

教程：创建项模板

2019/06/25 • 👤 👤

本文内容

[先决条件](#)

[创建所需的文件夹](#)

[创建项模板](#)

[创建模板配置](#)

[测试项模板](#)

[卸载模板](#)

[后续步骤](#)

使用 .NET Core，可以创建和部署可生成项目、文件甚至资源的模板。本教程是系列教程的第一部分，介绍如何创建、安装和卸载用于 `dotnet new` 命令的模板。

在本系列的这一部分中，你将了解如何：

- ✓ 为项模板创建类
- ✓ 创建模板配置文件夹和文件
- ✓ 从文件路径安装模板
- ✓ 测试项模板
- ✓ 卸载项模板

先决条件

- [.NET Core 2.2 SDK](#) 或更高版本。
- 阅读参考文章[为 dotnet new 自定义模板](#)。

参考文章介绍了有关模板的基础知识，以及如何将它们组合在一起。其中一些信息将在本文中重复出现。


- 打开终端并导航到 `working\templates` 文件夹。

创建所需的文件夹

本系列使用包含模板源的“working 文件夹”和用于测试模板的“testing 文件夹”。working 文件夹和 testing 文件夹应位于同一父文件夹下。

首先，创建父文件夹，名称无关紧要。然后，创建一个名为“working”的子文件夹。在 working 文件夹内，创建一个名为“templates”的子文件夹。


接下来，在名为“test”的父文件夹下创建一个文件夹。文件夹结构应如下所示：

console	 复制
<pre>parent_folder ├── test └── working └── templates</pre>	


创建项模板

项模板是包含一个或多个文件的特定类型的模板。当你想要生成类似于配置、代码或解决方案文件的内容时，这些类型的模板非常有用。在本例中，你将创建一个类，该类将扩展方法添加到字符串类型中。

在终端中，导航到 working\templates 文件夹，并创建一个名为“extensions”的新子文件夹。进入文件夹。

console	 复制
<pre>working └── templates └── extensions</pre>	

创建一个名为“CommonExtensions.cs”的新文件，并使用你喜爱的文本编辑器打开它。此类将提供一个用于反转字符串内容的名为 Reverse 的扩展方法。粘贴以下代码并保存文件：

C#	 复制
<pre>using System; namespace System { public static class StringExtensions { public static string Reverse(this string value) { var tempArray = value.ToCharArray(); Array.Reverse(tempArray); return new string(tempArray); } } }</pre>	

现在你已经创建了模板的内容，需要在模板的根文件夹中创建模板配置。

创建模板配置

模板在 .NET Core 中通过模板根目录中的特殊文件夹和配置文件进行识别。在本教程中，你的模板文件夹位于 `working\templates\extensions`。

创建模板时，除特殊配置文件夹外，模板文件夹中的所有文件和文件夹都作为模板的一部分包含在内。此配置文件夹名为 `.template.config`。

首先，创建一个名为 `.template.config` 的新子文件夹，然后进入该文件夹。然后，创建一个名为 `template.json` 的新文件。文件夹结构应如下所示：

console	复制
<pre>working ├── templates │ └── extensions │ ├── .template.config │ └── template.json</pre>	

使用你喜爱的文本编辑器打开 `template.json` 并粘贴以下 JSON 代码，然后保存：

JSON	复制
<pre>{ "\$schema": "http://json.schemastore.org/template", "author": "Me", "classifications": ["Common", "Code"], "identity": "ExampleTemplate.StringExtensions", "name": "Example templates: string extensions", "shortName": "stringext", "tags": { "language": "C#", "type": "item" } }</pre>	

此配置文件包含模板的所有设置。可以看到基本设置，例如 `name` 和 `shortName`，除此之外，还有一个设置为 `item` 的 `tags/type` 值。这会将你的模板归类为项模板。你创建的模板类型不存在限制。`item` 和 `project` 值是 .NET Core 建议使用的通用名称，便于用户轻松筛选正在搜索的模板类型。

`classifications` 项表示你在运行 `dotnet new` 并获取模板列表时看到的“标记”列。用户还可以根据分类标记进行搜索。不要将 `*.json` 文件中的 `tags` 属性与 `classifications`

