.NET Conf China 2020

2020 中国 .NET 开发者大会

.NET Conf China 2020
2020 中国 .NET 开发者大会

Dependency Injection in Xunit

Weihan Li(李卫涵)
iHerb - .NET Develop Engineer
2020-12-19

Intro

一个好的项目，它的背后有一系列的测试用例

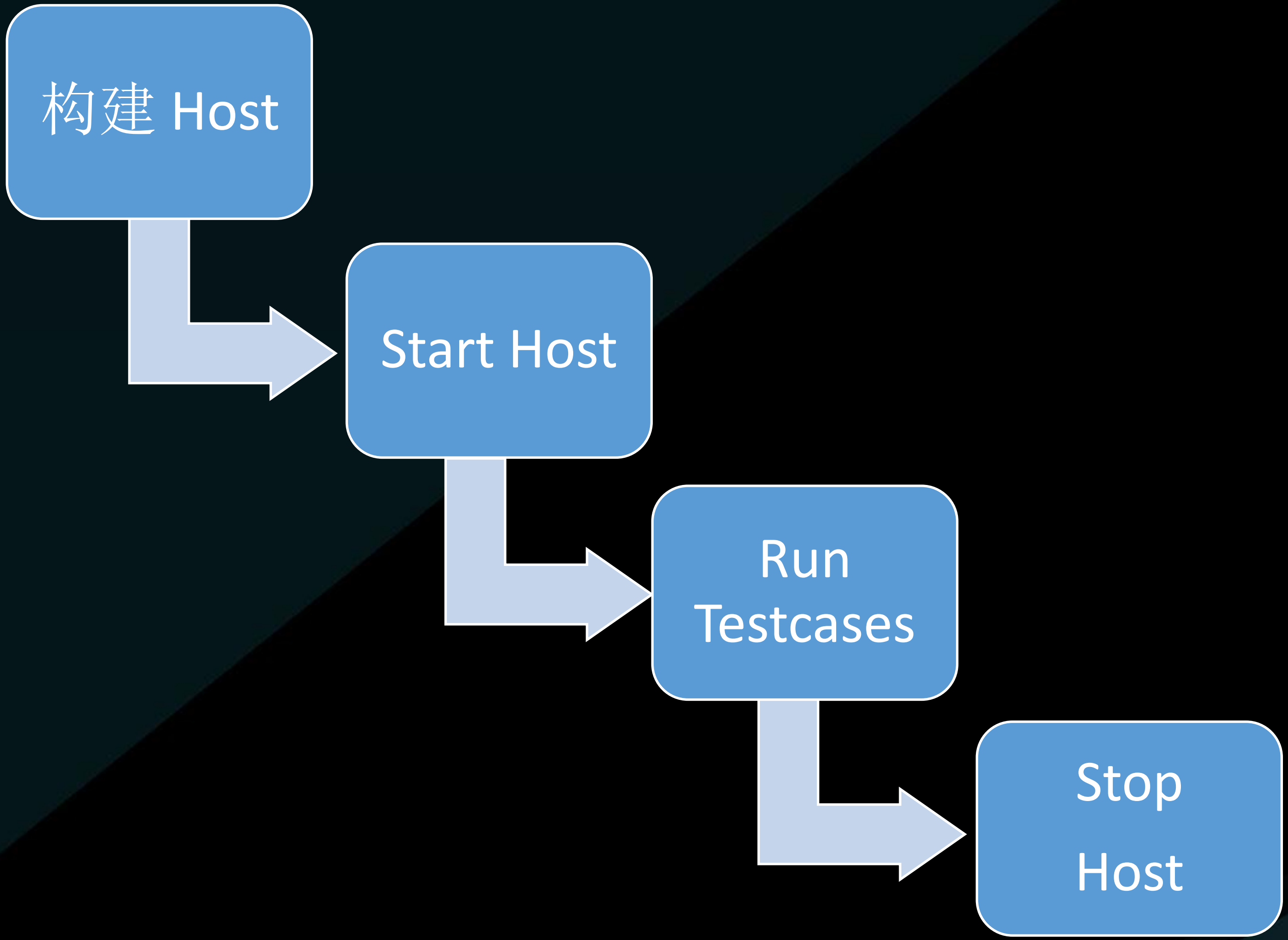
测试覆盖率是高质量项目的重要指标

依赖注入已经是现代化应用的标配，asp.net core 也是从一开始就集成了依赖注入

对于测试项目，依赖注入也不能缺席，依赖注入可以使得测试项目更容易

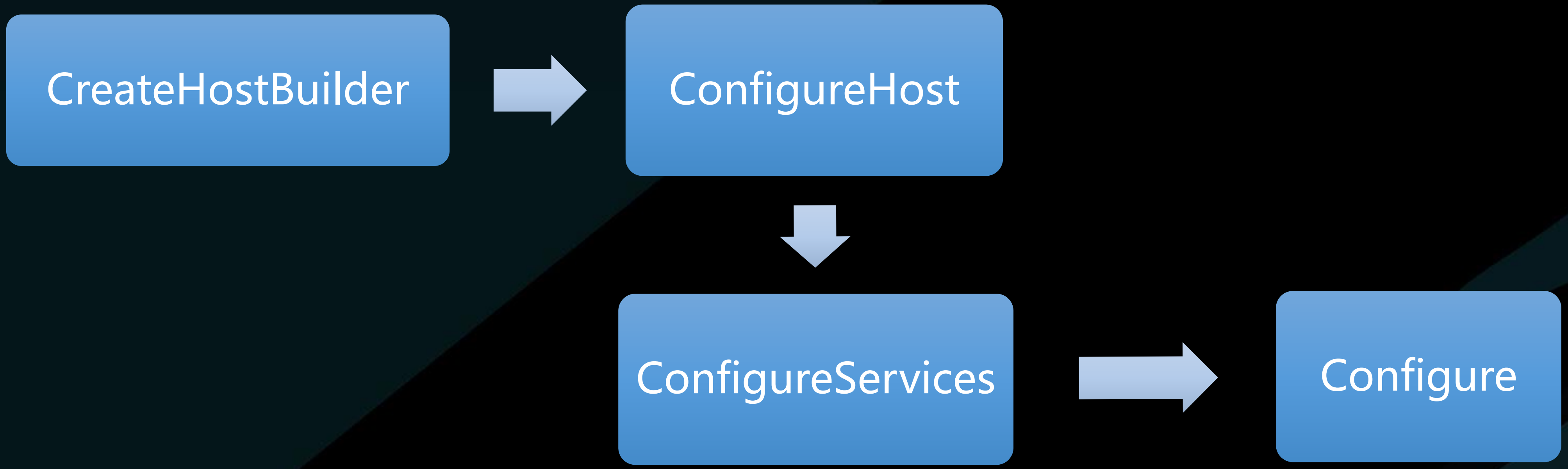
Xunit.DependencyInjection 是基于微软的 GenericHost 来实现的 xunit 扩展，能够让你更方便的在测试项目中使用依赖注入，可以更好的和 .net core 集成

