Testing ASP.NET Core Applications

Prepared for Vth semester DDU-CE students
2022-23 WAD

Apurva A Mehta

Introduction

- Unit testing is a form of testing in which individual components are isolated from the rest of the application so their behavior can be thoroughly validated.
- ASP.NET Core has been designed to make it easy to create unit tests, and there is support for a wide range of unit testing frameworks.

Windows PowerShell

```
PS D:\> dotnet new web --no-https --output Testing/SimpleApp --framework netcoreapp3.1
The template "ASP.NET Core Empty" was created successfully.

Processing post-creation actions...
Running 'dotnet restore' on Testing/SimpleApp\SimpleApp.csproj...
  Determining projects to restore...
  Restored D:\Testing\SimpleApp\SimpleApp.csproj (in 172 ms).

Restore succeeded.

PS D:\> dotnet new sln -o Testing
The template "Solution File" was created successfully.
PS D:\> dotnet sln Testing add Testing/SimpleApp
Project `SimpleApp\SimpleApp.csproj` added to the solution.
PS D:\>
```

This PC > D (D:) > Testing

Name	Date modified	Type	Size
SimpleApp	04-09-2020 03:41 ...	File folder	
Testing.sln	04-09-2020 03:41 ...	Microsoft Visual St...	2 KB

This PC > D (D:) > Testing > SimpleApp

Name	Date modified	Type	Size
obj	04-09-2020 03:41 ...	File folder	
Properties	04-09-2020 03:41 ...	File folder	
appsettings.Development.json	04-09-2020 03:41 ...	JSON File	1 KB
appsettings.json	04-09-2020 03:41 ...	JSON File	1 KB
Program.cs	04-09-2020 03:41 ...	Visual C# Source F...	1 KB
SimpleApp.csproj	04-09-2020 03:41 ...	Visual C# Project fi...	1 KB
Startup.cs	04-09-2020 03:41 ...	Visual C# Source F...	2 KB

Startup.cs | Output

SimpleApp | SimpleApp.Startup | Configure(IApplicationBuilder

```
7   using Microsoft.AspNetCore.Http;
8   using Microsoft.Extensions.DependencyInjection;
9   using Microsoft.Extensions.Hosting;
10
11  namespace SimpleApp
12  {
        1 reference
13      public class Startup
14      {
15          // This method gets called by the runtime. Use this method to add services
16          // For more information on how to configure your application, visit https:/
            0 references
17          public void ConfigureServices(IServiceCollection services)
18          {
19              services.AddControllersWithViews();
20          }
21
22          // This method gets called by the runtime. Use this method to configure the
            0 references
23          public void Configure(IApplicationBuilder app, IWebHostEnvironment env)
24          {
25              if (env.IsDevelopment())
```

Startup.cs | Output

SimpleApp | SimpleApp.Startup | ConfigureServices(IServiceColle

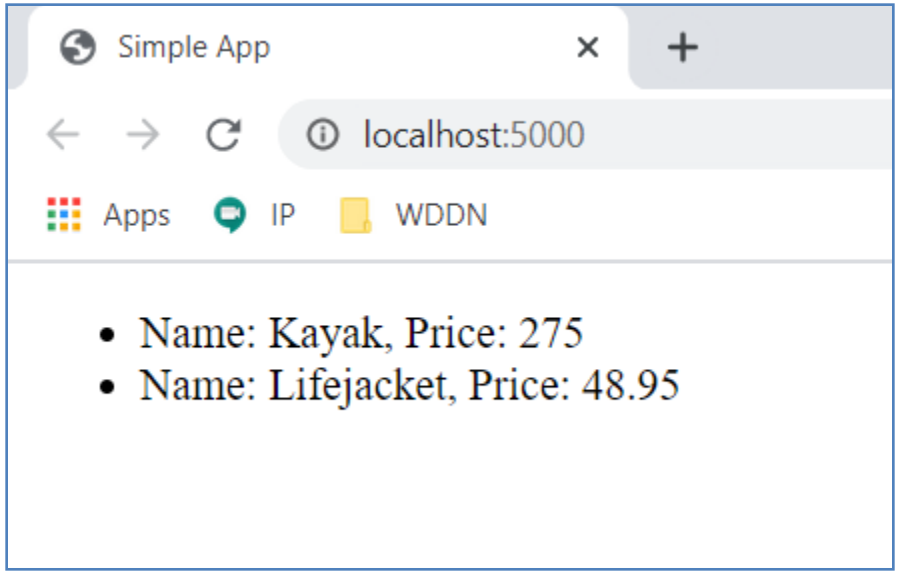
```
21
22          // This method gets called by the runtime. Use this method to configure the
            0 references
23          public void Configure(IApplicationBuilder app, IWebHostEnvironment env)
24          {
25              if (env.IsDevelopment())
26              {
27                  app.UseDeveloperExceptionPage();
28              }
29
30              app.UseRouting();
31
32              app.UseEndpoints(endpoints =>
33              {
34                  endpoints.MapDefaultControllerRoute();
35                  //endpoints.MapGet("/", async context =>
36                  //{
37                  //    await context.Response.WriteAsync("Hello World!");
38                  //});
39              });
40          }
41      }
42  }
```

```
namespace SimpleApp.Models
{
    6 references
    public class Product
    {
        2 references
        public string Name { get; set; }
        2 references
        public decimal? Price { get; set; }
        0 references
        public static Product[] GetProducts()
        {
            Product kayak = new Product
            {
                Name = "Kayak",
                Price = 275M
            };
            Product lifejacket = new Product
            {
                Name = "Lifejacket",
                Price = 48.95M
            };
            return new Product[] { kayak, lifejacket };
        }
    }
}
```

```
namespace SimpleApp.Controllers
{
    0 references
    public class HomeController : Controller
    {
        0 references
        public IActionResult Index()
        {
            return View(Product.GetProducts());
        }
    }
}
```

Index.cshtml | HomeController.cs* | Product.cs | Startup.cs | Output

```
1   @using SimpleApp.Models
2   @model IEnumerable<Product>
3   @{ Layout = null; }
4   <!DOCTYPE html>
5   <html>
6   <head>
7       <meta name="viewport" content="width=device-width" />
8       <title>Simple App</title>
9   </head>
10  <body>
11      <ul>
12          @foreach (Product p in Model)
13          {
14              <li>Name: @p.Name, Price: @p.Price</li>
15          }
16      </ul>
17  </body>
18  </html>
```


Simple App
localhost:5000
Apps
IP
WDDN
Name: Kayak, Price: 275
Name: Lifejacket, Price: 48.95

Creating a Unit Test Project

- For ASP.NET Core applications, you generally create a separate Visual Studio project to hold the unit tests, each of which is defined as a method in a C# class.
- Using a separate project means you can deploy your application without also deploying the tests.

Cont.

- The .NET Core SDK includes templates for unit test projects using three popular test tools.

Name	Description
mstest	This template creates a project configured for the MS Test framework, which is produced by Microsoft.
nunit	This template creates a project configured for the NUnit framework.
xunit	This template creates a project configured for the XUnit framework.

