FPT Education

FPT POLYTECHNIC

LẬP TRÌNH C# 6

BÀI 8: UNIT TEST ASP.NET CORE

P8.1

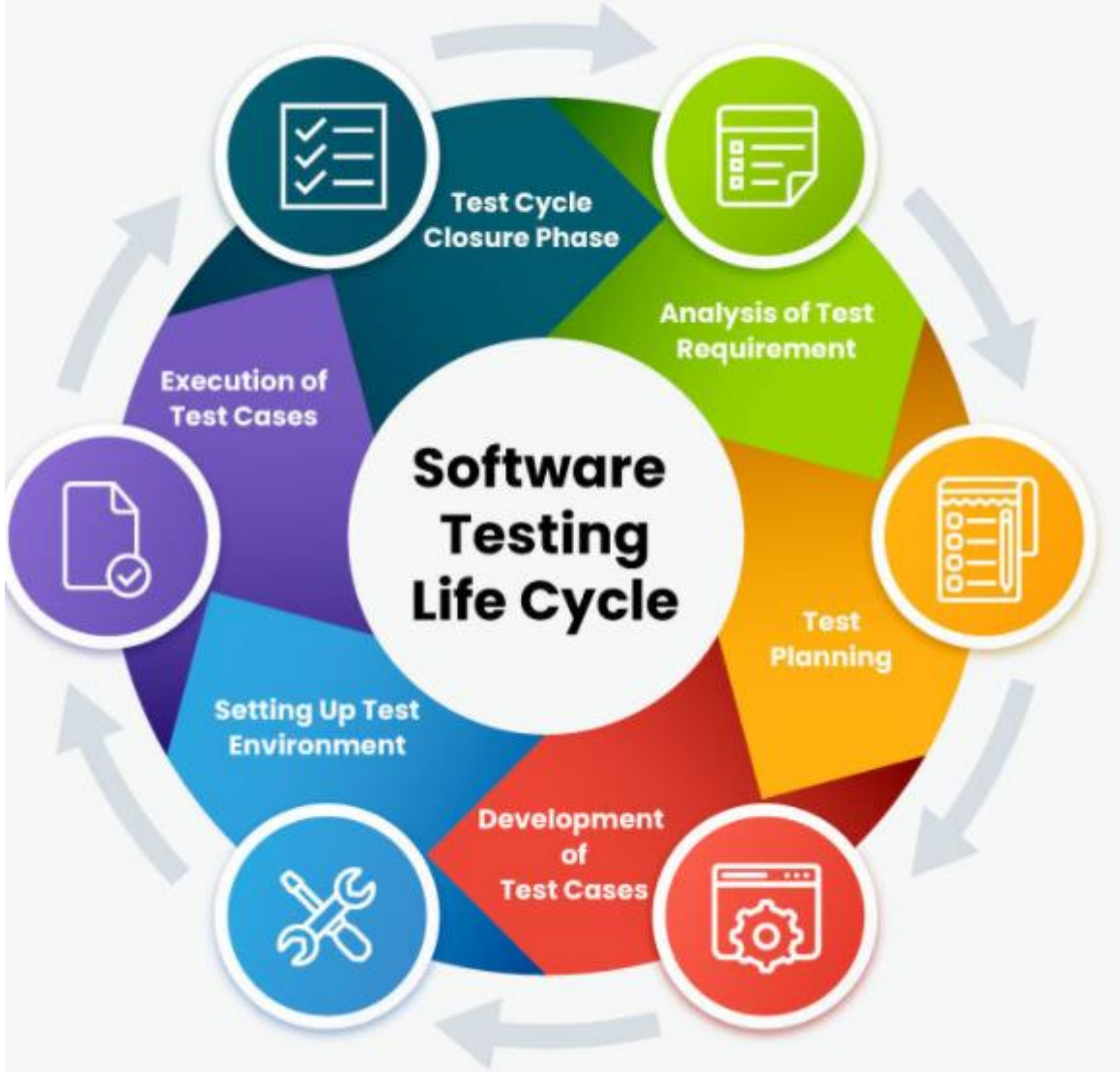
Mục tiêu

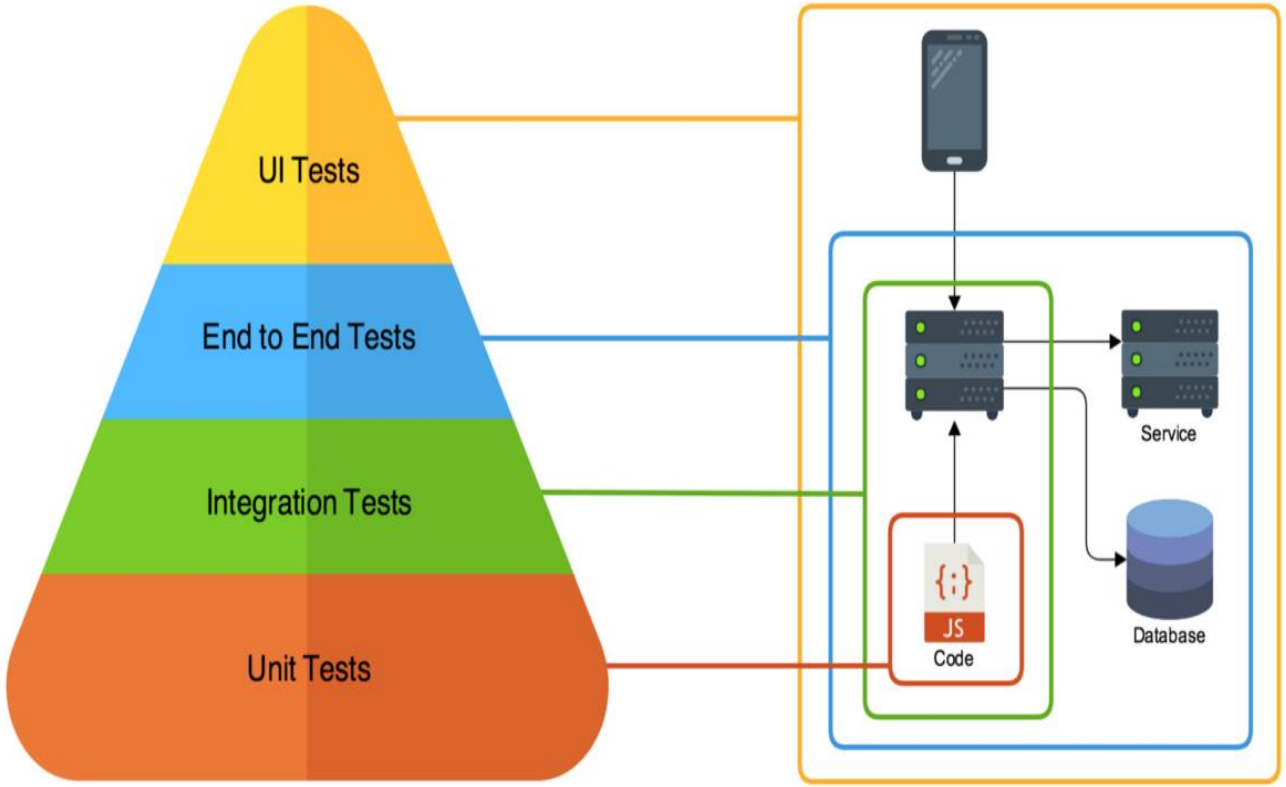
- Tổng quan Unit test .Net
- Implement xUnit.net
- Mock object - Moq Library
- Blazor Unit Testing

SOFTWARE TESTING

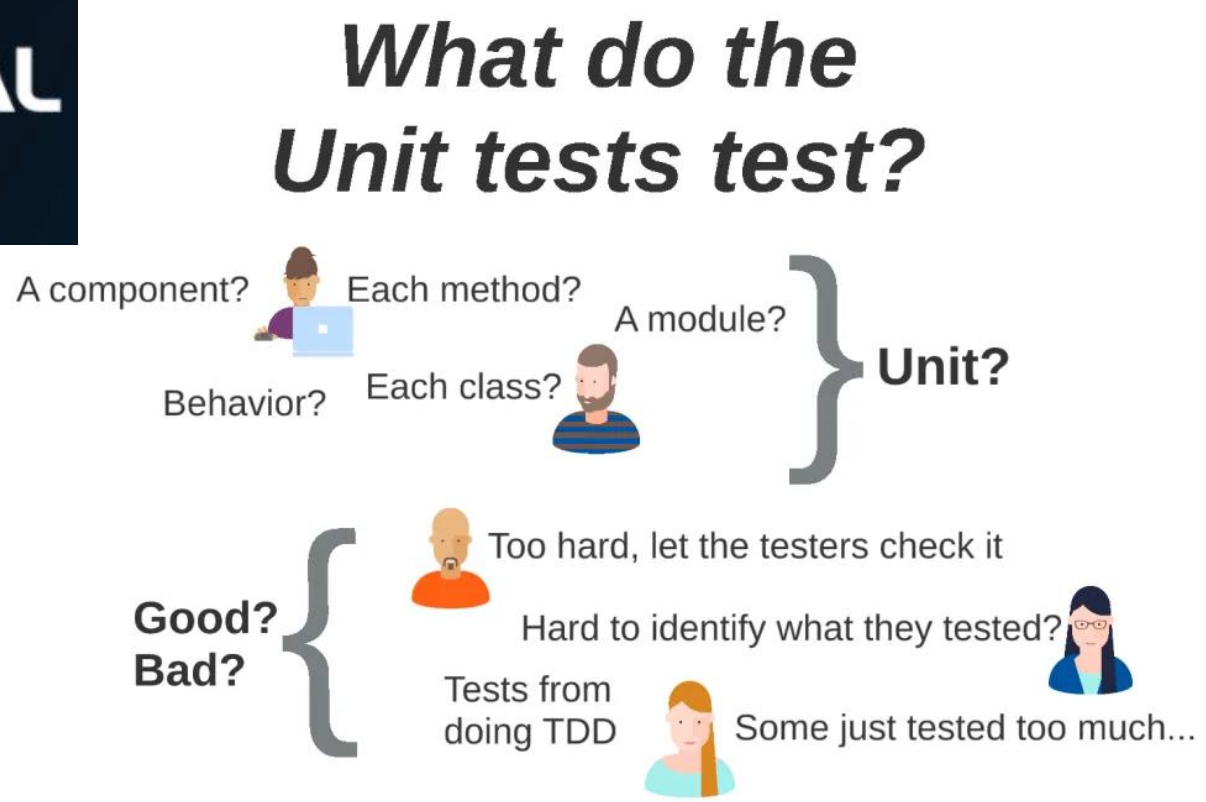
- To Gain Customer Confidence
- To Check Software Adaptability
- To Identify Errors
- To Avoid Extra Costs
- To Accelerate Software Development
- To Avoid Risks
- To Optimise Business