执行流程



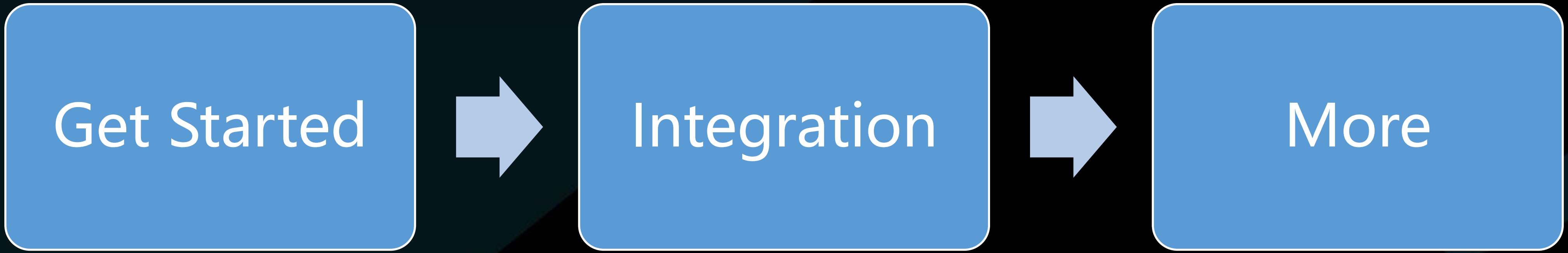
Host 构建流程

在测试项目中创建一个 Startup 类来控制 Host 的构建

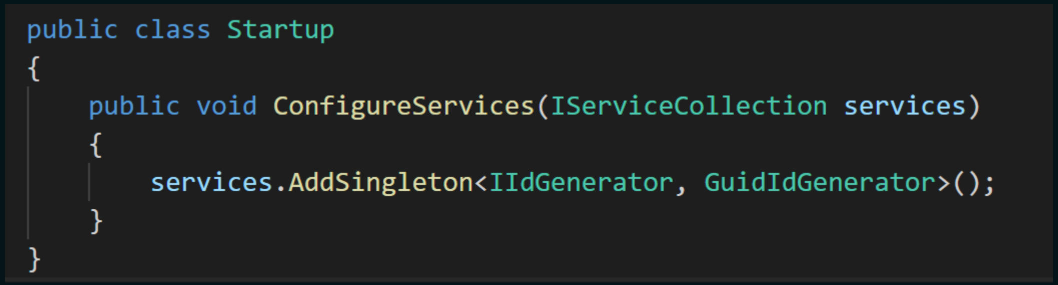


Samples

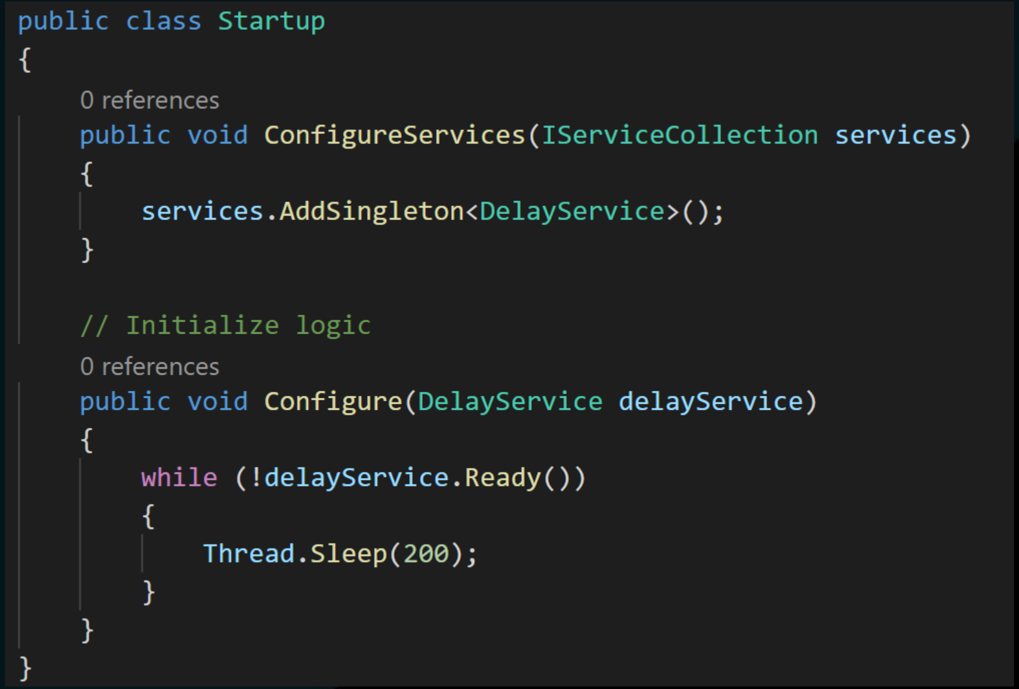
.NET Conf China 2020
2020 中国 .NET 开发者大会



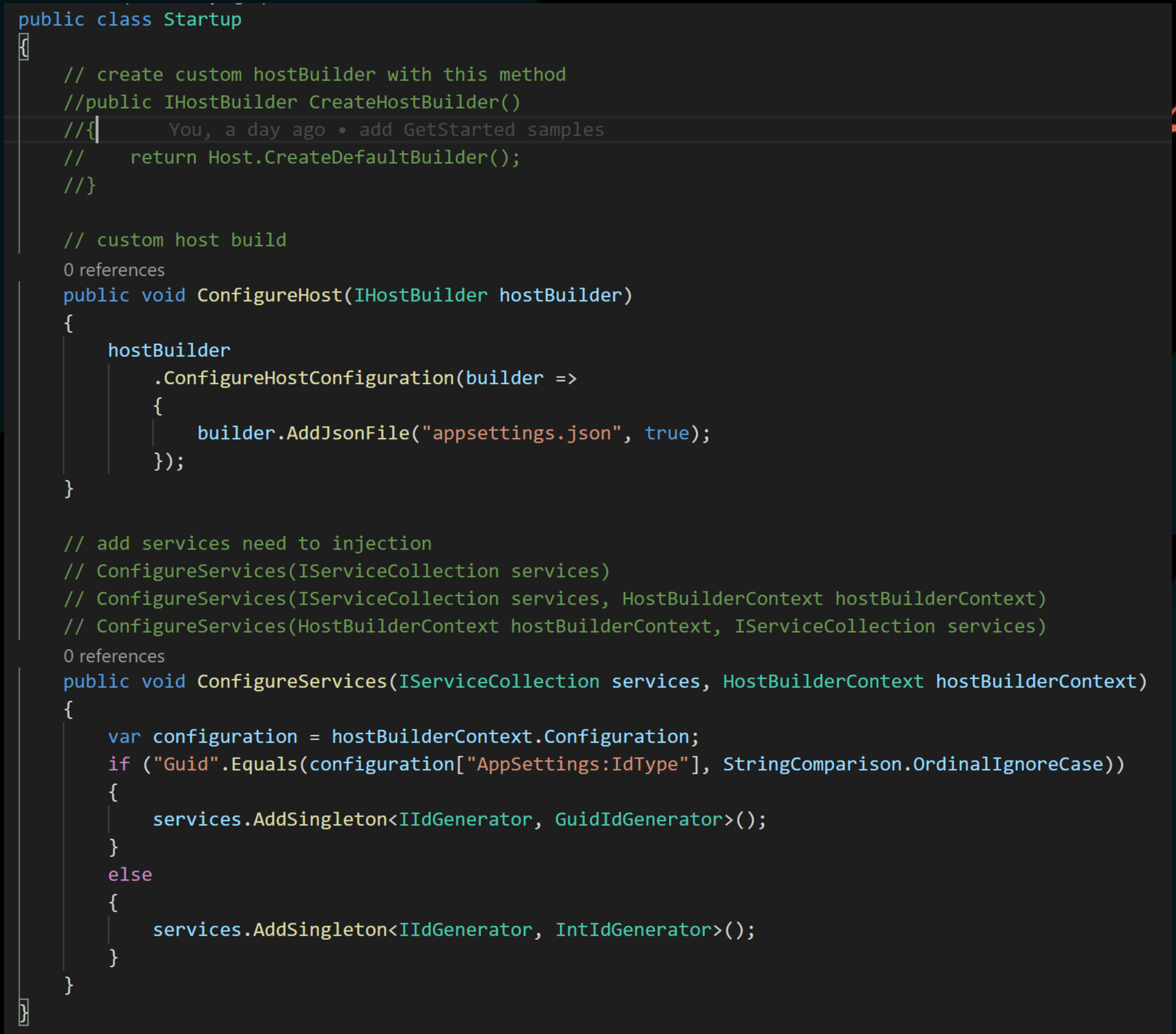
Get Started



```
public class Startup
{
    public void ConfigureServices(IServiceCollection services)
    {
        services.AddSingleton<IIdGenerator, GuidIdGenerator>();
    }
}
```