Windows PowerShell

```
PS D:\> cd .\Testing\
PS D:\Testing> dotnet new xunit -o SimpleApp.Tests
The template "xUnit Test Project" was created successfully.

Processing post-creation actions...
Running 'dotnet restore' on SimpleApp.Tests\SimpleApp.Tests.csproj...
  Determining projects to restore...
  Restored D:\Testing\SimpleApp.Tests\SimpleApp.Tests.csproj (in 544 ms).

Restore succeeded.

PS D:\Testing> dotnet sln add SimpleApp.Tests
Project `SimpleApp.Tests\SimpleApp.Tests.csproj` added to the solution.
PS D:\Testing> dotnet add SimpleApp.Tests reference SimpleApp
Reference `..\SimpleApp\SimpleApp.csproj` added to the project.
PS D:\Testing>
```

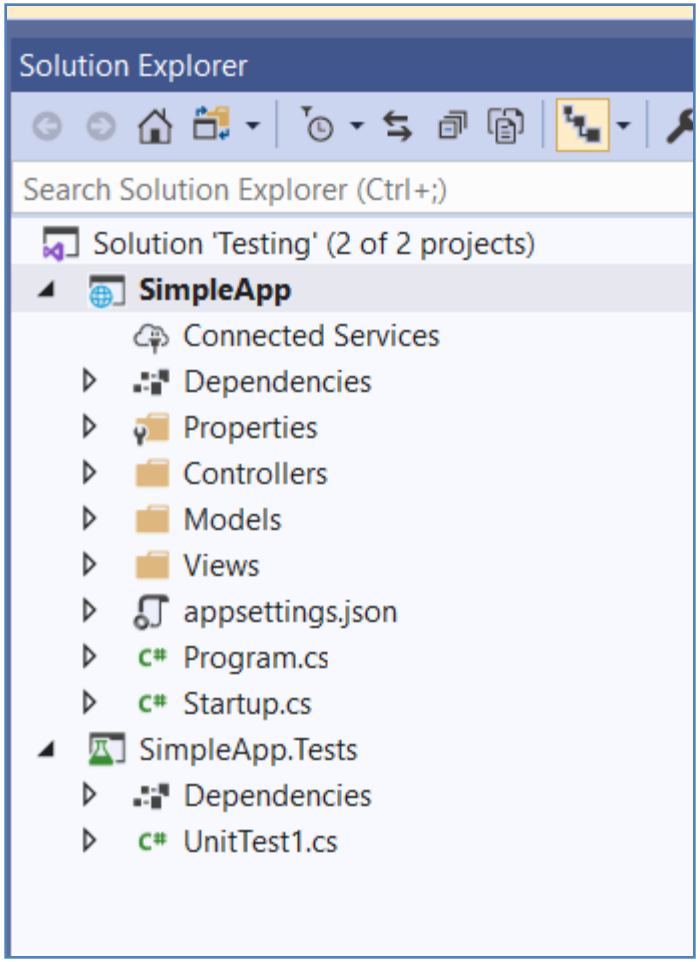
```
<head>
    <meta name="viewport" content="width=device-width" />
    <title>Simple App</title>
</head>
<body>
    <ul>
        @t
        {

        }
    </ul>
</body>
</html>
```

File Modification Detected

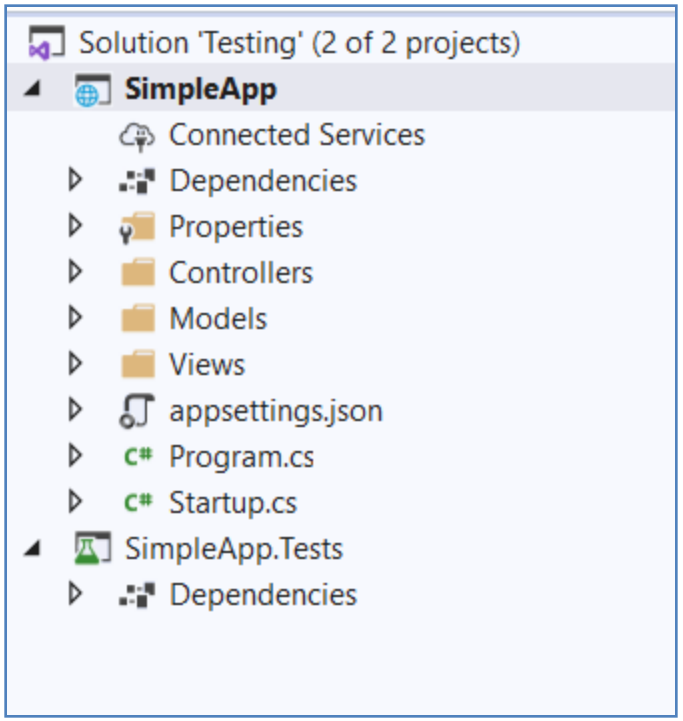
The solution 'Testing' has been modified outside the environment.

Press Reload to load the updated solution from disk.
Press Ignore to ignore the external changes. The changes will be used the next time you open the solution.

Reload | Ignore


Solution Explorer
Search Solution Explorer (Ctrl+;)
Solution 'Testing' (2 of 2 projects)
SimpleApp
Connected Services
Dependencies
Properties
Controllers
Models
Views
appsettings.json
Program.cs
Startup.cs
SimpleApp.Tests
Dependencies
UnitTest1.cs

```
PS D:\Testing> Remove-Item SimpleApp.Tests/UnitTest1.cs
PS D:\Testing>
```



ProductTests.cs | Output | Index.cshtml | HomeController.cs | Product.cs | Startup.cs

SimpleApp.Tests | SimpleApp.Tests.ProductTests | CanChangeProductPrice()

```
6
7   namespace SimpleApp.Tests
8   {
        0 references
9       public class ProductTests
10      {
11          [Fact]
            0 references
12          public void CanChangeProductName()
13          {
14              // Arrange
15              var p = new Product { Name = "Test", Price = 100M };
16              // Act
17              p.Name = "New Name";
18              //Assert
19              Assert.Equal("New Name", p.Name);
20          }
21          [Fact]
            0 references
22          public void CanChangeProductPrice()
23          {
24              // Arrange
25              var p = new Product { Name = "Test", Price = 100M };
26              // Act
27              p.Price = 200M;
```

ProductTests.cs | Output | Index.cshtml | HomeController.cs | Product.cs | Startup.cs

SimpleApp.Tests | SimpleApp.Tests.ProductTests | CanChangeProductPrice()

```
6
7   namespace SimpleApp.Tests
8   {
        0 references
9       public class ProductTests
10      {
11          [Fact]
            0 references
12          public void CanChangeProductName()
13          {
14              // Arrange
15              var p = new Product { Name = "Test", Price = 100M };
16              // Act
17              p.Name = "New Name";
18              //Assert
19              Assert.Equal("New Name", p.Name);
20          }
21          [Fact]
            0 references
22          public void CanChangeProductPrice()
23          {
24              // Arrange
25              var p = new Product { Name = "Test", Price = 100M };
26              // Act
27              p.Price = 200M;
```

ProductTests.cs | Output | Index.cshtml | HomeController.cs | Product.cs | Startup.cs

SimpleApp.Tests | SimpleApp.Tests.ProductTests | CanChangeProductPrice()

```
17            p.Name = "New Name";
18            //Assert
19            Assert.Equal("New Name", p.Name);
20        }
21        [Fact]
          0 references
22        public void CanChangeProductPrice()
23        {
24            // Arrange
25            var p = new Product { Name = "Test", Price = 100M };
26            // Act
27            p.Price = 200M;
28            //Assert
29            Assert.Equal(100M, p.Price);
30        }
31    }
32 }
```

Name	Description
Equal(expected, result)	This method asserts that the result is equal to the expected outcome. There are overloaded versions of this method for comparing different types and for comparing collections. There is also a version of this method that accepts an additional argument of an object that implements the IEqualityComparer<T> interface for comparing objects.
NotEqual(expected, result)	This method asserts that the result is not equal to the expected outcome.
True(result)	This method asserts that the result is true.
False(result)	This method asserts that the result is false.
IsType(expected, result)	This method asserts that the result is of a specific type.
IsNotType(expected, result)	This method asserts that the result is not a specific type.