标记列表混淆。 它们虽然具有类似的名称，但截然不同。 template.json 文件的完整架构位于 [JSON 架构存储](#)。 有关 template.json 文件的详细信息，请参阅 [dotnet 创建模板 wiki](#)。

现在你已有一个有效的 .template.config/template.json 文件，可以安装模板了。 在终端中，导航到 extensions 文件夹，并运行以下命令以安装位于当前文件夹的模板：

- 在 Windows 上：dotnet new -i .\
- 在 Linux 或 macOS 上：dotnet new -i ./

此命令输出安装的模板列表，其中应包括你的模板。

console

复制

```
C:\working\templates\extensions> dotnet new -i .\
Usage: new [options]

Options:
  -h, --help            Displays help for this command.
  -l, --list            Lists templates containing the specified name. If no
                        name is specified, lists all templates.

... cut to save space ...

Templates                                     Short Name
Language          Tags
-----
Example templates: string extensions          stringext      [C#]
Common/Code
Console Application                          console
[C#], F#, VB      Common/Console
Class library                                      classlib
[C#], F#, VB      Common/Library
WPF Application                                   wpf
[C#], VB          Common/WPF
Windows Forms (WinForms) Application          winforms
[C#], VB          Common/WinForms
Worker Service                                   worker          [C#]
Common/Worker/Web
```

测试项模板

现在你已安装了项模板，可对其进行测试。 导航到 test/ 文件夹，使用 dotnet new console 创建新的控制台应用程序。 这将生成一个可以使用 dotnet run 命令轻松测试的工作项目。

console


复制

```
C:\test> dotnet new console
The template "Console Application" was created successfully.

Processing post-creation actions...
Running 'dotnet restore' on C:\test\test.csproj...
    Restore completed in 54.82 ms for C:\test\test.csproj.

Restore succeeded.
```


console

 复制

```
C:\test> dotnet run
Hello World!
```

接下来，运行 `dotnet new stringext` 以从模板生成 `CommonExtensions.cs`。


console

 复制

```
C:\test> dotnet new stringext
The template "Example templates: string extensions" was created
successfully.
```

更改 `Program.cs` 中的代码以使用模板提供的扩展方法反转 `"Hello World"` 字符串。


C#

 复制

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

再次运行程序，将看到结果已反转。

console

 复制


```
C:\test> dotnet run
!dlroW olleH
```

祝贺你！你已使用 .NET Core 创建并部署了项模板。为准备学习本系列教程的下一部分，必须卸载已创建的模板。确保同时删除 `test` 文件夹中的所有文件。这将回到干净状态，为本教程的下一个主要部分做好准备。

卸载模板

由于模板是按文件路径安装的，因此，必须使用绝对文件路径将其卸载。可以通过运行 `dotnet new -u` 命令看到已安装的模板列表。你的模板应列在最后。使用列出的路径，通过执行 `dotnet new -u <ABSOLUTE PATH TO TEMPLATE DIRECTORY>` 命令卸载模板。

console

 复制


```
C:\working> dotnet new -u
Template Instantiation Commands for .NET Core CLI

Currently installed items:
  Microsoft.DotNet.Common.ItemTemplates
    Templates:
      dotnet gitignore file (gitignore)
      global.json file (globaljson)
      NuGet Config (nugetconfig)
      Solution File (sln)
      Dotnet local tool manifest file (tool-manifest)
      Web Config (webconfig)

... cut to save space ...

NUnit3.DotNetNew.Template
  Templates:
    NUnit 3 Test Project (nunit) C#
    NUnit 3 Test Item (nunit-test) C#
    NUnit 3 Test Project (nunit) F#
    NUnit 3 Test Item (nunit-test) F#
    NUnit 3 Test Project (nunit) VB
    NUnit 3 Test Item (nunit-test) VB
C:\working\templates\extensions
  Templates:
    Example templates: string extensions (stringext) C#
```

console

 复制

```
C:\working> dotnet new -u C:\working\templates\extensions
```

后续步骤

在本教程中，你创建了一个项模板。若要了解如何创建项目模板，请继续学习本系列教程。

[创建项目模板](#)

此页面有帮助吗？

 是  否