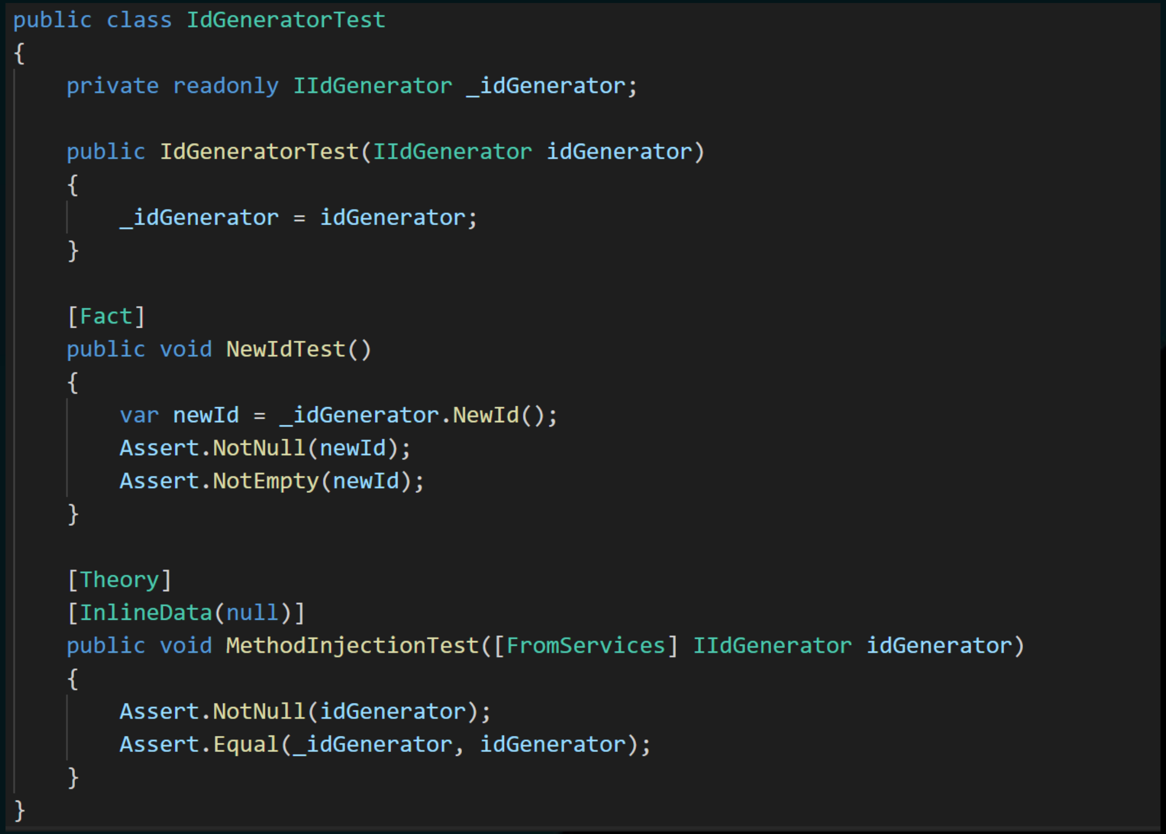


```
public class Startup
{
    0 references
    public void ConfigureServices(IServiceCollection services)
    {
        services.AddSingleton<DelayService>();
    }

    // Initialize logic
    0 references
    public void Configure(DelayService delayService)
    {
        while (!delayService.Ready())
        {
            Thread.Sleep(200);
        }
    }
}
```

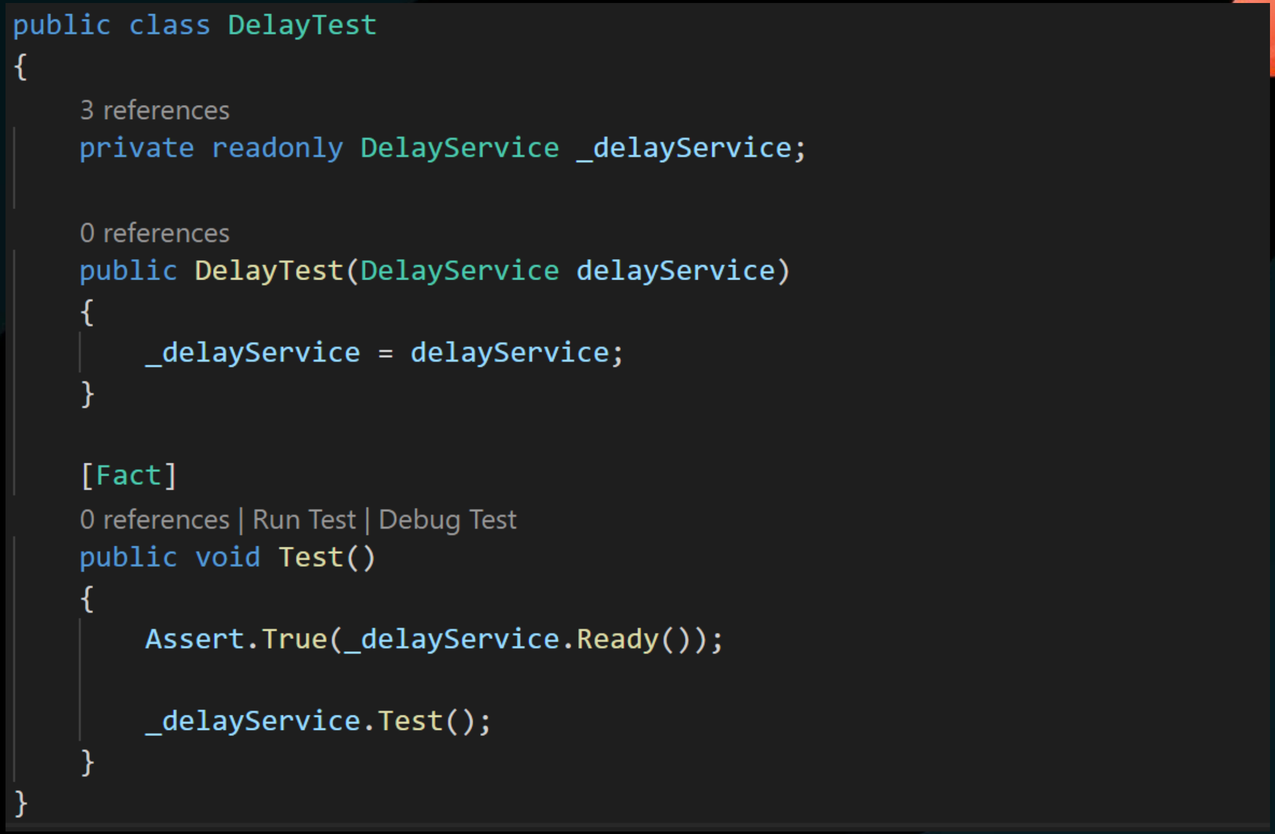


```
public class Startup
{
    // create custom hostBuilder with this method
    //public IHostBuilder CreateHostBuilder()
    //{       You, a day ago • add GetStarted samples
    //    return Host.CreateDefaultBuilder();
    //}

    // custom host build
    0 references
    public void ConfigureHost(IHostBuilder hostBuilder)
    {
        hostBuilder
            .ConfigureHostConfiguration(builder =>
            {
                builder.AddJsonFile("appsettings.json", true);
            });
    }

    // add services need to injection
    // ConfigureServices(IServiceCollection services)
    // ConfigureServices(IServiceCollection services, HostBuilderContext hostBuilderContext)
    // ConfigureServices(HostBuilderContext hostBuilderContext, IServiceCollection services)
    0 references
    public void ConfigureServices(IServiceCollection services, HostBuilderContext hostBuilderContext)
    {
        var configuration = hostBuilderContext.Configuration;
        if ("Guid".Equals(configuration["AppSettings:IdType"], StringComparison.OrdinalIgnoreCase))
        {
            services.AddSingleton<IIdGenerator, GuidIdGenerator>();
        }
        else
        {
            services.AddSingleton<IIdGenerator, IntIdGenerator>();
        }
    }
}
```

Get Started



```
public class IdGeneratorTest
{
    private readonly IIdGenerator _idGenerator;

    public IdGeneratorTest(IIdGenerator idGenerator)
    {
        _idGenerator = idGenerator;
    }

    [Fact]
    public void NewIdTest()
    {
        var newId = _idGenerator.NewId();
        Assert.NotNull(newId);
        Assert.NotEmpty(newId);
    }

    [Theory]
    [InlineData(null)]
    public void MethodInjectionTest([FromServices] IIdGenerator idGenerator)
    {
        Assert.NotNull(idGenerator);
        Assert.Equal(_idGenerator, idGenerator);
    }
}
```



