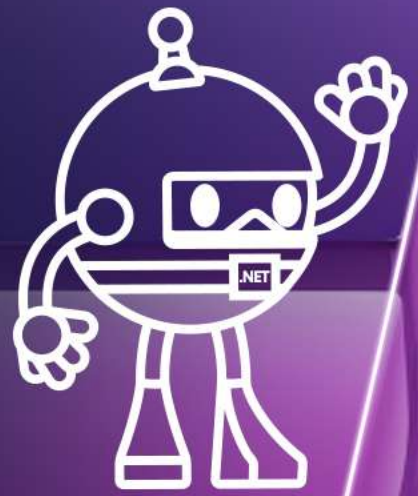
.NET Conf China 2023

2023/12/16
09:30 - 18:00

中国 • 北京



.NET中文社区

中国 · 北京

.NET Conf China 2023

构建更简单的 C# dotnet-exec

李卫涵/WeihanLi

iHerb .NET 开发工程师/微软 MVP

Microsoft® Most Valuable Professional

目录
AGENDA

01 C# Evolution

02 What’s it

03 How it works

04 How to use

C# Simplify

Program.cs > ...

```
using System;

class Program
{
    public static void Main()
    {
        Console.WriteLine("Hello World");
    }
}
```

```
Console.WriteLine("Hello World");
```

Top Level Statement

Implicit Using

ProjectFile Simplify

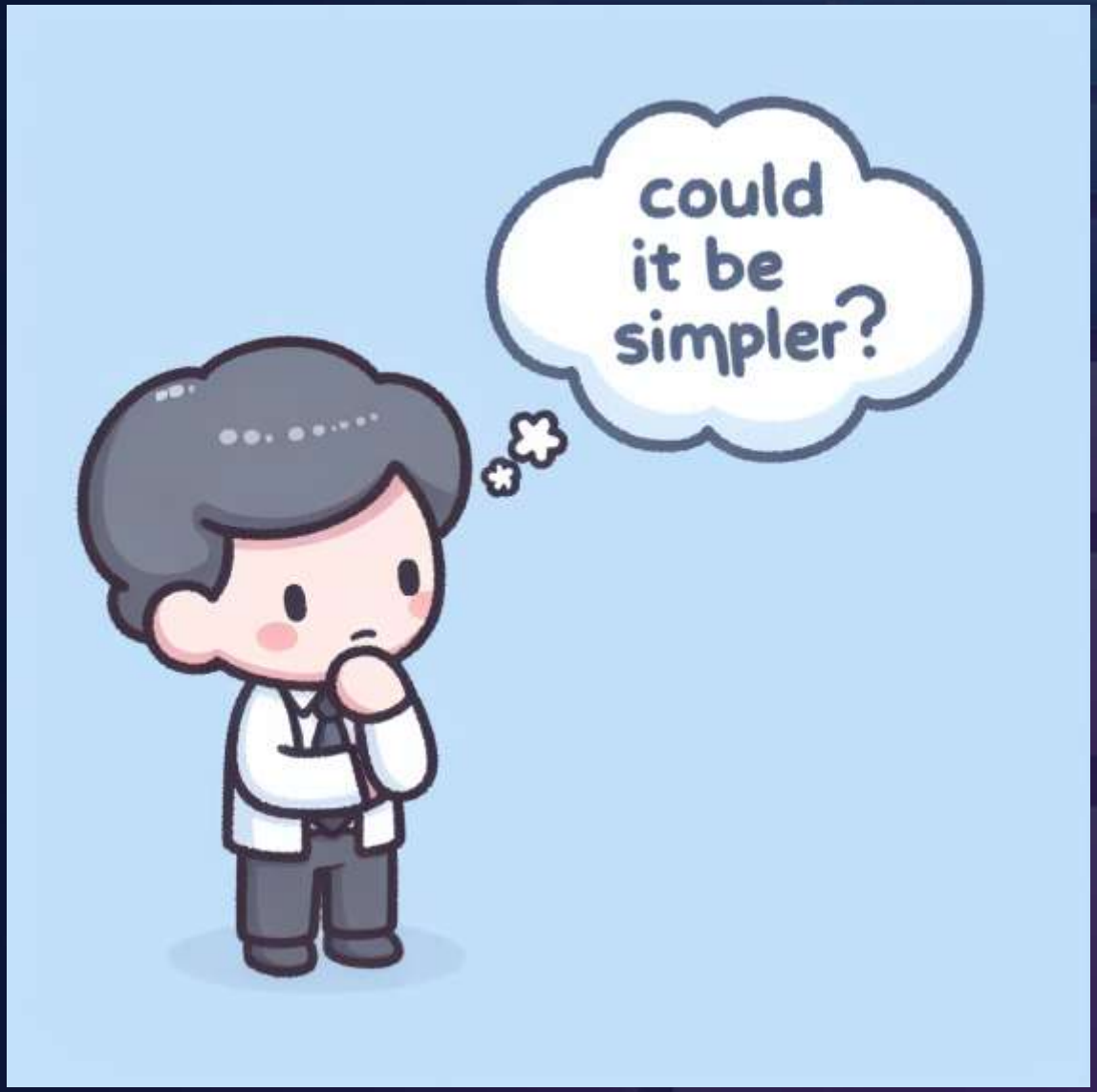
```
<?xml version="1.0" encoding="utf-8"?>
<Project ToolsVersion="14.0" DefaultTargets="Build" xmlns="http://schemas.microsoft.com/developer/msbuild/2003">
  <Import Project="$(MSBuildExtensionsPath)\$(MSBuildToolsVersion)\Microsoft.Common.props" Condition="Exists('$(MSBuildExtensionsPath)\$(MSBuildT
  <PropertyGroup>
    <Configuration Condition=" '$(Configuration)' == '' ">Debug</Configuration>
    <Platform Condition=" '$(Platform)' == '' ">AnyCPU</Platform>
    <ProjectGuid>{E00527DF-512D-47B6-B341-C090F0D01DA7}</ProjectGuid>
    <AssemblyName>HelloWorld</AssemblyName>
    <OutputType>Exe</OutputType>
    <TargetFrameworkVersion>v4.8</TargetFrameworkVersion>
  </PropertyGroup>
  <PropertyGroup Condition=" '$(Configuration)|$(Platform)' == 'Debug|AnyCPU' ">
    <PlatformTarget>AnyCPU</PlatformTarget>
    <DebugSymbols>true</DebugSymbols>
    <DebugType>full</DebugType>
    <Optimize>false</Optimize>
    <OutputPath>bin\Debug\</OutputPath>
    <DefineConstants>DEBUG;TRACE</DefineConstants>
    <ErrorReport>prompt</ErrorReport>
    <WarningLevel>4</WarningLevel>
  </PropertyGroup>
  <PropertyGroup Condition=" '$(Configuration)|$(Platform)' == 'Release|AnyCPU' ">
    <PlatformTarget>AnyCPU</PlatformTarget>
    <DebugType>pdbonly</DebugType>
    <Optimize>true</Optimize>
    <OutputPath>bin\Release\</OutputPath>
    <DefineConstants>TRACE</DefineConstants>
    <ErrorReport>prompt</ErrorReport>
    <WarningLevel>4</WarningLevel>
  </PropertyGroup>
  <ItemGroup>
    <Reference Include="System" />
  </ItemGroup>
  <ItemGroup>
    <Compile Include="Program.cs" />
  </ItemGroup>
  <Import Project="$(MSBuildToolsPath)\Microsoft.CSharp.targets" />
</Project>
```

