老孔发的.net Core

s

学习新的开发框架是一项巨大的投资。您需要学习如何在新框架中编写，构建，测试，部署和维护应用程序。作为开发人员，有许多框架可供选择，很难知道什么是最适合的。即使您正在使用.NET开发软件，.NET Core也不相同，您需要花时间学习使用它的工具和技术。为了理解为什么.NET Core值得一看，这有助于知道你从哪里开始。

**.NETCore的开源协议**

.NET Core从属于.NET基金会，由微软进行官方支持***。使用最宽松的MIT和Apache 2开源协议，文档协议遵循CC-BY***。这将允许任何人任何组织和企业任意处置，包括使用，复制，修改，合并，发表，分发，再授权，或者销售。唯一的限制是，软件中必须包含上述版 权和许可提示，***后者协议将会除了为用户提供版权许可之外，还有专利许可，并且授权是免费，无排他性的(任何个人和企业都能获得授权)并且永久不可撤销，相较于oracle对java和mysql的开源协议微软做出了最大的诚意***

**如果您是.NET Framework开发人员**

.NET Core是为了重新启动某些Framework组件而为其他人提供跨平台工作的机会。由于.NET Framework主要以托管（C＃）代码为基础构建，因此这些部分不需要更改代码即可移至.NET Core。依赖于Windows特定组件的库必须被移除或重构以使用跨平台替代方案。这同样适用于您的应用程序。

**您的.NET应用程序可以是跨平台的**

您现有的.NET Framework应用程序可以在其他操作系统上工作。对于希望扩大类库的受众平台，或者希望在分布式应用程序的其他领域使用相同代码的开发人员来说，这是非常好的选择。即使你想用你亮瞎眼的MacBook上开发.NET，而无需双启动到Windows。

并不是所有的框架都被移植到.NET Core，但主要的部分。存在一些API差异。例如，如果您使用了大量的反射，则可能需要重构代码才能使用Core。有关更多信息，请看文章末尾的的两者差异部分。

**ASP.NETCore胜过框架ASP.NET**

***ASP.NET Core和ASP.NET之间的性能差异是几个数量级***。大部分的ASP.NET被传统的System.Web库所限制。.NET Framework支持旧版本的ASP.NET项目，而且这个约束限制了ASP.NET的发展。微软决定重写整个架构。这意味着打破变化，但结果是值得的。

**.NETCore是创新的焦点**

向后兼容性是一把双刃剑。这意味着您的应用程序可以继续得到新版本框架的支持，确保在新版本框架中所做的更改不会破坏现有的应用程序。

所有的努力避免变化都限制了框架的创新。框架的变化需要彻底的理由（通常来自客户），详尽的测试以及来自多个层次产品组的批准。

使用.NET Core，团队可以更容易专注的在.net core上工作。比如核心类库（如System.Collections）的更改仍然需要与.NET Framework相同的活力，但是ASP.NET Core或Entity Framework Core可以更轻松地进行实质性更改，而不受向后兼容性的限制。这允许更大的创新。

.NET Framework作为一个整体产品发布，但是.NETCore被分解成多个部分。现在开发人员可以选择使用哪个版本的库。***只要它在.NET标准库之外，.NET Core团队就可以用较少的难度进行创新***。这就是为什么在将来你只能看到错误修复的框架; .net core将获得所有新功能。

**发布周期更快**

如果您曾在框架中遇到错误，并将其报告给Microsoft，则您将知道发布修复需要多长时间。这个框架有很长的发布周期，通常至少要测量一年，而且在这些周期中还有很小的窗口用于特性工作。每个代码更改都可能会导致框架中其他位置出现意外的问题。为了给每个团队足够的时间来测试框架，有很多时候代码更改是受限制的或者严格审查的。如果您在.NET中发现了一个错误，***最好找到一个解决方法，而不是等待更新（这也是.NETCore开源带来的好处，您不需要完全依赖微软支持才能解决问题，这对于大型企业更是有利，可以利用技术资源来对.NETCore进行本企业定制化，成为.NETCore的分支来运行）***。

