Functional Testing - xUnit

CSCI E-94

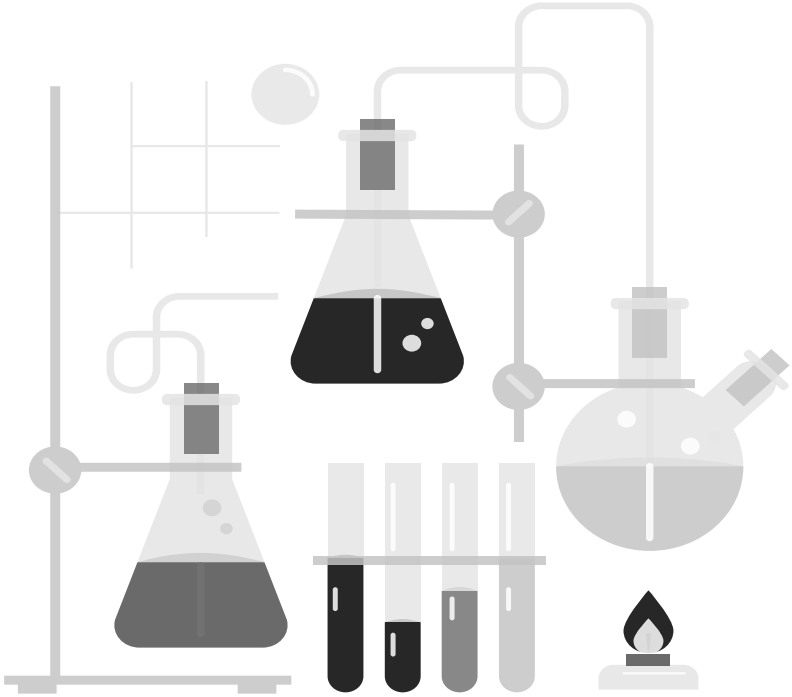
Fundamentals of Cloud Computing - Azure

Joseph Ficara

Agenda

- Functional Testing
  - Creating a client SDK from a REST Service
  - Creating a test project and functional tests

Testing



Functional Tests

- Functional Tests are:
  - Tests that verify the high-level function
    - Include positive and negative tests
      - Typically, not full negative tests seen in **Contract Tests**
  - Test the implementation of the REST interface
    - Pros
      - Can perform integration tests
      - Can support contract tests
    - Cons
      - Slower than unit tests
      - Require dependencies are properly configured

Functional Tests

- Why do functional tests?
  - Enable end to end functional verification
    - Can also be used for performance testing
  - Not as expensive to write as contract tests
- What about Unit Tests?
  - Excellent for testing business logic
  - Faster than functional test
  - More on these later…
    - See this link

Contract Tests

- What are “Contract Tests”?
  - Contract tests verify the full contract meaning:
    - The fully published contract
      - Including detailed error response validation
  - Contract tests are typically expensive to write
    - Can be cost effective in the long run

Contract Tests

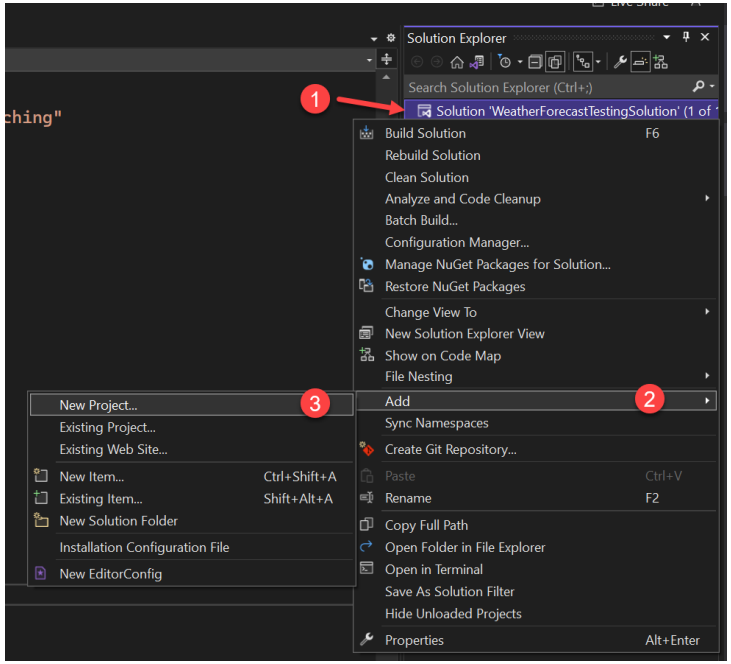
- Why Contract Tests?
  - Enable full verification of the published contract
  - Detect contract violations
    - That break client code
  - Finding violations
    - Reduces support costs
    - Improves client retention