Project File Simplify

```
<Project Sdk="Microsoft.NET.Sdk">
  <PropertyGroup>
    <TargetFramework>net8.0</TargetFramework>
    <OutputType>Exe</OutputType>
    <ImplicitUsings>enable</ImplicitUsings>
    <AssemblyName>HelloWorld</AssemblyName>
  </PropertyGroup>
</Project>
```

DefaultItems

Sdk/Framework Reference


could
it be
simpler?

更简单的 C# ？

```
[pipeline@weihanli-asia-01 scripts]$ echo 'Console.WriteLine("Hello dotnet-exec");' > hello.cs
[pipeline@weihanli-asia-01 scripts]$ cat hello.cs
Console.WriteLine("Hello dotnet-exec");
[pipeline@weihanli-asia-01 scripts]$ dotnet-exec hello.cs
Hello dotnet-exec
[pipeline@weihanli-asia-01 scripts]$
```

```
[pipeline@weihanli-asia-01 scripts]$ dotnet-exec "Console.WriteLine(Guid.NewGuid());"
31f7f5d4-9aa9-4d5f-ad93-8b1885a0e1bf
[pipeline@weihanli-asia-01 scripts]$
```

```
[pipeline@weihanli-asia-01 scripts]$ dotnet-exec "Guid.NewGuid()"
[a645fa6c-0a96-4b7f-855b-9748e4de02c6]
[pipeline@weihanli-asia-01 scripts]$
```

更简单的 C# ？

```
[pipeline@weihanli-asia-01 scripts]$ cat ./TestClass.cs
public class TestClass
{
        public static void TestMethod1()
        {
                Console.WriteLine("TestMethod1 invoked");
        }

        public static void TestMethod2()
        {
                Console.WriteLine("TestMethod2 invoked");
        }
}
[pipeline@weihanli-asia-01 scripts]$ dotnet-exec ./TestClass.cs  --entry TestMethod1
TestMethod1 invoked
[pipeline@weihanli-asia-01 scripts]$ dotnet-exec ./TestClass.cs  --entry TestMethod2
TestMethod2 invoked
[pipeline@weihanli-asia-01 scripts]$
```

What’s it – dotnet tool

- Use as a dotnet global tool

```
dotnet tool update -g dotnet-execute
```

```
dotnet tool update -g dotnet-execute --prerelease
```