.NET Core遵循更快的发布步调。开发人员可以使用夜间构建来尽早测试。不属于.NET标准库的库可以按自己的步调发布。***因为所有东西都是开源的，如果微软没有足够快的响应，任何开发者都可以提出修补。如果解决方法不被接受，讨论就会公开进行，所有人都可以看到为什么做出这个决定***

**如果你是.NET新手**

在Windows平台上，.NET框架没有太多的竞争。微软可以对从操作系统内核层到高级.NET库的所有内容进行更改。通过将.NET引入其他平台，竞争环境发生了变化。.NET现在必须与其他所有的开发框架竞争。

**C＃是一个了不起的语言**

.NET的旗舰语言C＃具有许多独特的功能，例如语言集成查询和异步构造，这使得它强大且易于使用。相较于保守的java(我们也看到Oracle旗下的老牌java已经已经越来越难推出新特性，这是历史包袱所导致，在将来模块化的开发中很难进行更新维护)C＃也在不断创新。C＃团队公开设计语言，因为他们希望任何人提出建议或参与讨论。编译器（Roslyn）完全是模块化和可扩展的。

**ASP.NET Core性能与顶级Web平台相当**

如果您正在编写Web应用程序或服务，那么ASP.NET Core是一个很好的搭建平台。它具有出色的性能和低内存占用。许多功能可以使您的应用程序更容易开发和维护。

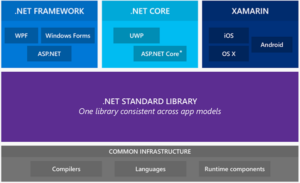
**.NET Core不是从头开始的**

.NET在2000年以前就已经出现了。框架代码在过去几年中已经得到了巩固，开发人员也从中受益。***已经被移植到Core的大部分Framework代码都没有改变。这使.NET Core在构建应用程序的可靠框架方面领先一步***。.NET Core也完全由Microsoft支持还有越来越活跃的开源社区例如红帽组织。这降低了为您的应用程序使用Core的风险。

**什么是.NETCore？**

为了理解.NET Core，它有助于理解.NET Framework。微软在21世纪初发布了.NET Framework。.NET Framework是一个仅限于Windows的开发框架，它在最底层提供了内存管理，安全性，异常处理和许多其他功能。.NET框架附带一大堆库，可执行从XML解析到HTTP请求的各种功能。它还支持几种语言，并将它们编译成相同的通用中间语言; 任何语言都可以使用任何其他语言构建的库。这些关键概念也出现在.NET Core中。

2016年，微软收购了Xamarin并发布了.NET Core 1.0。之前Xamarin已经移植了.NET框架的大部分内容来运行在基于Linux / Unix的操作系统上。一些代码可以在.NET Framework，Xamarin和新的.NET Core之间共享，但编译后的二进制文件不能。.NET Core的一部分努力是创建一个标准化的平台，允许所有.NET实现共享相同的库。



**图1** .NET框架，.NET Core和Xamarin都共享一个称为.NET标准库的标准化平台

以前的Xamarin和.NET Framework之间是无法共享二进制文件的。随着.NET标准库和通用基础架构的推出，这两个框架现在成为统一的.NET生态系统的一部分。

什么是.NETCore？在上图中，似乎.NET Core是另一个包含UWP（通用Windows平台）和ASP.NET Core的框架。为了使.NET Core成为现实，创建了.NET标准库和通用基础架构。

