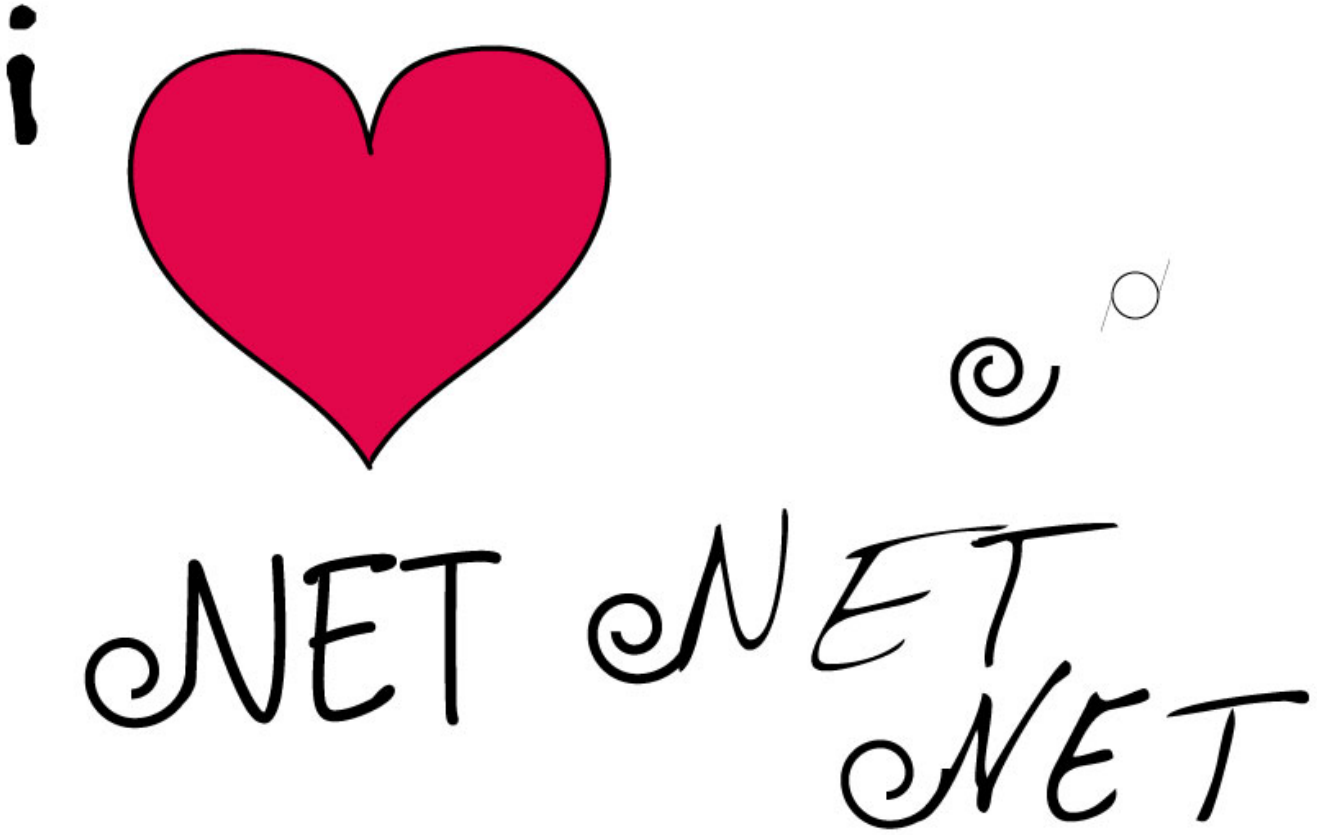
从 Mono、.NET Core 说起 - 文章 - 伯乐在线



前段时间，被问了这样一个问题：.NET 应用程序是怎么运行的？

当时大概愣了好久，好像也没说出个所以然，得到的回复是：这是 .NET 程序员最基本的。。。呵呵！

微软开源，其实不只是对 .NET 本身有利，从另一方面讲，对于 .NET 程序员来说，因为开源，你可以想了解到你想要的任何事。在之前的岁月，你可以“平凡”做一个默默无闻的 C# 代码撰写者，可以不用考虑任何事，使用宇宙最强大的 IDE - Visual Studio 编写代码后，发布到 IIS 即可，就是这么简单，甚至你不需要知道 IIS 是怎么运行的，代码是怎么托管的等等，你只需要写好你的 C# 代码即可，因为其余的一切工作，微软已经帮你考虑并完成了。

这样虽然简单并且高效，但对于 .NET 程序员来说，这就像“井底之蛙”一样，你看不到外面的世界，你也没有任何动力去跳出这个“深井”，而只是“舒适”的观望着外面你所认为的“狭隘世界”。但因为开源、跨平台，这些平静的日子都将被打破了，微软开源出来的一个个项目、发表的一篇篇文章，怎么越来越看不懂了？最后才发现，现在的微软已经不是原来我们熟悉的那个微软了，.NET 亦是如此。

我大概花了几天的时间，去真正阅读 .NET 开源的一些文章，虽然英文很烂，但我都坚持读了下来，阅读过程中，越来越被 .NET 的魅力所吸引，并时不时的被一些“特殊表达“所逗笑，但这都只是在阅读的过程中，读完、思考完，回到上面的那个问题，我还是不能很好的去表述、回答它，这时候我就意识到：脑子不好使，需要博客记录，这篇博文可以当做一篇阅读笔记来看（再一次的阅读）。

在很久很久之前，如果提到 .NET 跨平台，你首先想到的应该是 Mono，那 Mono 到底是什么？它为什么可以跨平台？在 .NET 开源之前，我对这些是一无所知，理解微软这次 .NET 跨平台，需要首先了解 Mono，了解 Mono，首先你需要仔细阅读下这篇文章：http://www.wikipedia.org/wiki/Mono。