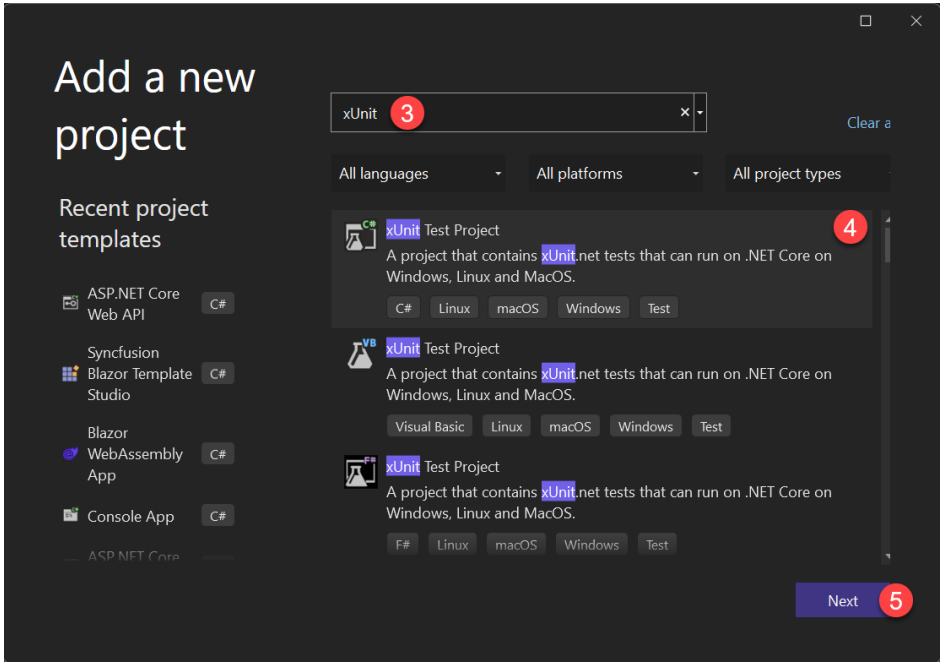
Testing

- To test a REST interface using xUnit
  - Add an xUnit Test project
  - Add a connected service
  - Perform the typical **AAA** pattern
    - **A**rrange
      - Setup the test data
    - **A**ct
      - Call the resource
    - **A**ssert
      - Verify the result

Functional Tests

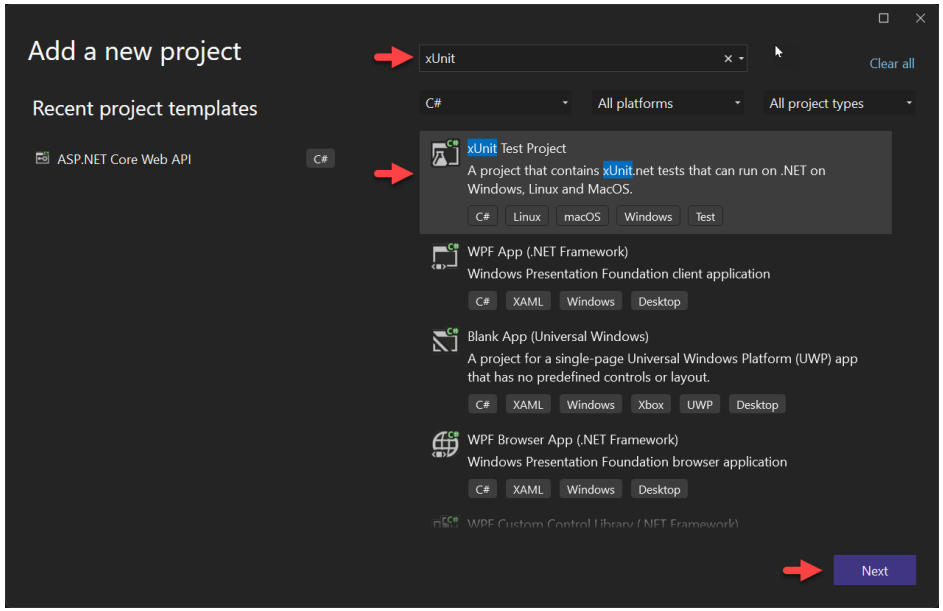
- To add an xUnit test project
  - Right Click on the solution
    - Add->New Project->