What’s it – container image

```
docker run --rm weihanli/dotnet-exec:latest "Guid.NewGuid()"
```

```
[pipeline@weihanli-asia-01 ~]$ docker run --rm --pull=always weihanli/dotnet-exec:latest dotnet-exec "ApplicationHelper
.RuntimeInfo"
latest: Pulling from weihanli/dotnet-exec
Digest: sha256:d3ce1a911100d8385257036e069b70efef9ee6690a4795e297421409709927c7
Status: Downloaded newer image for weihanli/dotnet-exec:latest
RuntimeInfo { FrameworkDescription=".NET 9.0.0-preview.1.24080.9", IsInContainer=true, IsInKubernetes=false, LibraryHas
h="1d1bf92fcf43aa6981804dc53c5174445069c9e4", LibraryVersion="9.0.0-preview.1.24080.9", MachineName="61c4966ef30f", OSA
rchitecture="X64", OSDescription="Alpine Linux v3.19", OSVersion="Unix 4.18.0.193", ProcessId=1, ProcessorCount=2, Proc
essPath="/app/dotnet-exec", RepositoryUrl="https://github.com/dotnet/runtime", RuntimeIdentifier="linux-musl-x64", Vers
ion="9.0.0", WorkingDirectory="/app" }
```

What’s it – container image

TAG

0.18.0-web

Last pushed 6 hours ago by weihanli

```
docker pull weihanli/dotnet-exec:0.18.0-web
```
Copy

Digest	OS/ARCH	Compressed Size
6a22d51cc17e	linux/amd64	53.38 MB
19ca1d1d7ed1	linux/arm/v7	50.85 MB
bb7cbbf309df	linux/arm64	51.59 MB

TAG

0.18.0

Last pushed 6 hours ago by weihanli

```
docker pull weihanli/dotnet-exec:0.18.0
```
Copy

Digest	OS/ARCH	Compressed Size
b0664c8e768b	linux/amd64	42.87 MB
130416bd3648	linux/arm/v7	40.4 MB
d63ed3c92c1e	linux/arm64	41.4 MB

What’s it – container image

```
[pipeline@weihanli-asia-01 ~]$ docker run --rm weihanli/dotnet-exec:0.18.0 "ApplicationHelper.RuntimeInfo"
RuntimeInfo { FrameworkDescription=".NET 9.0.0-preview.1.24080.9", IsInContainer=true, IsInKubernetes=false, LibraryHash
="1d1bf92fcf43aa6981804dc53c5174445069c9e4", LibraryVersion="9.0.0-preview.1.24080.9", MachineName="a12c595e03ad", OSArc
hitecture="X64", OSDescription="Alpine Linux v3.19", OSVersion="Unix 4.18.0.193", ProcessId=1, ProcessorCount=2, Process
Path="/app/dotnet-exec", RepositoryUrl="https://github.com/dotnet/runtime", RuntimeIdentifier="linux-musl-x64", Version=
"9.0.0", WorkingDirectory="/app" }
```

```
weihanli@s0-arm-01:~$ docker run --rm weihanli/dotnet-exec:0.18.0 "ApplicationHelper.RuntimeInfo"
RuntimeInfo { FrameworkDescription=".NET 9.0.0-preview.1.24080.9", IsInContainer=true, IsInKubernetes=false, LibraryHash
="1d1bf92fcf43aa6981804dc53c5174445069c9e4", LibraryVersion="9.0.0-preview.1.24080.9", MachineName="3a5aee4b6745", OSArc
hitecture="Arm64", OSDescription="Alpine Linux v3.19", OSVersion="Unix 5.15.0.1054", ProcessId=1, ProcessorCount=2, Proc
essPath="/app/dotnet-exec", RepositoryUrl="https://github.com/dotnet/runtime", RuntimeIdentifier="linux-musl-arm64", Ver
sion="9.0.0", WorkingDirectory="/app" }
```