摘录

Mono 是一个由 Xamarin 公司（先前是 Novell，最早为 Ximian）所主持的自由开放源代码项目。该项目的目标是创建一系列符合 ECMA 标准（Ecma-334 和 Ecma-335）的 .NET 工具，包括 C# 编译器和通用语言架构。与微软的 .NET Framework（共通语言运行平台）不同，Mono 项目不仅可以运行于 Windows 系统上，还可以运行于 Linux，FreeBSD，Unix，OS X 和 Solaris，甚至一些游戏平台，例如：Playstation 3，Wii 或 XBox 360。

重要的两点：

- C# 编译器：Mono 的 C# 编译器及其相关工具发布于 GNU 通用公共许可证（GPL）之下，其运行时库发布于 GNU 宽通用公共许可证（LGPL）之下，其类库发布于 MIT 许可证之下。这些均是开源协议，因此 Mono 是一个开源软件。
- 通用语言架构：微软开发了一个称为通用语言架构（Shared Source Common Language Infrastructure，Shared Source CLI；即今 ECMA—通用语言架构）的可用于 FreeBSD，Windows 和 Mac OS X 的 .NET 实现版本。微软的共享源代码协议并不是开源软件协议，且可能对于社区来说也是不足够的（它明文禁止了对软件的商业用途）。

再来看两个概念：

- 公共语言基础（Common Language Infrastructure，CLI）是一套标准（ECMA335），公共语言运行时（Common Language Runtime）即 CLR 是 CLI 标准的实现，Mono 是实现者之一。该运行时用于执行已编译的 .NET 应用程序。公共语言基础已被 ECMA 定义为标准 ECMA-335。要运行一个 .NET 应用程序，你必须使用相应的参数调用运行时。通用语言基础架构定义了构成 .NET Framework 基础结构的可执行码以及代码的运行时环境的规范，它定义了一个语言无关的跨体系结构的运行环境，这使得开发者可以用规范内定义的各种高级语言来开发软件，并且无需修正即可将软件运行在不同的计算机体系结构上。

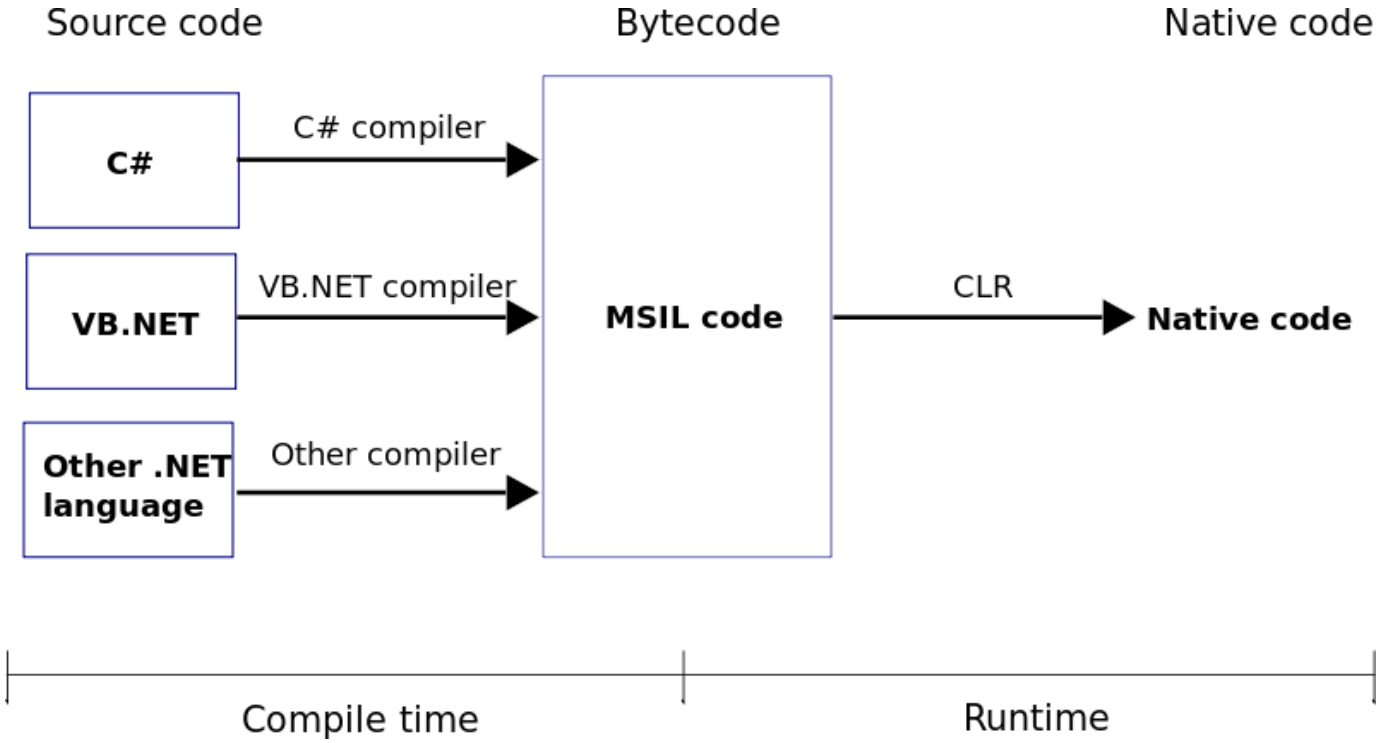
- 公共语言规范（Common Language Specification，CLS）定义了提供给公共语言基础的接口，例如对于枚举类型的隐含表示类型的协定。Mono 的编译器负责生成符合公共语言规范的映射代码，即公共中间语言（Common Intermediate Language，CIL）。Mono 的运行时将运行这类代码。ECMA 标准先前还定义了一个符合公共语言规范的程序库作为应用框架。