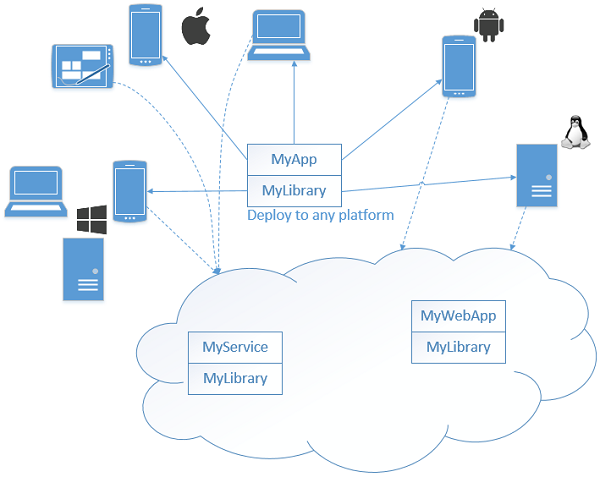
**.NETCore的关键功能**

.NET Core借鉴了.NET Framework的最佳实践，并将软件工程的最新进展结合在一起。这些是.NET Core的一些显着特征。

**扩大您类库的覆盖面**

使用.NET Core，您可以使用.NET标准库编写应用程序或库。

然后它可以在许多平台上共享。



**图2** .NETCore开发

同一个库可以在后台服务“本地”或云中运行，也可以在手机，平板电脑或桌面上运行的客户端应用程序中运行。与其为iOS，Android和Windows构建单独的应用程序，您可以构建一个适用于所有平台的应用程序。.NET Core是小型和完美的容器，可以轻松扩展并缩短开发时间。

.NET Core和.NET Standard Library建立了一个通用平台。过去，当新版本的操作系统或新设备出现时，开发人员有责任重新构建新平台的应用程序或库，并分发更新。使用.NET Core，不需要重建和重新分配。只要新的平台支持你所有的依赖库，它就支持你的应用程序。

**在任何平台上简单部署**

Microsoft产品往往具有复杂的安装过程。COM组件，注册表项，特殊文件夹，GAC - 都是为了利用Windows的特性而设计的。.NET框架依赖于这些构造，这使得它不适合其他操作系统。

在发布依赖于.NET Framework的应用程序时，安装程 序必须足够聪明才能检测是否安装了错误的.NET Framework版本，并为用户提供正确的方法。大多数现代Windows版本都包含.NET Framework。这使得某些应用程序更容易安装，但是如果应用程序使用默认情况下未安装的功能（如ASP.NET与IIS或WCF组件的集成），则可能会导致复杂的问题。

另一个复杂的问题来自补丁。包括错误修复或安全更新的修补程序可以通过Windows更新或通过Microsoft下载中心分发给客户。您测试应用程序的.NET Framework可能与客户使用的补丁程序不同。当您假定.NET Framework对于所有客户都是一样的时候，通常很难确定在应用程序中导致奇怪行为的原因。

.NET Core的模块化设计意味着您只包含所需的依赖关系。所有这些依赖关系与您的应用程序进入相同的文件夹。部署应用程序与复制文件夹一样简单。这种方法的另一个优点是可以有多个版本并行运行。这个策略对于使所有平台的部署体验保持一致至关重要。

**云和容器**

在云系统中，用更少的硬件为更高密度的用户提供服务是非常重要的。应用程序的占位面积越小，密度越高。虚拟机已经在云端普遍存在多年，但是它们有几个问题：

* 大小 - 一个典型的虚拟机文件是千兆字节，如果不是几十千兆字节。这使得它们跨网络传输非常耗时，并且对磁盘空间有很大的要求。
* 启动时间 - 启动虚拟机意味着启动操作系统。对于Windows来说，这是一个挑战，因为启动新机器需要花费时间。这可以使处理突发交通困难。
* 内存 - 虚拟机需要将整个操作系统与应用程序一起加载到内存中。这意味着很多主机的内存被浪费了。
* 不一致性 - 相同的虚拟机可以复制到多个主机，主机必须提供相同的虚拟化硬件，这可能依赖于物理硬件。无法保证虚拟机在任何给定的主机上运行相同的操作。