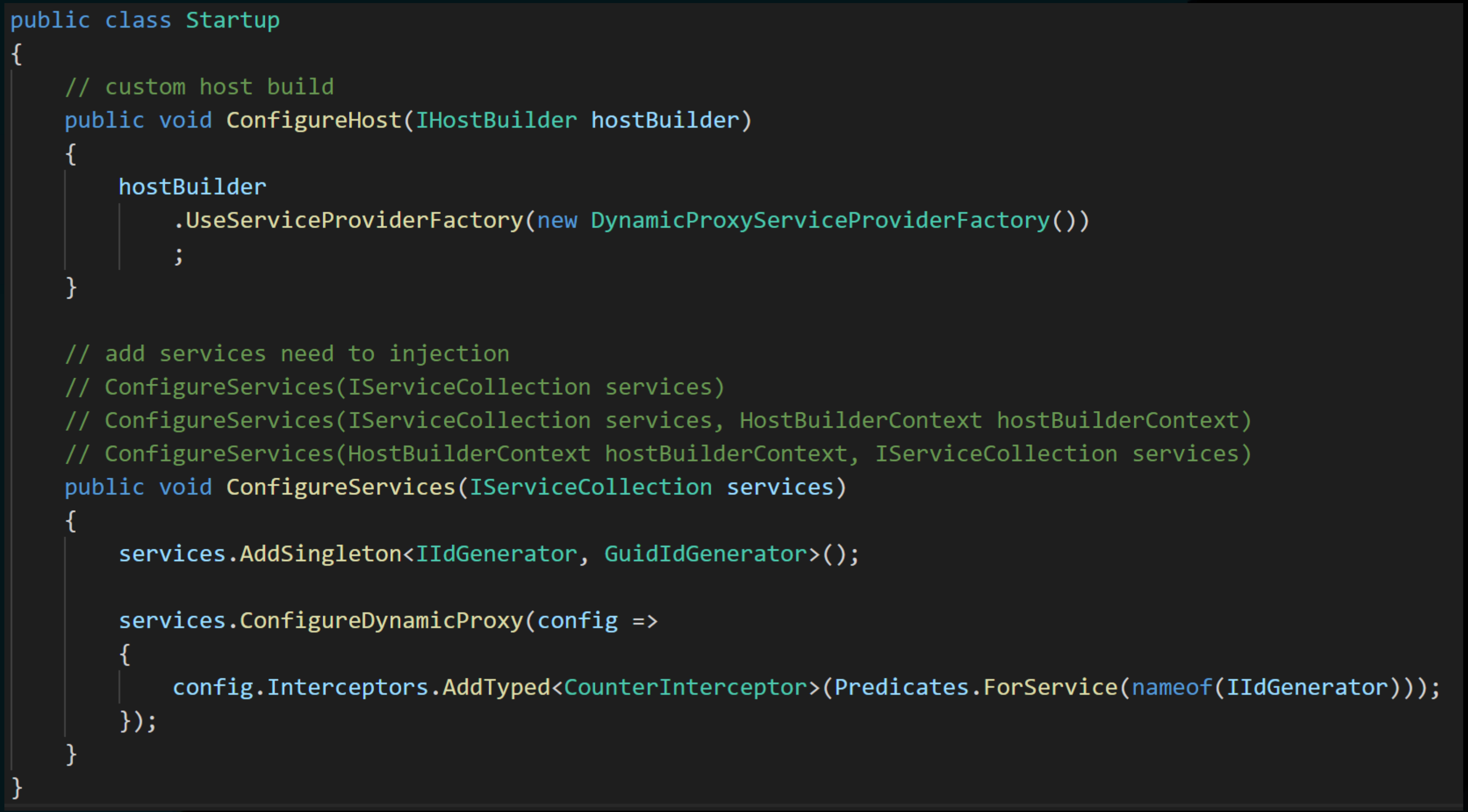
```
public class DelayTest
{
    3 references
    private readonly DelayService _delayService;

    0 references
    public DelayTest(DelayService delayService)
    {
        _delayService = delayService;
    }

    [Fact]
    0 references | Run Test | Debug Test
    public void Test()
    {
        Assert.True(_delayService.Ready());

        _delayService.Test();
    }
}
```

Autofac Integration

```csharp
public class Startup
{
    // custom host build
    public void ConfigureHost(IHostBuilder hostBuilder)
    {
        hostBuilder.UseServiceProviderFactory(new AutofacServiceProviderFactory(builder =>
        {
            builder.RegisterType<GuidIdGenerator>()
                .As<IIdGenerator>()
                .SingleInstance()
                ;
        }));
    }
}
```

AspectCore Integration

.NET Conf China 2020
2020 中国 .NET 开发者大会



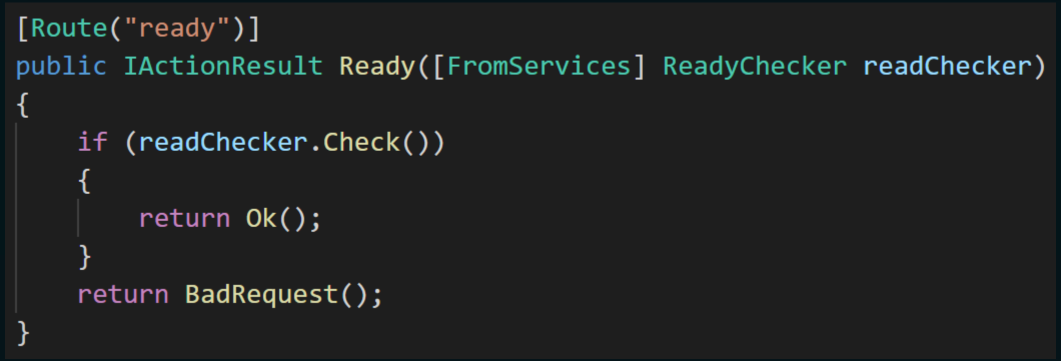
```csharp
public class Startup
{
    // custom host build
    public void ConfigureHost(IHostBuilder hostBuilder)
    {
        hostBuilder
            .UseServiceProviderFactory(new DynamicProxyServiceProviderFactory())
            ;
    }

    // add services need to injection
    // ConfigureServices(IServiceCollection services)
    // ConfigureServices(IServiceCollection services, HostBuilderContext hostBuilderContext)
    // ConfigureServices(HostBuilderContext hostBuilderContext, IServiceCollection services)
    public void ConfigureServices(IServiceCollection services)
    {
        services.AddSingleton<IIdGenerator, GuidIdGenerator>();

        services.ConfigureDynamicProxy(config =>
        {
            config.Interceptors.AddTyped<CounterInterceptor>(Predicates.ForService(nameof(IIdGenerator)));
        });
    }
}
```

TestServer Integration

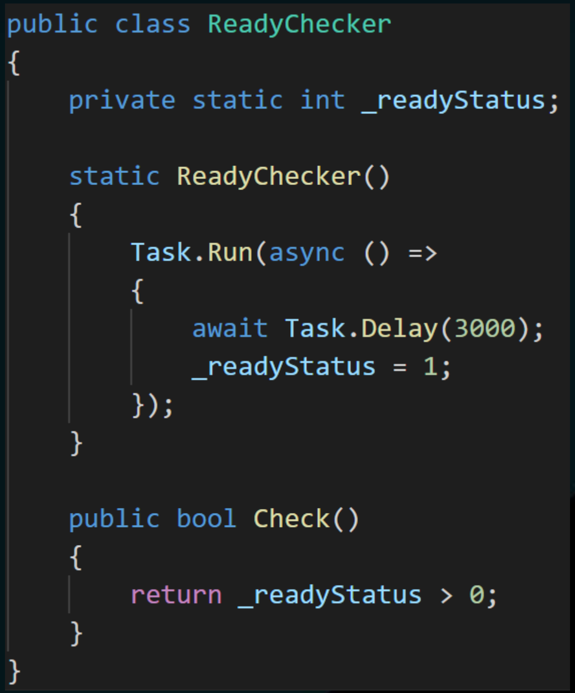
```csharp
public class Startup
{
    // custom host build
    public void ConfigureHost(IHostBuilder hostBuilder)
    {
        hostBuilder
            .ConfigureWebHostDefaults(builder =>
            {
                builder.UseStartup<TestWebApp.Startup>();

                builder.UseTestServer();
                builder.ConfigureServices(services =>
                {
                    services.AddSingleton(sp => sp.GetRequiredService<IHost>()
                        .GetTestClient()
                    );
                });
            })
            ;
    }
}
```

```csharp
public class ApiTest
{
    private readonly HttpClient _httpClient;

    public ApiTest(HttpClient httpClient)
    {
        _httpClient = httpClient;
    }

    [Fact]
    public async Task GetTest()
    {
        var response = await _httpClient.GetAsync("api/test");
        Assert.True(response.IsSuccessStatusCode);

        var responseText = await response.Content.ReadAsStringAsync();
        Assert.NotEmpty(responseText);

        var result = JsonSerializer.Deserialize<Result<bool>>(responseText, new JsonSerializerOptions()
        {
            PropertyNamingPolicy = JsonNamingPolicy.CamelCase,
        });
        Assert.NotNull(result);
        Assert.True(result.Data);
    }
}
```