理解

对于上面一大堆的协议或概念，理解起来是需要花点时间，首先 Mono 的各个部分都是基于各种开源协议，也就是说它是一个彻彻底底的开源项目，再来大致理一下协议及其实现：

- GPL、LGPL 协议： C# 编译器及其相关工具实现。
- CLI（公共语言基础）：MS CLR、Mono（CLR 虚拟机），CLR 全称为 Common Language Runtime（通用语言运行时），需要注意的是，CLR 和 .NET Framework（.NET 框架）完全不是一个概念，.NET Framework 是以一种采用系统虚拟机运行的编程平台，以 CLR 为基础，支持多种语言（C#、VB.NET、C++、Python等）的开发库。
- CLS（公共语言规范）：MSIL（微软中间语言），现已更改为 CIL，也就是通过编译器生成的中间语言。

可以用来回答博文一开始那个问题的一张图：



开发人员使用高级编程语言撰写程序，接下来编译器将代码编译成微软的中间语言（MSIL），运行的时候 CLR 会将 MSIL 代码转换为操作系统的原生代码（Native code），CLR 内置有实时编译器（中间代码转化为原生代码的编译器）。

是不是还是有点晕？其实说简单也简单，抛开应用程序和编译器，.NET 的运行机制组成，其实就只有 CIL 和 CLR，CIL 是编译器生成的中间语言，CLR 的作用就是将它再编译成原始机器代码。你可能还有一个疑问，我们常说的 .NET Framework 到底是什么？它在 .NET 运行机制中到底扮演什么角色？在上面演示图中，其实 Source Code 并不包含 .NET Framework，Source Code 指的是你应用程序的源代码，在上面有提到，.NET Framework 是以 CLR 为基础的开发库，你可以把它看作是编译好的 CIL，它并没有通过编译器的再次编译过程，而是直接通过 CLR 中的编译器编译为原始机器代码。

简要总结：

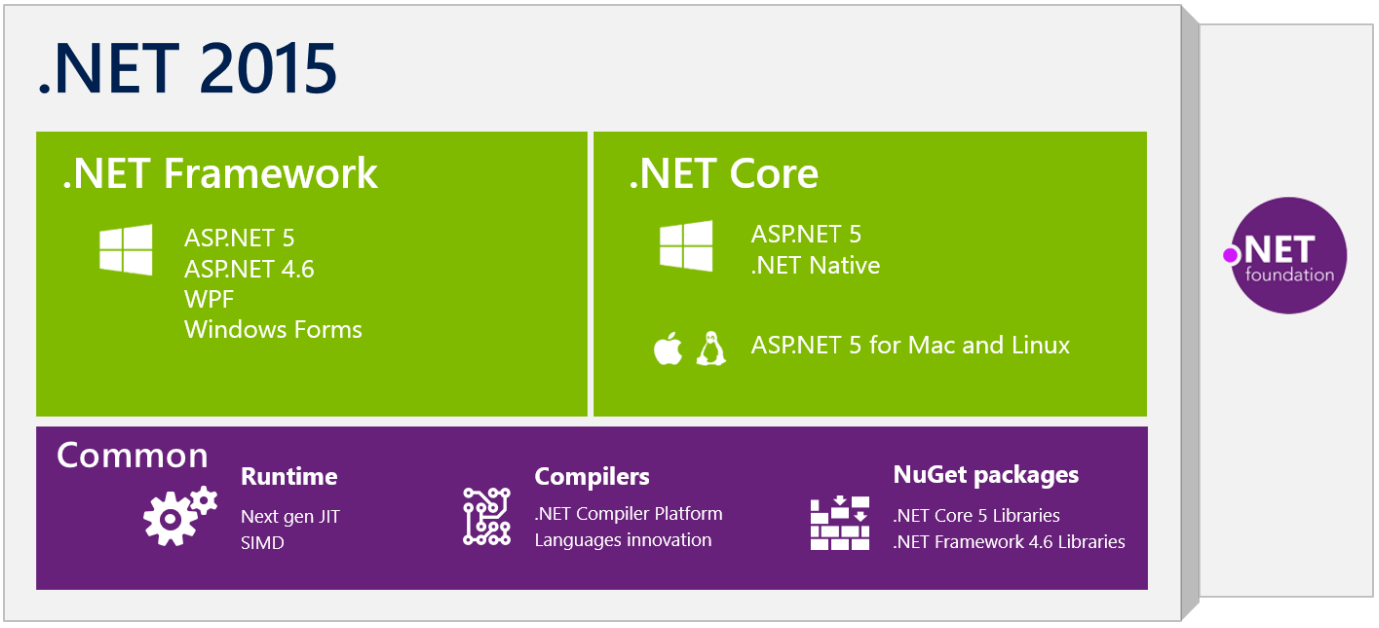
- CLI -> CLR
- CLS -> CIL
- CLR 运行 CIL

好了，我们再猜想一下，如果把 .NET 运行机制的各个组件进行分开重组，除去 .NET Framework 是微软提供的，其余的组件，比如 C# 编译器、CIL、CLR 等，都是 Mono 基于各种协议进行实现的组件，那如果你的应用程序代码引用了 .NET Framework（微软实现），很显然，这个应用程序是无法运行的，因为实现的 CLR 并不是微软的 CLR，所以针对 .NET Framework 来说，因为它的不开源，所以 Mono 需要另行实现基于自己 CLR 的基础类库，但这是一个庞大而复杂的工程，所以现在的 Mono 基础类库并不是很完整，很多东西和微软的 .NET Framework 并不能很好的兼容，所以你如果使用 Visual Studio 开发的项目，想迁移到 Mono Develop 下，是非常复杂的，当然，如果你没有使用任何的 .NET Framework，是非常简单的，但现实是，敲个最简单的“Hello World”输出代码，都得引用 mscorlib.dll 程序集（System.Console）。