SOFTWARE TESTING



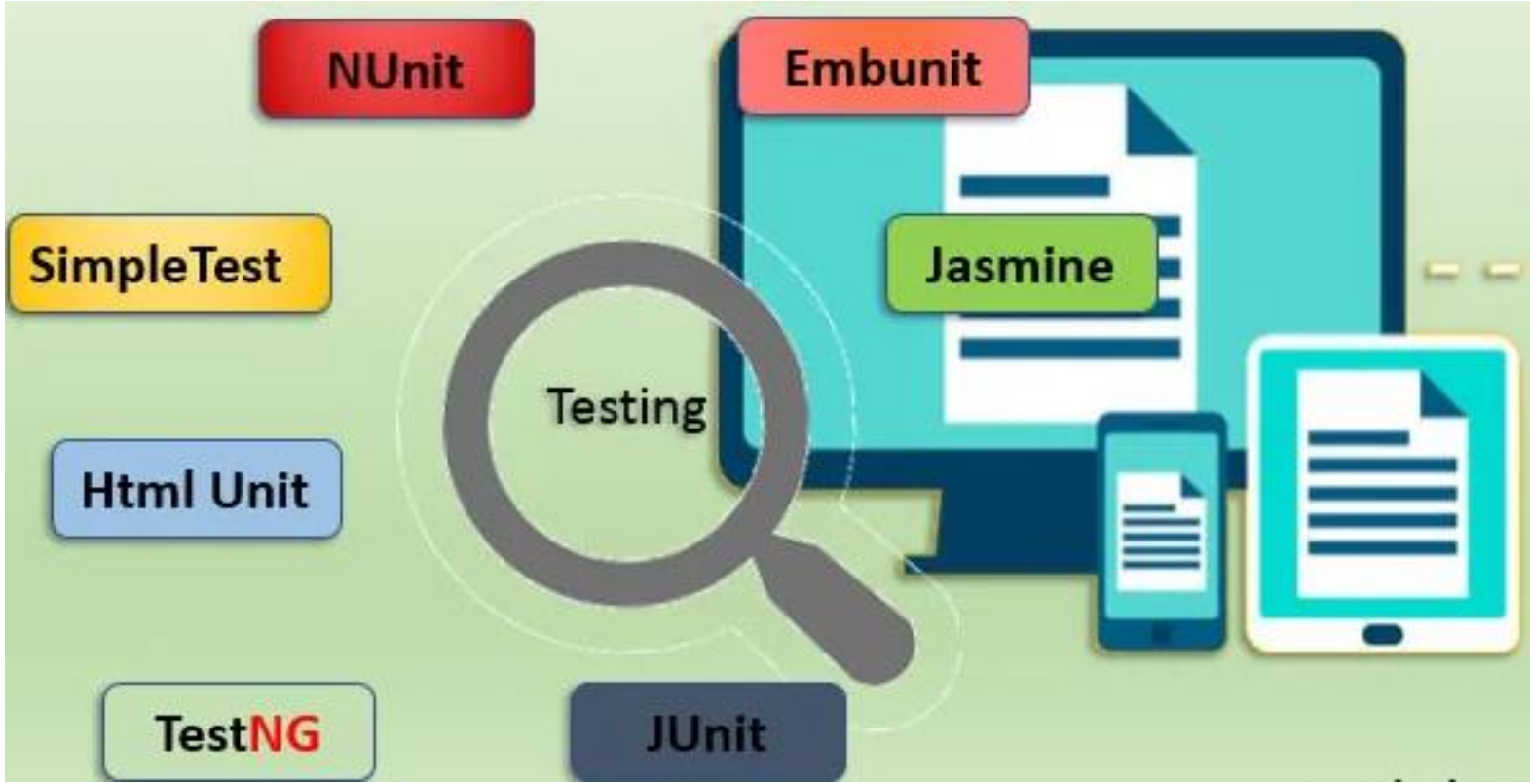



AUTOMATION TEST
VS
MANUAL TEST


What do the Unit tests test?
A component?
Each method?
A module?
Behavior?
Each class?
Unit?
Good?
Bad?
Too hard, let the testers check it
Hard to identify what they tested?
Tests from doing TDD
Some just tested too much...

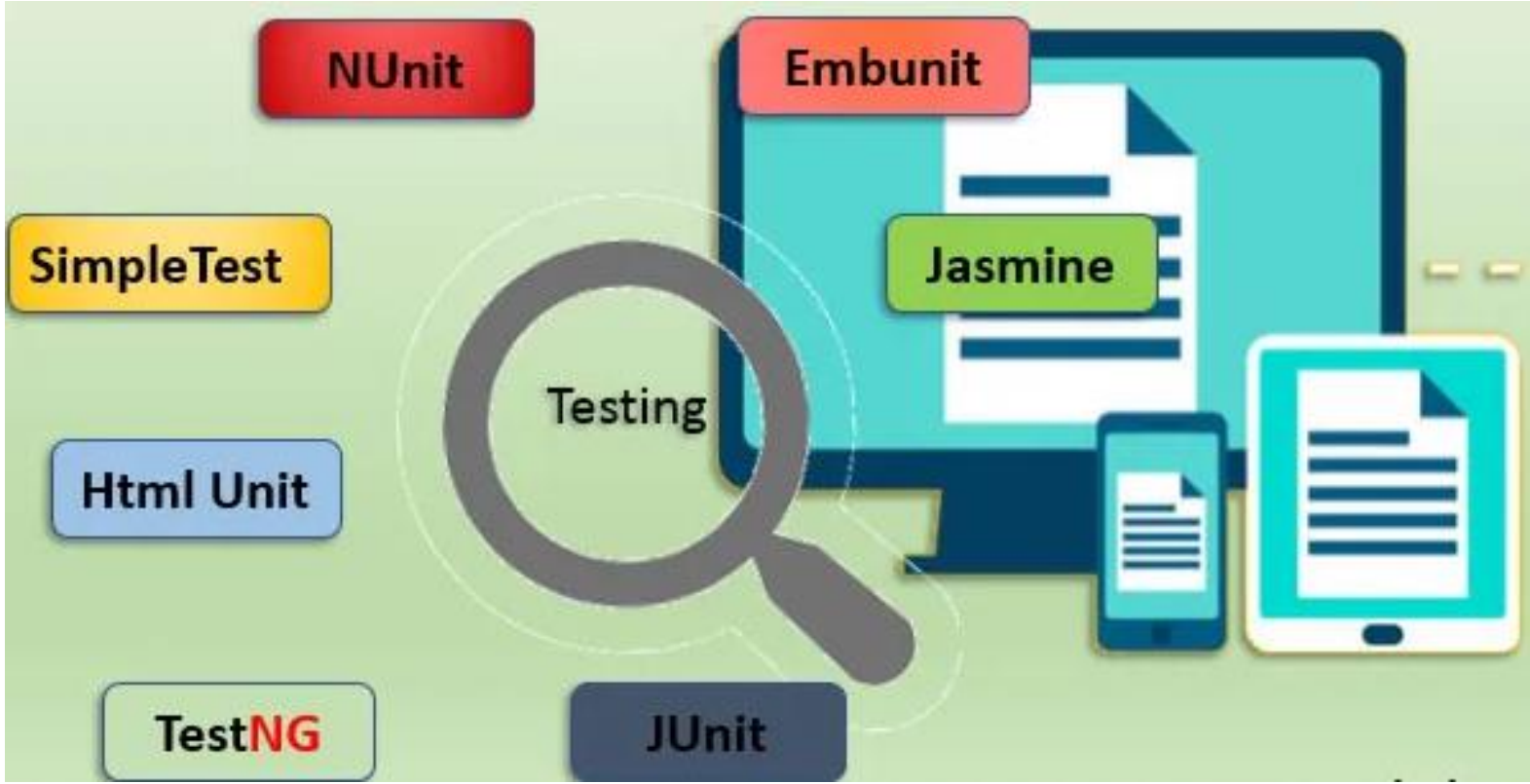
Unit Testing Tools

- Nunit
- TestNG
- Xunit
- Bunit
- Selenium
- Postman
- Swagger
- ...