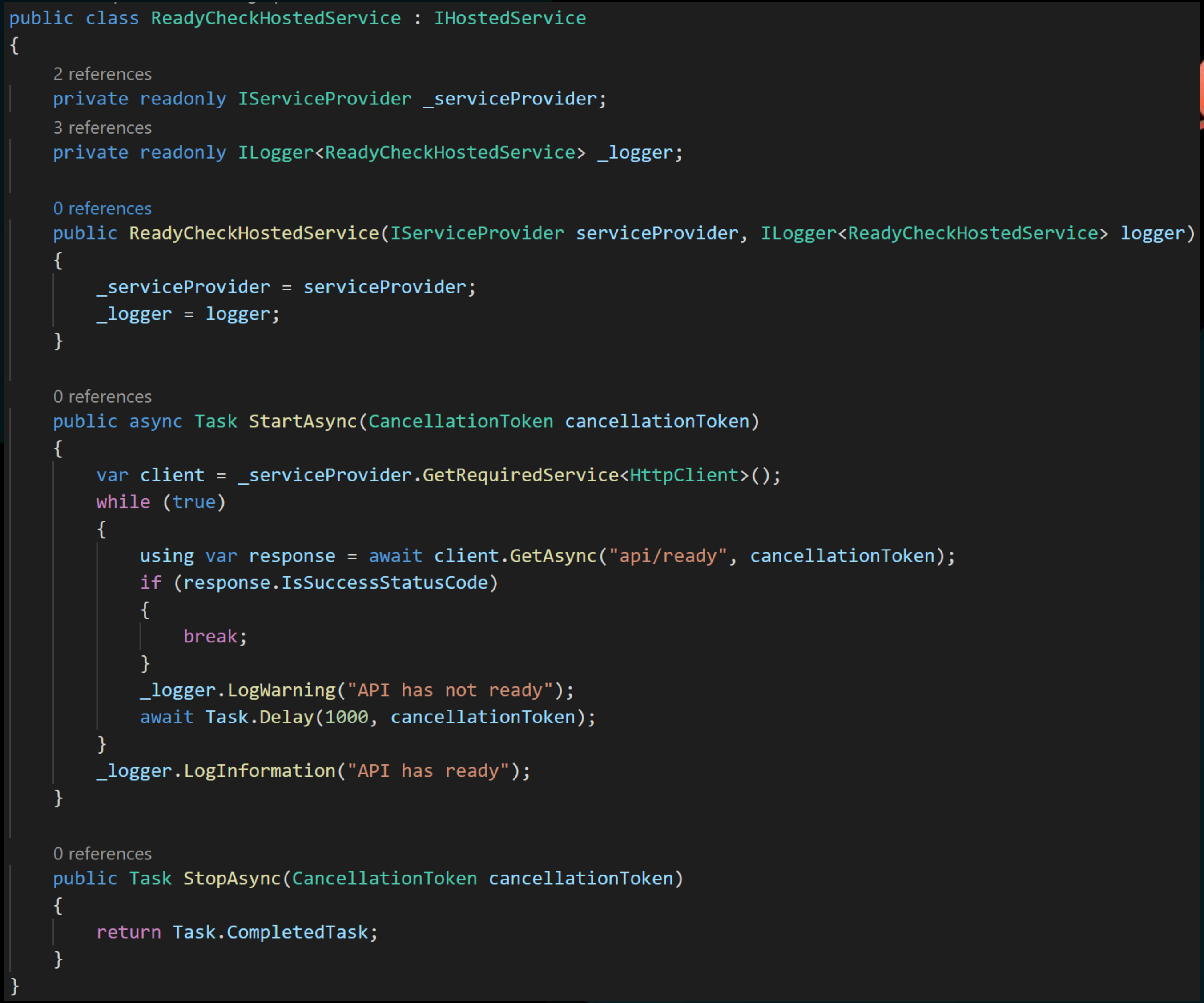
Hosted Service



```csharp
[Route("ready")]
public IActionResult Ready([FromServices] ReadyChecker readChecker)
{
    if (readChecker.Check())
    {
        return Ok();
    }
    return BadRequest();
}
```

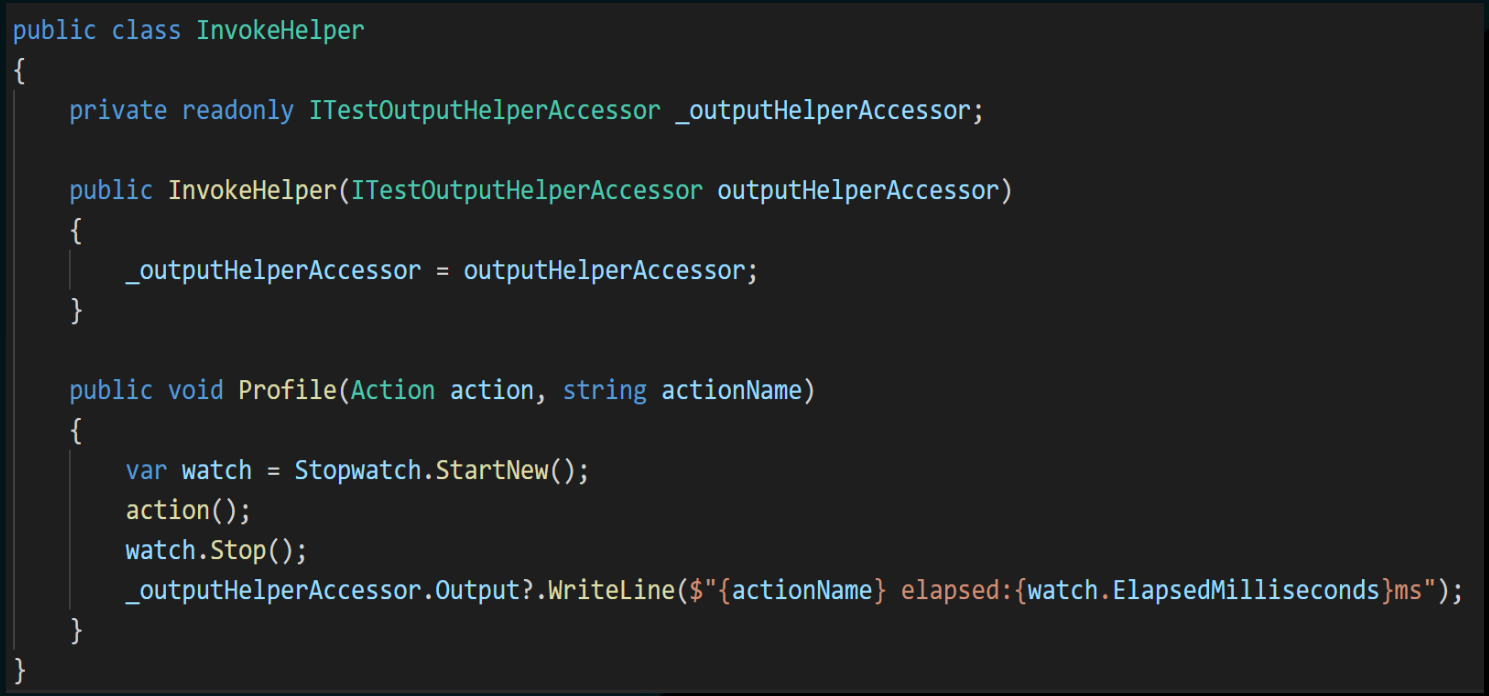


```csharp
public class ReadyChecker
{
    private static int _readyStatus;

    static ReadyChecker()
    {
        Task.Run(async () =>
        {
            await Task.Delay(3000);
            _readyStatus = 1;
        });
    }

    public bool Check()
    {
        return _readyStatus > 0;
    }
}
```