上面描述那么多，我想表达什么呢？其实我们现在在 Windows 平台下开发的 .NET 应用程序，是深深依赖于 .NET Framework（深深的那种），你的应用程序代码基本上都是在它之上完成的，而 .NET

Framework 又是深深依赖于 Windows 平台下的 CLR（也是深深的那种），在这种情况下，根本就无法使你的应用程序跨平台，因为微软紧紧的抱住 Windows 平台，妄想 Windows 可以实现“大一统”，但现实是很残酷的，这次的 .NET 开源、跨平台，其实也是微软的无奈之举。但就是在这种背景下，Mono 出现了，并且在微软的各种“排挤”下坚持了下来，这是非常不容易的，其实实现 .NET 跨平台的三个关键是：编译器、CLR 和基础类库，而 Mono 实质上就是把他们三个进行跨平台实现了，一个很小团队完成了一个巨头需要完成的工作，而且还是在这个巨头的“排挤”下，其实这就是开源和社区的力量。

Mono 就说到这，有些疑问先放下，后面再进行讲解，下面我们来说一下 .NET Core 的相关内容。



需要认真阅读的一篇文章：[.NET Core is Open Source]

这是 .NET 开源的一篇官方说明文章，内容很多（包括评论），如果你真正花时间去读的话，这篇文章的信息量是非常大的，下面是我搜刮的一些片段。

.NET Core 开源地址：https://github.com/dotnet/corefx。

.NET Core 包含：

编译器、ASP.NET：

1. What is .NET Core?

答：.NET Core is a modular development stack that is the foundation of all future .NET platforms It’s already used by ASP.NET 5 and .NET Native.

关键词：Modular（模块化），现在的 .NET Core 只是增加了 .NET Core Framework 的一些代码，.NET Runtime 并未添加任何代码，这个需要时间，既然代码都未添加完善，那为什么 ASP.NET 5 and .NET Native 现在可以跨平台了？因为他们没有使用 .NET Core，而是使用独立的 Runtime—KRuntime，现在已更名为 XRE。

2. Why do we open source .NET Core?

比较“官方”的回答：

1. Lay the foundation for a cross platform .NET.
2. Build and leverage a stronger ecosystem.

其实翻译过来就是“形势所逼”，但事实真是如此吗？其实也不尽然，看这一段叙述就明白一些了：

> The challenge is that the Windows implementation has one code base while Mono has a completely separate code base. The Mono community was essentially forced to re-implement .NET because no open source implementation was available. Sure, the source code was available since Rotor but we didn’t use an OSI approved open source license, which made Rotor a non-starter. Customers have reported various mismatches, which are hard to fix because neither side can look at the code of the other side. This also results in a lot of duplicated work in areas that aren’t actually platform specific. A recent example is immutable collections.

这段叙述是很有意思的，因为 .NET 的不开源，Mono 社区重写了跨平台的 .NET，又因为 Rotor 没有基于任何 OSI 开源协议实现，所以导致 Mono 的实现充满了困难，这些都是微软亲自承认的，想想在以前，微软会说这些吗？相反它会高傲的抬起它的头，其实从这一方面就可以看出，Mono 社区这么多年是多么的不容易，微软这次的改变也是多么的不容易。

为了加深你的印象，微软列举了一个最新例子：immutable collections，Mono 团队是这样写的说明：Unfortunately Microsoft didn’t release their implementation under a license useable in mono I needed to reimplement the interfaces.，其实你可以看出一种无奈，Immutable Collections 是 .NET Framework 的基础核心类库之一，这还是之一，当然还有很多的实现，试想一下，如果 .NET 早就开放 .NET Framework 的源代码，即使微软没有对它进行跨平台，也会减少 Mono 团队的开发难度，以及尽量减少与 Windows 平台实现应用程序的兼容问题，当然这都是后话。

Build and leverage a stronger ecosystem.
关于这部分内容的解读，微软主要说了两点：Nuget 和 Interact，关于 Nuget，dudu 之前好像吐槽了好多次，但是微软说了，我们花了两年的时间去做这个东西，并得到了很好的反馈，但好像得到的反馈没有来自国内，使用 Nuget 的用意，其实就是之前说到的 Modular（模块化、组件化），这样的好处就是很好的扩展和维护，Interact 可以理解为开源带来的交互和进步，可以让开发者参与，并且反馈问题，利用社区的力量，让 .NET 更加完善。

很有意思的是，在最后作者举了一个生活的例子：我认为这就像驾驶一辆汽车，方向盘小的调整比大的调整，更加有效果，而且也更加安全，这个小的调整可以看作是来自社区的那一份份“微小”的力量，大的调整，当然是微软自己单搞，想想就知道哪个比较靠谱点。

3. Our choice of using GitHub

这部分也是非常有意思的，首先，微软做了一个调查，发现大部分的 .NET 社区都活跃在 GitHub 上，Don’t believe it?（看到这，我大笑了三声），不相信的话，微软给你举了一个真实的例子，nuproj（作者的一个个人项目），在 CodePlex 上放了两年，仅仅得到了一个 Pull 请求（想想也挺伤心的），然后把它移到 GitHub 上，仅仅五天后，就收到了三个 Pull 请求和两个提交，三个月的时间，总共收到了 16 个 Pull 请求和一些很有价值的提交。

作者的感受：one of the first ones was around adding unit testing, how awesome is that? 看来他是非常的激动！

需要注意的是，CodePlex 是微软构建的 Open Source 网站，它的竞争对手就是 GitHub，而微软却如此讽刺自己的“孩子”，并且放弃了它，选择了别人家的孩子，并把自己的“财产”传给它，这是需要多么大的勇气啊，虽然是迫不得已，但让一个巨头这样低头也是一件很难得的事（此处应该有掌声）。