How it works

获取代码 → 代码编译 → 代码执行

How it works

- Console
  - Main
- DLL
  - Custom Entry
  - Execute
- Script

How to use

```
PS C:\Users\weihan> dotnet-exec --help
Description:
  dotnet-exec, execute C# script/program from command line

Usage:
  dotnet-exec <script> [command] [options]

Arguments:
  <script>  CSharp script to execute

Options:
  -f, --framework <net6.0|net7.0|net8.0|net9.0>   Target framework [default: net9.0]
  --startup-type <startup-type>                   The startup type to use for finding the correct entry
  -e, --entry <entry>                             Custom entry point('MainTest' by default)
  --compiler, --compiler-type <simple|workspace>  The compiler to use [default: workspace]
  --executor, --executor-type <default>           The executor to use [default: default]
  --preview                                       Use preview language feature and enable preview features
  -c, --configuration <Debug|Release>             Compile configuration/OptimizationLevel
  --args, --arguments <arguments>                 Input arguments, please use `-- <args[0]> <args[1]>` instead
  --debug                                         Enable debug logs for debug
  --ref-compile                                   Use Ref assemblies for compile, when not found from local download
                                                  from nuget
  --project <project>                             The project file path to exact references and usings
  --wide                                          Includes widely-used
                                                  references(Microsoft.Extensions.Configuration/DependencyInjection/Logg
                                                  ing,Newtonsoft.Json,WeihanLi.Common) [default: True]
  -w, --web                                       Includes Web SDK references
  -r, --reference <reference>                     Additional references
  -u, --using <using>                             Namespace usings
  --ad, --addition <addition>                     Additional script path
  --generator                                     Enable the source generator support
  --profile <profile>                             The config profile to use
  --compile-symbol <compile-symbol>               Preprocessor symbol names for parsing and compiling
  --compile-feature <compile-feature>             Features for parsing and compiling
  --dry-run                                       Dry-run, would not execute script and output debug info
  --nuget-config <nuget-config>                   NuGet config file path to use
  --env <env>                                     Set environment variable for process, usage example: --env name=test
                                                  --env value=123
  --info                                          Tool version and runtime info
  --compile-out <compile-out>                     Compiled dll output path
  --version                                       Show version information
  -?, -h, --help                                  Show help and usage information


Commands:
  profile  Configure user config profile
```

Local Code file

```
PS C:\projects\sources\SamplesInPractice\CSharp11Sample> cat .\NameOfSample.cs
using System.ComponentModel.DataAnnotations;
using System.Reflection;
using System.Runtime.CompilerServices;

namespace CSharp11Sample;
internal static class NameOfSample
{
    [Display(Name = nameof(MainTest))]
    public static void MainTest()
    {
        var displayName = MethodBase.GetCurrentMethod()
            ?.GetCustomAttribute<DisplayAttribute>()
            ?.Name;
        Console.WriteLine(displayName);

        Hello(1 + 1 > 2);
    }

    private static void Hello(bool condition, [CallerArgumentExpression(nameof(condition))] string? expression = null)
    {
        Console.WriteLine($"{expression} : {condition}");
    }
}
PS C:\projects\sources\SamplesInPractice\CSharp11Sample> dotnet-exec .\NameOfSample.cs
MainTest
1 + 1 > 2 : False
```

Remote Code file

```
PS C:\projects\sources\SamplesInPractice\CSharp11Sample> dotnet-exec https://github.com/WeihanLi/SamplesInPractice/blob/
master/CSharp11Sample/NameOfSample.cs
MainTest
1 + 1 > 2 : False
```

```
PS C:\projects\sources\SamplesInPractice\CSharp11Sample> dotnet-exec https://raw.githubusercontent.com/WeihanLi/SamplesI
nPractice/master/CSharp11Sample/NameOfSample.cs
MainTest
1 + 1 > 2 : False
```

dotnet-exec test script

print-date.cs

Raw

```
Console.WriteLine($"Current date: {DateTime.Now.Date: yyyy-MM-dd}");
```

```
PS C:\Users\weihan> dotnet-exec https://gist.github.com/WeihanLi/7b4032a32f1a25c5f2d84b6955fa83f3
Current date:  2024-03-01
```

test.cs 33 Bytes

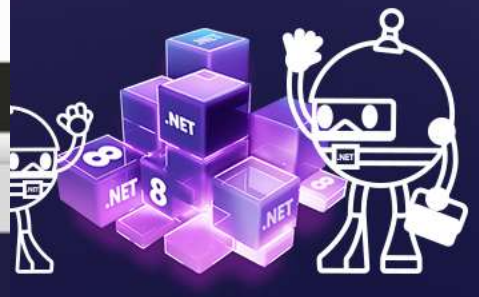
一键复制 编辑 Web IDE 原始数据

WeihanLi 提交于 刚刚 . add test/test.cs.

```
Console.WriteLine("Hello world");
```

PowerShell

```
PS C:\projects\sources\dotnet-exec> dotnet-exec https://gitee.com/weihanli/storage/blob/master/test/test.cs
Hello world
```



Raw code execution

```
PS C:\projects\sources\SamplesInPractice> dotnet-exec 'Guid.NewGuid()'
[d3b83962-bbae-4f6a-a62b-926376295080]
```

```
PS C:\projects\sources\SamplesInPractice> dotnet-exec 'Console.WriteLine("Hello world!");'
Hello world!
```

```
PS C:\projects\sources\SamplesInPractice> dotnet-exec 'WebApplication.Create().Run();' -w
info: Microsoft.Hosting.Lifetime[14]
      Now listening on: http://localhost:5000
info: Microsoft.Hosting.Lifetime[0]
      Application started. Press Ctrl+C to shut down.
info: Microsoft.Hosting.Lifetime[0]
      Hosting environment: Production
info: Microsoft.Hosting.Lifetime[0]
      Content root path: C:\projects\sources\SamplesInPractice
info: Microsoft.Hosting.Lifetime[0]
      Application is shutting down...
```