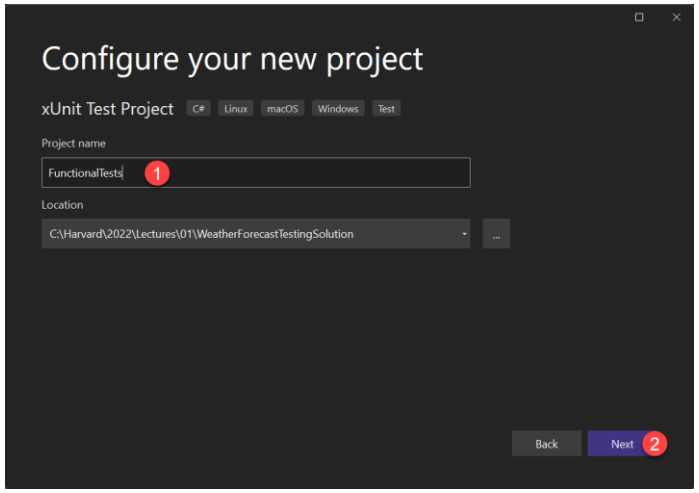


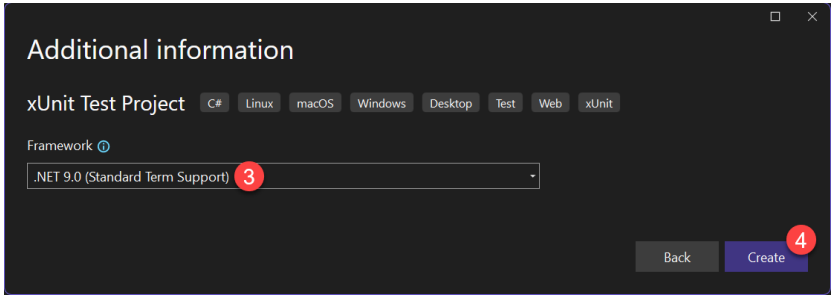


Functional Tests

- Add an xUnit project

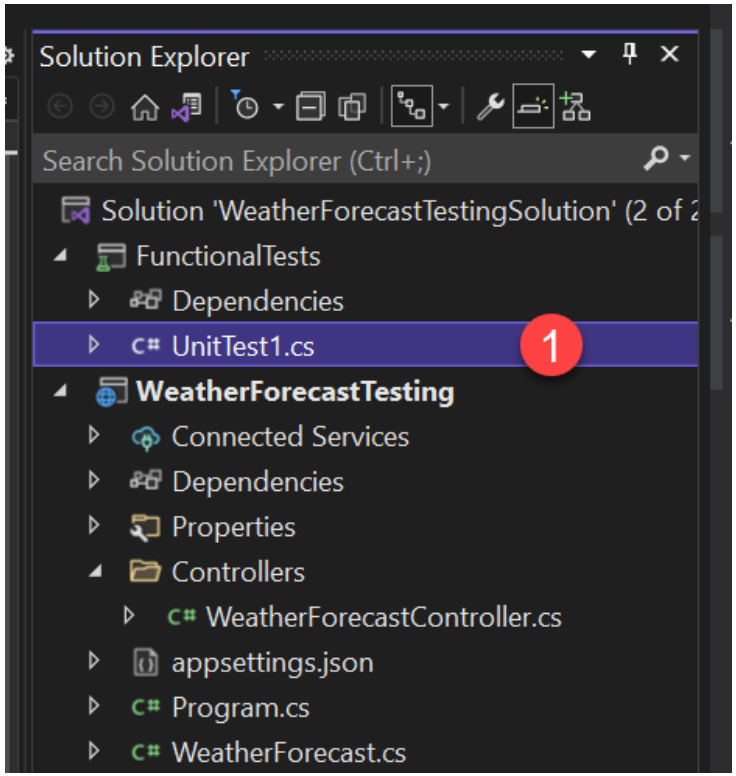


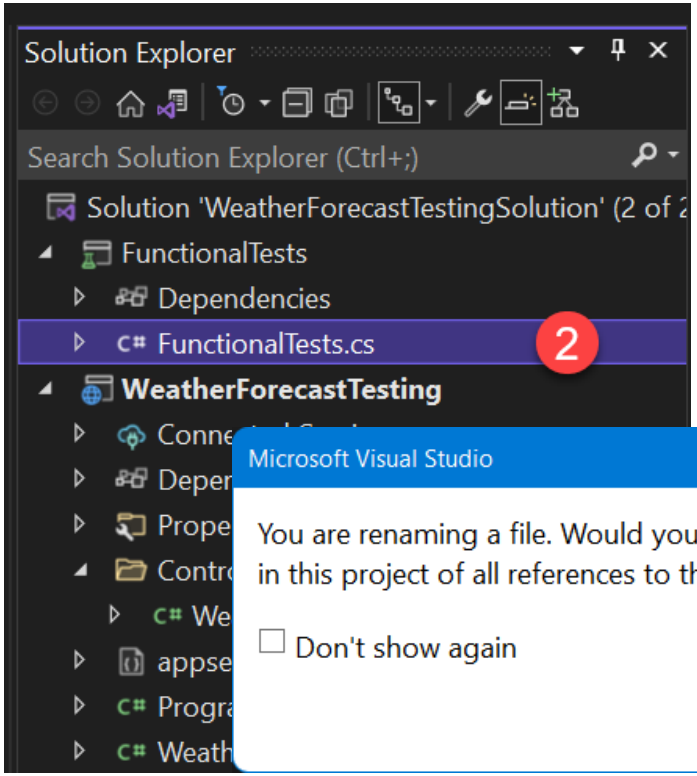




Functional Tests

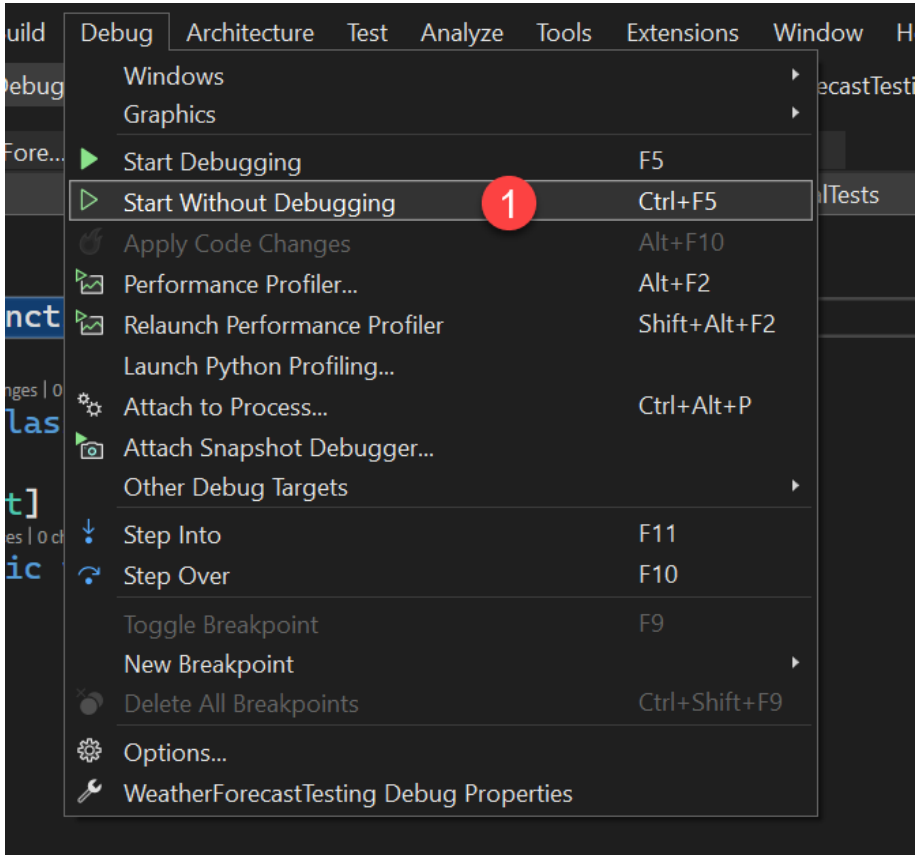
- Rename the UnitTest1.cs to
  - FunctionalTests.cs