Name	Description
InRange(result, low, high)	This method asserts that the result falls between low and high.
NotInRange(result, low, high)	This method asserts that the result falls outside low and high.
Throws(exception, expression)	This method asserts that the specified expression throws a specific exception type.
IsNull(result)	This method asserts that the result is null.
IsNotNull(result)	This method asserts that the result is not null.

File Edit View Project Build Debug Test Analyze Tools Extensions Window Help Search (Ctrl+Q) Testing

Debug

Run All Tests Ctrl+R, A
Repeat Last Run Ctrl+R, L
Debug All Tests Ctrl+R, Ctrl+A
Debug Last Run Ctrl+R, D
Configure Run Settings
Processor Architecture for AnyCPU Projects
Test Explorer Ctrl+E, T
Options...

You can improve 'Startup' performance by auto hiding or c Don't show this message again

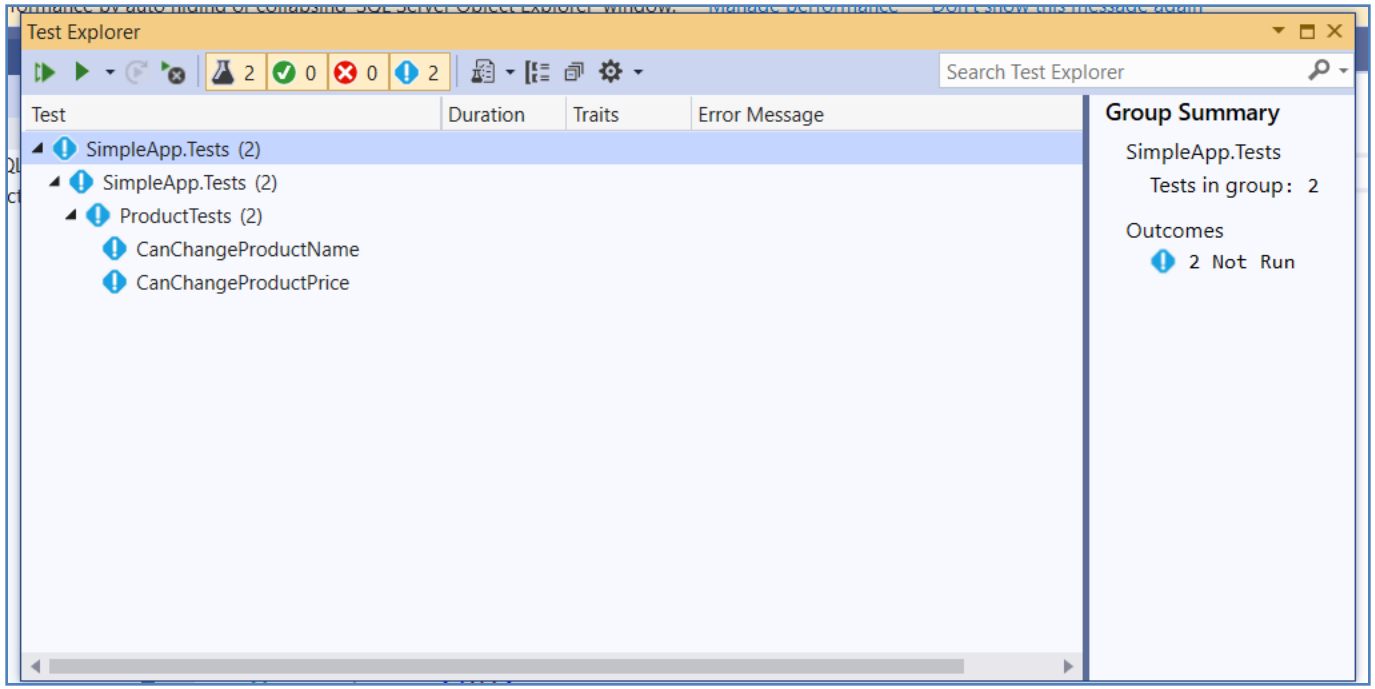
SQL Server Object Explorer

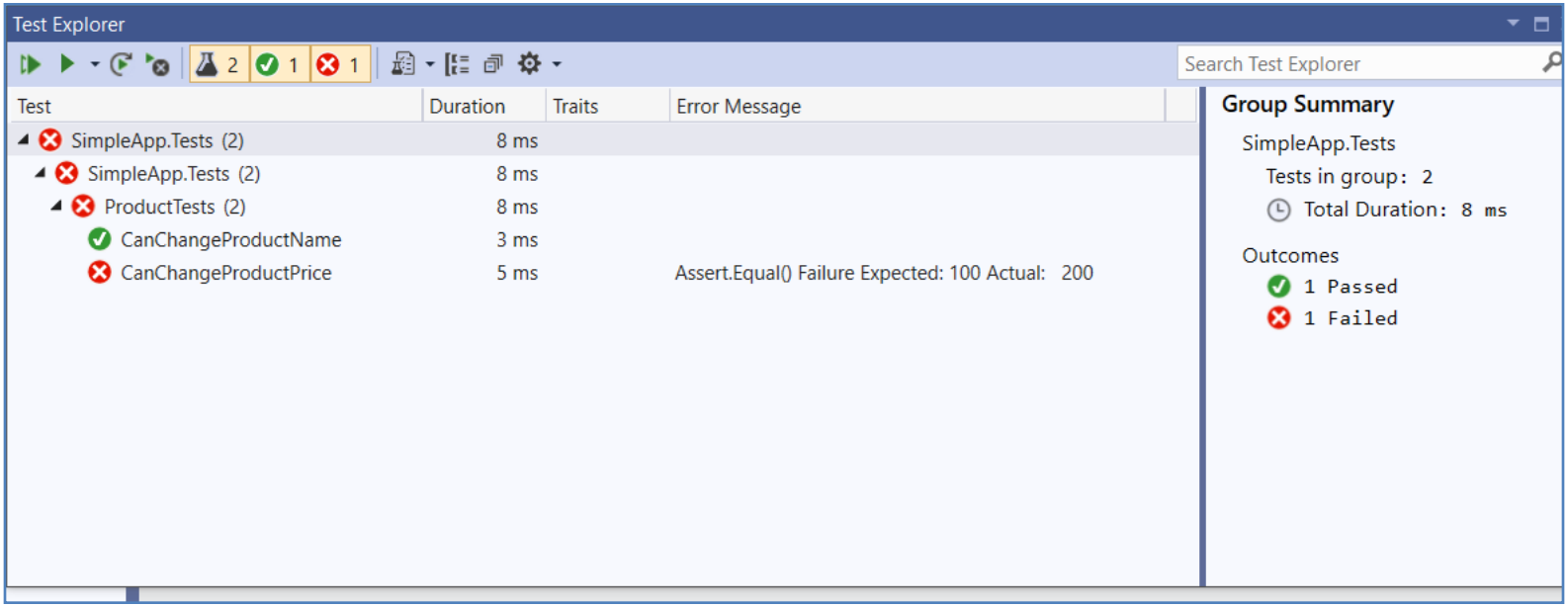
- SQL Server
  - (localdb)\MSSQLLocalDB (SQ
  - (localdb)\ProjectsV13 (SQL Se
- Projects - Testing