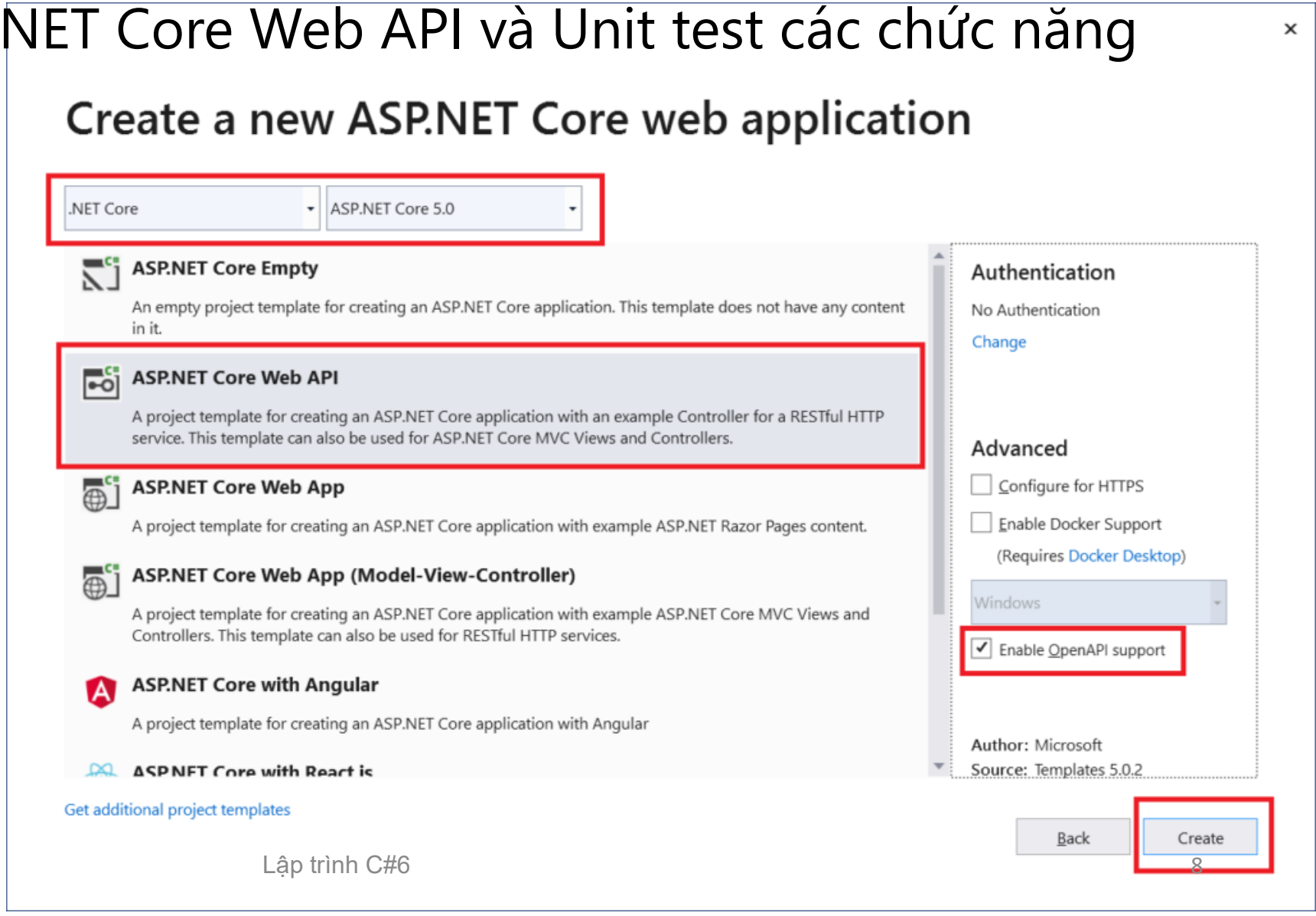


Unit Testing Tools

- Nunit
- TestNG
- Xunit
- Bunit
- Moq
- Selenium
- Postman
- Swagger
- …



- Tạo ứng dụng ASP.NET Core Web API và Unit test các chức năng toán học



interface & class for Maths Service

```
public interface IMathsService
{
    3 references | 0/1 passing
    double Add(double x1, double x2);
    3 references | 0/1 passing
    double Subtract(double x1, double x2);
    3 references | 0/1 passing
    double Multiply(double x1, double x2);
    5 references | 0/3 passing
    double Divide(double x1, double x2);
}
```

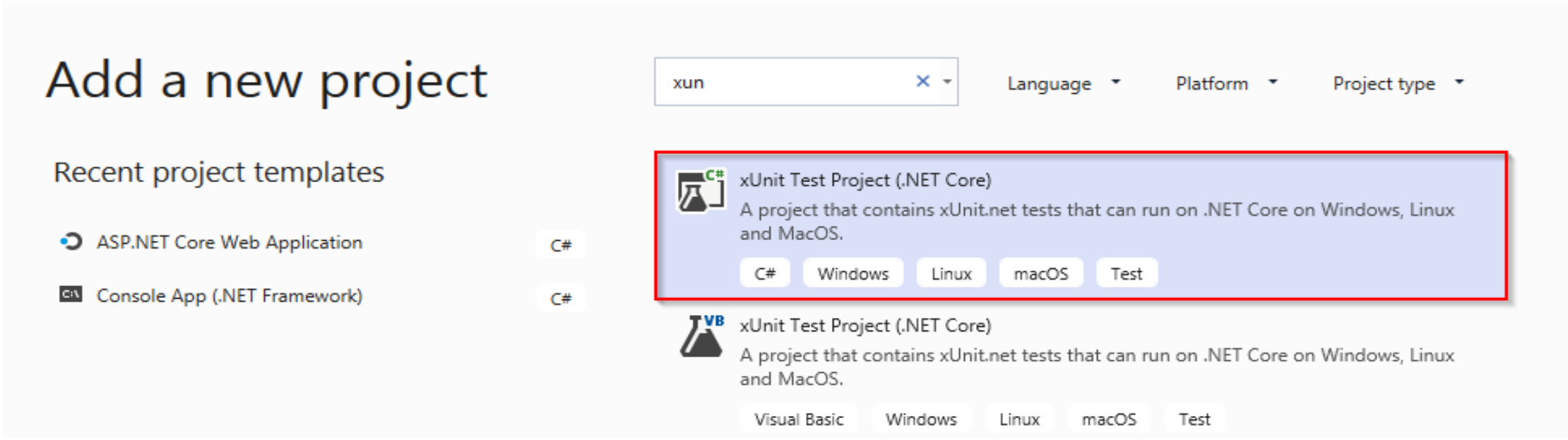
```
public class MathsService : IMathsService
{
    3 references | 0/1 passing
    public double Add(double x1, double x2)
    {
        return (x1 + x2);
    }
    5 references | 0/3 passing
    public double Divide(double x1, double x2)
    {
        if (x2 == 0)
        {
            throw new DivideByZeroException("x2 cannot be zero");
        }
        return (x1 / x2);
    }
    3 references | 0/1 passing
    public double Multiply(double x1, double x2)
    {
        return (x1 * x2);
    }
    3 references | 0/1 passing
    public double Subtract(double x1, double x2)
    {
        return (x1 - x2);
    }
}
```

Controller

```
[Route("api/[controller]")]
[ApiController]
1 reference
public class MathsController : ControllerBase
{
    private IMathsService _mathService = null;
    0 references
    public MathsController(IMathsService mathService)
    {
        _mathService = mathService;
    }
    [HttpPost]
    [Route("Add")]
    0 references
    public double Add(double x1, double x2)
    {
        return _mathService.Add(x1, x2);
    }
```

```
[HttpPost]
[Route("Divide")]
0 references
public double Divide(double x1, double x2)
{
    return _mathService.Divide(x1, x2);
}
[HttpPost]
[Route("Multiply")]
0 references
public double Multiply(double x1, double x2)
{
    return _mathService.Multiply(x1, x2);
}
[HttpPost]
[Route("Subtract")]
0 references
public double Subtract(double x1, double x2)
{
    return _mathService.Subtract(x1, x2);
}
```

❑ Tạo Testing Project



xunit by James Newkirk,Brad Wilson
xUnit.net is a developer testing framework, built to support Test Driven Development.

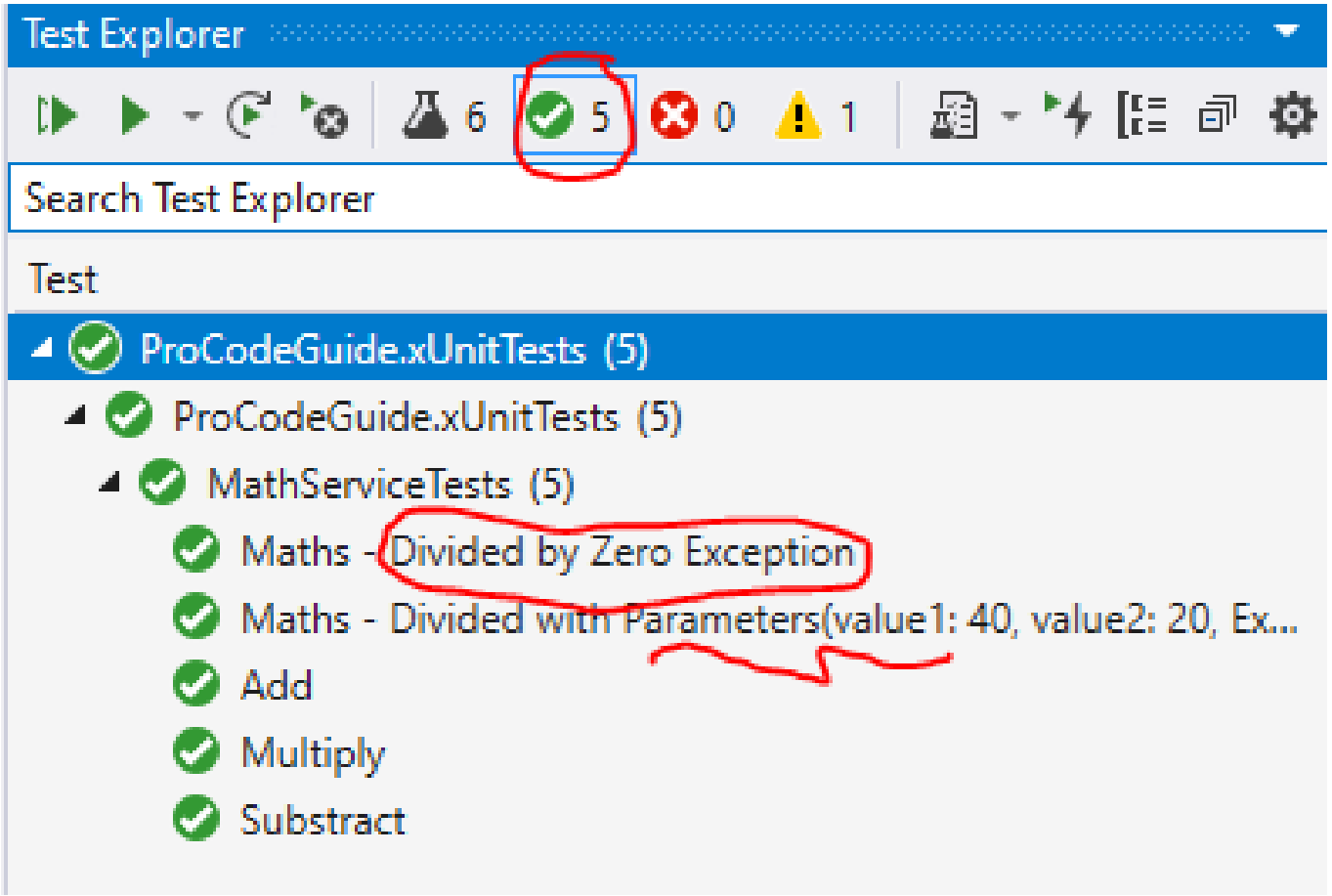
xunit.runner.visualstudio by James Newkirk, Brad Wilson
Visual Studio 2012+ Test Explorer runner for the xUnit.net framework. Capable of running xUnit.net v1.9.2 and v2.0+ tests. Supports .NET 2.0 or later, .NET Core 1.0 or later, and Universal Windows 10.0 or later.