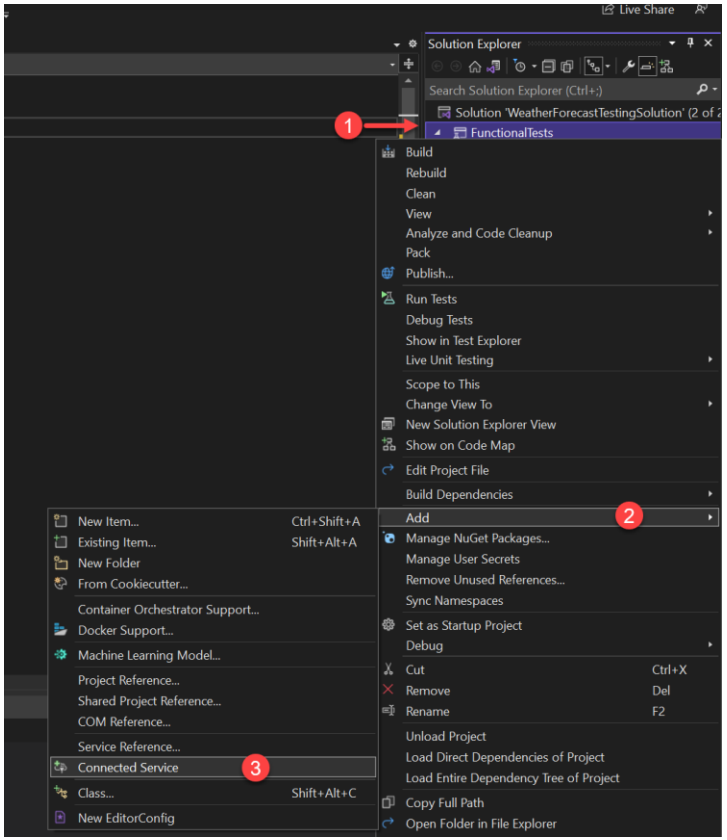
Functional Tests

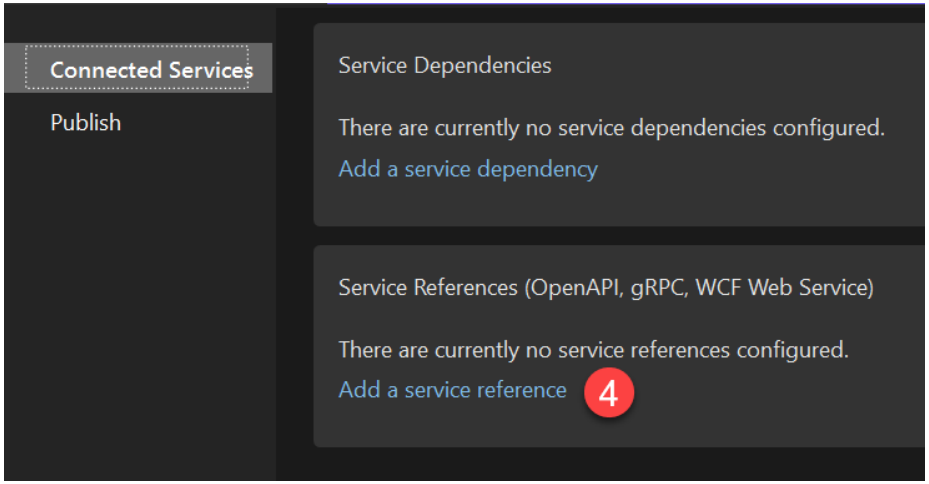
- Start without debugging



Functional Tests

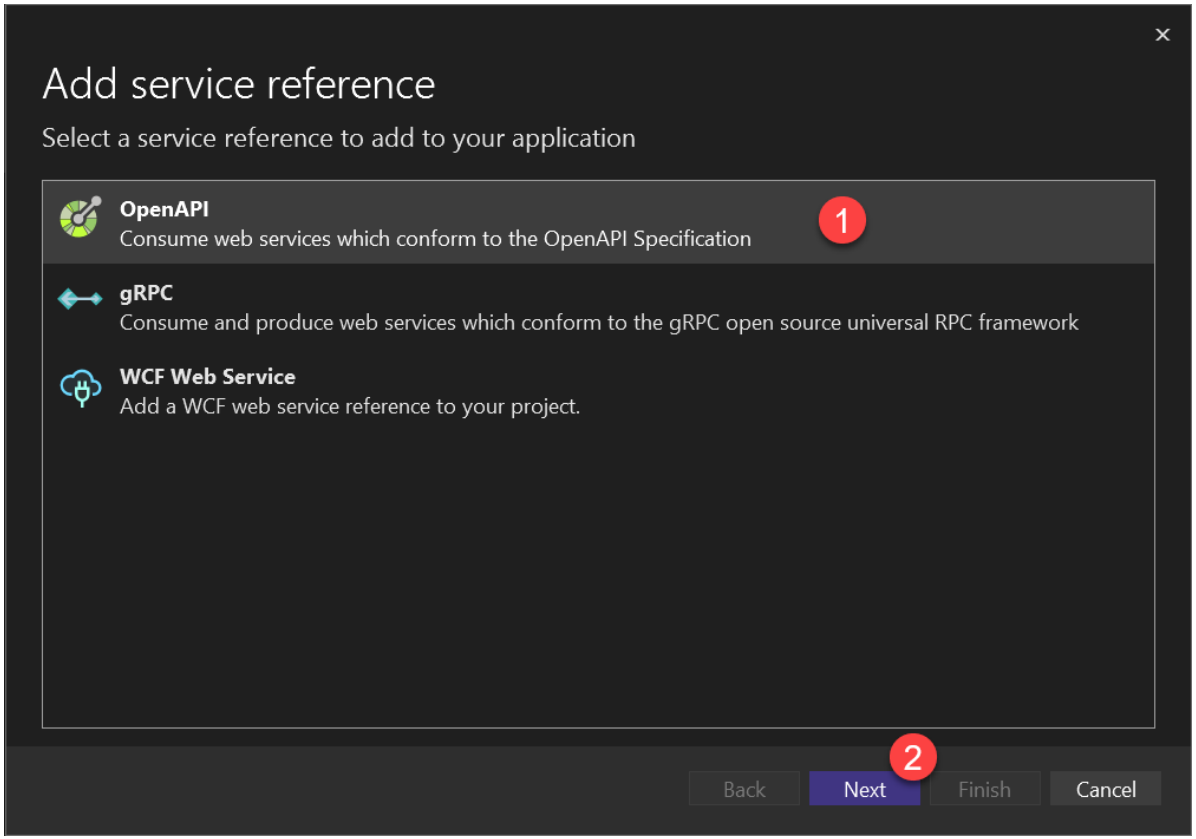
- Add a connected service





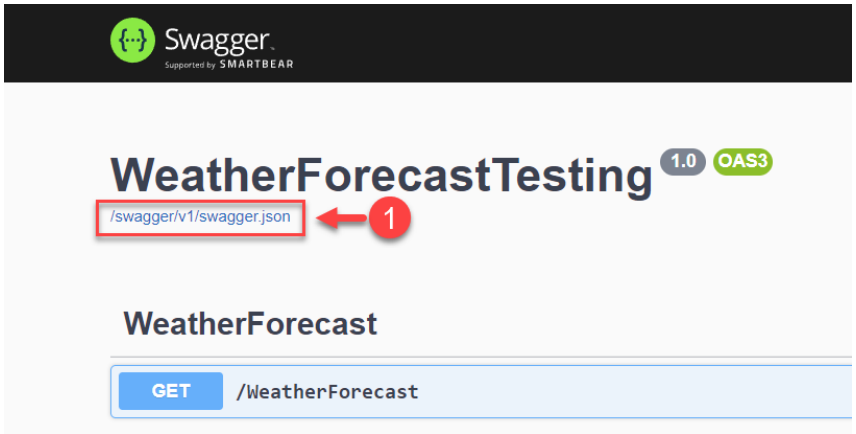
Functional Tests

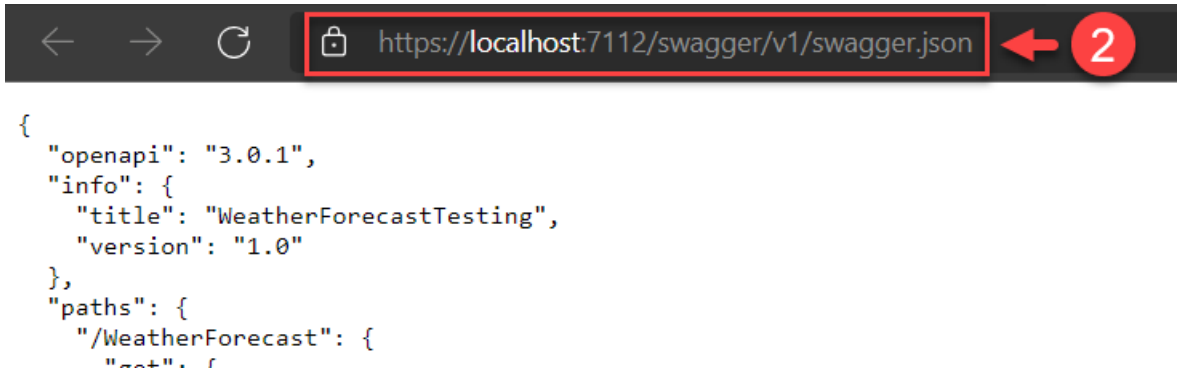
