# An introduction to NuGet

对于 .NET（包括 .NET Core），Microsoft 支持的共享代码的机制为 NuGet，它定义了如何创建、托管和使用面向 .NET 的包，并针对每个角色提供适用工具。

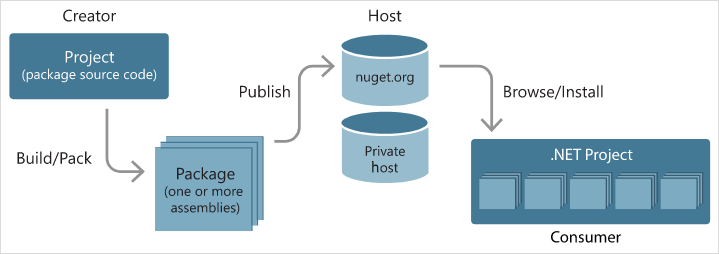
简单来说，NuGet 包是一个以 .nupkg 为扩展的单个 ZIP 文件，此扩展包含编译代码 (DLLs)、与该代码相关的其他文件以及描述性清单（包含包版本号等信息）。 开发人员创建一个包并将它发布到一个共有或者私有的Host上以共享代码。 包使用者从适合的主机获取这些包，将它们添加到项目，然后在其项目代码中调用包的功能。 NuGet 负责处理所有中间的细节。

由于 NuGet 支持公用 nuget.org 宿主与私有宿主一起使用，因此，可以使用 NuGet 包来共享组织或工作组专用的代码。 此外，你还可以使用 NuGet 包作为一种便捷的方式，将自己的代码用于除你自己项目之外的任何其他项目。简而言之，NuGet 包是可共享的代码单元，但不需要暗示任何特定的共享方式。

## The flow of packages between creators, hosts, and consumers

作为公用宿主，NuGet 自身在 [nuget.org](https://www.nuget.org/) 中维护包含 100,000 多个唯一包的中央存储库。这些包每天供数以百万的 .NET/.Net Core 开发人员使用。 NuGet 还支持在云中（例如在 Visual Studio Team Services 中）、在私有网络中或者甚至直接在本地文件系统以私密方式托管包。 通过这样做，这些程序包仅对那些有权访问主机的开发人员可用，使你能够将程序包提供给特定的一组用户。 [托管自己的 NuGet 源](https://docs.microsoft.com/zh-cn/nuget/hosting-packages/overview)中提供了对相关选项的说明。 通过配置选项，你还可以精确控制任何给定计算机可以访问的主机，从而确保程序包是从特定源（而不是像 nuget.org 这样的公用存储库）获取的。

无论主机的本质是什么，它都可作为包创建者和包使用者之间的连接点。 创建者生成有用的 NuGet 包并将其发布到主机。 然后，使用者可以在可访问的主机上搜索有用且兼容的包，下载包并将其包含在项目中。 在项目中安装包后，包的 API 将可用于其余项目代码。



## Package targeting compatibility

“兼容”包指：此包所包含的程序集应至少针对与使用项目的目标框架兼容的一个目标 .NET Framework 而生成。与 UWP 控件一样，开发人员可以创建特定于一个框架的程序包，也可以支持更广泛的目标。为了最大限度地利用程序包的兼容性，开发人员的目标是所有 .NET 和 .NET Core 项目都可以使用的 [.NET Standard](https://docs.microsoft.com/zh-cn/dotnet/standard/net-standard)。 对于创建者和使用者而言，这是最有效的方式，因为单个包（通常包含单个程序集）适用于所有使用项目。

另一方面，需要 .NET Standard 之外的 API 的程序包开发人员会为他们希望支持的不同目标框架创建单独的程序集，并将所有这些程序集包含在同一个程序包中（称为“多目标”）。 使用者安装此类包时，NuGet 将仅提取项目需要的程序集。 这能将包在该项目生成的最终应用程序和/或程序集中的占用量降到最低。 当然，多目标包对创建者来说更难维护。