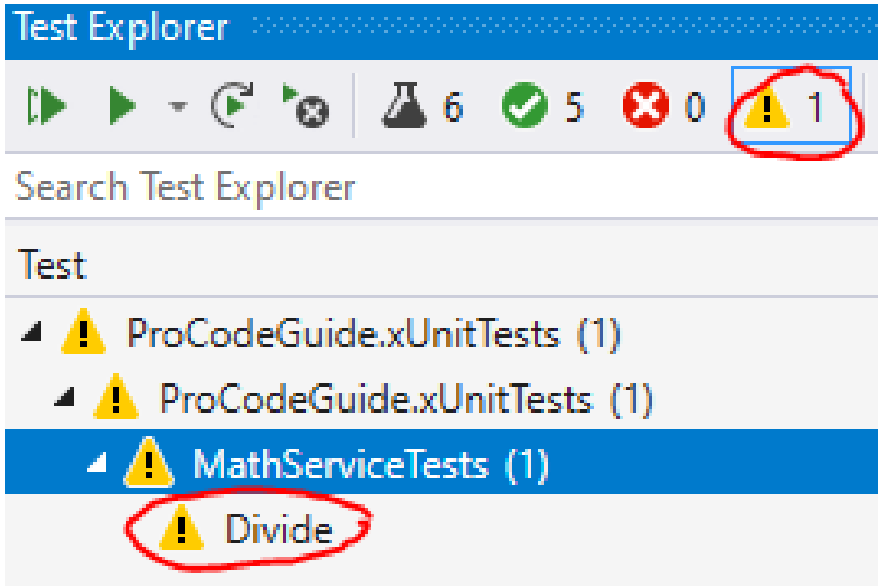
Tạo lớp testcase

```
private MathsService _unitUnderTesting = null;
0 references
public MathServiceTests()
{
    if (_unitUnderTesting == null)
    {
        _unitUnderTesting = new MathsService();
    }
}
[Fact]
0 references
public void Add()
{
    double x1 = 5;
    double x2 = 8;
    double expected = 13;

    var actual = _unitUnderTesting.Add(x1, x2);

    Assert.Equal(expected, actual, 0);
}
```

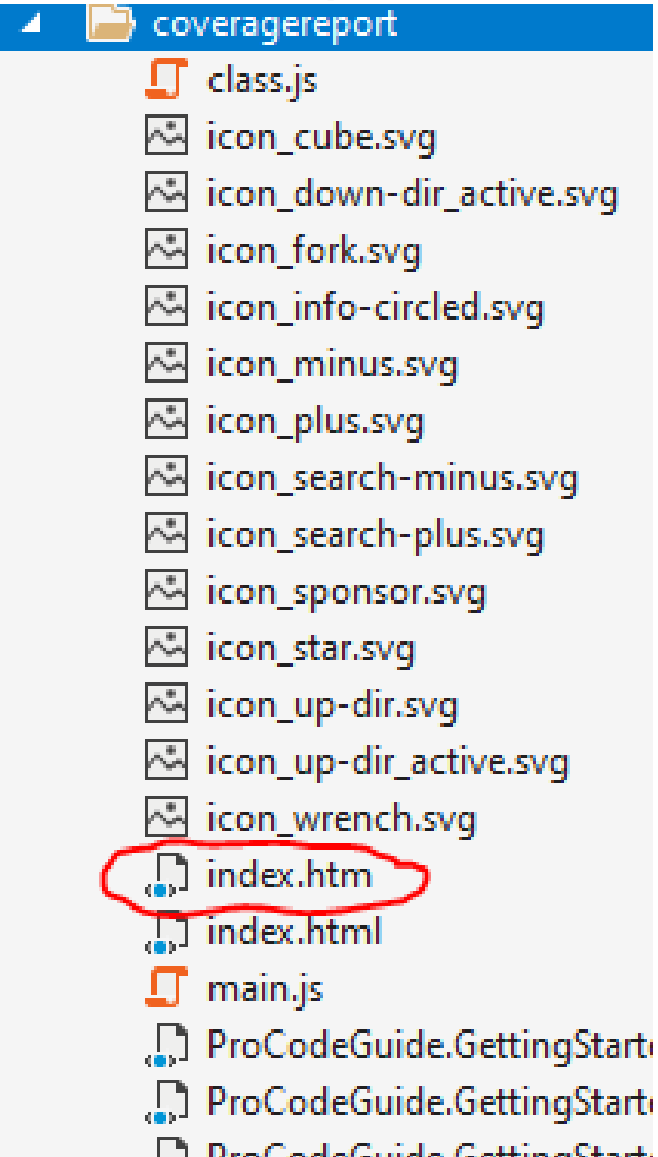
```
[Fact(Skip = "Dont run now")]
0 references
public void Divide()
{
    double x1 = 100;
    double x2 = 10;
    double expected = 10;
    var actual = _unitUnderTesting.Divide(x1, x2);
    Assert.Equal(expected, actual, 0);
}
[Fact(DisplayName = "Maths - Divided by Zero Exception")]
0 references
public void DivideByZeroException()
{
    double x1 = 100;
    double x2 = 0;
    Action act = () => _unitUnderTesting.Divide(x1, x2);
    Assert.Throws<DivideByZeroException>(act);
}
[Theory(DisplayName = "Maths - Divided with Parameters")]
[InlineData(40, 20, 2)]
0 references
public void DivideWithParameter(double value1, double value2, double ExpectedValue)
{
    double x1 = value1;
    double x2 = value2;
    double expected = ExpectedValue;
    var actual = _unitUnderTesting.Divide(x1, x2);
    Assert.Equal(expected, actual, 0);
}
```

Run lớp testcase





Run lớp testcase



Generated on:	13-03-2021 - 00:29:58
Parser:	CoberturaParser
Assemblies:	1
Classes:	5
Files:	5
Covered lines:	15
Uncovered lines:	56
Coverable lines:	71
Total lines:	183
Line coverage:	21.1% (15 of 71)
Covered branches:	2
Total branches:	4
Branch coverage:	50% (2 of 4)

isk Hotspots

No risk hotspots found.

overage

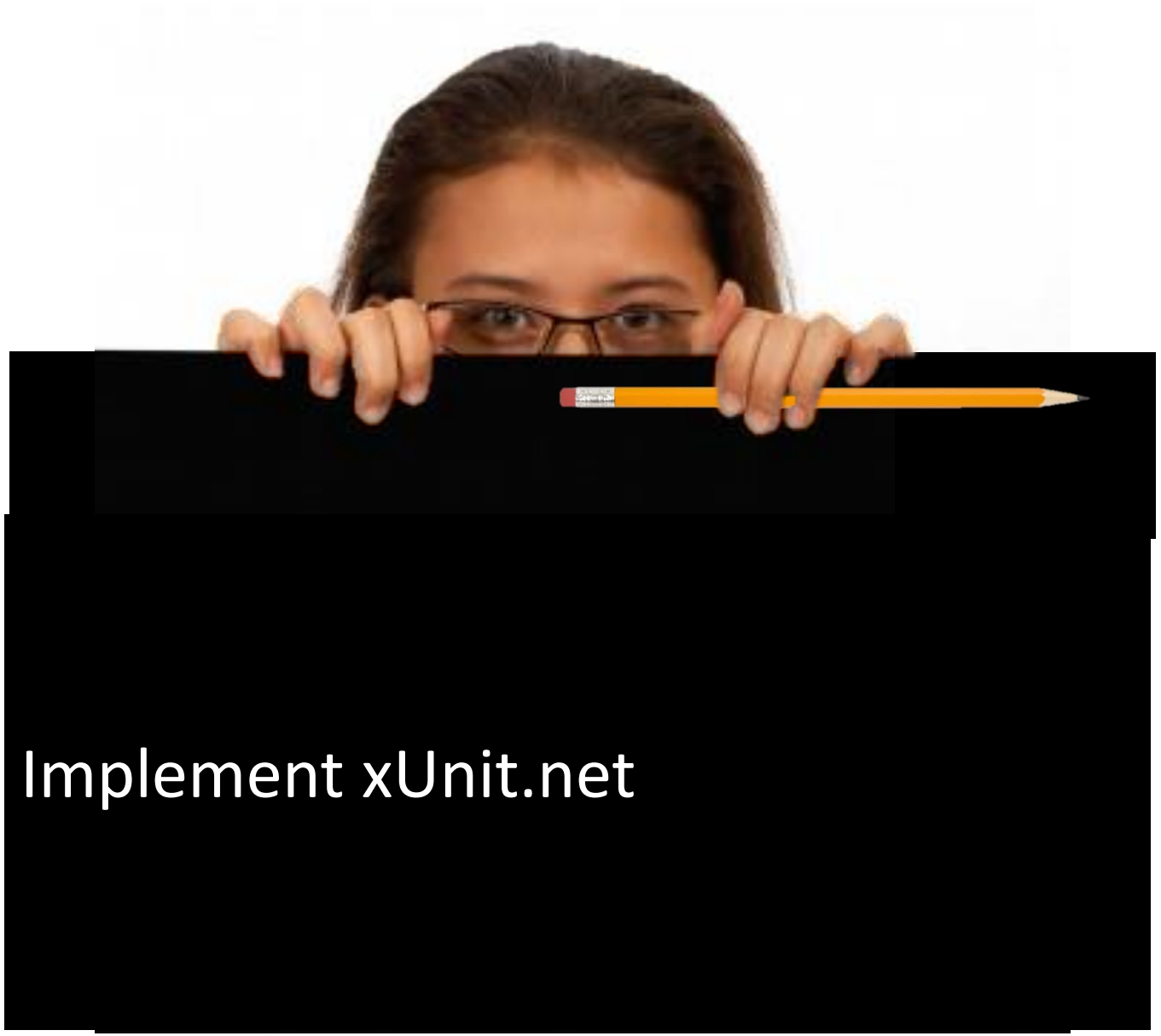
Collapse all | Expand all

By assembly
Grouping:

Filter:

Name	Covered	Uncovered	Coverable	Total	Line coverage	Covered	Total	Branch coverage
ProCodeGuide.GettingStarted.xUnit	15	56	71	183	21.1%	2	4	50%
ProCodeGuide.GettingStarted.xUnit.Controllers.MathsController	0	17	17	49	0%	0	0	
ProCodeGuide.GettingStarted.xUnit.Program	0	8	8	26	0%	0	0	
ProCodeGuide.GettingStarted.xUnit.Services.MathsService	15	0	15	34	100%	2	2	100%
ProCodeGuide.GettingStarted.xUnit.Startup	0	27	27	59	0%	0	2	0%
ProCodeGuide.GettingStarted.xUnit.WeatherForecast	0	4	4	15	0%	0	0	

DEMO 8.1



FPT Education

FPT POLYTECHNIC

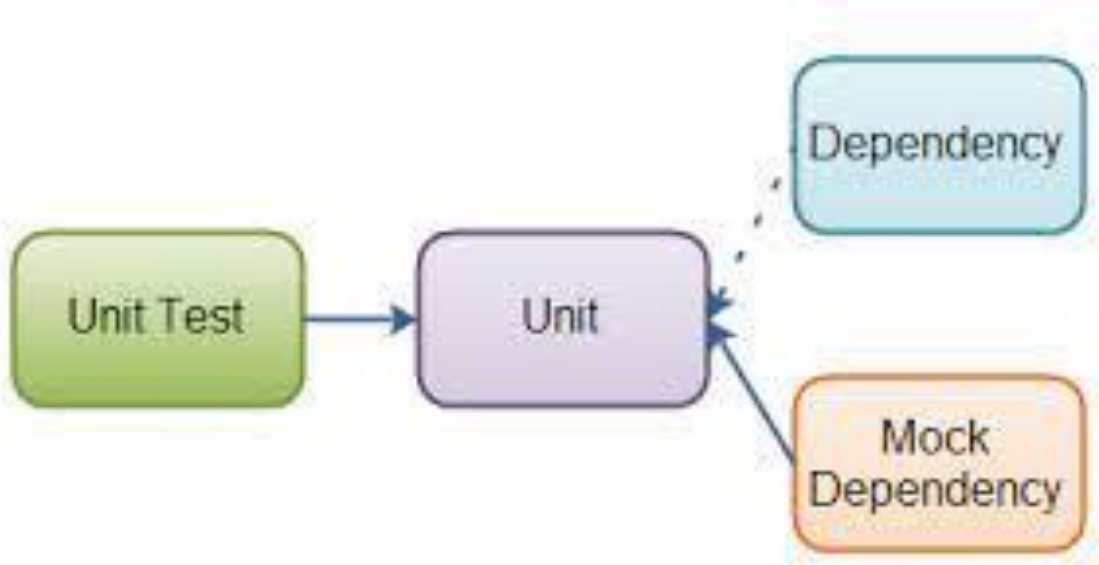
LẬP TRÌNH C# 6

BÀI 8: UNIT TEST ASP.NET CORE

P8.2

MOQ LIBRARY-TESTING MVC CONTROLLERS

- Mock Object
- Mock Unit Test



MOQ LIBRARY-TESTING MVC CONTROLLERS

DI

```
public class Employee
{                                   1
    5 references
    public Guid Id { get; set; }
    [Required(ErrorMessage = "Name is required")]
    11 references | 2/2 passing
    public string Name { get; set; }
    [Required(ErrorMessage = "Age is required")]
    15 references | 4/4 passing
    public int Age { get; set; }
    [Required(ErrorMessage = "Account number is required")]
    15 references | 3/3 passing
    public string AccountNumber { get; set; }
}
```

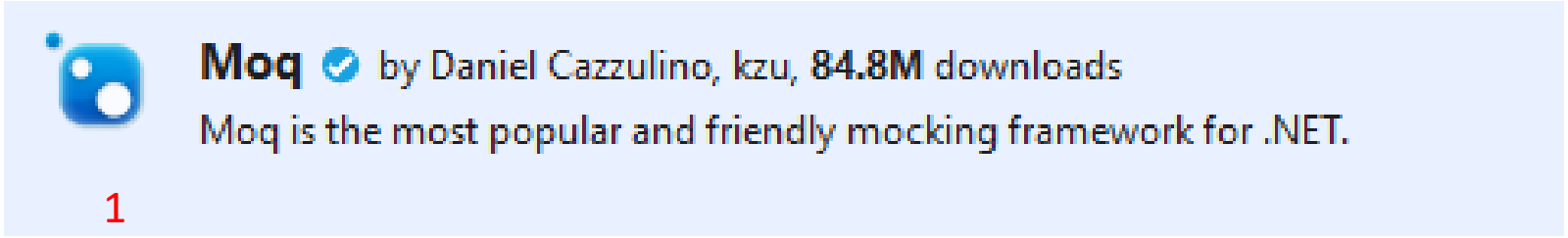
```
public interface IEmployeeRepository
{                                   2
    3 references | 1/1 passing
    IEnumerable<Employee> GetAll();
    1 reference
    Employee GetEmployee(Guid id);
    5 references | 2/2 passing
    void CreateEmployee(Employee employee);
}
```

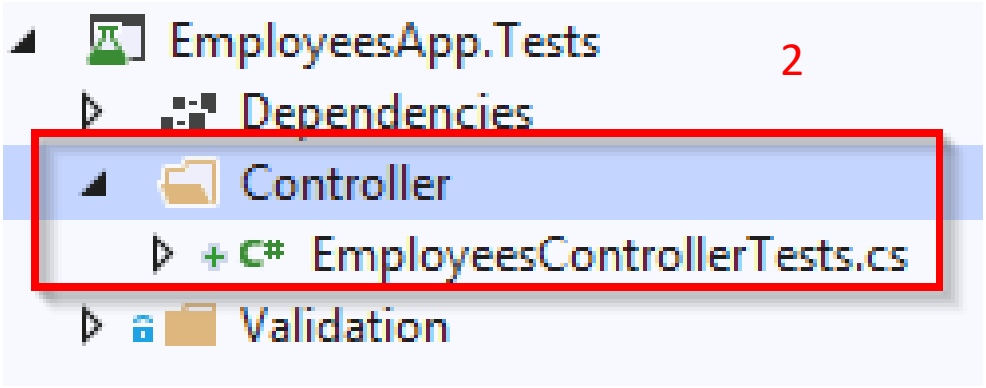
```
public class EmployeeRepository : IEmployeeRepository
{                                                   2
    3 references | 1/1 passing
    public IEnumerable<Employee> GetAll() => _context.Employees.ToList();
    5 references | 2/2 passing
    public void CreateEmployee(Employee employee)...
```

```
public class EmployeesController : Controller
{
    private readonly IEmployeeRepository _repo;
    private readonly AccountNumberValidation _validation;

                                                    3
    1 reference | 0 exceptions
    public EmployeesController(IEmployeeRepository repo)
    {
        _repo = repo;
        _validation = new AccountNumberValidation();
    }
```

MOQ LIBRARY-TESTING MVC CONTROLLERS

- Moq library





- Tạo Mock Object

```
private readonly Mock<IEmployeeRepository> _mockRepo;
private readonly EmployeesController _controller;
0 references
public EmployeesControllerTests()
{
    _mockRepo = new Mock<IEmployeeRepository>();
    _controller = new EmployeesController(_mockRepo.Object);
}
```

3

MOQ LIBRARY-TESTING MVC CONTROLLERS

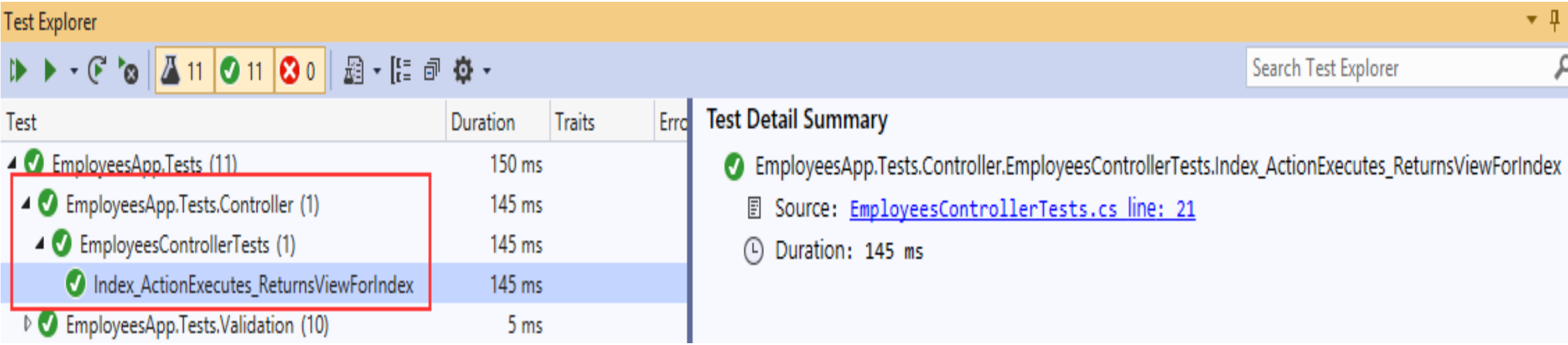
Testing Index Action

```
[Fact]
0 references
public void Index_ActionExecutes_ReturnsViewForIndex()
{
    var result = _controller.Index();

    Assert.IsType<ViewResult>(result);
}
```

