

教程：创建模板包

2019/12/10 • 👤 👤

本文内容

[先决条件](#)

[创建模板包项目](#)

[生成和安装](#)

[卸载模板包](#)

[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三部分，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：

- ✓ 创建一个 *.csproj 项目以生成模板包
- ✓ 配置项目文件以进行打包
- ✓ 从 NuGet 包文件安装模板
- ✓ 按包 ID 卸载模板

先决条件

- 完成本系列教程的[第 1 部分](#)和[第 2 部分](#)。

本教程使用本教程前两部分中创建的两个模板。只要将不同的模板作为文件夹复制到 `working\templates\` 文件夹中，就可以使用该模板。

- 打开终端并导航到 `working\` 文件夹。

创建模板包项目

模板包是打包到文件中的一个或多个模板。安装或卸载程序包时，将分别添加或删除包中包含的所有模板。本系列教程的前几部分仅适用于各自的模板。若要共享非打包模板，必须复制模板文件夹并通过该文件夹进行安装。由于模板包中可以包含多个模板，并且是单个文件，因此共享更容易。

模板包由 NuGet 包 (.nupkg) 文件表示。并且，与任何 NuGet 包一样，可以将模板包上传到 NuGet 源。`dotnet new -i` 命令支持从 NuGet 包源安装模板包。此外，可以直接从 .nupkg 文件安装模板包。

通常情况下，使用 C# 项目文件来编译代码并生成二进制文件。但是，该项目也可用于生成模板包。通过更改 .csproj 的设置，可以阻止它编译任何代码，而是将模板的所有资产都包含在内作为资源。生成此项目后，它会生成模板包 NuGet 包。

将要创建的包将包含先前创建的[项模板](#)和[包模板](#)。由于我们将两个模板分组到 working\templates\ 文件夹中，因此可以使用 .csproj 文件的 working 文件夹。

在终端中，导航到 working 文件夹。创建一个新项目，将名称设置为 templatepack，并将输出文件夹设置为当前文件夹。

.NET Core CLI	复制
<pre>dotnet new console -n templatepack -o .</pre>	

-n 参数将 .csproj 文件名设置为 templatepack.csproj。-o 参数将在当前目录中创建文件。应看到类似于以下输出的结果。

console	复制
<pre>C:\working> dotnet new console -n templatepack -o . The template "Console Application" was created successfully. Processing post-creation actions... Running 'dotnet restore' on .\templatepack.csproj... Restore completed in 52.38 ms for C:\working\templatepack.csproj. Restore succeeded.</pre>	

接下来，在你喜爱的编辑器中打开 templatepack.csproj 文件，并将内容替换为以下 XML：

XML	复制
<pre><Project Sdk="Microsoft.NET.Sdk"> <PropertyGroup> <PackageType>Template</PackageType> <PackageVersion>1.0</PackageVersion> <PackageId>AdatumCorporation.Utility.Templates</PackageId> <Title>AdatumCorporation Templates</Title> <Authors>Me</Authors> <Description>Templates to use when creating an application for Adatum Corporation.</Description> <PackageTags>dotnet-new;templates;contoso</PackageTags> <TargetFramework>netstandard2.0</TargetFramework> <IncludeContentInPack>true</IncludeContentInPack> <IncludeBuildOutput>>false</IncludeBuildOutput> </PropertyGroup> </Project></pre>	

```

    <ContentTargetFolders>content</ContentTargetFolders>
  </PropertyGroup>

  <ItemGroup>
    <Content Include="templates\**\*"
      Exclude="templates\**\bin\*;templates\**\obj\*" />
    <Compile Remove="**\*" />
  </ItemGroup>

</Project>

```

上面的 XML 中的 <PropertyGroup> 设置分为三组。第一组处理 NuGet 包所需的属性。三个 <Package 设置与 NuGet 包属性相关，用于在 NuGet 源上标识你的包。具体来说，<PackageId> 值用于通过单个名称而不是目录路径卸载模板包。它还可用于从 NuGet 源安装模板包。其余的设置，如 <Title> 和 <PackageTags>，它们与 NuGet 源上显示的元数据相关。有关 NuGet 设置的详细信息，请参阅 [NuGet 和 MSBuild 属性](#)。


必须设置 <TargetFramework> 设置，以便在运行 pack 命令编译和打包项目时 MSBuild 正常运行。

最后三个设置与正确配置项目有关，以便在创建 NuGet 包时将模板包含在该包中的相应文件夹中。


<ItemGroup> 包含两个设置。首先，<Content> 设置的内容包含 templates 文件夹中的所有内容。它还设置为排除任何 bin 文件夹或 obj 文件夹，以防止包含任何已编译的代码（如果已测试和编译模板）。其次，<Compile> 设置将所有代码文件排除在编译范围之外，无论它们位于何处都是如此。此设置可阻止用于创建模板包的项目尝试编译 templates 文件夹层次结构中的代码。

生成和安装

保存此文件，然后运行 pack 命令

.NET Core CLI	 复制
<code>dotnet pack</code>	

此命令将生成项目并在其中创建一个 NuGet 包，具体应为 working\bin\Debug 文件夹。

console	 复制
<pre> C:\working> dotnet pack Microsoft (R) Build Engine version 16.2.0-preview-19278-01+d635043bd for .NET Core Copyright (C) Microsoft Corporation. All rights reserved. </pre>	

```
Restore completed in 123.86 ms for C:\working\templatepack.csproj.
```

```
templatepack -> C:\working\bin\Debug\netstandard2.0\templatepack.dll  
Successfully created package  
'C:\working\bin\Debug\AdatumCorporation.Utility.Templates.1.0.0.nupkg'.
```

接下来，使用 `dotnet new -i PATH_TO_NUPKG_FILE` 命令安装模板包文件。

console

 复制


```
C:\working> dotnet new -i  
C:\working\bin\Debug\AdatumCorporation.Utility.Templates.1.0.0.nupkg  
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  
Example templates: async project             consoleasync       [C#]  
Common/Console/C#8  
Class library                                classlib  
[C#], F#, VB      Common/Library
```

如果将 NuGet 包上传到 NuGet 源，可以使用 `dotnet new -i PACKAGEID` 命令，其中，`PACKAGEID` 与 `.csproj` 文件中的 `<PackageId>` 设置相同。此包 ID 与 NuGet 包标识符相同。

卸载模板包

无论如何安装模板包，即无论是直接使用 `.nupkg` 文件还是通过 NuGet 源安装，删除模板包的操作都是一样的。使用要卸载的模板的 `<PackageId>`。可以通过运行 `dotnet new -u` 命令获取已安装的模板列表。

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
AdatumCorporation.Utility.Templates
  Templates:
    Example templates: async project (consoleasync) C#
    Example templates: string extensions (stringext) C#
```

运行 `dotnet new -u AdatumCorporation.Utility.Templates` 以卸载模板。 `dotnet new` 命令将输出应忽略先前安装的模板的帮助信息。

祝贺你！你已安装，并卸载了模板包。

后续步骤

若要了解有关模板的详细信息（你已经了解了大部分相关信息），请参阅[为 dotnet new 自定义模板](#)一文。

- [dotnet/templating GitHub 存储库 Wiki](#)
- [dotnet/dotnet-template-samples GitHub 存储库](#)
- [JSON 架构存储中的 template.json 架构](#)

此页面有帮助吗？

 是  否