Usings

- Namespace using

```
PS C:\projects\sources\SamplesInPractice> dotnet-exec 'JsonConvert.SerializeObject(new{Name="Xiaoming"})' --using Newton
soft.Json
"{\"Name\":\"Xiaoming\"}"
```

- Static using

```
PS C:\projects\sources\SamplesInPractice> dotnet-exec 'SerializeObject(new{Name="Xiaoming"})' --using 'static Newtonsoft
.Json.JsonConvert'
"{\"Name\":\"Xiaoming\"}"
```

- Alias using

```
PS C:\projects\sources\SamplesInPractice> dotnet-exec 'MyJson.SerializeObject(new{Name="Xiaoming"})' --using 'MyJson = N
ewtonsoft.Json.JsonConvert'
"{\"Name\":\"Xiaoming\"}"
```

References

Local file dll reference

```
PS C:\projects\sources\dotnet-exec\tests\IntegrationTest\bin\Debug\net7.0> dotnet-exec 'ApplicationHelper.GetLibraryInfo
(typeof(WeihanLi.Npoi.CsvHelper))' --reference ".\out\WeihanLi.Npoi.dll"
LibraryInfo { LibraryHash="4875c1619ff4ae811411cb0ef2c4d93273426fa5", LibraryVersion=[2.0.0], RepositoryUrl="https://git
hub.com/WeihanLi/WeihanLi.Npoi" }
```

Local folder dll reference

```
PS C:\projects\sources\dotnet-exec\tests\IntegrationTest\bin\Debug\net7.0> dotnet-exec 'ApplicationHelper.GetLibraryInfo
(typeof(WeihanLi.Npoi.CsvHelper))' --reference "folder: .\out"
LibraryInfo { LibraryHash="4875c1619ff4ae811411cb0ef2c4d93273426fa5", LibraryVersion=[2.0.0], RepositoryUrl="https://git
hub.com/WeihanLi/WeihanLi.Npoi" }
```

References

NuGet package reference

```
PS C:\projects\sources\SamplesInPractice> dotnet-exec 'ApplicationHelper.GetLibraryInfo(typeof(WeihanLi.Npoi.CsvHelper))
' --reference "nuget: WeihanLi.Npoi"
LibraryInfo { LibraryHash="8e2c1dee6efee9b7b4b12f16272f266c9ad09233", LibraryVersion=[2.4.2], RepositoryUrl="https://git
hub.com/WeihanLi/WeihanLi.Npoi" }
PS C:\projects\sources\SamplesInPractice> dotnet-exec 'ApplicationHelper.GetLibraryInfo(typeof(WeihanLi.Npoi.CsvHelper))
' --reference "nuget: WeihanLi.Npoi, 2.4.2"
LibraryInfo { LibraryHash="8e2c1dee6efee9b7b4b12f16272f266c9ad09233", LibraryVersion=[2.4.2], RepositoryUrl="https://git
hub.com/WeihanLi/WeihanLi.Npoi" }
```

Project reference

```
dotnet-exec ./build/build.cs --wide false --reference
"project:./src/WeihanLi.Common/WeihanLi.Common.csproj" --using "WeihanLi.Common" --using
"WeihanLi.Extensions" --using "WeihanLi.Common.Helpers"
```

References

Framework reference

```
PS C:\Users\weihan> dotnet-exec "WebApplication.Create().Run();" --web
info: Microsoft.Hosting.Lifetime[14]
      Now listening on: http://localhost:5000
info: Microsoft.Hosting.Lifetime[0]
      Application started. Press Ctrl+C to shut down.
info: Microsoft.Hosting.Lifetime[0]
      Hosting environment: Production
info: Microsoft.Hosting.Lifetime[0]
      Content root path: C:\Users\weihan
info: Microsoft.Hosting.Lifetime[0]
      Application is shutting down...
PS C:\Users\weihan> dotnet-exec "WebApplication.Create().Run();" --reference "framework:web"
info: Microsoft.Hosting.Lifetime[14]
      Now listening on: http://localhost:5000
info: Microsoft.Hosting.Lifetime[0]
      Application started. Press Ctrl+C to shut down.
info: Microsoft.Hosting.Lifetime[0]
      Hosting environment: Production
info: Microsoft.Hosting.Lifetime[0]
      Content root path: C:\Users\weihan
info: Microsoft.Hosting.Lifetime[0]
      Application is shutting down...
```

References

Framework reference

```
PS C:\Users\weihan> dotnet-exec 'MessageBox.Show("Hello, Shanghai post .NET Conf China 2023")' --reference "framework:WindowsDesktop"
```