4. Development in the open

这部分主要是讲微软如何做开源开发，提到了他们之前做的一个开源项目—Managed Extensibility Framework (MEF)，他们认为这是失败的，原因很简单，就是缺少社区的参与，刚才打开这个项目看了一下，基本上冷冷清清的，没有一个 Pull 请求（可以认为没人理它），文中提到了一个词 code bombs，不知道翻译为“代码炸弹”合不合适？什么是 code bombs 呢？结合文中的叙述，谈一下自己的理解，当开发一个开源项目，其实最重要的就是社区的参与，以及把每个代码提交都尽可能的细化、规范和公开，因为你不是自己在开发这个项目，是整个开源社区在做，你需要考虑的更加全面，但往往现实是这样的，维护团队因为得不到社区的反馈，开发或修复一个功能没有及时提交，总是隔很长的时间进行“整坨”的提交，这时候如果项目出现什么问题，提交的代码将会非常的难以查看和修复，而且提交的过程中不注意规范，最后变成“一坨X”，也就是“代码炸弹”，为了避免这种情况的发生，所以微软做了以下总结：

- Code reviews. We also want to have all code reviews the team is doing to be in the public as well, via GitHub’s pull request model.
- Design documents & discussions. We’ll also share design notes, specs, and implementation specific documentation. We’ll need to figure out exactly what format we’ll be using. At a minimum you can expect Markdown based documents, similar to Mad’s C# design notes. Another idea we had was recording our design meetings and share them on Channel 9. We need to figure out how we can do this on a somewhat regular cadence.

其实总结就是一个词：规范化，这样才会让项目更加“健壮”，之后提到了如果出现一些代码问题，该如何反馈，以及微软如何进行解决，这个就略过了，如果你发现了一些问题，可以使用 GitHub issue 进行反馈，微软说了，我们会立即处理，呵呵。

后面又讲述了两点：We accept contributions、Building and running your own forks，主要说明你如何进行代码提交，奈何自己功力还没达到这个境界，就此略过。

5. .NET Foundation

.NET Foundation（.NET 基金会）：http://www.dotnetfoundation.org/
.NET Foundation Projects：http://www.dotnetfoundation.org/projects

说明：The .NET Core project is under the stewardship of the .NET Foundation. We believe that to be a critical part in promoting and advancing the .NET Core stack. We’re closely working with Miguel de Icaza from Xamarin/Mono in order to create a shared code base that can become a cross-platform implementation of .NET Core.

上面提到了一个人名：Miguel de Icaza（米格尔·德伊卡萨），他是谁呢？他就是 Mono 的发起人，也是 Mono 团队的负责人，微软成立 .NET Foundation 第一步的目的，其实说白了，就是要把 Mono 拉过来，共同把 .NET Core 给维护起来。

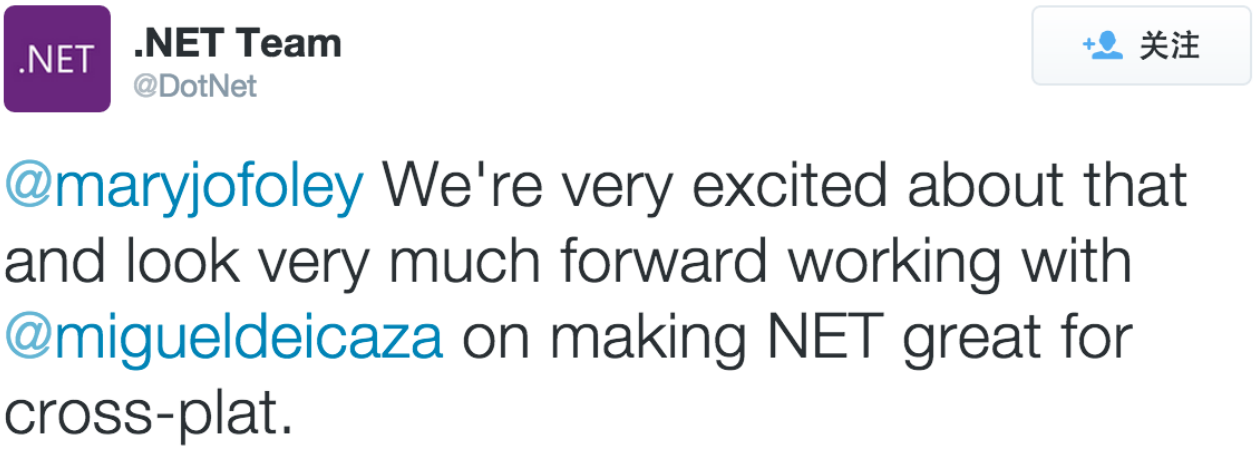
很多人可能会有这样的疑问：那 Mono 咋办？在之前曾提到这样一段话：The best way to build a cross-platform stack is to build a single stack, in a collaborative manner. And the best way to do exactly that is by open sourcing it.，关键词：single、collaborative manner，这两个词就很好的说明了，现在 Mono 与 .NET Core 之间的关系，也就是说他们相互合作，共同维护和管理单一的 .NET 跨平台实现（.NET Core），这也是社区的共同心愿。

6. The relationship between Mono and .NET Core

关于这个问题，上面可以看到一些答案，我们再来看一下更多的内容。

首先，在这篇博文中，有人这样评论：I hope you join forces with the Mono team. Miguel de Icaza has been promoting this for years!

作者回复的是一个 twitter 链接，内容：



如果有时间，可以再阅读下这几篇文章：

- .NET Source Code Integration（来自 Mono 官网）
- Introducing .NET Core（跳过前面内容，直接查看 Mono 部分）

摘录自 .NET Source Code Integration 的部分内容：

> We want to contribute some of the cross platform code from Mono, as well as chunks that we port from the .NET Framework Reference Source to the .NET Core effort.
> We will update this page with information as the porting process at Microsoft evolves and we can volunteer some of our/their code back to the .NET Core effort.

