the application

Comprise: Controller → Service → DB (In-Memory or real) Are endpoint checks working as expected?

EIU

Integration Test Tool in ASP.NET Core

- Microsoft.AspNetCore.Mvc.Testing
- WebApplicationFactory<TStartup> – Create a real app to test
- HttpClient – Submit a mock request
- SQLite InMemory, Testcontainers – Lightweight DB test

Integration Test API Example

```
[Fact]
public async Task PostOrder_Returns201Created()
{
    var client = _factory.CreateClient();
    var response = await client.PostAsJsonAsync("/api/orders", new OrderDto(...));

    Assert.Equal(HttpStatusCode.Created, response.StatusCode);
}
```

- Check the API /api/orders works correctly
- Test data stored in DB In-Memory

Advantages of Integration Test

- Close to the actual operation of the system
- Coordinated logic error detection
- Can be falsely detected when mapping model → entity → DB

Disadvantages of Integration Test

- Longer runtime than Unit Test
- More complicated setup
- If you don't isolate your tests → data conflicts

Advantages of Integration Test

- Close to the actual operation of the system
- Coordinated logic error detection
- Can be falsely detected when mapping model → entity → DB

How to combine the two types of tests?

Type of Test	Percent
Unit Test	70%
Integration Test	30%

Visual Comparison

Criteria	Unit Test	Integration Test
Scope	Function, small class	Full API Flow
Database Dependencies	Not	Yes/InMemory
Speed	Very fast	Average
Complexity	Easy	Harder

Conclusion

- Unit Test: test logic Core – Fast, Separate
- Integration Test: Endpoint coordination test – close to reality
- Combine the two for optimal quality for ASP.NET Core applications

Install xUnit, MOQ, and FluentAssertions

- dotnet add package xunit
- dotnet add package Moq
- dotnet add package FluentAssertions
- dotnet add package Microsoft.NET.Test.Sdk
- dotnet add package xunit.runner.visualstudio
- dotnet add package coverlet.collector

You can check the installed packages in MyApp.Tests.csproj.

(Optional) Install test integration with WebApplicationFactory

- dotnet add package Microsoft.AspNetCore.Mvc.Testing

Used for Integration Test (emulating API requests via TestServer).

Configure MyApp.Tests.csproj (if needed)

```
<PropertyGroup>
  <TargetFramework>net8.0</TargetFramework>
  <IsPackable>false</IsPackable>
</PropertyGroup>

<ItemGroup>
  <PackageReference Include="Moq" Version="4.18.4" />
  <PackageReference Include="xunit" Version="2.4.2" />
  <PackageReference Include="xunit.runner.visualstudio" Version="2.4.5" />
  <PackageReference Include="FluentAssertions" Version="6.11.0" />
  <PackageReference Include="coverlet.collector" Version="3.2.0" />
  <PackageReference Include="Microsoft.AspNetCore.Mvc.Testing"
Version="7.0.10" />
</ItemGroup>
```

Test run

Create file SampleTest.cs in MyApp.Tests:

```
using Xunit;

namespace MyApp.Tests;

public class SampleTest
{
    [Fact]
    public void SimpleAddition_WorksCorrectly()
    {
        int result = 2 + 3;
        Assert.Equal(5, result);
    }
}
```

Run the test:

```
dotnet test
```

Configure test coverage (optional)

Thêm vào file MyApp.Tests.csproj:

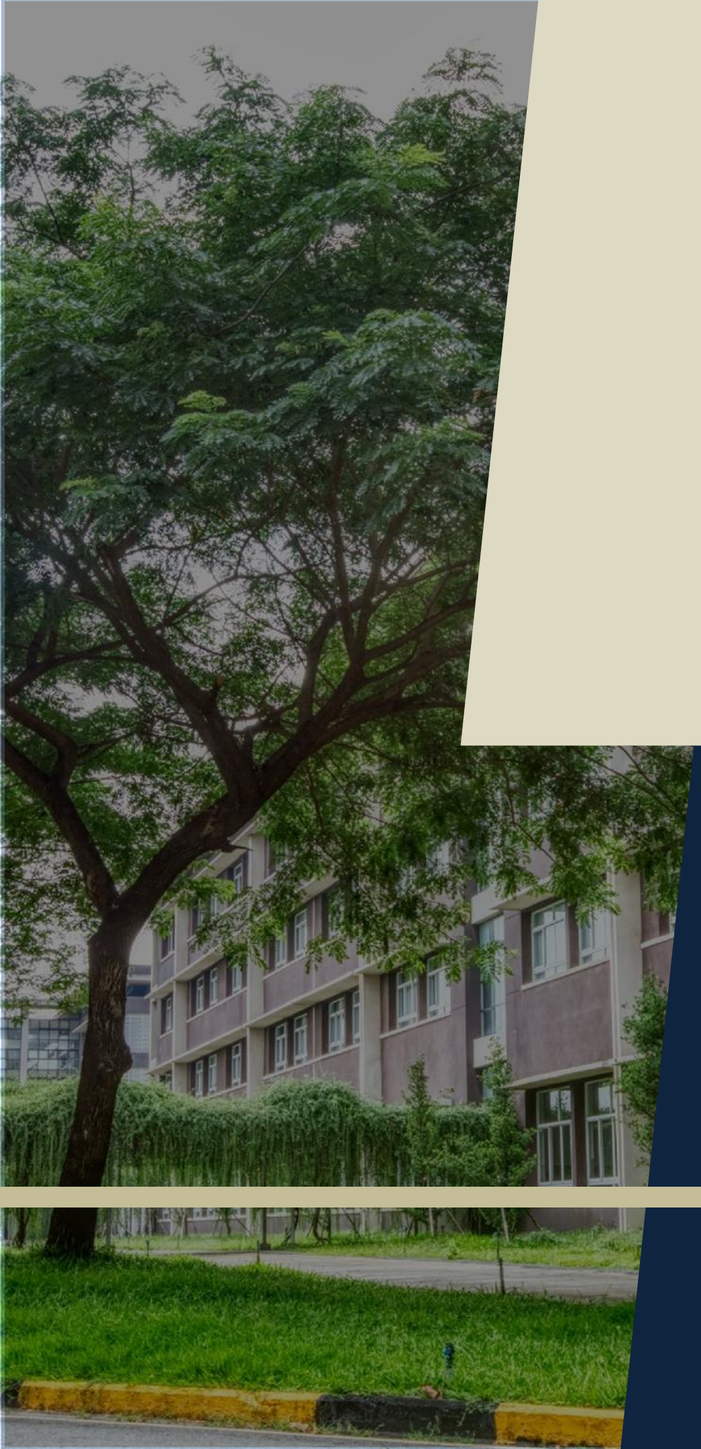
```
<CollectCoverage>true</CollectCoverage>
<CoverletOutputFormat>opencover</CoverletOutputFormat>
```

Professional Test Directory Organization

```
MyApp.Tests/
├── Services/
│   └── OrderServiceTests.cs
├── Controllers/
│   └── OrdersControllerTests.cs
├── Integration/
│   └── OrdersApiTests.cs
```

Summary of main packages

Package	Purpose
xUnit	Framework Main Testing
Moq	Interface and dependency forgery
FluentAssertions	Write asserts more clearly
Microsoft.AspNetCore.Mvc.Testing	Test Integration API
coverlet.collector	Calculate coverage when running tests



EIU
TRƯỜNG ĐẠI HỌC
QUỐC TẾ
MIỀN ĐÔNG
EASTERN
INTERNATIONAL
UNIVERSITY

Start your future at EIU

Thank You