ProductTests Product.cs Startup.cs

SimpleA CanChangePr

```
17
18
19                  e", p.Name);
20              }
21              [Fact]
                0 references
22              public void CanChangeProductPrice()
23              {
24                  // Arrange
25                  var p = new Product { Name = "Test", Price =
26                  // Act
27                  p.Price = 200M;
28                  //Assert
29                  Assert.Equal(100M, p.Price);
30              }
31          }
32      }
33
```


Test Explorer
2
0
0
2
Search Test Explorer
Test
Duration
Traits
Error Message
SimpleApp.Tests (2)
SimpleApp.Tests (2)
ProductTests (2)
CanChangeProductName
CanChangeProductPrice
Group Summary
SimpleApp.Tests
Tests in group: 2
Outcomes
2 Not Run


Test Explorer
2
1
1
Search Test Explorer
Test
Duration
Traits
Error Message
SimpleApp.Tests (2)
8 ms
SimpleApp.Tests (2)
8 ms
ProductTests (2)
8 ms
CanChangeProductName
3 ms
CanChangeProductPrice
5 ms
Assert.Equal() Failure Expected: 100 Actual: 200
Group Summary
SimpleApp.Tests
Tests in group: 2
Total Duration: 8 ms
Outcomes
1 Passed
1 Failed

Windows PowerShell

```
PS D:\Testing> dotnet test
Test run for D:\Testing\SimpleApp.Tests\bin\Debug\netcoreapp3.1\SimpleApp.Tests.dll(.NETCo
Microsoft (R) Test Execution Command Line Tool Version 16.6.0
Copyright (c) Microsoft Corporation.  All rights reserved.

Starting test execution, please wait...

A total of 1 test files matched the specified pattern.
[xUnit.net 00:00:00.68]     SimpleApp.Tests.ProductTests.CanChangeProductPrice [FAIL]
  X SimpleApp.Tests.ProductTests.CanChangeProductPrice [5ms]
  Error Message:
   Assert.Equal() Failure
Expected: 100
Actual:   200
  Stack Trace:
     at SimpleApp.Tests.ProductTests.CanChangeProductPrice() in D:\Testing\SimpleApp.Tests
 29

Test Run Failed.
Total tests: 2
     Passed: 1
     Failed: 1
 Total time: 1.4931 Seconds
PS D:\Testing>
```

Build Debug Test Analyze Tools Extensions Window Help Search (Ctrl+Q) Testing

Debug | Any CPU | SimpleApp | SimpleApp

y auto hiding or collapsing 'SQL Server Object Explorer' window. Manage performance Don't show this message again

ProductTests.cs | Output | Index.cshtml | HomeController.cs | Product.cs | Startup.cs

SimpleApp.Tests | SimpleApp.Tests.ProductTests | CanChangeProductPrice()

```
0 references
public class ProductTests
{
    [Fact]
    | 0 references
    public void CanChangeProductName()
    {
        // Arrange
```

SimpleApp.Tests.ProductTests.CanChangeProductPrice

Duration: 5 ms

Message:

```
Assert.Equal() Failure
Expected: 100
Actual:   200
```

Run | Debug

```
    | 0 references
    public void CanChangeProductPrice()
    {
        // Arrange
        var p = new Product { Name = "Test", Price = 100M };
        // Act
        p.Price = 200M;
        //Assert
        Assert.Equal(100M, p.Price);
    }
}
```

Solution E

Search Sc

Solu

Solution

Properties

```
0 references
public class ProductTests
{
    [Fact]
    0 references
    public void CanChangeProductName()
    {
        // Arrange
        var p = new Product { Name = "Test", Price = 100M };
        // Act
        p.Name = "New Name";
        //Assert
        Assert.Equal("New Name", p.Name);
    }
    [Fact]
    0 references
    public void CanChangeProductPrice()
    {
        // Arrange
        var p = new Product { Name = "Test", Price = 100M };
        // Act
        p.Price = 200M;
        //Assert
        Assert.Equal(200M, p.Price);
    }
}
```

Test Explorer

2 2 0

Search Test Explorer

Test	Duration	Traits	Error Message
SimpleApp.Tests (2)	8 ms		
SimpleApp.Tests (2)	8 ms		
ProductTests (2)	8 ms		
CanChangeProductName	3 ms		
CanChangeProductPrice	5 ms		