2

```
public IActionResult Index()
{
    var employees = _repo.GetAll();
    return View(employees);
}
```

1



MOQ LIBRARY-TESTING MVC CONTROLLERS

Testing Index Action

```
[Fact]
0 references
public void Index_ActionExecutes_ReturnsExactNumberOfEmployees()
{
    _mockRepo.Setup(repo => repo.GetAll())
        .Returns(new List<Employee>() { new Employee(), new Employee() });

    var result = _controller.Index();

    var viewResult = Assert.IsType<ViewResult>(result);
    var employees = Assert.IsType<List<Employee>>(viewResult.Model);
    Assert.Equal(2, employees.Count);
}
```

```
public IActionResult Index()
{
    var employees = _repo.GetAll();
    return View(employees);
}
```

1

2

Test Explorer

12 12 0

Search Test Explorer

Test	Duration	Traits
EmployeesApp.Tests (12)	157 ms	
EmployeesApp.Tests.Controller (2)	152 ms	
EmployeesControllerTests (2)	152 ms	
Index_ActionExecutes_ReturnsExactNumberOfEmployees	150 ms	
Index_ActionExecutes_ReturnsViewForIndex	2 ms	
EmployeesApp.Tests.Validation (10)	5 ms	

Test Detail Summary

EmployeesApp.Tests.Controller.EmployeesControllerTests.Index_ActionExecutes_ReturnsExactNumberOfE

Source: EmployeesControllerTests.cs line: 31

Duration: 150 ms

MOQ LIBRARY-TESTING MVC CONTROLLERS

Testing Create Actions

```
public IActionResult Create()
{
    return View();
}
```

1

```
[Fact]
0 references
public void Create_ActionExecutes_ReturnsViewForCreate()
{
    var result = _controller.Create();
    Assert.IsType<ViewResult>(result);
}
```

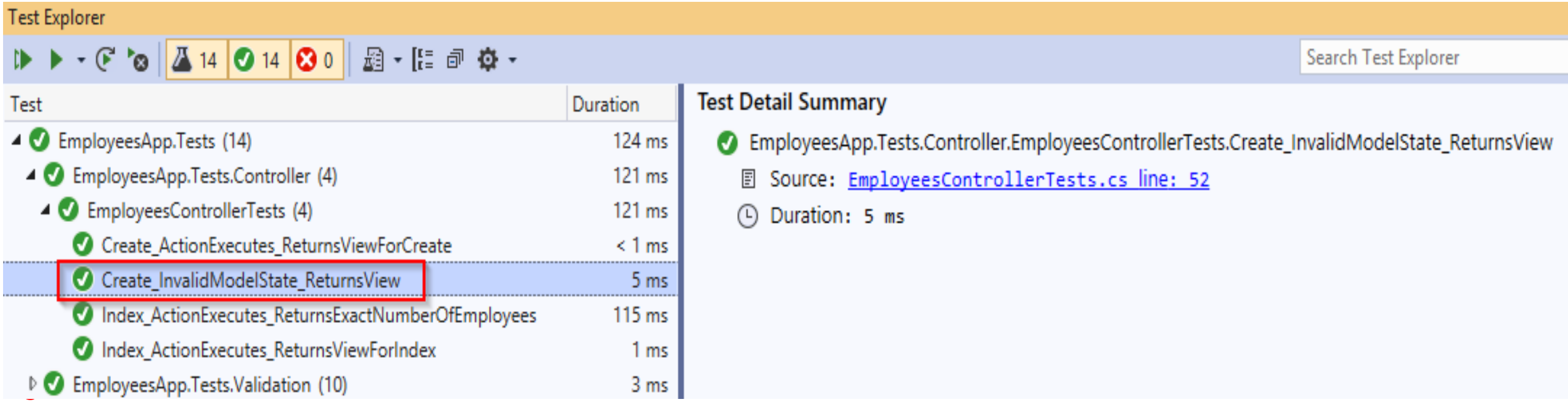
2

```
[Fact]
0 references
public void Create_InvalidModelState_ReturnsView()
{
    _controller.ModelState.AddModelError("Name", "Name is required");

    var employee = new Employee { Age = 25, AccountNumber = "255-8547963214-41" };

    var result = _controller.Create(employee);

    var viewResult = Assert.IsType<ViewResult>(result);
    var testEmployee = Assert.IsType<Employee>(viewResult.Model);
    Assert.Equal(employee.AccountNumber, testEmployee.AccountNumber);
    Assert.Equal(employee.Age, testEmployee.Age);
}
```

3

MOQ LIBRARY-TESTING MVC CONTROLLERS

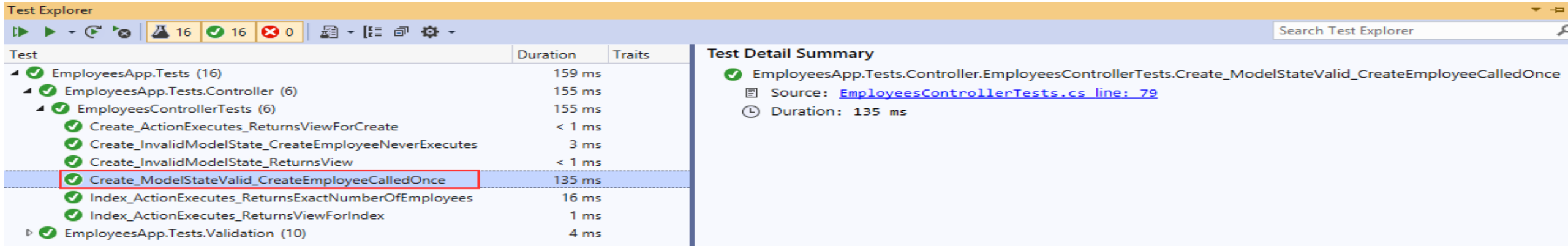
Testing Create Actions



MOQ LIBRARY-TESTING MVC CONTROLLERS

- Testing Create Actions

```
[Fact]
0 references
public void Create_ModelStateValid_CreateEmployeeCalledOnce()
{
    Employee emp = null;
    _mockRepo.Setup(r => r.CreateEmployee(It.IsAny<Employee>()))
        .Callback<Employee>(x => emp = x);
    var employee = new Employee
    {
        Name = "Thepv",
        Age = 36,
        AccountNumber = "123-5435789603-21"
    };
    _controller.Create(employee);
    _mockRepo.Verify(x => x.CreateEmployee(It.IsAny<Employee>()), Times.Once);
    Assert.Equal(emp.Name, employee.Name);
    Assert.Equal(emp.Age, employee.Age);
    Assert.Equal(emp.AccountNumber, employee.AccountNumber);
}
```