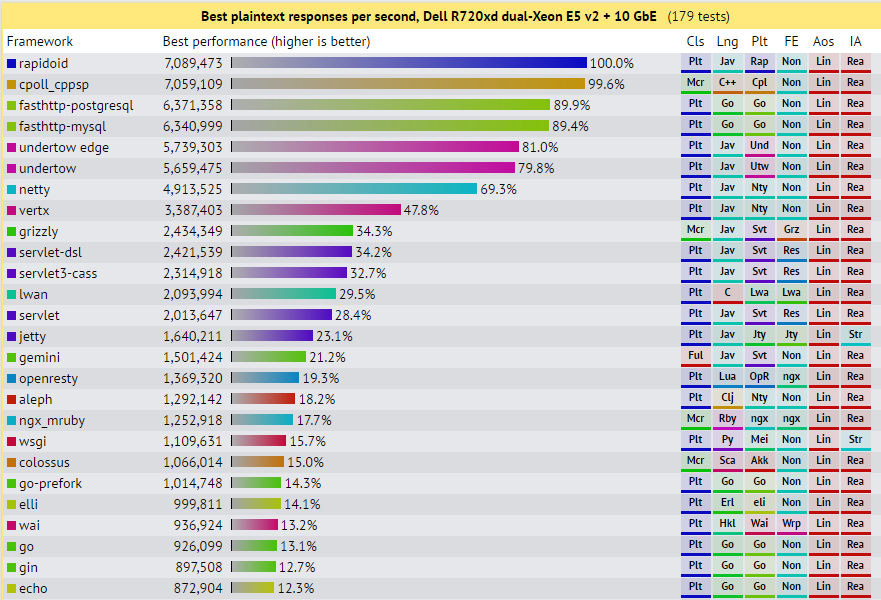
容器通过虚拟化操作系统来解决虚拟机的问题。容器只包含应用程序及其依赖项。文件大小要小很多倍，启动时间以秒为单位，只有应用程序加载到内存中，容器保证在任何主机上工作。

内置于Windows的.NET Framework不能在容器上运行。鉴于容器的明显优势，***.NET Core的设计决定之一就是使其成为模块化***。这意味着你的.NETCore应用程序可以被“发布”，***使得它和它的所有依赖关系在一个地方，这很容易放入容器***。

**ASP.NET性能**

如果您不熟悉ASP.NET，则是.NET Framework中内置的Web应用程序平台。***ASP.NET被许多有影响力的组织所使用，包括Stack Overflow还有绝大多数的可靠性很高的金融证券行业领域***。ASP.NET是2002年发布的第一个.NET Framework版本，并不断发展。尽管ASP.NET取得了成功，但在ASP.NET团队中却感觉到他们正在失去开发人员，因为ASP.NET的性能没有竞争力，只能在Windows平台上运行。

一家名为TechEmpower的公司每隔几个月就会运行一次Web应用程序平台的基准测试，并提供一个分为几类的排名。基准测试在Linux上运行，仅包含Windows平台。对于ASP.NET团队来说，这很麻烦。许多平台都用于编写跨平台的Web应用程序，并且其性能数据令人印象深刻。此外，一些Java框架发布了天文数字，例如每秒570万个明文请求，或者490万个。