Group Summary

SimpleApp.Tests

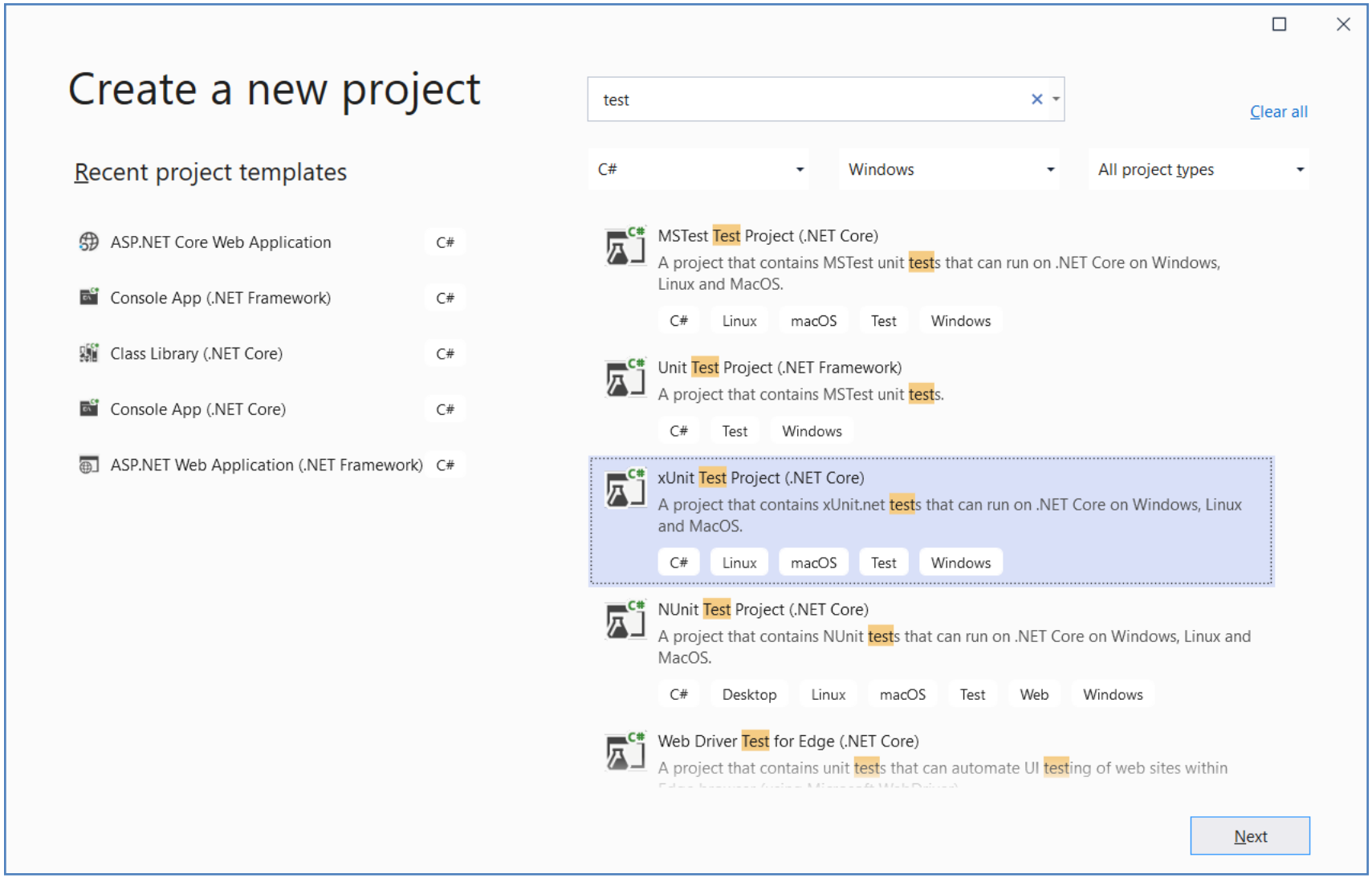
Tests in group: 2

Total Duration: 8 ms

Outcomes

2 Passed

Unit testing with xUnit: One more…



XUnitDemo.csproj | UnitTest1.cs | Output

```
<Project Sdk="Microsoft.NET.Sdk">

  <PropertyGroup>
    <TargetFramework>netcoreapp3.1</TargetFramework>

    <IsPackable>false</IsPackable>
  </PropertyGroup>

  <ItemGroup>
    <PackageReference Include="Microsoft.NET.Test.Sdk" Version="16.5.0" />
    <PackageReference Include="xunit" Version="2.4.0" />
    <PackageReference Include="xunit.runner.visualstudio" Version="2.4.0" />
    <PackageReference Include="coverlet.collector" Version="1.2.0" />
  </ItemGroup>

</Project>
```

The xUnit test framework

```
3
4   namespace XUnitDemo
5   {
        0 references
6       public class UnitTest1
7       {
8           [Fact]
            0 references
9           public void Test1()
10          {
11
12          }
13      }
14  }
15
```

Configure your new project

ASP.NET Core Web Application Cloud C# Linux macOS Service Web Windows

Project name

ExchangeRates.Web

Location

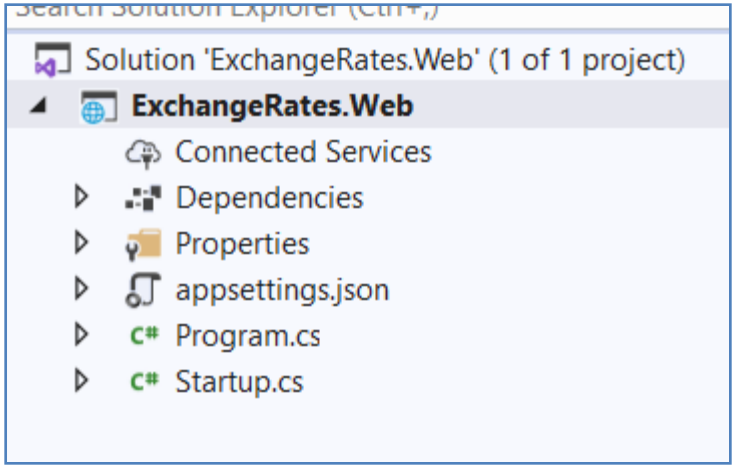
C:\Users\AAM\source\repos ...

Solution name

ExchangeRates.Web

☐ Place solution and project in the same directory

Back Create


Search Solution Explorer (Ctrl+;)
Solution 'ExchangeRates.Web' (1 of 1 project)
ExchangeRates.Web
Connected Services
Dependencies
Properties
appsettings.json
Program.cs
Startup.cs