- Add a service reference



Functional Tests

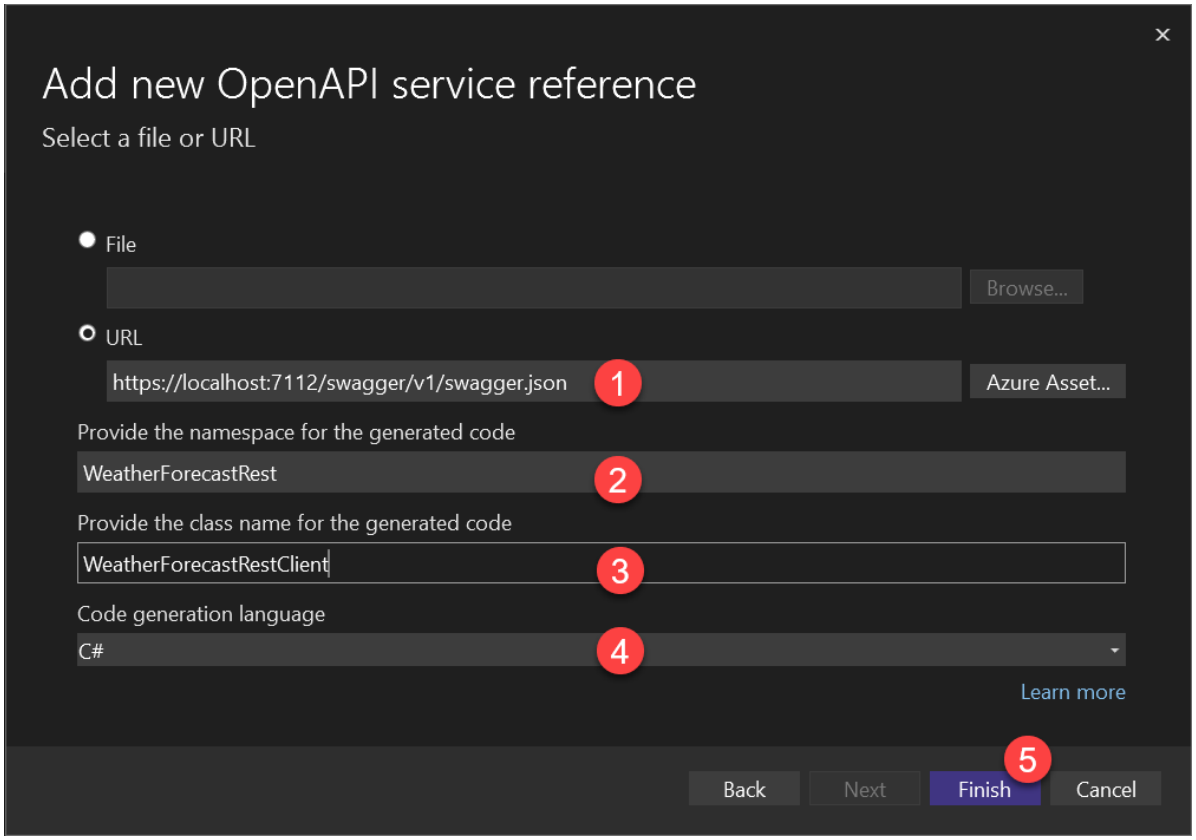
- Copy the URL to the swagger.json file





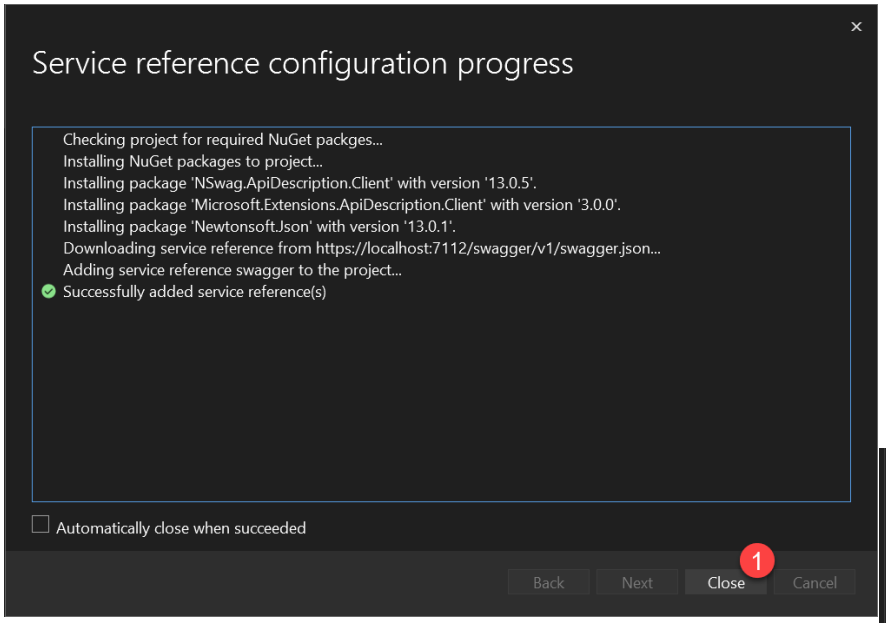
Functional Tests

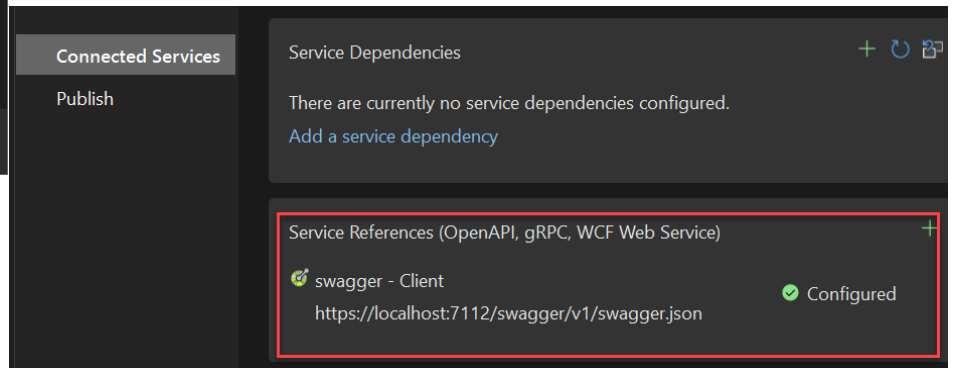
- Add new OpenAPI service reference



Functional Tests

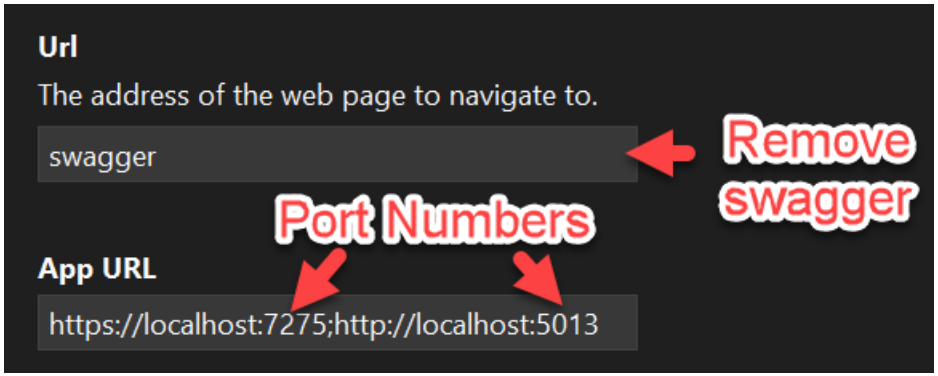