Embeded Reference && using

```
// reference: nuget:WeihanLi.Npoi, 2.4.2
// using: WeihanLi.Npoi

Console.WriteLine(new[] { 1, 2, 3 }.GetCsvText());
```

```
// r: "nuget: CliWrap, 3.6.4"

using CliWrap;
using Newtonsoft.Json;

//
var target = Guard.NotNull(Argument("target", "Default"));
var apiKey = Argument("apiKey", "");
var stable = ArgumentBool("stable", false);
var noPush = ArgumentBool("noPush", false);
var branchName = Environment.GetEnvironmentVariable("BUILD_SOURCEBRANCHNAME") ?? "local";
stable |= branchName is "master" or "main";
```

Reference && using from project file

```
PS C:\projects\sources\dotnet-exec> dotnet-exec 'WriteLine(MyFile.Exists("appsettings.json"));' --project
'https://raw.githubusercontent.com/WeihanLi/SamplesInPractice/master/net6sample/ImplicitUsingsSample/Impli
citUsingsSample.csproj'
False
```

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

  <PropertyGroup>
    <OutputType>Exe</OutputType>
    <TargetFramework>net6.0</TargetFramework>
    <ImplicitUsings>enable</ImplicitUsings>
    <Nullable>enable</Nullable>
  </PropertyGroup>

  <ItemGroup>
    <PackageReference Include="WeihanLi.Common" Version="1.0.46" />
  </ItemGroup>

  <ItemGroup>
    <Using Include="System.Console" Static="true" />
    <Using Include="WeihanLi.Common.Helpers" />
    <Using Include="System.IO.File" Alias="MyFile" />
    <Using Remove="System" />
  </ItemGroup>
</Project>
```

Config profile

```
PS C:\Users\weihan> dotnet-exec profile --help
Description:
  Configure user config profile

Usage:
  dotnet-exec <script> profile [command] [options]

Arguments:
  <script>  CSharp script to execute

Options:
  -?, -h, --help  Show help and usage information


Commands:
  set <profile-name>  Configure config profile
  get <profile-name>  Get config profile
  rm <profile-name>   Remove config profile
  ls                  List the config profiles configured
```

Config profile

```
PS C:\Users\weihan> dotnet-exec profile set web -r "nuget:WeihanLi.Web.Extensions" -u 'WeihanLi.Web.Extensions' -u 'Weiha
nLi.Extensions' --web --wide false
PS C:\Users\weihan> dotnet-exec "WebApplication.Create().Chain(_ => _.MapRuntimeInfo()).Run();" --profile web
info: Microsoft.Hosting.Lifetime[14]
      Now listening on: http://localhost:5000
info: Microsoft.Hosting.Lifetime[0]
      Application started. Press Ctrl+C to shut down.
info: Microsoft.Hosting.Lifetime[0]
      Hosting environment: Production
info: Microsoft.Hosting.Lifetime[0]
      Content root path: C:\Users\weihan
info: Microsoft.AspNetCore.Hosting.Diagnostics[1]
      Request starting HTTP/1.1 GET http://localhost:5000/runtime-info - - -
info: Microsoft.AspNetCore.Routing.EndpointMiddleware[0]
      Executing endpoint 'HTTP: GET /runtime-info'
info: Microsoft.AspNetCore.Routing.EndpointMiddleware[1]
      Executed endpoint 'HTTP: GET /runtime-info'
info: Microsoft.AspNetCore.Hosting.Diagnostics[2]
      Request finished HTTP/1.1 GET http://localhost:5000/runtime-info - 200 - application/json;+charset=utf-8 209.9513ms
info: Microsoft.AspNetCore.Hosting.Diagnostics[1]
      Request starting HTTP/1.1 GET http://localhost:5000/runtime-info - - -
info: Microsoft.AspNetCore.Routing.EndpointMiddleware[0]
      Executing endpoint 'HTTP: GET /runtime-info'
info: Microsoft.AspNetCore.Routing.EndpointMiddleware[1]
      Executed endpoint 'HTTP: GET /runtime-info'
info: Microsoft.AspNetCore.Hosting.Diagnostics[2]
      Request finished HTTP/1.1 GET http://localhost:5000/runtime-info - 200 - application/json;+charset=utf-8 1.2672ms
```

Config profile

```
PS C:\Users\weihan> dotnet-http :5000/runtime-info
HTTP/1.1 200 OK
Content-Type: application/json; charset=utf-8
Date: Fri, 01 Mar 2024 10:22:07 GMT
Server: Kestrel
Transfer-Encoding: chunked

{
  "version": "9.0.0",
  "frameworkDescription": ".NET 9.0.0-preview.1.24080.9",
  "processorCount": 4,
  "osArchitecture": "X64",
  "osDescription": "Microsoft Windows 10.0.26063",
  "osVersion": "Microsoft Windows NT 10.0.26063.0",
  "machineName": "WEIHANLI-SURFAC",
  "runtimeIdentifier": "win-x64",
  "workingDirectory": "C:\\Users\\weihan",
  "processId": 29944,
  "processPath": "C:\\Users\\Weiha\\.dotnet\\tools\\dotnet-exec.exe",
  "isInContainer": false,
  "isInKubernetes": false,
  "libraryVersion": "9.0.0-preview.1.24080.9",
  "libraryHash": "1d1bf92fcf43aa6981804dc53c5174445069c9e4",
  "repositoryUrl": "https://github.com/dotnet/runtime"
}
```

Config profile