Configure your new project

xUnit Test Project (.NET Core) C# Linux macOS Test Windows

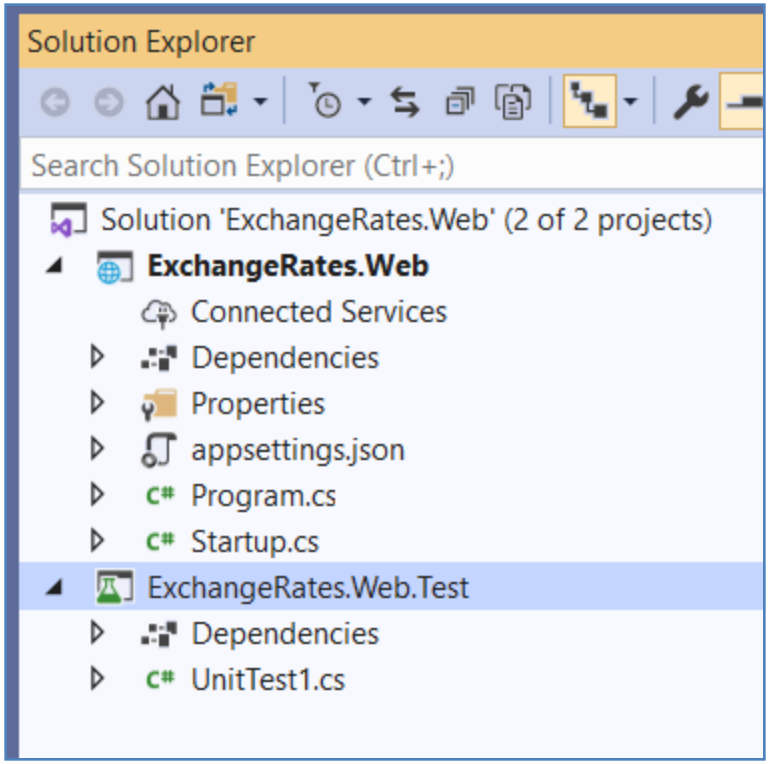
Project name

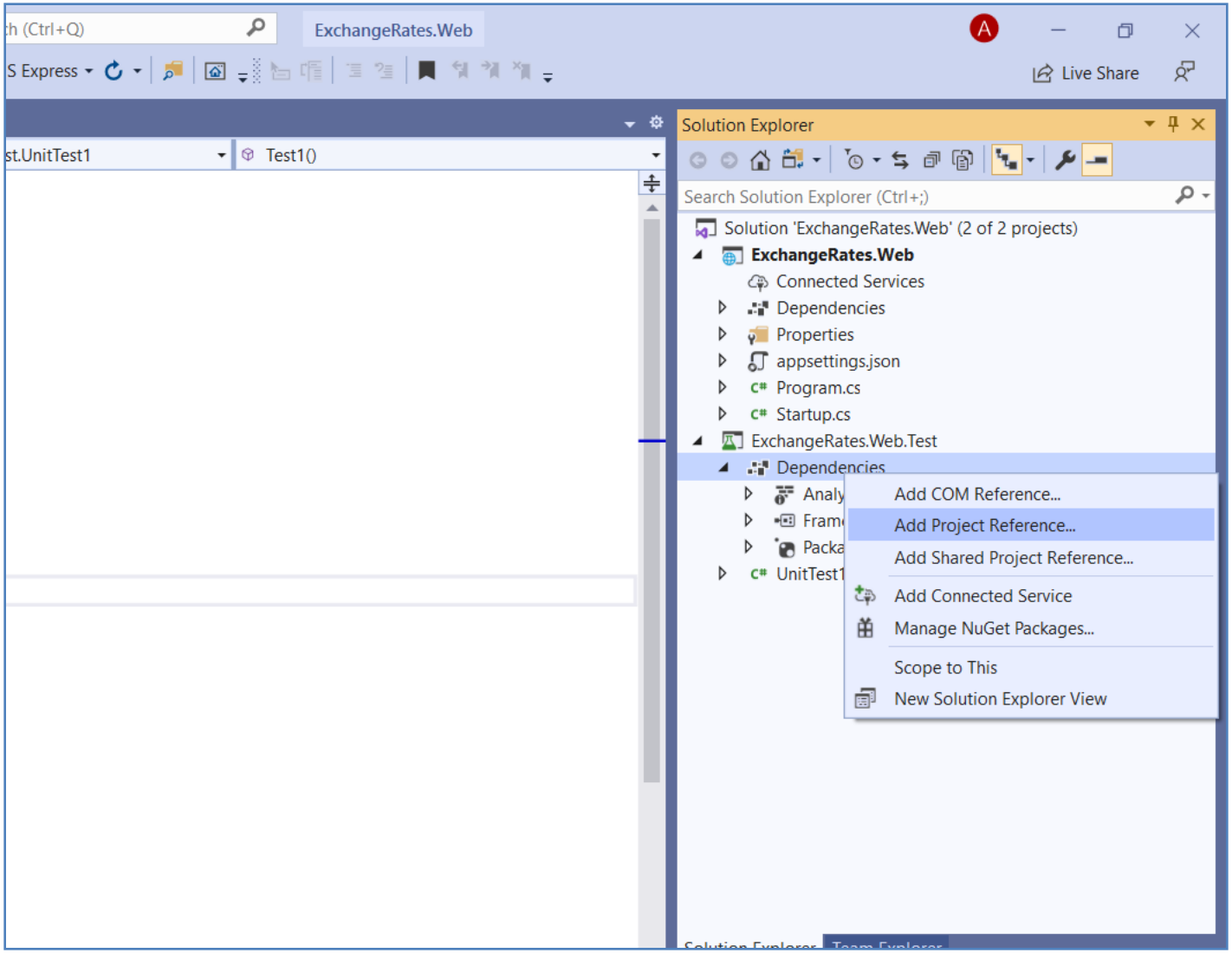
ExchangeRates.Web.Test

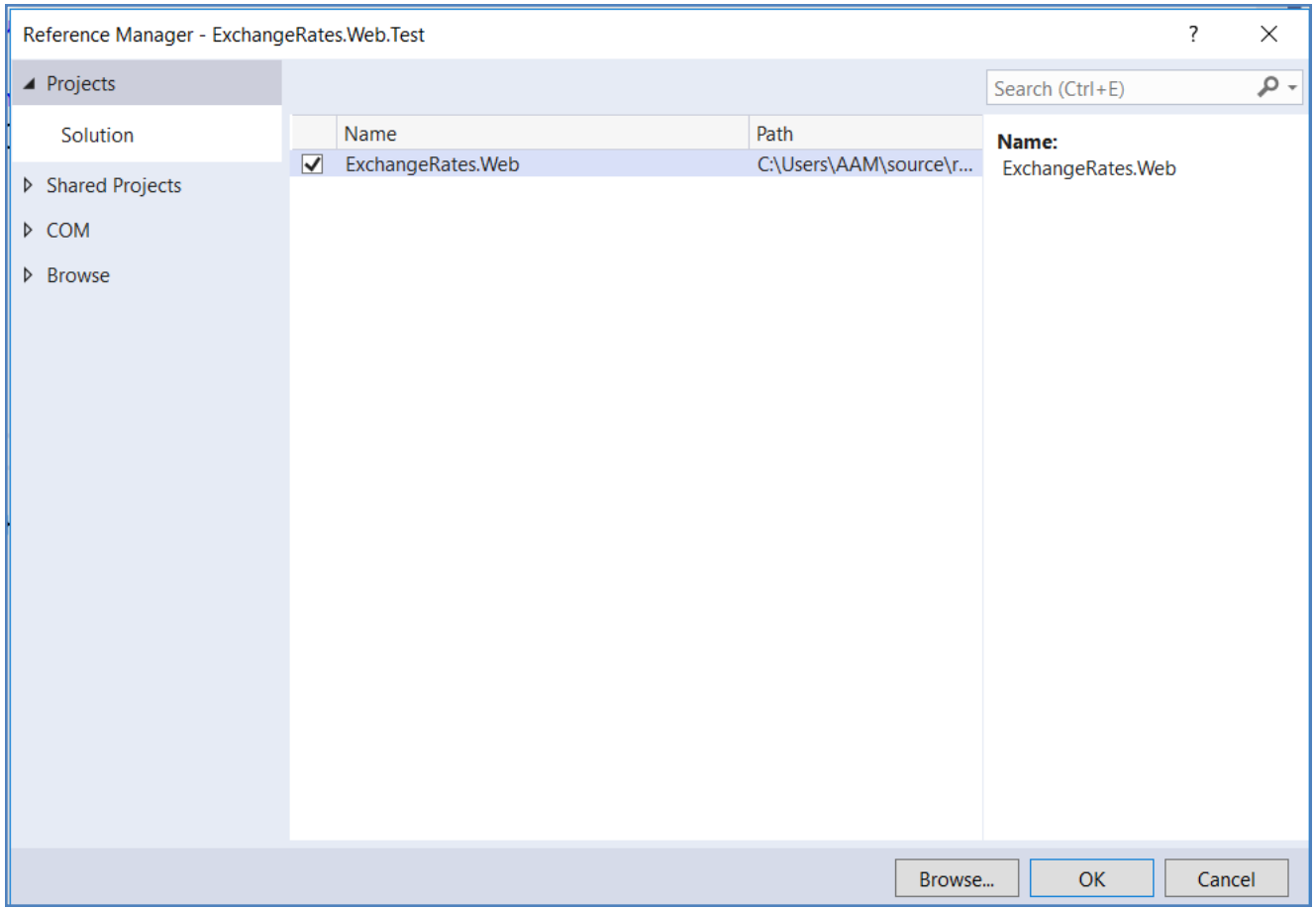
Location

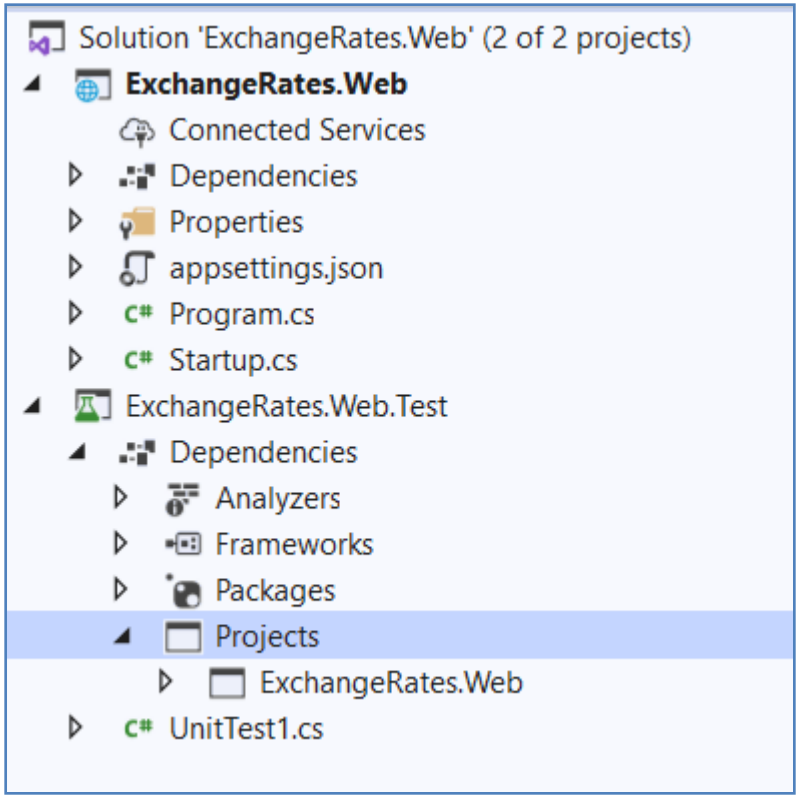
C:\Users\AAM\source\repos\ExchangeRates.Web

Back Create


Solution Explorer
Search Solution Explorer (Ctrl+;)
Solution 'ExchangeRates.Web' (2 of 2 projects)
ExchangeRates.Web
Connected Services
Dependencies
Properties
appsettings.json
Program.cs
Startup.cs
ExchangeRates.Web.Test
Dependencies
UnitTest1.cs


ch (Ctrl+Q)
ExchangeRates.Web
S Express
Live Share
st.UnitTest1
Test1()
Solution Explorer
Search Solution Explorer (Ctrl+;)
Solution 'ExchangeRates.Web' (2 of 2 projects)
ExchangeRates.Web
Connected Services
Dependencies
Properties
appsettings.json
Program.cs
Startup.cs
ExchangeRates.Web.Test
Dependencies
Analy
Fram
Packa
UnitTest1
Add COM Reference...
Add Project Reference...
Add Shared Project Reference...
Add Connected Service
Manage NuGet Packages...
Scope to This
New Solution Explorer View


Reference Manager - ExchangeRates.Web.Test
Projects
Solution
Shared Projects
COM
Browse
Search (Ctrl+E)
Name
Path
ExchangeRates.Web
C:\Users\AAM\source\r...
Name:
ExchangeRates.Web
Browse...
OK
Cancel


Solution 'ExchangeRates.Web' (2 of 2 projects)
ExchangeRates.Web
Connected Services
Dependencies
Properties
appsettings.json
Program.cs
Startup.cs
ExchangeRates.Web.Test
Dependencies
Analyzers
Frameworks
Packages
Projects
ExchangeRates.Web
UnitTest1.cs

ExchangeRates.Web.Test.csproj* | UnitTest1.cs | ExchangeRates.Web | Output

```xml
<Project Sdk="Microsoft.NET.Sdk">

  <PropertyGroup>
    <TargetFramework>netcoreapp3.1</TargetFramework>

    <IsPackable>false</IsPackable>
  </PropertyGroup>

  <ItemGroup>
    <PackageReference Include="Microsoft.NET.Test.Sdk" Version="16.5.0" />
    <PackageReference Include="xunit" Version="2.4.0" />
    <PackageReference Include="xunit.runner.visualstudio" Version="2.4.0" />
    <PackageReference Include="coverlet.collector" Version="1.2.0" />
  </ItemGroup>

  <ItemGroup>
    <ProjectReference Include="..\ExchangeRates.Web\ExchangeRates.Web.csproj" />