> The code is available today from http://github.com/Microsoft/referencesource. Mono will be able to use as much a it wants from this project.
> We have a project underway that already does this. We are replacing chunks of Mono code that was either incomplete, buggy, or not as fully featured as it should be with Microsoft’s code.
> With the Mono project, we have spent 14 years working on open source .NET. Having Microsoft release .NET and issue a patent covenant will ensure that we can all cooperate and build a more vibrant, richer, and larger .NET community.

仔细看 Miguel de Icaza 写的博文内容，你会感受到他当时的心情，是多么的激动（I am currently in NY celebrating :-)），后面大部分内容介绍了 Mono 需要做的工作，下面我们来侦查一下他们是如何进行交互工作的。

文中提到，微软开源了 Reference Source，referencesource 是什么？它其实就是 .NET Framework，只不过 GitHub 上并没有全部开源，详情请查看：http://referencesource.microsoft.com/，需要注意的是，这个 .NET Framework 是 Windows 平台下的，并不是我们所说的 .NET Core 的一部分，Mono 也 Fork 了这个项目，地址为：https://github.com/Microsoft/referencesource。

我们再具体随便看一下 referencesource 项目中一个具体类 System.ComponentModel.DataAnnotations.AssociationAttribute：

```
C##if !SILVERLIGHT
using System.Diagnostics.CodeAnalysis;
```

```
namespace System.ComponentModel.DataAnnotations {
        [AttributeUsage(AttributeTargets.Property | AttributeTargets.Field, AllowMultiple =
false)]
        [SuppressMessage("Microsoft.Performance", "CA1813:AvoidUnsealedAttributes",
Justification = "We want users to be able to extend this class")]
        public class ScaffoldColumnAttribute : Attribute {
                public bool Scaffold { get; private set; }
                public ScaffoldColumnAttribute(bool scaffold) {
                        Scaffold = scaffold;
                }
        }
        [AttributeUsage(AttributeTargets.Class, AllowMultiple = false)]
        [SuppressMessage("Microsoft.Performance", "CA1813:AvoidUnsealedAttributes",
Justification = "We want users to be able to extend this class")]
        public class ScaffoldTableAttribute : Attribute {
                public bool Scaffold { get; private set; }
                public ScaffoldTableAttribute(bool scaffold) {
                        Scaffold = scaffold;
                }
        }
}#endifC#// Copyright (c) Microsoft. All rights reserved.// Licensed under the MIT license.
See LICENSE file in the project root for full license information.
namespace System.ComponentModel.DataAnnotations
{
        [AttributeUsage(AttributeTargets.Property | AttributeTargets.Field, AllowMultiple =
false)]
        public class ScaffoldColumnAttribute : Attribute
        {
                public ScaffoldColumnAttribute(bool scaffold)
                {
                        Scaffold = scaffold;
                }
                public bool Scaffold { get; private set; }
        }
}
```

你会发现，.NET Core 中的实现完全变了，还需要注意的是，在每个类的上面都有了开源协议说明（MIT license），这个和之前 referencesource 项目是完全不同的，关于代码更改，这只是项目中的冰山一角，具体你可以对比查看两个项目代码到底有什么不同，有的被移除了，有的被重写了，这样改变的目的就是跨平台。

我们再看下 Mono 中这部分的代码（Windows 平台下），tree 目录：mono/mono/tree/master/mcs/class/System.ComponentModel.DataAnnotations，查看里面的目录结

构，没有发现一个源代码类文件，只有一些文档说明和单元测试代码，难道 Mono 没有实现吗？

答案当然不是的，查看一下最近的一个代码提交记录 https://github.com/mono/mono/commit/c7405e6ca0b5749433c9c7acd49a5a8521ed9094，然后在 System.ComponentModel.DataAnnotations.dll.sources 文件中，发现 Mono 进行了如下更改：

```
C#
../../build/common/Locale.cs
../../build/common/MonoTODOAttribute.cs
Assembly/AssemblyInfo.cs
-System.ComponentModel.DataAnnotations/AssociatedMetadataTypePropertyDescriptor.cs
-System.ComponentModel.DataAnnotations/AssociatedMetadataTypeTypeDescriptor.cs
-System.ComponentModel.DataAnnotations/AssociatedMetadataTypeTypeDescriptionProvider.cs
-...
+../../../external/referencesource/System.ComponentModel.DataAnnotations/DataAnnotations/AssociatedMetadataTypeTypeDescriptionProvider.cs
+../../../external/referencesource/System.ComponentModel.DataAnnotations/DataAnnotations/AssociatedMetadataTypeTypeDescriptor.cs
+../../../external/referencesource/System.ComponentModel.DataAnnotations/DataAnnotations/AssociationAttribute.cs
+...
```

返回上级、上级、上级，进入 external/referencesource 目录，你看到了熟悉的东西，查找到这，你也应该懂了，因为 referencesource 的开源（可以认为是 Windows 平台下的 .NET Framework），所以 Mono 没必要用以前自己实现的代码了，直接使用微软开源出来的代码，这是给 Mono 带来的一个改变，关于上面 .NET Core 这部分的实现代码，我想微软也应该参考了 Mono 跨平台的实现代码（非 Windows 平台），从这一方面讲，微软和 Mono 社区是互惠的，就像 Miguel de Icaza 所讲的那样，we can all cooperate and build a more vibrant, richer, and larger .NET community. 其实大家都是为了 .NET 社区的更好发展，说到这，在他的这篇文章中，有一个有意思的评论（Miguel de Icaza 的回复也是很有意思）：

Stuart Quinn · 2 months ago

"Microsoft has stated that they do not currently plan on taking patches back or engaging into a full open source community style development of this code base, as the requirements for backwards compatibility on Windows are very high."

Is this the main reason that Mono continues to exist at all? Can you see a time where Xamarin contribute directly to MS.net and abandon Mono altogether? In other words, now that .net itself is open-sourced, what reasons does Mono have to justify its existence?