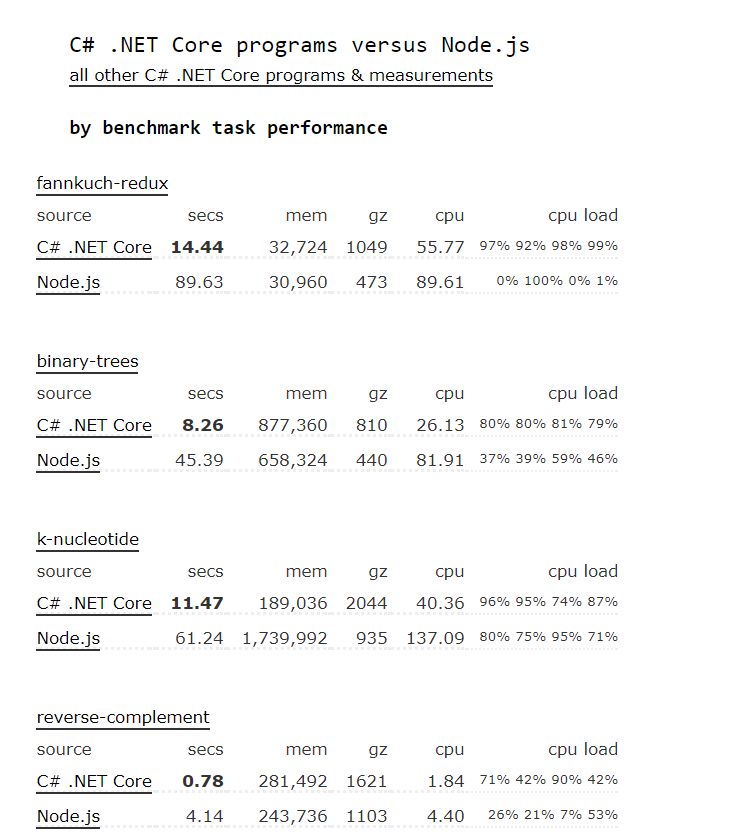
**图3** TechEmpower基准，第12轮

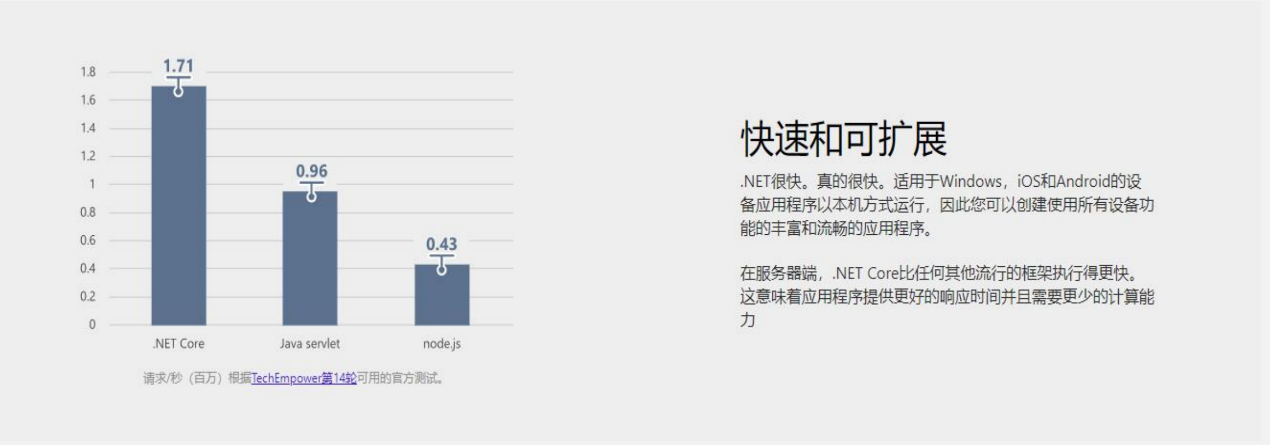
在TechEmpower基准测试的第十一轮中，Mono平台上的ASP.NET MVC被纳入测试。***结果不好。Mono上的ASP.NET每秒产生一个微小的2000个明文请求。Mono不是由Microsoft创建的，它不会像普通的.NET Framework那样得到相同数量的性能调整。为了获得更公平的比较，ASP.NET团队决定在与TechEmpower相同的硬件上运行.NET 4.6的基准测试。结果是每秒约50,000个请求。仍然没有接近NodeJS（每秒320,000个请求），或TechEmpower列表上的其他任何顶级框架。***