  </ItemGroup>

</Project>
```

```
public class CurrencyConverter
{

    public decimal ConvertToINR(decimal value, decimal exchangeRate, int decimalPlaces)
    {
        if (exchangeRate <= 0)
        {
            throw new ArgumentException("Exchange rate must be greater than zero",
            nameof(exchangeRate));
        }
        var valueInInr = value * exchangeRate;
        return decimal.Round(valueInInr, decimalPlaces);
    }
}
```

Adding Fact and Theory unit tests

- When you write unit tests, you should target one of three different paths through the method under test:
  - The happy path—Where typical arguments with expected values are provided
  - The error path—Where the arguments passed are invalid and tested for
  - Edge cases—Where the provided arguments are right on the edge of expected values

```
namespace ExchangeRates.Web.Test
{
    0 references
    public class ConvertToINRTests
    {
        [Fact]
        0 references
        public void ConvertToINR_ConvertsCorrectly()
        {
            var converter = new CurrencyConverter();
            decimal value = 5.0m;
            decimal rate = 70m;
            int dp = 4;
            decimal expected = 350;
            var actual = converter.ConvertToINR(value, rate, dp);
            Assert.Equal(expected, actual);
        }
    }
}
```

Test Explorer

1 1 0

Search Test Explorer

Test	Duration	Traits	Error Message
ExchangeRates.Web.Test (1)	11 ms		
ExchangeRates.Web.Test (1)	11 ms		
ConvertToINRTests (1)	11 ms		
ConvertToINR_ConvertsCorrectly	11 ms		

Group Summary

ExchangeRates.Web.Test

Tests in group: 1

Total Duration: 11 ms

Outcomes

1 Passed

```
namespace ExchangeRates.Web.Test
{
    0 references
    public class ConvertToINRTests
    {
        [Fact]
        0 references
        public void ConvertToINR_ConvertsCorrectly()
        {
            var converter = new CurrencyConverter();
            decimal value = 5.0m;
            decimal rate = 70m;
            int dp = 4;
            decimal expected = 350.01m;
            var actual = converter.ConvertToINR(value, rate, dp);
            Assert.Equal(expected, actual);
        }
    }
}
```