6 · Reply · Share ›

migueldeicaza Mod ➦ Stuart Quinn · 2 months ago

Until the open sourced .NET becomes available everywhere where Mono matters or where some specifics of Mono are used, Mono will continue to be used.

Feel free to stop using Mono today, if all your needs are covered.

24 · Reply · Share ›

7. The relationship between .NET Core(corefx, CoreCLR) and XRE

我们先来看一下，微软在写那篇文章的时候所列出的一些计划：

- More libraries. Consider the subset we have today a down-payment on what is to come. Our goal is to open source the entire .NET Core library stack by Build 2015.
- Building and running on non-Windows platforms. We currently only provide the ability to build and run on Windows. We intend to build a public working group between us and the Mono community once we have enough code out there.
- .NET Core Runtime (CoreCLR). We’re currently figuring out the plan for open sourcing the runtime. Stay tuned!

再来看一下现在完成的进度：

- corefx：This repository contains the foundational libraries that make up the .NET Core development stack.
- Roslyn：The .NET Compiler Platform (“Roslyn”) provides open-source C# and Visual Basic compilers with rich code analysis APIs.
- XRE：The .NET Cross-Platform Runtime SDK (.NET XRE) contains the code required to bootstrap and run an application, including the compilation system, SDK tools, and the native CLR hosts.
- ….

对比计划和开源的项目，你可以往上面套一套了，首先，.NET Core 包含 .NET Core Framework（corefx，部分完成）和 .NET Core Runtime（CoreCLR，暂无），Roslyn 为跨平台的编译器，XRE 是什么？它是不是微软未完成的 CoreCLR，其实并不是的，我们再来看开发者提给微软的一些问题。

1. 链接地址：.NET Core Runtime

问：I’ve wanted to learn about the .NET runtime for a while and have tried studying Rotor and the spec but haven’t gotten very far. I was thinking of retrying with the .NET Core runtime but I can’t seem to find it in the GitHub repository.

答：Excellent question. It is really two questions.

- Where/when is CoreCLR? It is in a private repo on GitHub, but it isn’t ready yet. We’ll switch to public in the first part of next year. The team is focussed on making this happen.
  We’ve been thinking about how to make it easy for folks to learn about the runtime, in order to understand the code and make quality PRs. We have a bunch of internal documentation that we were thinking about making available in a WH wiki. Sound like a good idea?
- We have a bunch of requirements that are really important to satisfy. On another thread in the forums here, we’ve been talking about the relationship between .NET Core and the

> .NET Framework. We need an efficient way to flow code back and forth between .NET Core and the .NET Framework. Getting that right takes time, but will also pay dividends and allow us to take PRs more efficiently once we go public.

We had some of that infrastructure in place when we released the corefx repo, but we learned that it wasn't quite enough. We're taking more time upfront with the coreclr repo.

I'm glad to hear that you and others are excited to see the coreclr repo show up. We are too.

解读：这个提问者很犀利，直接问为什么在 GitHub 上，找不到 .NET Core Runtime 的项目，原来它被藏起来了（Private），还没有被开放出来，微软正在加紧时间开发这个东西（最重视的项目），后面说明了 .NET Core and the .NET Framework 之间的关系，目前 corefx 是 .NET Core 的部分实现，等到 coreclr 正式发布出来，才可以说它是完整的。

2. 链接地址：.NET Core CLR Questions!

问：Can I embed .NET Core and invoke the runtime like I can with "Embedded Mono"?

答：The CLR has had a public hosting API since its inception. SQL Server and IIS are great examples of CLR hosts. CoreCLR also has a hosting API, which ASP.NET 5 uses. This will be included in the CoreCLR repo, when it shows up.

问：Will .NET Native be open sourced as well so we could port it to other platforms and CPU types? (This would be important to compile .NET on iOS).

答：Right now, we are focussed on CoreCLR. I can imagine .NET Native following. Right now, CoreCLR has all of our attention, so we'll re-visit the .NET Native question after we've got CoreCLR on GitHub.

解读：第一个问题是，现在能不能像 Mono 一样启动 .NET Core，答案是如果使用 ASP.NET 5 是可以的，因为 ASP.NET 5 已经实现了，那就是 XRE，但 XRE 并不是 CoreCLR，它只是 CoreCLR 的一部分（included），第二个问题的答案在后半部分，我没有贴出来，但这部分答案会让你对 CoreCLR 有所期待，微软正在集中力量去开发它，我们只能期待它早点正式发布。

8. 有关 .NET 开源的一些其他疑问（来自文章评论）

1. 问：Does this mean yes or no to open-sourced WPF?

答：We currently don't plan on open sourcing the client stacks, which includes Windows Forms as well as WPF.

暂时没有计划开放 Windows Forms、WPF 源代码，是不是意味着被抛弃？

2. 问：Will I ever be able to run Visual Studio on Linux?

答：I’m not working for Visual Studio but I haven’t heard of any plans to offer a cross-platform version of Visual Studio itself.

其实我还想问下 on Mac？呵呵，多希望回答的是 Absolutely，但是还是很期待微软可以实现跨平台的 Visual Studio，这样才真正是宇宙最强大的 IDE！

3. 问：Is there a roadmap to “upgrade” WPF to .Net Core 5?

答：There are currently no plans to port either WinForms or WPF to .NET Core.

和第一个问题一样，微软没打算把 WinForms 和 WPF 的代码添加到 .NET Core 中。

4. 问：does anyone know is plain old WCF a part of CORE?

答：The client side of WCF will be included in .NET Core.

和 WinForms、WPF 相反，WCF 的代码将会被添加到 .NET Core 中。

5. 问：Immo, ok so all of this is leaving me a bit confused.What does all of this mean for me as an LOB developer?

答：The key difference is that .NET Core will be a single, cross platform stack.