DEMO 8.2

FPT Education

FPT POLYTECHNIC

LẬP TRÌNH C# 6

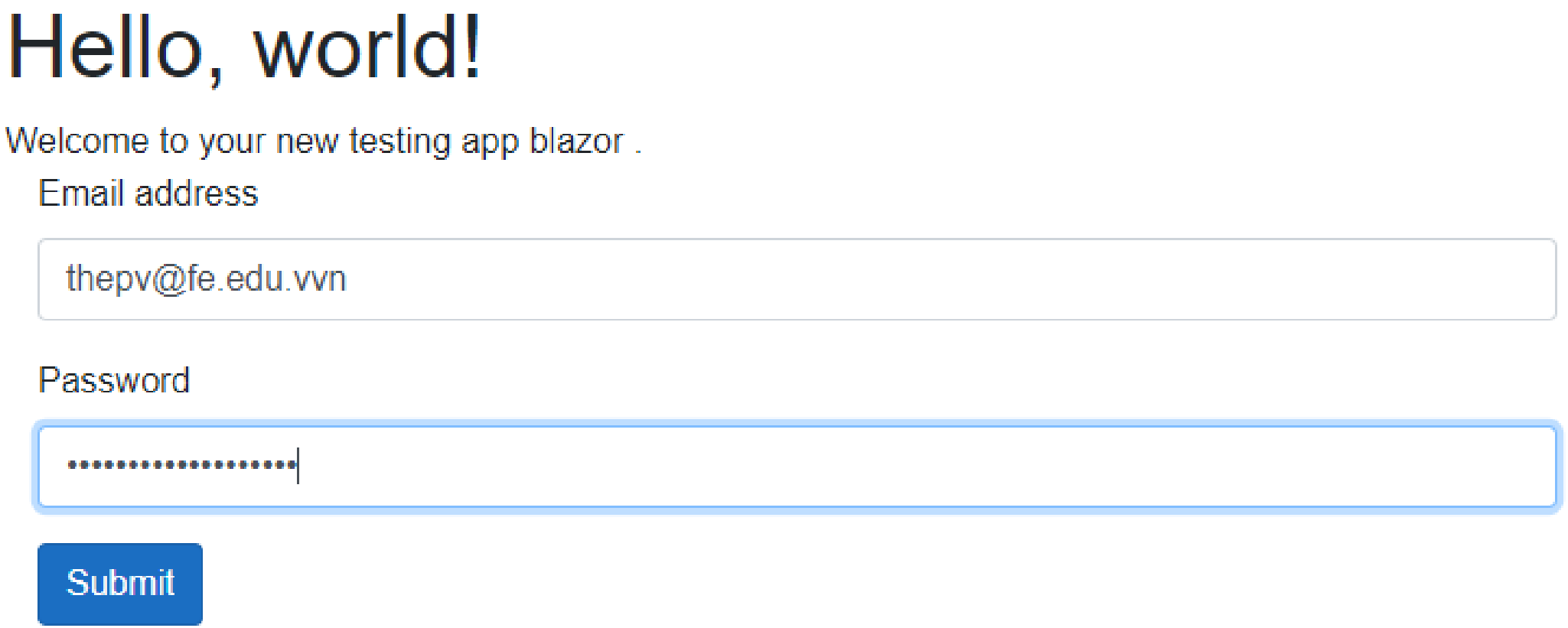
BÀI 8: UNIT TEST ASP.NET CORE

P8.3

BLAZOR UNIT TESTING

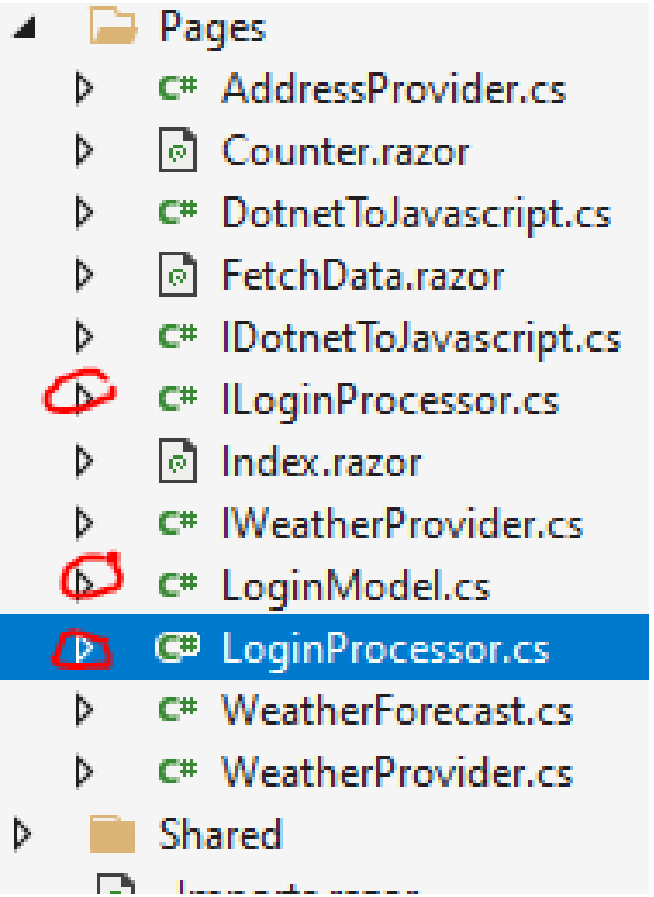
- Nunit
- bUnit

Tạo dự án Blazor



BLAZOR UNIT TESTING

model



1

```
public class LoginModel
{
    5 references
    public string Email { get; set; }
    5 references
    public string Password { get; set; }
}
```

2

```
public interface ILoginProcessor
{
    5 references | 1/2 passing
    bool Login(string email, string password);
}
```

3

```
public class LoginProcessor : ILoginProcessor
{
    5 references | 1/2 passing
    public bool Login(string email, string password)
    {
        return true;
    }
}
```

- Index.razor

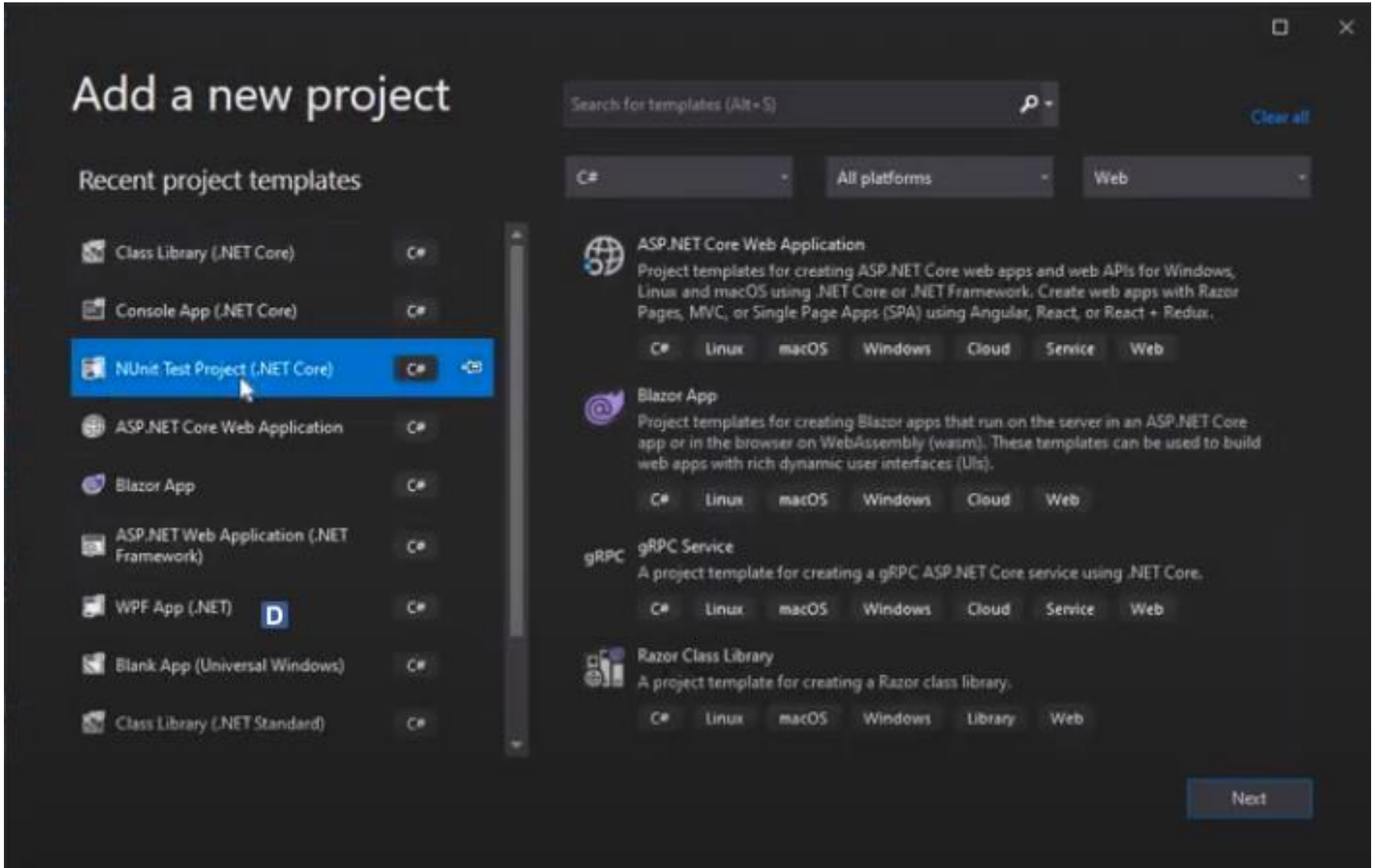
```
@page "/"
<h1>Hello, world!</h1>
Welcome to your new testing app blazor .
<div class="container-fluid">
    <div class="row">
        <div class="col-md-12">
            <form role="form">
                <div class="form-group">

                    <label for="exampleInputEmail1">
                        Email address
                    </label>
                    <input type="email" class="form-control" id="agentEmail" @bind-value="login.Email" />
                </div>
                <div class="form-group">

                    <label for="exampleInputPassword1">
                        Password
                    </label>
                    <input type="password" class="form-control" id="agentPassword" @bind-value="login.Password" />
                </div>
                <button type="button" class="btn btn-primary" @onclick="Login">
                    Submit
                </button>
            </form>
            <div class="alert alert-primary" hidden="@(errorMessage == string.Empty)" role="alert">
                @errorMessage
            </div>
```

BLAZOR UNIT TESTING

- Index.razor.cs

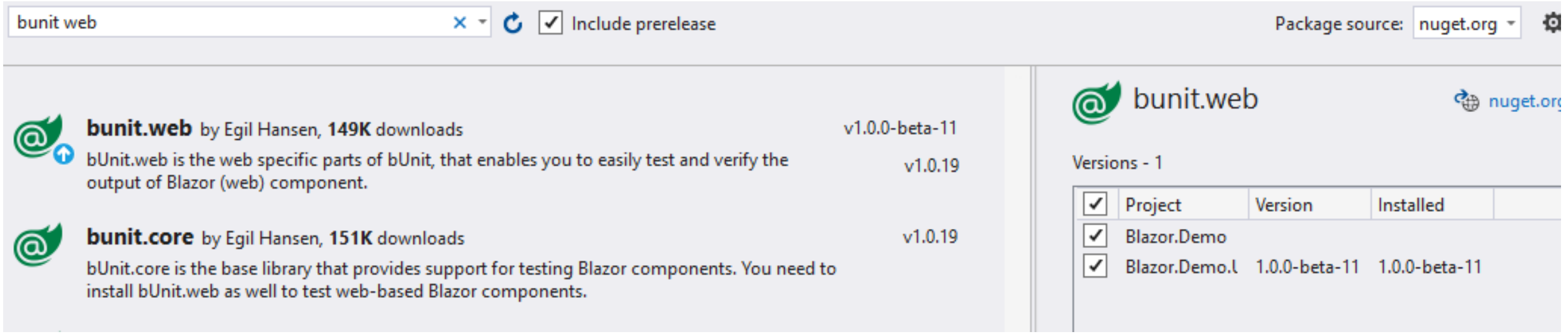
```
partial class Index
{
    [Inject]
    1 reference
    private ILoginProcessor LoginProcessor { get; set; }
    private LoginModel login = new();
    private string errorMessage = string.Empty;
    1 reference
    private bool ValidateEmail()
    {
        return !string.IsNullOrEmpty(login.Email);
    }
    1 reference
    private bool ValidatePassword()
    {
        return !string.IsNullOrEmpty(login.Password);
    }
    1 reference
    private async Task Login()
    {
        if(ValidateEmail() && ValidatePassword())
        {
            if(!LoginProcessor.Login(login.Email, login.Password))
            {
                errorMessage = "Invalid login";
            }
        }
        else
        {
            errorMessage = "Email/Password Invalid";
        }
    }
```