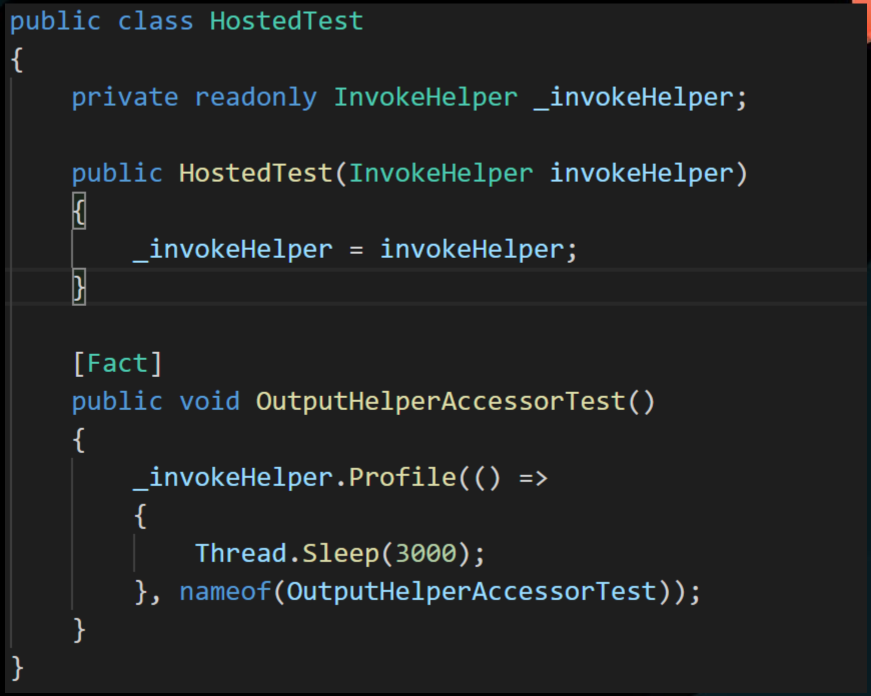


```csharp
public class ReadyCheckHostedService : IHostedService
{
    2 references
    private readonly IServiceProvider _serviceProvider;
    3 references
    private readonly ILogger<ReadyCheckHostedService> _logger;

    0 references
    public ReadyCheckHostedService(IServiceProvider serviceProvider, ILogger<ReadyCheckHostedService> logger)
    {
        _serviceProvider = serviceProvider;
        _logger = logger;
    }

    0 references
    public async Task StartAsync(CancellationToken cancellationToken)
    {
        var client = _serviceProvider.GetRequiredService<HttpClient>();
        while (true)
        {
            using var response = await client.GetAsync("api/ready", cancellationToken);
            if (response.IsSuccessStatusCode)
            {
                break;
            }
            _logger.LogWarning("API has not ready");
            await Task.Delay(1000, cancellationToken);
        }
        _logger.LogInformation("API has ready");
    }

    0 references
    public Task StopAsync(CancellationToken cancellationToken)
    {
        return Task.CompletedTask;
    }
}
```

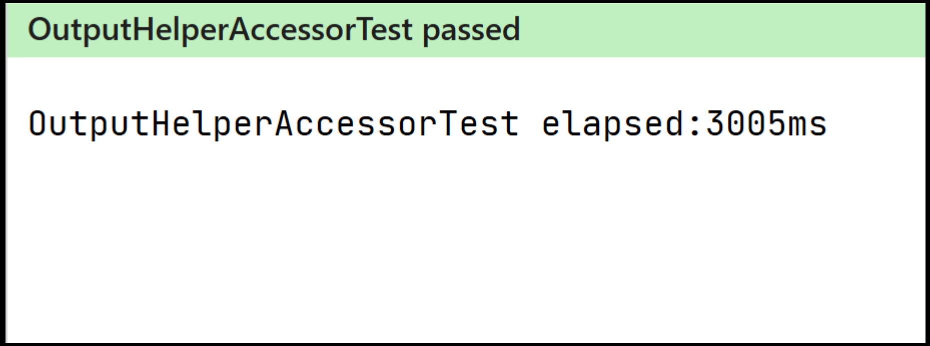
TestOutputHelperAccessor



```csharp
public class InvokeHelper
{
    private readonly ITestOutputHelperAccessor _outputHelperAccessor;

    public InvokeHelper(ITestOutputHelperAccessor outputHelperAccessor)
    {
        _outputHelperAccessor = outputHelperAccessor;
    }

    public void Profile(Action action, string actionName)
    {
        var watch = Stopwatch.StartNew();
        action();
        watch.Stop();
        _outputHelperAccessor.Output?.WriteLine($"{actionName} elapsed:{watch.ElapsedMilliseconds}ms");
    }
}
```



```csharp
public class HostedTest
{
    private readonly InvokeHelper _invokeHelper;

    public HostedTest(InvokeHelper invokeHelper)
    {
        _invokeHelper = invokeHelper;
    }

    [Fact]
    public void OutputHelperAccessorTest()
    {
        _invokeHelper.Profile(() =>
        {
            Thread.Sleep(3000);
        }, nameof(OutputHelperAccessorTest));
    }
}
```

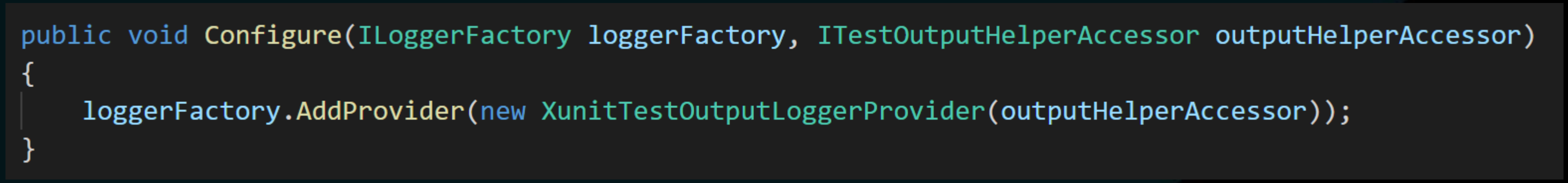


OutputHelperAccessorTest passed

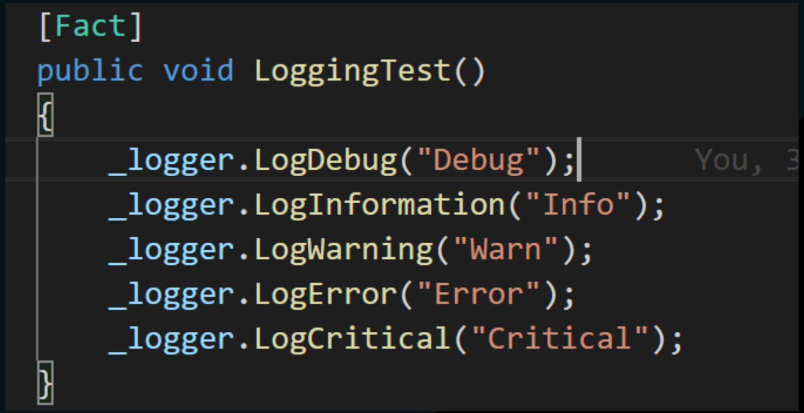
```
OutputHelperAccessorTest elapsed:3005ms
```