Let’s contrast this with .NET Framework and Mono today. Mono is a full reimplementation of the .NET Framework. This means that certain features aren’t supported or implemented differently enough to cause behavioral changes that can break your application. Also, Microsoft doesn’t support running on Linux or MacOS today.

.NET Core on the other hand will be supported by Microsoft on Windows, Linux and MacOS. We’ll have a single code base that we plan on working on together with the Mono community.

For you that means that targeting other platforms will be more reliable and innovation will happen faster. It also means that you’ll be able to find out sooner what’s coming next because all the design work is happening in public.

Finally, you’ll also be able to take a look at the kitchen and engage in design discussion.

这个回答有点啰嗦，主要强调了 Mono 的贡献，然后再拉着 Mono 一块做单一的跨平台 .NET Core，最后就是过程开放。

插一段：including the entire CoreCLR runtime (JIT, GC, etc) as well as a set of class libraries. 可以看出 .NET Core 大致包含哪些东西。

6. 问：Will everything eventually have an MIT license? Or the Apache 2.0 license? Does it now? I’m confused.

答：Your confusion is very understandable. Some of the .NET open source projects (such as Roslyn) use Apache 2.0. Newer projects (such as .NET Core) use the MIT license. Both licenses are OSI approved open source licenses so either one is indicative of OSS.

Moving forward, we plan on exclusively using the MIT license. I don’t know whether we’ll switch the existing projects from Apache 2.0 to MIT.

这个是关于开源协议的，我们可以通过查看开源项目中，比如类文件的最上面有协议说明，ASP.NET 中大部分项目都是基于 Apache 2.0，最新的开源项目（比如 corefx）是基于 MIT，不知道这两个协议有什么不同，看作者的意思，微软是想把 Apache 2.0 协议转到 MIT 下，不知道可不可行。

7. 问：WebForms on Linux and Mac OS?

答：We’re making the .NET Core Framework cross plat, not the .NET Framework and .NET Core doesn’t include WebForms.

很简单，No。

8. 问：does this mean, i will be able to run .net desktop apps on linux, without the original developer having to port to mono?

答：We currently don’t plan to open source the client technologies (WinForms, WPF), nor taking them cross plat. So I’m afraid the answer would be no.

很简单，No plan。

9. 问：What is the relation between .NET Core and .NET Native?

答：.NET Native is two things: it’s a tool chain that pre-compiles IL to native and it’s a runtime that executes that code. .NET Core is CLI compliant stack that runs both, JIT based applications (via CoreCLR) as well as ahead-of-time compiled (AOT) applications (produced by .NET Native).

这真是一个好问题，什么是 .NET Native？它最大的特点，是可以直接将应用程序代码编译成本地机器码，这是和 .NET Core 最大的区别，所以作者提到了 CLI compliant，这部分在上面 Mono 中有说明。

10. 问：You said that the client side of WCF will be included in .NET Core. What about the server side of WCF? I mean .NET Core will be a Server side technology if I understand correctly

答：That’s an execellent question for the WCF team. I’ll forward that question.

这个上面有个问题搞错了，“client side of WCF will be included in .NET Core”，注意是 Client，而并不是 Server，这个两个不同的概念，Client 的作用就是为了兼容之前开发的 WCF 项目。

11. 问：.NET Core Runtime is the same runtime which .NET Framework uses?

答：.NET Core has two runtimes: CoreCLR and .NET Native.

CoreCLR isn’t identical to the CLR but it’s very close. The key pieces are virtually identical, so same GC, same JIT, same type system etc.

In .NET Native, it’s obviously a lot more different as .NET Native doesn’t have a JIT, but the GC, for example, is the same.

How close is .NET Core Runtime to Mono Runtime?

They are different implementations, done by different people. But both implement the same ECMA CLI standard, so they are quite similar in what they do, but most likely not in how they do it.

好问题，好回答，CoreCLR 和 .NET Native 是完全不同的模式，.NET Native 因为直接可以讲程序代码编译成本地机器码，所以根本不需要 JIT，另外一个问题是 .NET Core Runtime 和 Mono Runtime，首先他们是不同的实现，但他们都是基于 ECMA CLI（公共语言基础），他们的相似之处都是将 CIL 编译成本地机器码。

前段时间，dudu 写了一篇这样的博文：挡不住的好奇心：ASP.NET 5是如何通过XRE实现跨平台的，最后产生了这样一个疑问：在 Core CLR 被加载，Microsoft.AspNet.Hosting 被执行之后，为什么还要用 Mono Runtime 加载一些 dotnet.host 相关的程序集？为什么不直接用 Core CLR 加载呢？

其实 XRE 并不是真正的 .NET Core Runtime，可以说是它的一部分，或者是一个先行版本，为什么要借助 Mono Runtime？很简单，因为 .NET Core Runtime 并不完善，我相信微软和 Mono 社区现在正在加紧完善这个最重要的项目 coreclr，至于 .NET Core Framework（corefx）的开发，有了微软的 Windows .NET Framework 和 Mono 自行实现的跨平台 .NET Framework，其实移植到 corefx 的难度并不大，重要的还是 coreclr，请注意这个链接地址：https://github.com/dotnet/coreclr，说不准过几天就不是“404”了，期待吧！

如果你能坚持看到这里，我真是感激涕零！！！

- 小组 - 好的话题、有启发的回复、值得信赖的圈子
- 头条 - 分享和发现有价值的内容与观点
- 相亲 - 为IT单身男女服务的征婚传播平台
- 资源 - 优秀的工具资源导航
- 翻译 - 翻译传播优秀的外文文章
- 文章 - 国内外的精选文章
- 设计 - UI,网页，交互和用户体验
- iOS - 专注iOS技术分享
- 安卓 - 专注Android技术分享
- 前端 - JavaScript, HTML5, CSS
- Java - 专注Java技术分享
- Python - 专注Python技术分享