可怜的低分并不意外。如前所述，ASP.NET知道改变现状只能通过重写整个架构来清除障碍。这正是发生的事情。

ASP.NET团队着手构建ASP.NET Core，几个月后，该团队庆祝asp.net core每秒超过100万个请求。最新asp.netcore与node.js的性能对比更是让asp.net core的性能优势继续扩大

**ASP.NET Core VS Node.js**





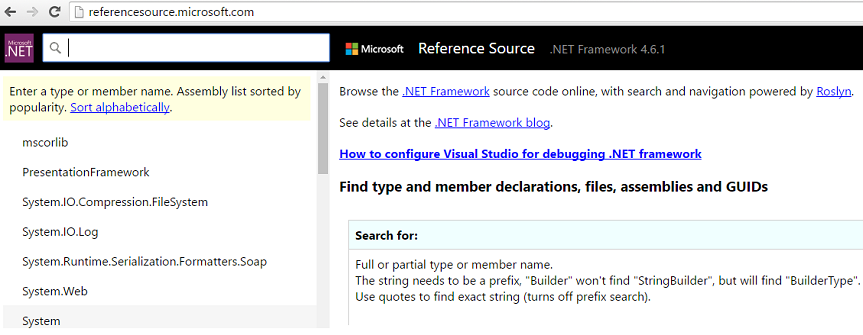
上图中可看出asp.net core作为复杂web开发的性能已经相较于之前有质的提高， [***ASP.NET Core VS node.js***](http://benchmarksgame.alioth.debian.org/u64q/compare.php?lang=csharpcore&lang2=node)

***ASP.NET Core表明了微软思维的转变。微软意识到它必须具有竞争力才能赢得开发者。它还必须在Windows以外的平台上竞争。这是创建.NET Core的原动力。***

**开源**

现代软件开发人员并不满足于寻求功能。当已经有一个满足他们需求的开源项目的时候，尤其如此。当大公司热衷于开源软件时，即使是最忠实的微软开发者也会转向其他框架和库，以便在预算内按时完成自己的项目。如果微软关心“开发人员，他们必须做出改变。

公开.NET Framework的源代码是第一步。.NET Framework源代码已经在referencesource.microsoft.com 和GitHub上公开了好几年了。



**图4** .NET Framework参考源

从单独的开源代码到接受外来的贡献是完全不同的。.NET Core开发人员不仅希望获得外部贡献，还希望将社区纳入设计和开发。这导致了更多的透明度。***ASP.NET Core团队每周都会在***[***http://live.asp.net***](http://live.asp.net/)***上举行一次社区直播会议***。.NET Core的代码从一开始就在GitHub上公开，任何人都可以提出请求。社区成员可以在GitHub中创建错误和功能。.NET Core已经标志着微软在开源方面的重大转变。

**与.NET Framework的差异**

.NET Core不仅是Linux和Mac的.NET Framework。微软并没有移植所有的.NET Framework，对于ASP.NET客户来说，ASP.NET团队是.NET Core的驱动因素。MVC，Web API和SignalR都可以在.NET Core中找到，或者在路线图上。