Logging

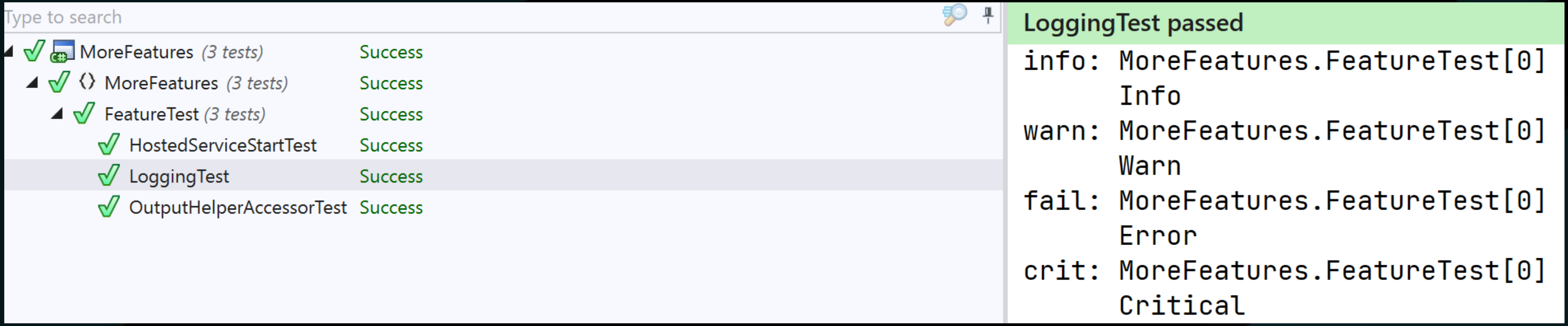
.NET Conf China 2020
2020 中国 .NET 开发者大会



```
public void Configure(ILoggerFactory loggerFactory, ITestOutputHelperAccessor outputHelperAccessor)
{
    loggerFactory.AddProvider(new XunitTestOutputLoggerProvider(outputHelperAccessor));
}
```



```
[Fact]
public void LoggingTest()
{
    _logger.LogDebug("Debug");        You, 3
    _logger.LogInformation("Info");
    _logger.LogWarning("Warn");
    _logger.LogError("Error");
    _logger.LogCritical("Critical");
}
```



Type to search

```
MoreFeatures (3 tests)          Success
  MoreFeatures (3 tests)        Success
    FeatureTest (3 tests)       Success
      HostedServiceStartTest    Success
      LoggingTest               Success
      OutputHelperAccessorTest  Success
```

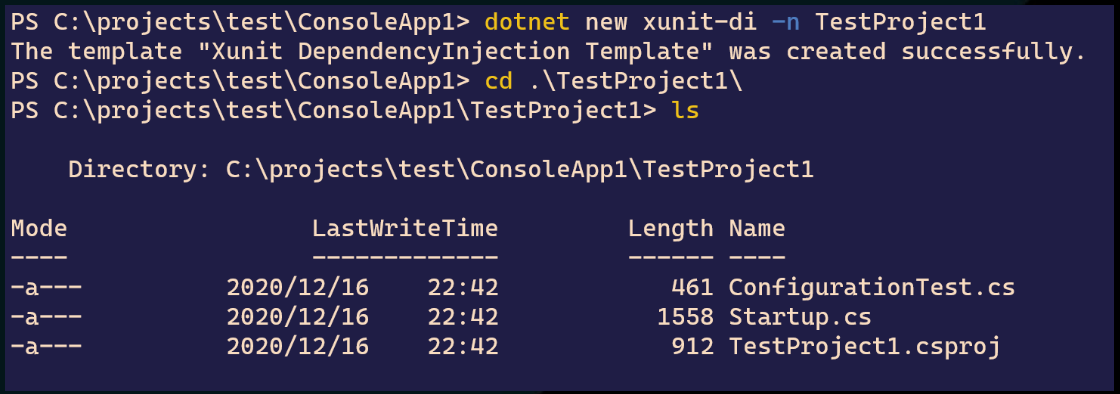
LoggingTest passed

```
info: MoreFeatures.FeatureTest[0]
      Info
warn: MoreFeatures.FeatureTest[0]
      Warn
fail: MoreFeatures.FeatureTest[0]
      Error
crit: MoreFeatures.FeatureTest[0]
      Critical
```

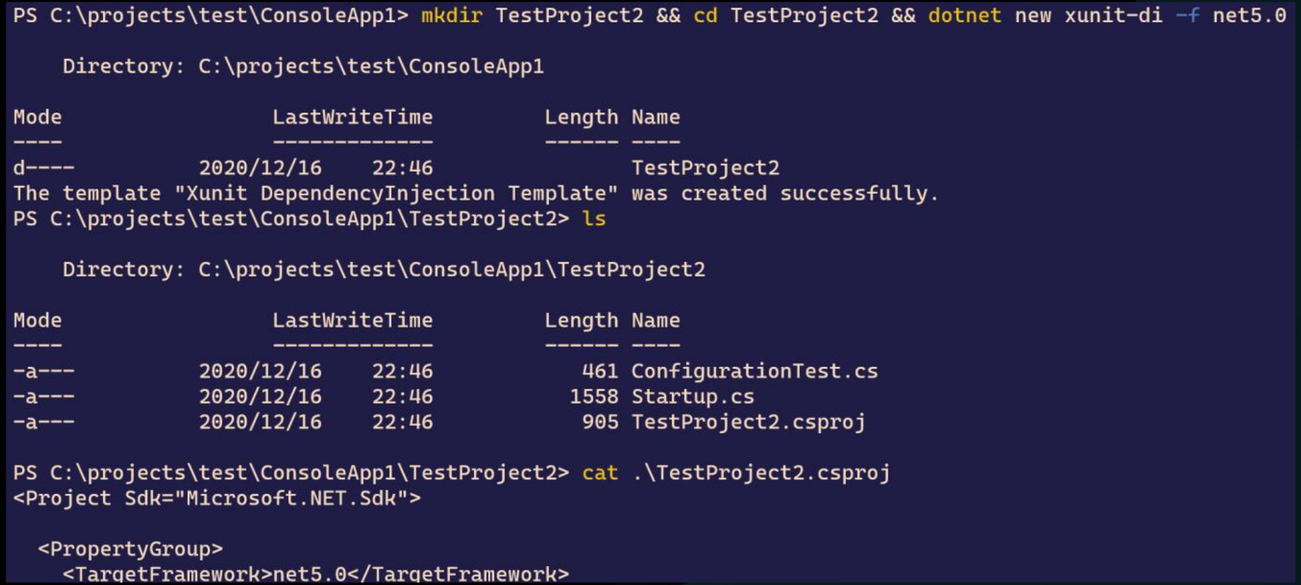
Project Template

安装项目模板：dotnet new -i Xunit.DependencyInjection.Template

创建项目:
dotnet new xunit-di –n TestProjectName
dotnet new xunit-di –n TestProject1 –f net5.0
makedir TestProject2 && cd TestProject2 && dotnet new xunit-di –f net5.0