- Tạo dự án kiểm thử blazor

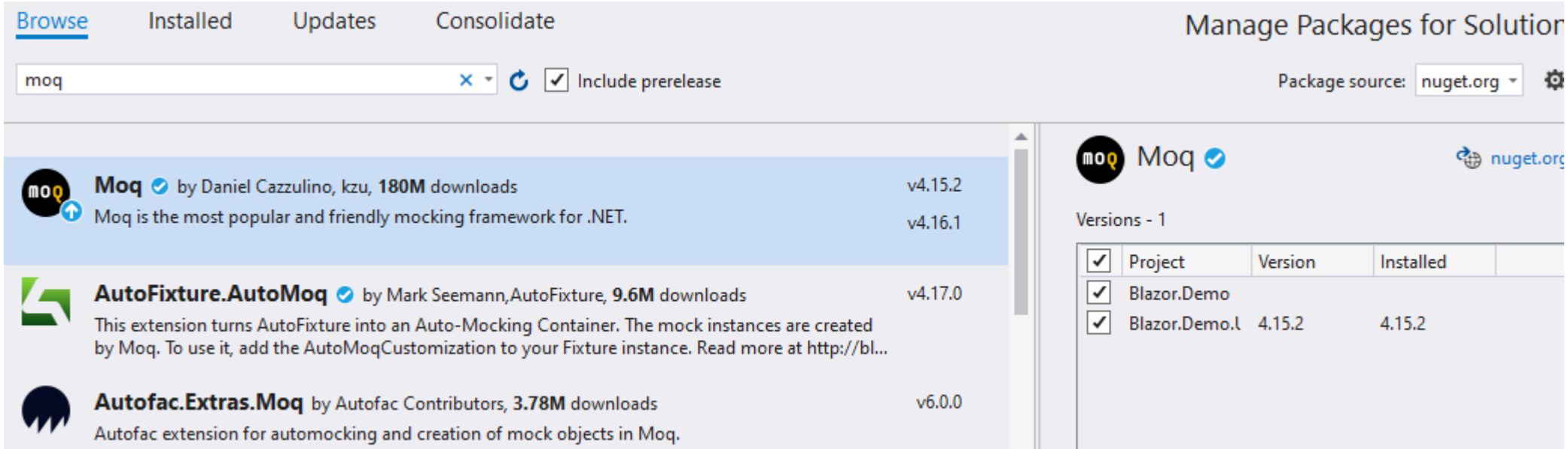


BLAZOR UNIT TESTING

Cài bunit



Cài Moq



- Khởi tạo bunit và mockup object

```
public class Tests
{
    private Bunit.TestContext testContext;
    private Mock<ILoginProcessor> loginProcessor;

    [SetUp]
    0 references
    public void Setup()
    {
        testContext = new Bunit.TestContext();
        loginProcessor = new Mock<ILoginProcessor>();
    }

    [TearDown]
    0 references
    public void Teardown()
    {
        testContext.Dispose();
    }
```

Viết test method No_Email_Password_Condition

```
[Test]
0 references
public void No_Email_Password_Condition()
{
    testContext.Services.AddScoped(x => loginProcessor.Object);

    var component = testContext.RenderComponent<Index>();

    Assert.IsTrue(component.Markup.Contains("Welcome to your new testing app blazor"));

    var buttons = component.FindAll("button");
    Assert.AreEqual(1, buttons.Count);

    var submit = buttons.FirstOrDefault(b => b.OuterHtml.Contains("Submit"));
    Assert.IsNotNull(submit);

    submit.Click();
    loginProcessor.Verify(l => l.Login(It.IsAny<string>(), It.IsAny<string>()), Times.Never);
}
```

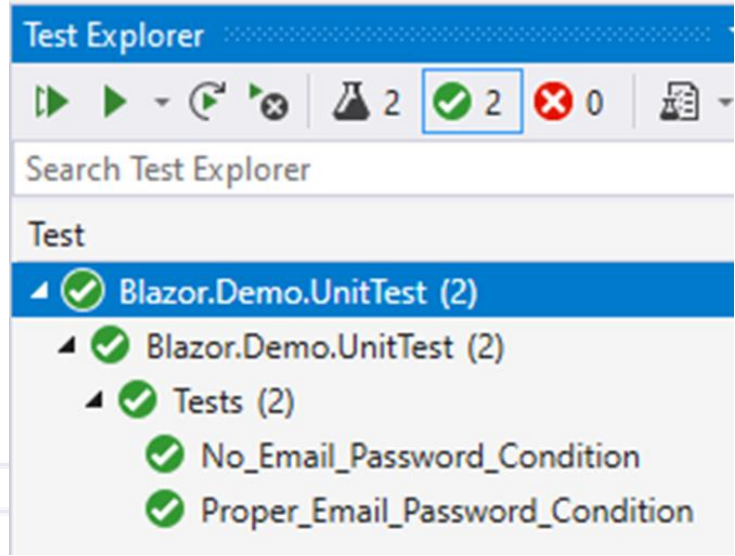
BLAZOR UNIT TESTING

Viết test method Proper_Email_Password_Condition

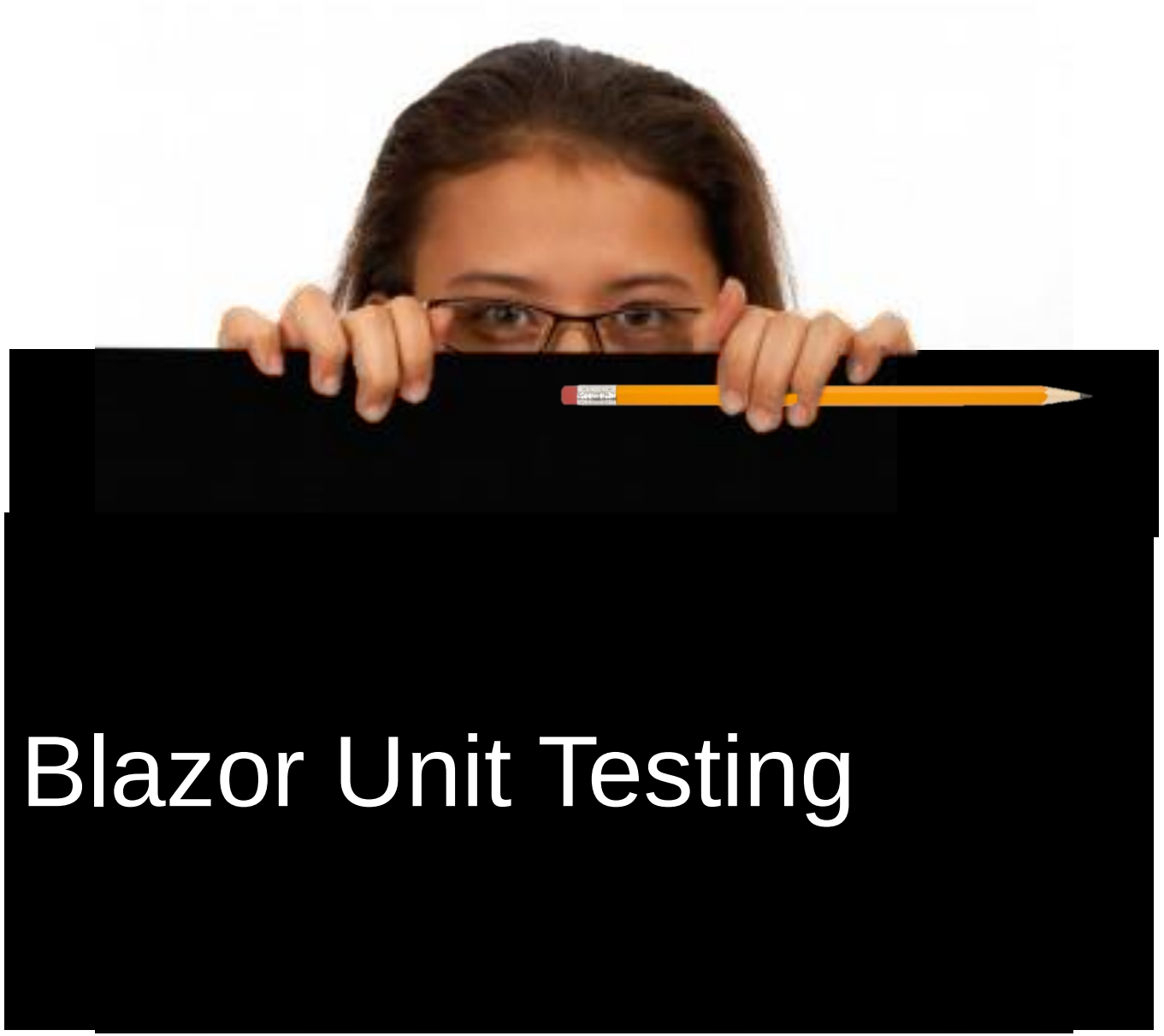
Index.razor.cs

```
private async Task Login()
{
    if(ValidateEmail() && ValidatePassword())
    {
        if(!LoginProcessor.Login(login.Email, login.Password))
        {
            errorMessage = "Invalid login";
        }
    }
```

```
[Test]
0 references
public void Proper_Email_Password_Condition()
{
    testContext.Services.AddScoped(x => loginProcessor.Object);
    var component = testContext.RenderComponent<Index>();
    var buttons = component.FindAll("button");
    var submit = buttons.FirstOrDefault(b => b.OuterHtml.Contains("Submit"));
    var email = component.Find("#agentEmail");
    email.Change("testemail");
    var password = component.Find("#agentPassword");
    password.Change("TestPassword");
    submit.Click();
    loginProcessor.Verify(l => l.Login(It.IsAny<string>(), It.IsAny<string>()), Times.Once);
    var alert = component.Find("div.alert");
    Assert.AreEqual("Invalid login", alert.InnerHtml);
}
```



DEMO 8.3



Tổng kết bài học

- Tổng quan Unit test .Net
- Implement xUnit.net
- Mock object - Moq Library
- Blazor Unit Testing

KẾT THÚC