备注：

目标 .NET Standard 取代了以前使用各种可移植类库 (PCL) 目标的方法。 因此，此文档着重于为 .NET Standard 创建程序包。

## NuGet tools

除托管支持外，NuGet 还提供各种供创建者和使用者使用的工具。 有关如何获取特定工具的信息，请参阅[安装 NuGet 客户端工具](https://docs.microsoft.com/zh-cn/nuget/install-nuget-client-tools)。

| 工具 | 平台 | 适用方案 | 描述 |
| --- | --- | --- | --- |
| [nuget.exe CLI](https://docs.microsoft.com/zh-cn/nuget/tools/nuget-exe-cli-reference) | 全部 | 创建、使用 | 提供所有 NuGet 功能，包括一些专门适用于包创建者、仅适用于使用者和适用于两者的命令。 例如，包创建者使用 nuget pack 命令通过各种程序集和相关文件创建包，包使用者使用 nuget install 在项目文件夹中包含包，而所有人都可使用 nuget config 设置 NuGet 配置变量。 作为与平台无关的工具，NuGet CLI 不会与 Visual Studio 项目交互。 |
| [dotnet CLI](https://docs.microsoft.com/zh-cn/nuget/tools/dotnet-commands) | 全部 | 创建、使用 | 直接在 .NET Core 工具链中提供特定 NuGet CLI 功能。 与 NuGet CLI 一样，dotnet CLI 不会与 Visual Studio 项目交互。 |
| [包管理器控制台](https://docs.microsoft.com/zh-cn/nuget/tools/package-manager-console) | Windows 版 Visual Studio | 使用 | 提供用于在 Visual Studio 项目中安装和管理包的 [PowerShell 命令](https://docs.microsoft.com/zh-cn/nuget/tools/powershell-reference)。 |
| [包管理器 UI](https://docs.microsoft.com/zh-cn/nuget/tools/package-manager-ui) | Windows 版 Visual Studio | 使用 | 提供用于在 Visual Studio 项目中安装和管理包的易用 UI。 |
| [管理 NuGet UI](https://docs.microsoft.com/zh-cn/visualstudio/mac/nuget-walkthrough) | Visual Studio for Mac | 使用 | 提供用于在 Visual Studio for Mac 项目中安装和管理包的易用 UI。 |
| [MSBuild](https://docs.microsoft.com/zh-cn/nuget/reference/msbuild-targets) | Windows | 创建、使用 | 支持创建包和还原项目中直接通过 MSBuild 工具链使用的包。 |

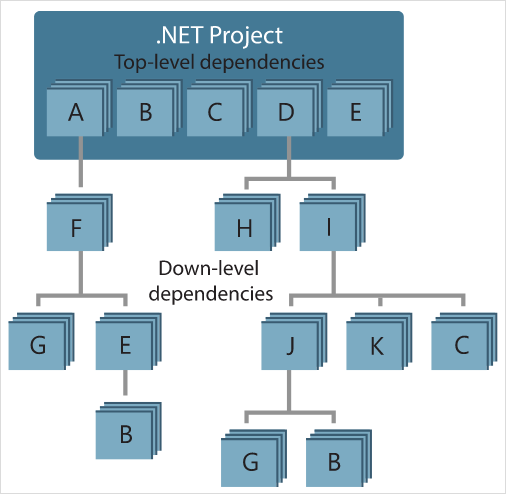
如上文所述，你使用的 NuGet 工具很大程度上取决于用户要创建、使用还是发布程序包以及你所使用的平台。 包创建者通常也是使用者，因为他们以其他 NuGet 包中已存在的功能作为生成基础。 当然，这些包反过来可能也需要依赖其他包。

有关详细信息，请从[包创建工作流](https://docs.microsoft.com/zh-cn/nuget/create-packages/overview-and-workflow)和[包使用工作流](https://docs.microsoft.com/zh-cn/nuget/consume-packages/overview-and-workflow)文章开始。

## Managing dependencies

在其他人的工作基础上轻松生成，这是程序包管理系统最强大的特性之一。 相应地，大部分 NuGet 的用途就是代表项目管理该依赖关系树或“关系图”。 简单来说，你仅需要关注在项目中直接使用的包。 如果任何这些包本身使用其他包（这些包仍可以使用其他包），NuGet 将负责所有这些下层依赖性。

下图显示一个依赖于五个包的项目，这些包反过来也依赖于许多其他包。



请注意，某些包在依赖项关系图中多次出现。 例如，包 B 有三个不同的使用者，并且每个使用者可能为该包指定不同版本（未显示）。 这是一种常见情况，特别是对于广泛使用的程序包。 幸运的是，NuGet 将执行所有困难的工作来确定包 B 的哪一个版本可满足所有使用者。 随后，无论依赖项关系图多么复杂，NuGet 都将对所有包执行相同操作。

有关 NuGet 如何实现此服务的更多详细信息，请参阅[依赖项解析](https://docs.microsoft.com/zh-cn/nuget/consume-packages/dependency-resolution)。

## Tracking references and restoring packages