```
PS C:\Users\weihan> dotnet-exec profile get web
{
  "Usings": [
    "WeihanLi.Web.Extensions",
    "WeihanLi.Extensions"
  ],
  "References": [
    "nuget:WeihanLi.Web.Extensions"
  ],
  "IncludeWebReferences": true,
  "IncludeWideReferences": false,
  "EntryPoint": null,
  "EnablePreviewFeatures": false
}
```

Arguments

```
PS C:\Users\weihan> dotnet-exec "args.Dump();" --args "hello world"
["hello","world"]
PS C:\Users\weihan>
```

```
PS C:\Users\weihan> dotnet-exec "WebApplication.Create(args).Run();" --web -- --urls="http://localhost:12000"
info: Microsoft.Hosting.Lifetime[14]
      Now listening on: http://localhost:12000
info: Microsoft.Hosting.Lifetime[0]
      Application started. Press Ctrl+C to shut down.
info: Microsoft.Hosting.Lifetime[0]
      Hosting environment: Production
info: Microsoft.Hosting.Lifetime[0]
      Content root path: C:\Users\weihan
```

Environment Variable

```
PS C:\Users\weihan> dotnet-exec [env:name=test] '$"Hello {Environment.GetEnvironmentVariable("name")}".Dump();'
Hello test
PS C:\Users\weihan>
```

```
PS C:\Users\weihan> dotnet-exec '$"Hello {Environment.GetEnvironmentVariable("name")}".Dump();' --env name=test
Hello test
PS C:\Users\weihan>
```

Export compilation output

```
PS C:\tmp\temp> dotnet-exec 'Console.WriteLine("Hello World");' --compile-out ./
Hello World
PS C:\tmp\temp> dotnet-exec 'Console.WriteLine("Hello World");' --compile-out ./hello.dll
Hello World
PS C:\tmp\temp> ls

    Directory: C:\tmp\temp

Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
-a---            3/1/2024    23:15           2560 dotnet-exec_d6588b18497346a09818e3f028b8825e.dll
-a---            3/1/2024    23:15           2560 hello.dll
```

Program

```
// dotnet-exec_3216fcae47eb4b72ae776be78d57771f, Version=0.0.0.0, Culture=neutral, PublicKeyToken=null
// Program
using System;
using System.Runtime.CompilerServices;

[CompilerGenerated]
internal class Program
{
    private static void <Main>$(string[] args)
    {
        Console.WriteLine("Hello World");
    }
}
```

Private NuGet

```
PS C:\projects> dotnet-exec "typeof(ICMSUtility).FullName.Dump();" -r "nuget:iHerb.CMS.SDK.Redis" -u "iHerb.CMS.SDK.Redis
" --nuget-config .\iHerb\NuGet.Config
iHerb.CMS.SDK.Redis.ICMSUtility
```

```
PS C:\projects> dotnet-exec "typeof(ICMSUtility).FullName.Dump();" -r "nuget:iHerb.CMS.SDK.Redis" -u "iHerb.CMS.SDK.Redis
" --nuget-config .\iHerb\NuGet.Config --debug
dbug: dotnet-exec[0]
      options: {
        "Script": "typeof(ICMSUtility).FullName.Dump();",
        "TargetFramework": "net9.0",
        "StartupType": null,
        "EntryPoint": "MainTest",
        "Arguments": [],
        "ProjectPath": "",
        "IncludeWideReferences": true,
        "IncludeWebReferences": false,
        "References": [
          "nuget: iHerb.CMS.SDK.Redis"
        ],
        "Usings": [
          "iHerb.CMS.SDK.Redis"
        ],
```

Private NuGet

```
info: NuGetClient[0]
        CACHE https://iherb.jfrog.io/iherb/api/nuget/nuget/Packages(Id='iHerb.CMS.SDK.Redis',Version='3.1.0')
info: NuGetClient[0]
        GET https://iherb.jfrog.io/artifactory/api/nuget/nuget/Download/iHerb.CMS.SDK.Redis/3.1.0
info: NuGetClient[0]
        OK https://iherb.jfrog.io/artifactory/api/nuget/nuget/Download/iHerb.CMS.SDK.Redis/3.1.0 561ms
info: NuGetClient[0]
      Installed iHerb.CMS.SDK.Redis 3.1.0 from https://iherb.jfrog.io/iherb/api/nuget/nuget to [redacted]\.nuget\pack
ages\iherb.cms.sdk.redis\3.1.0 with content hash HBxImYLrgS[redacted]EPUlUu0l64Ffx6P5Mau29TxmG9Kv2vJUOt6y1ZvfBPz4N
MKa3iw2QDsi3nA==.
info: NuGetClient[0]
      Package(iHerb.CMS.SDK.Redis.3.1.0) downloaded to C:\Us[redacted]kages\iherb.cms.sdk.redis\3.1.0 from jfro
g
dbug: dotnet-exec[0]
      Compile elapsed: 00:00:07.4701138
```

More

Source Generator	Interceptor
Symbols	more…

Usage example

dev WeihanLi.Common / build / build.cs

Code Blame 120 lines (110 loc) · 4.73 KB Code 55% faster with GitHub Copilot