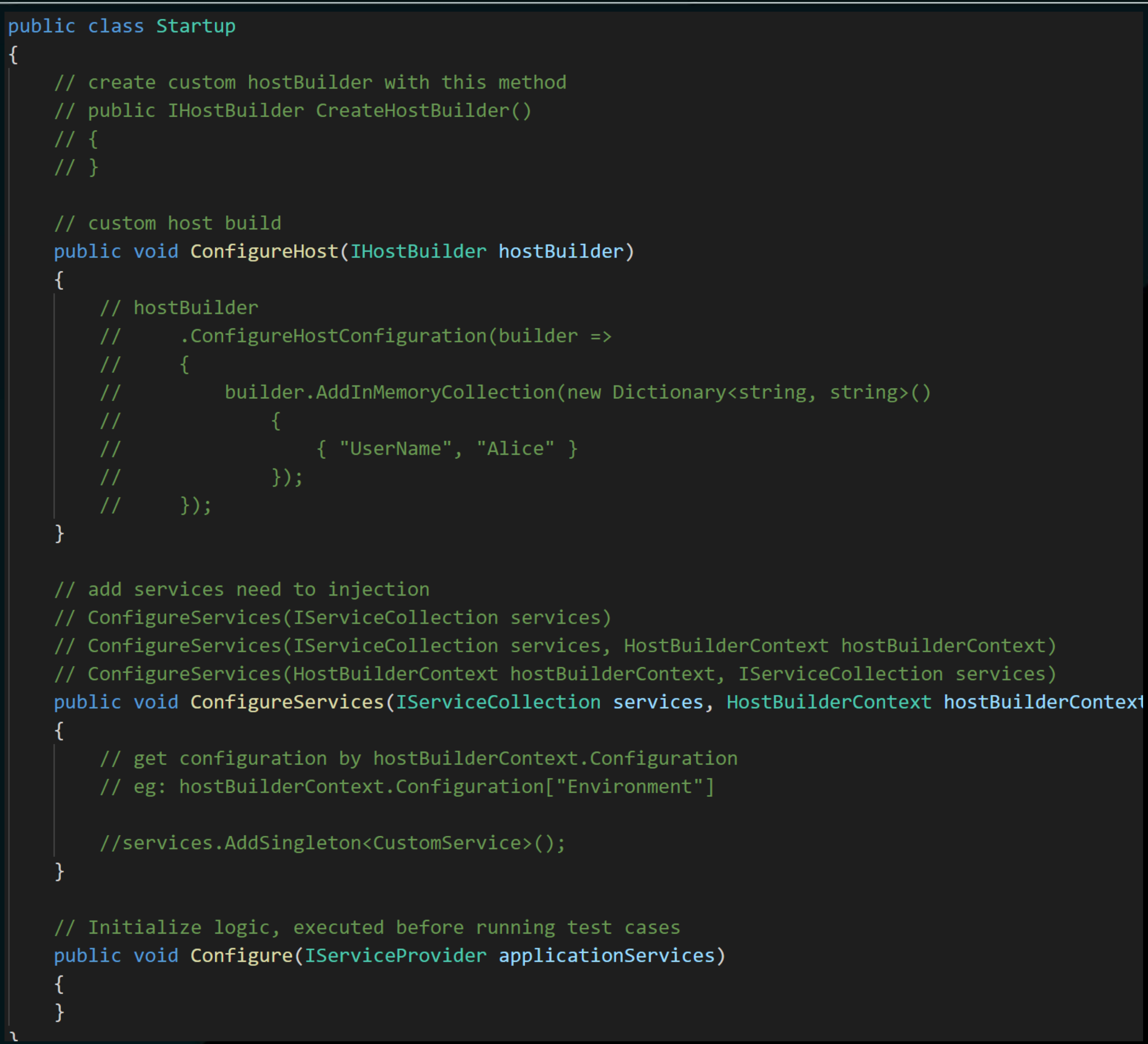


```
PS C:\projects\test\ConsoleApp1> dotnet new xunit-di -n TestProject1
The template "Xunit DependencyInjection Template" was created successfully.
PS C:\projects\test\ConsoleApp1> cd .\TestProject1\
PS C:\projects\test\ConsoleApp1\TestProject1> ls

    Directory: C:\projects\test\ConsoleApp1\TestProject1

Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
-a---          2020/12/16     22:42            461 ConfigurationTest.cs
-a---          2020/12/16     22:42           1558 Startup.cs
-a---          2020/12/16     22:42            912 TestProject1.csproj
```

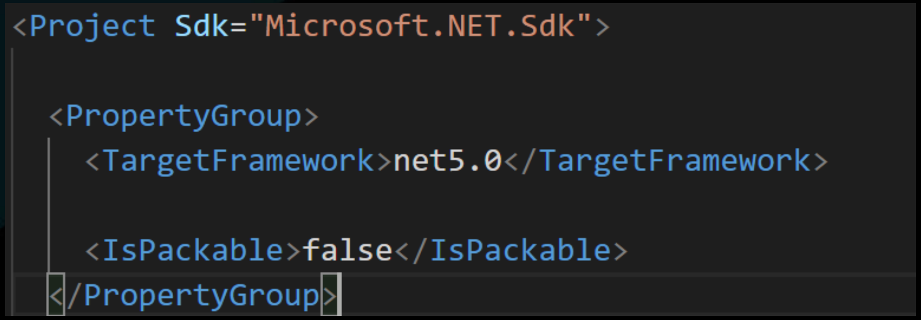


```
PS C:\projects\test\ConsoleApp1> mkdir TestProject2 && cd TestProject2 && dotnet new xunit-di -f net5.0

    Directory: C:\projects\test\ConsoleApp1

Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
d----          2020/12/16     22:46                TestProject2
The template "Xunit DependencyInjection Template" was created successfully.
PS C:\projects\test\ConsoleApp1\TestProject2> ls

    Directory: C:\projects\test\ConsoleApp1\TestProject2

Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
-a---          2020/12/16     22:46            461 ConfigurationTest.cs
-a---          2020/12/16     22:46           1558 Startup.cs
-a---          2020/12/16     22:46            905 TestProject2.csproj

PS C:\projects\test\ConsoleApp1\TestProject2> cat .\TestProject2.csproj
<Project Sdk="Microsoft.NET.Sdk">

  <PropertyGroup>
    <TargetFramework>net5.0</TargetFramework>
```

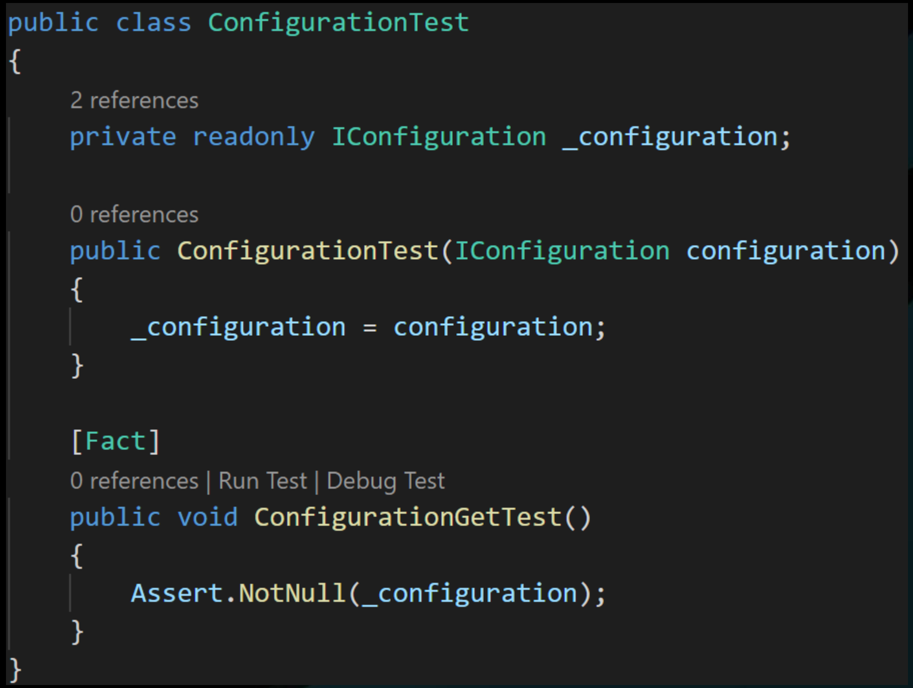
Project Template



```
public class Startup
{
    // create custom hostBuilder with this method
    // public IHostBuilder CreateHostBuilder()
    // {
    // }

    // custom host build
    public void ConfigureHost(IHostBuilder hostBuilder)
    {
        // hostBuilder
        //     .ConfigureHostConfiguration(builder =>
        //     {
        //         builder.AddInMemoryCollection(new Dictionary<string, string>()
        //             {
        //                 { "UserName", "Alice" }
        //             });
        //     });
    }

    // add services need to injection
    // ConfigureServices(IServiceCollection services)
    // ConfigureServices(IServiceCollection services, HostBuilderContext hostBuilderContext)
    // ConfigureServices(HostBuilderContext hostBuilderContext, IServiceCollection services)
    public void ConfigureServices(IServiceCollection services, HostBuilderContext hostBuilderContex
    {
        // get configuration by hostBuilderContext.Configuration
        // eg: hostBuilderContext.Configuration["Environment"]

        //services.AddSingleton<CustomService>();
    }

    // Initialize logic, executed before running test cases
    public void Configure(IServiceProvider applicationServices)
    {
    }
}
```



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

  <PropertyGroup>
    <TargetFramework>net5.0</TargetFramework>

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



```
public class ConfigurationTest
{
    2 references
    private readonly IConfiguration _configuration;

    0 references
    public ConfigurationTest(IConfiguration configuration)
    {
        _configuration = configuration;
    }

    [Fact]
    0 references | Run Test | Debug Test
    public void ConfigurationGetTest()
    {
        Assert.NotNull(_configuration);
    }
}
```

Reference

- https://github.com/pengweiqhca/Xunit.DependencyInjection
- https://github.com/WeihanLi/XunitDependencyInjection.Samples
- https://www.nuget.org/packages/Xunit.DependencyInjection
- https://www.nuget.org/packages/Microsoft.AspNetCore.TestHost
- https://www.nuget.org/packages/Xunit.DependencyInjection.Logging
- https://www.nuget.org/packages/Xunit.DependencyInjection.Template/

Thank You