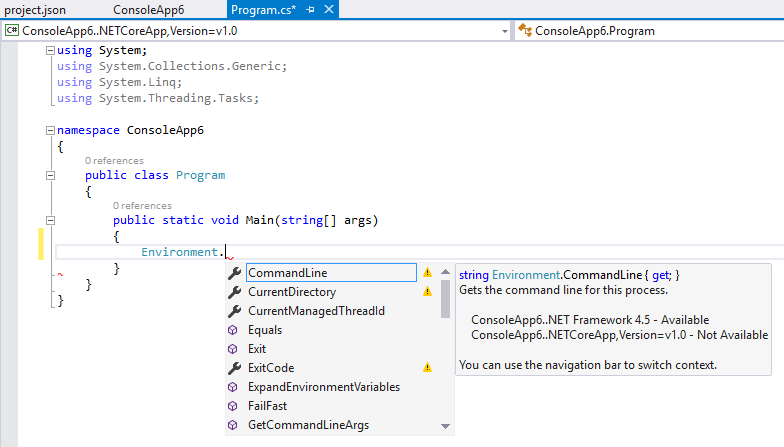
**框架功能没有移植到.NETCore**

我给这个列表提供了可以改变的知识。某些功能不适用于非Windows平台，其他某些功能也将不会由Microsoft在将来推出，因为有更好的替换或功能在某些方面有问题（不安全，难以维护等 ）。

* WPF / XAML - Windows Presentation Foundation仅适用于用户界面。.NET标准库不包括用户界面库，.NET Core不会提供跨平台的UI框架。这意味着像Windows窗体和系统的东西。绘图不会移植到.NET Core。
* 事务 - 这个库使创建分布式事务变得容易。它依赖于特定于Windows的组件，使其不易移植到.NET Core。
* AppDomains - 这些对于隔离程序集来说是非常有用的，而且不会终止进程，这对于允许插件的应用程序来说非常有用。它们依赖于某些Windows特定的构造，这些构造在其他操作系统上不起作用。
* .NET远程处理 - 远程对象已被REST服务成功。
* ASMX - 编写已被Web API取代的Web服务的旧方法。
* Linq to SQL - 由实体框架取代。
* WCF服务 - Windows Communication Foundation客户端功能在.NET Core中可用，但不能创建服务。
* WF - Windows Workflow Foundation依赖于XAML，WCF服务以及其他.NET Framework特性之间的事务。

**.NET Framework开发人员的微妙变化**

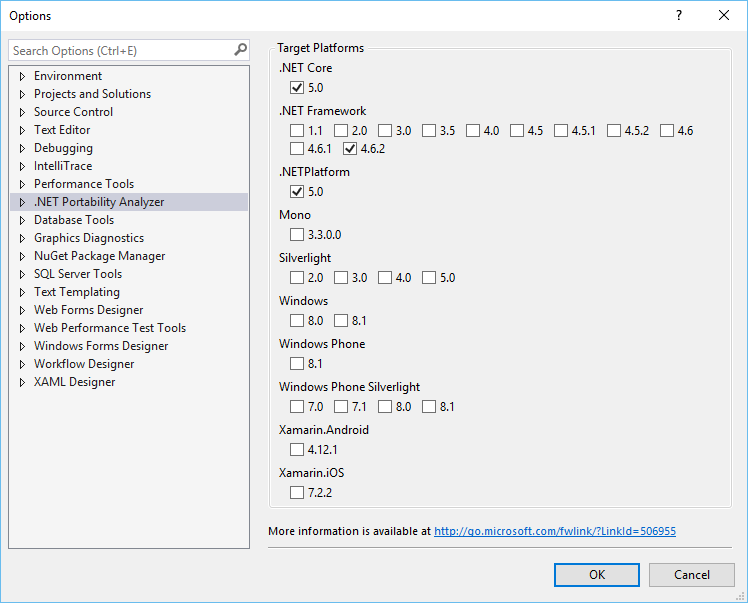
有经验的.NET Framework开发人员可能会在.NET Core中遇到一些惊喜。编写新的代码应该是相对直接的，因为你不太可能使用像HashTableor 这样的旧的结构ArrayList。Visual Studio的Intellisense也指出.NET Core是否支持类型，方法，属性等。