```
18  await new BuildProcessBuilder()
19      .WithSetup(() =>
20      {
21          // cleanup artifacts
22          if (Directory.Exists("./artifacts/packages"))
23              Directory.Delete("./artifacts/packages", true);
24
25          // args
26          Console.WriteLine("Arguments");
27          Console.WriteLine($"    {args.StringJoin(" ")}");
28      })
29      .WithTaskExecuting(task => Console.WriteLine($@"===== Task {task.Name} {task.Description} executing ======"))
30      .WithTaskExecuted(task => Console.WriteLine($@"===== Task {task.Name} {task.Description} executed ======"))
31      .WithTask("hello", b => b.WithExecution(() => Console.WriteLine("Hello dotnet-exec build")))
32      .WithTask("build", b =>
33      {
34          b.WithDescription("dotnet build")
35              .WithExecution(() => ExecuteCommandAsync($"dotnet build {solutionPath}"))
36              ;
37      })
38      .WithTask("test", b =>
39      {
40          b.WithDescription("dotnet test")
41              .WithDependency("build")
42              .WithExecution(async () =>
43              {
44                  foreach (var project in testProjects)
45                  {
46                      await ExecuteCommandAsync($"dotnet test --collect:\"XPlat Code Coverage;Format=cobertura,opencover;ExcludeByAttribute=ExcludeFromCode
47                  }
```

Usage example

dev dotnet-httpie / build.ps1

WeihanLi refactor: build script with dotnet-exec and C# script

Code Blame 5 lines (3 loc) · 175 Bytes Code 55% faster with GitHub Copilot

```
dotnet tool update -g dotnet-execute --prerelease

Write-Host 'dotnet-exec ./build/build.cs "--args=$ARGS"' -ForegroundColor GREEN

dotnet-exec ./build/build.cs --args $ARGS
```

```yaml
- name: dotnet-exec
  uses: WeihanLi/dotnet-exec-action@c6057670431a07419dbd8b779641340e711cd889
  with:
    script: "./scripts/build.cs"
    options: "--debug"
  env:
    DingBotWebhookUrl: ${{ secrets.DINGBOTWEBHOOKURL }}
```

Usage example

```yaml
- script: |
    dotnet tool update -g dotnet-execute
    export PATH="$PATH:$HOME/.dotnet/tools"
    dotnet-exec "https://github.com/OpenReservation/scripts/blob/main/deploy/azure-pipelines-notification.cs"
  displayName: 'Push notification'
  env:
    # https://learn.microsoft.com/en-us/azure/devops/pipelines/process/variables?view=azure-devops&tabs=yaml%2Cbatch#secret-variables
    # can not directly reference a secret value
    DingBotWebhookUrl: $(DingBotWebhookUrl)
```

Usage example

```
using System.Net.Http.Json;

var messageTemplate = """
The service {{$env SERVICENAME}} has been deployed with version {{$env IMAGENAME}}
RepoUrl: {{$env BUILD_REPOSITORY_URI}}
[Amazing]
""";
var message = await WeihanLi.Common.Template.TemplateEngine.CreateDefault()
    .RenderAsync(messageTemplate);
Console.WriteLine(message);

var webhookUrl = Environment.GetEnvironmentVariable("DingBotWebhookUrl");
if (string.IsNullOrEmpty(webhookUrl))
{
    Console.WriteLine("WebhookUrl not found");
    return;
}

using var response = await HttpHelper.HttpClient.PostAsJsonAsync(webhookUrl, new
    {
        msgtype = "text",
        text = new
        {
            content = message
        }
    });
response.EnsureSuccessStatusCode();
```

Usage example

```
FROM weihanli/dotnet-exec:0.17.0
WORKDIR /app
COPY entrypoint.cs ./
ENTRYPOINT ["dotnet-exec", "/app/entrypoint.cs"]
```

References

- https://github.com/WeihanLi/dotnet-exec
- https://www.nuget.org/packages/dotnet-execute
- https://hub.docker.com/r/weihanli/dotnet-exec
- https://github.com/WeihanLi/dotnet-exec-action
- https://github.com/marketplace/actions/dotnet-exec
- https://github.com/WeihanLi/dotnet-httpie/blob/dev/build.ps1
- https://github.com/WeihanLi/markdown-nice/blob/master/.github/workflows/docker.yaml
- https://github.com/dotnet/roslyn
- https://github.com/dotnet/command-line-api
- https://github.com/NuGet/NuGet.Client

Q&A
Thank you!