- Success looks like this





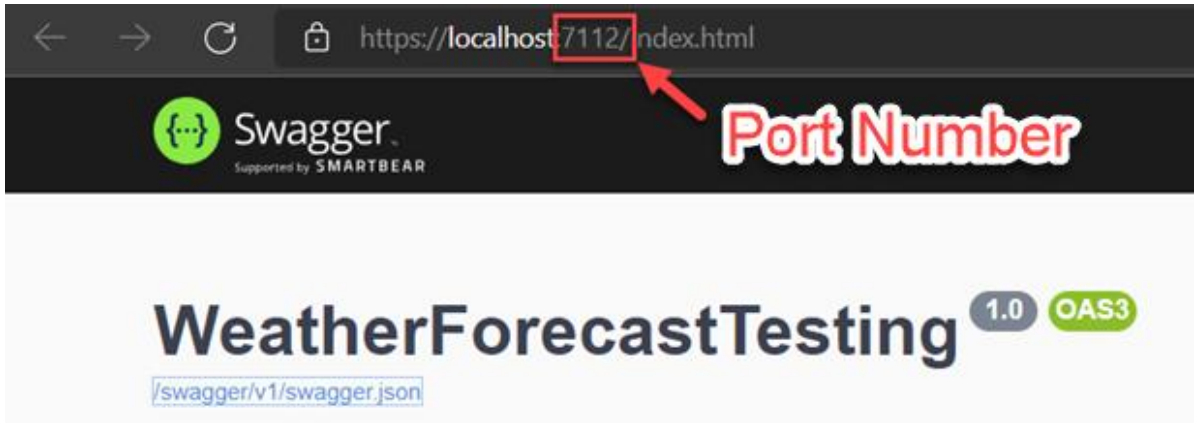
Functional Tests

- Add public methods for each scenario
  - Adorn them with the Fact Attribute
- Notes:
  - The test project is .NET 9 xUnit project
  - Remove swagger from project debug settings
  - Double check
    - Port numbers