Test Explorer

1 0 1

Search Test Explorer

Test	Duration	Traits	Error Message
ExchangeRates.Web.Test (1)	14 ms		
ExchangeRates.Web.Test (1)	14 ms		
ConvertToINRTests (1)	14 ms		
ConvertToINR_ConvertsCorrectly	14 ms		Assert.Equal() Failure Expected: 350.01 Actual: 350.0

Group Summary

ExchangeRates.Web.Test

Tests in group: 1

Total Duration: 14 ms

Outcomes

1 Failed

```
namespace ExchangeRates.Web.Test
{
    0 references
    public class ConvertToINRTests
    {
        [Fact]
        0 references
        public void ConvertToINR_ConvertsCorrectly()
        {
            var converter = new CurrencyConverter();
            decimal value = 5.0m;
            decimal rate = 70m;
            int dp = 4;
            decimal expected = 350;
            var actual = converter.ConvertToINR(value, rate, dp);
            Assert.Equal(expected, actual);
        }
    }
}
```

```
0 references
public class ConvertToINRTests
{
    [Theory]
    [InlineData(0, 70, 0)]
    [InlineData(5, 70, 350)]
    [InlineData(8.75, 70.2, 614.25)]
    0 references
    public void ConvertToINR_ConvertsCorrectly(decimal value, decimal rate, decimal expected)
    {
        var converter = new CurrencyConverter();
        int dps = 4;
        var actual = converter.ConvertToINR(value, rate, dps);
        Assert.Equal(expected, actual);
    }
}
```

Test Explorer

3 3 0

Search Test Explorer

Test	Duration	Traits	Error Message
ExchangeRates.Web.Test (3)	7 ms		
ExchangeRates.Web.Test (3)	7 ms		
ConvertToINRTests (3)	7 ms		
ConvertToINR_ConvertsCorrectly (3)	7 ms		
ConvertToINR_ConvertsCorrectly(value: 0, rate: 70, expected: 0)	7 ms		
ConvertToINR_ConvertsCorrectly(value: 5, rate: 70, expected: 350)	< 1 ms		
ConvertToINR_ConvertsCorrectly(value: 8.75, rate: 70.2, expected: 614.25)	< 1 ms		

Group Summary

ExchangeRates.Web.Test

Tests in group: 3

Total Duration: 7 ms

Outcomes

3 Passed

```
public class ConvertToINRTests
{
    [Fact]
    0 references
    public void ThrowsExceptionIfRateIsZero()
    {
        var converter = new CurrencyConverter();
        const decimal value = 5;
        const decimal rate = 0;
        const int dp = 2;
        var ex = Assert.Throws<ArgumentException>(
        () => converter.ConvertToINR(value, rate, dp));
        // Further assertions on the exception thrown, ex
    }
```

```
public class ConvertToINRTests
{
    [Fact]
    0 references
    public void ThrowsExceptionIfRateIsZero()
    {
        var converter = new CurrencyConverter();
        const decimal value = 5;
        const decimal rate = 0;
        const int dp = 2;
        var ex = Assert.Throws<ArgumentException>(
        () => converter.ConvertToINR(value, rate, dp));
        // Further assertions on the exception thrown, ex
    }
```

Test Explorer

4 4 0

Search Test Explorer

Test	Duration	Traits	Error...
ExchangeRates.Web.Test (4)	28 ms		
ExchangeRates.Web.Test (4)	28 ms		
ConvertToINRTests (4)	28 ms		
ConvertToINR_ConvertsCorrectly (3)	5 ms		
ThrowsExceptionIfRateIsZero	23 ms		

Test Detail Summary

ExchangeRates.Web.Test.Conv

Source: ConvertToINRTe

Duration: 23 ms

Hobby, that helps!

Unit testing MVC controllers

- Unit tests are all about isolating behavior; you want to test only the logic contained in the component itself, separate from the behavior of any dependencies.
- MvcMiddleware as a whole contains complex behavior in the form of a filter pipeline, routing, and model binding, but these are all external to the MVC controller.

Cont.

- MVC controllers themselves are responsible for only a limited number of things.
  - For invalid requests (that have failed validation, for example), return an appropriate IActionResult.
  - For valid requests, call the required business logic services and return an appropriate IActionResult (or alternatively an object to serialize, in the case of Web API Controllers).
  - Optionally, apply resource-based authorization as required.

Cont.

- MVC Controllers generally shouldn’t contain business logic themselves; instead, they should call out to other services.
- Think of an MVC controller as more of an orchestrator, serving as the intermediary between the HTTP interfaces your app exposes and your business logic services.

```
namespace ExchangeRates.Web.Test
{
    0 references
    public class HomeControllerTest
    {
        [Fact]
        0 references
        public void Index_ReturnsIndexViewModelInViewResult()
        {
            var controller = new HomeController();
            IActionResult result = controller.Index();
            Assert.IsType<ViewResult>(result);
            var viewModel = (result as ViewResult).Model;
            Assert.IsType<IndexViewModel>(viewModel);
        }
    }
}
```

Creates an instance of the HomeController to test

Invokes the Index method and captures the IActionResult returned

Asserts that the IActionResult is a ViewResult

Extracts the ViewModel from the ViewResult and asserts it's of the IndexViewModel type

```
namespace ExchangeRates.Web.Test
{
    0 references
    public class HomeControllerTest
    {
        [Fact]
        0 references
        public void Index_ReturnsIndexViewModelInViewResult()
        {
            var controller = new HomeController();
            IActionResult result = controller.Index();
            Assert.IsType<ViewResult>(result);
            var viewModel = (result as ViewResult).Model;
            Assert.IsType<IndexViewModel>(viewModel);
        }
    }
}
```

Creates an instance of the HomeController to test

Invokes the Index method and captures the IActionResult returned

Asserts that the IActionResult is a ViewResult

Extracts the ViewModel from the ViewResult and asserts it's of the IndexViewModel type

```
namespace ExchangeRates.Web.Controllers
{
    0 references
    public class CurrencyController : Controller
    {
        [HttpPost]
        0 references
        public IActionResult Convert(ConvertInputModel model)
        {
            if (!ModelState.IsValid)
            {
                return BadRequest(ModelState);
            }
            // Other processing + business logic
            return Json(new { Success = true });
        }
    }
}
```

The ConvertInputModel is automatically validated during model binding.

If the input model is invalid . . .

. . .return a 400 Bad Request including the ModelState.

The model was valid so process it and return a result.

```
namespace ExchangeRates.Web.Test
{
    0 references
    public class Convert
    {
        [Fact]
        0 references
        public void Convert_ReturnsBadRequestWhenInvalid()
        {
            var controller = new CurrencyController();
            var model = new ConvertInputModel
            {
                Value = 1,
                ExchangeRate = -2,
                DecimalPlaces = 2,
            };
            controller.ModelState.AddModelError(
            nameof(model.ExchangeRate),
            "Exchange rate must be greater than zero"
            );
            var result = controller.Convert(model);
            Assert.IsType<BadRequestObjectResult>(result);
        }
    }
}
```

Creates an instance of the Controller to test

Creates an invalid binding model. ExchangeRate should be a positive value.

Manually adds a model error to the Controller's ModelState. This sets ModelState.IsValid to false.

Invokes the action method, passing in the binding models

Verifies the action method returned a BadRequestObjectResult