**图5** Visual Studio IntelliSense指示.NET Core中是否有类或成员可用

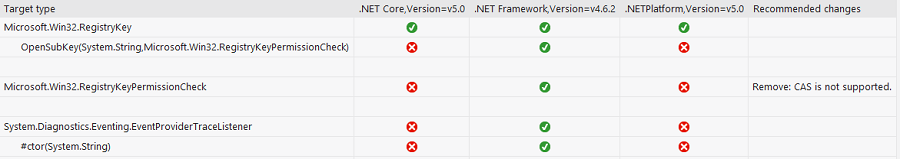
。NET可移植性分析器

如果您试图将现有的.NET应用程序转换为.NET Core，那么最好的地方就是.NET可移植性分析器。它既可以作为命令行应用程序，也可以作为Visual Studio插件使用。有许多.NET平台可供选择：



**图6** .NET可移植性分析器中提供的目标平台

尽可能使用此工具创建一个详细的报告，提供有用的建议：



**图7**。.NET可移植性分析器报告

**反射**

.NET Core中的反射工作方式与.NET Framework中的不同。最明显的区别是，Type类上正常的许多操作不再存在。有些人已经被调到一个新的TypeInfo类。其他操作在.NET标准库中不可用，并且需要额外的依赖性。

**.NET Framework实现反射**

.NET Framework实现反射 PropertyInfo targetProperty = null; foreach (var property in owningElement.GetType().GetProperties()) { if (property.Name == this.Name && property.PropertyType.IsGenericType) {

**The .NET Core 实现反射.**

PropertyInfo targetProperty = null; foreach (var property in owningElement.GetType().GetProperties()) { if (property.Name == this.Name && property.PropertyType .GetTypeInfo().IsGenericType) {

❶需要依赖于System.Reflection.TypeExtensions

❷ 许多类型操作移到了TypeInfo

**概要**

软件开发行业正在不断发展。从语言到框架，工具到方法论，一切都受到挑战和改进。.NET框架已经到了不得不做出改变的时候，以跟上竞争对手。.NETCore是.NET演进过程中必不可少的一步。它将最好的.NET Framework与现代软件开发中使用的实践相结合。诸如可移植性，简单部署，高性能，开放源代码和强大支持等特性使其值得一试。

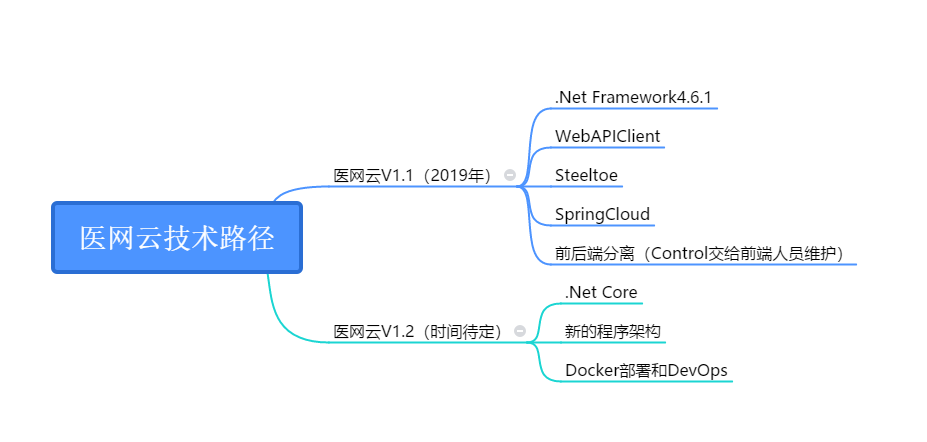
以上数据是.NETCore2.0数据，最新稳定版本是2.1.300，性能在2.0基础上又大幅提升，具体详情请Google

[Why Choose .NET Core?](http://freecontent.manning.com/why-choose-net-core/)

推荐免费高质量微服务书籍：[.NET微服务 容器化.NET应用架构指南](http://www.cnblogs.com/xiaoliangge/p/8566292.html)

.NET Core介绍 (PDF) <https://files.cnblogs.com/files/xiaoliangge/NETCore-chaunce.pdf>(可用于公司.netcore分享ppt使用)





医网云技术路径规划