Functional Tests

- Notes:
  - Check your port number



Functional Tests

Example test class

```
using Xunit;
using WeatherForecastRest;
using System.Net.Http;
using System.Threading.Tasks;
using System.Collections.Generic;
using System;

public class FunctionalTests
{
  //const string EndpointUrlString = "https://localhost:7112/";

  // DEMO: Testing against Azure instance
  const string EndpointUrlString = "https://localhost:7112/";

...
}
```

Functional Tests

Example test method

```
…
 [Fact]
 public async Task TestRetrievingWeatherForecasts()
 {
    // Arrange
    using HttpClient httpClient = new HttpClient();
    WeatherForecastRestClient weatherForecastRestClient = new
            WeatherForecastRestClient(EndpointUrlString, httpClient);

    // Act - Make the HTTP GET call
    ICollection<WeatherForecast>? weatherForecasts = await
             weatherForecastRestClient.GetWeatherForecastAsync();

    // Assert a result is returned
    Assert.NotNull(weatherForecasts);
…
```

Functional Tests

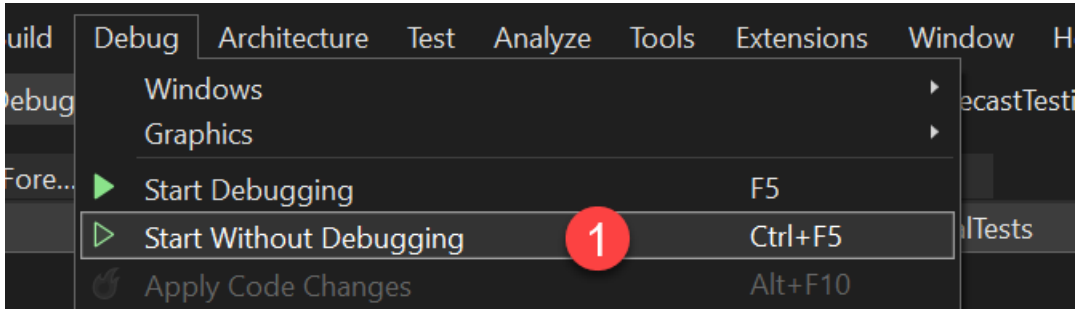
Example test method continued …

```
…
 // Assert - Verify exactly 5 returned
 Assert.Equal(5,weatherForecasts.Count);

 // Assert - Verify the date increments properly
 int index = 0;
 foreach (var item in weatherForecasts)
 {
   index++;
   Assert.Equal(DateTime.UtcNow.AddDays(index).Day, item.Date.Day);
 }
}
…
```

Functional Tests

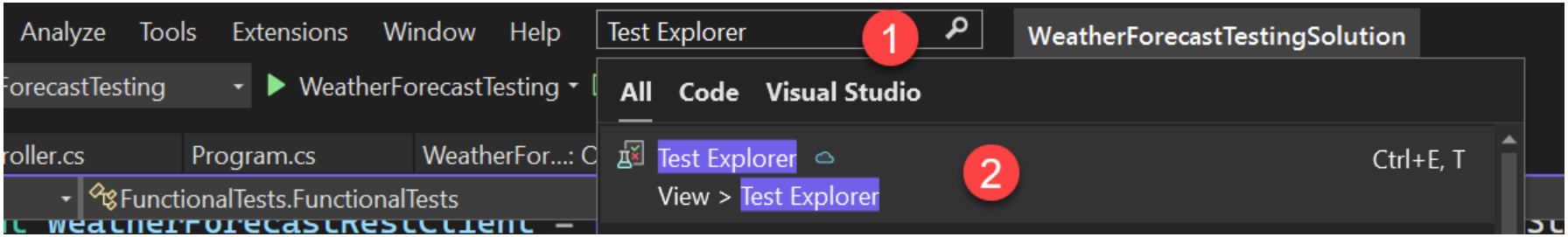
- To run tests locally
  - Start Web API project without debugging



    - Note: If you want to debug the Web API project
      - Launch another instance of Visual Studio
  - Run or Debug all or one test

Functional Tests

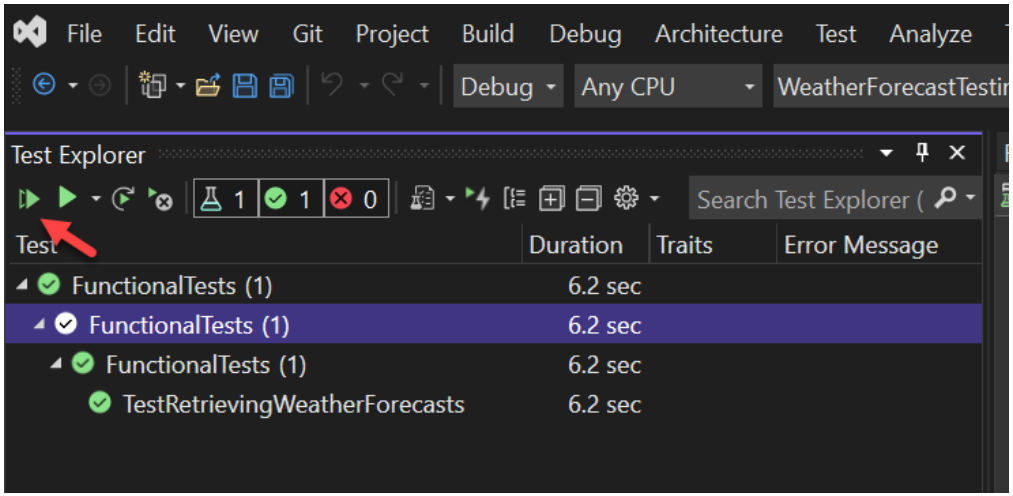
- To run all tests
  - Right click anywhere in FunctionalTests.cs file
    - Show the test explorer



Functional Tests

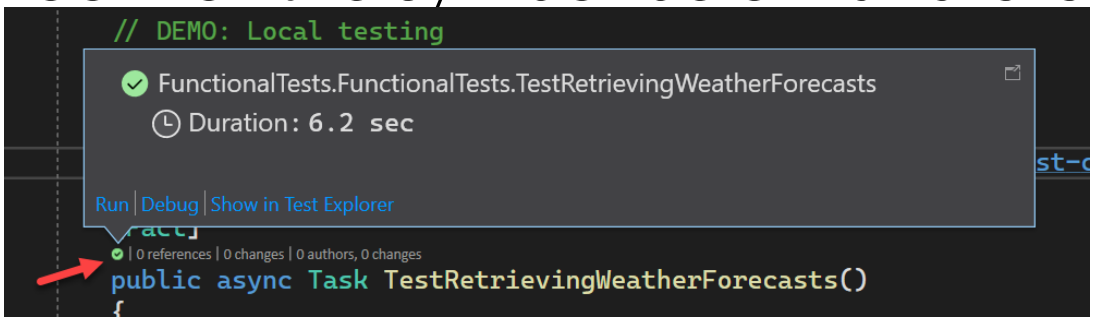
To run all tests …

- Bring up the “Test Explorer”
  - Note if you don’t see your tests
    - Try rebuilding all
    - Exit VS, relaunch VS and reload your solution
  - Click “Run All”

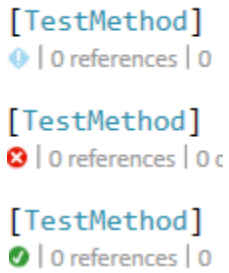


Functional Tests

- To run or debug a single test
- Look for the symbol below and click on it



- It will appear
  - Blue if the test has not run
  - Red if the test failed
  - Green if the test passed



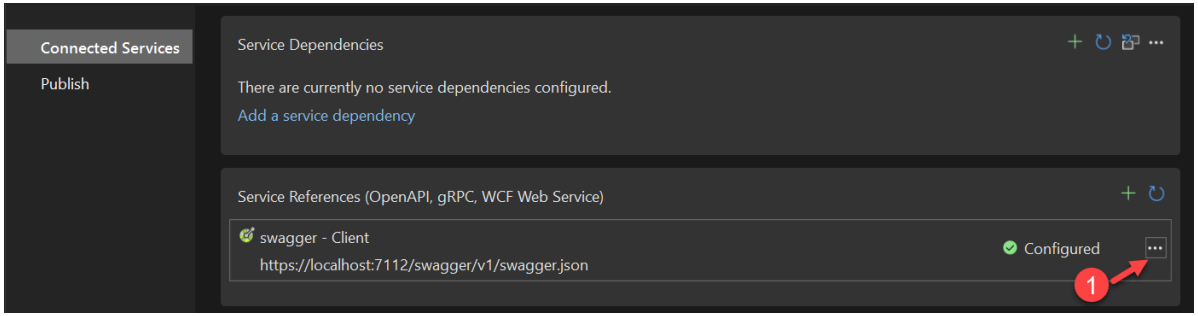
- Click Run or Debug

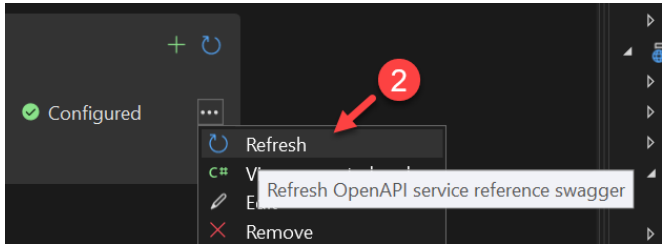
Functional Tests

- Notes:
  - When debugging the Web API
    - You must use a separate instance of Visual Studio
    - Otherwise test running will be disabled
  - Using the Connected Service SDK
    - Code coverage reports are about the SDK
      - Not your REST API
  - Code change detection and code coverage
    - Work best when performing unit tests
      - Not functional or contract type tests

Functional Tests

- Notes:
  - To regenerate the connected service SDK
    - Rebuild your Web API Project
    - Launch it without debugging
    - Refresh OpenAPI service reference swagger
    - Rebuild the test project





Demo

Generating REST Client and Functional Testing

```
WeatherForecastTesting REST Client based testing
WeatherForecastTestingSolution.sln
  WeatherForecastTesting
```

Best Practices

- Be stateless
- Be asynchronous
  - Execute I/O operations on non request thread
- Measure then optimize
- Cache as close to the wire as possible
  - Think carefully about your caching policy
- Servers shall be expendable
  - **They will fail, plan for it in your design**

Further Reading

- Azure for Developers: Implement rich Azure PaaS ecosystems using containers, serverless services, and storage solutions, 2nd Edition
  - Author: Kamil Mrzygłód
  - ISBN: 978-1803240091
  - Chapter 1

Further Reading

- Pro ASP.NET Core 6
  - Author: Adam Freeman
  - ISBN: 978-1484279564
  - Chapter 19

    -- *OR* --
- Pro ASP.NET Core 7
  - Author: Adam Freeman
  - ISBN: 1633437825
  - Chapter 19

Further Reading

- Building Cloud Apps with Microsoft Azure
  - Authors: Scott Guthrie, Mark Simms, Tom Dkystra, Rick Anderson, Mike Wasson
  - ASIN: B00LXAAMSG
  - Chapters: 4, 9, 11

Links

- Azure App Services (API and Web)
  - App Service documentation
- ASP.NET Core
  - ASP.NET documentation
- Create a web API with ASP.NET Core and Visual Studio for Windows
  - Tutorial: Create a web API with ASP.NET Core
  - Generate OpenAPI documents | Microsoft Learn

Links

- Publish an ASP.NET Core app to Azure with Visual Studio
- Publish an ASP.NET Core app to Azure with Visual Studio Code
- Publish an ASP.NET Core web app with CLI tools

Links

- Visual Studio 2022
  - Built in support for .http files
- Visual Studio Code
  - REST Client

REST Utilities

- Postman
  - Make REST calls from a richly featured UI
    - https://www.getpostman.com/
- Nightengale
    - https://nightingale.rest/
- cURL
  - Part of Windows 10 as of 1803 and Windows 11
  - Rest calls from the command line

REST Utilities

- Fiddler
  - Make REST Calls
  - Examine / Debug request/response
  - http://www.telerik.com/fiddler
- Firefox extension:
  - RESTClient