学号： 2020302111004

年级： 2020级

**武汉大学****计算机学院**

**Windows原理与应用**

**期末论文**

# .NET平台介绍

院 系 名 称：计算机学院

专 业 名 称：软件工程

课 程 名 称：Windows原理与应用

姓 名：李西洋

学 号：2020302111004

指 导 教 师：胡继承

**郑 重 声 明**

本人呈交的课程论文，是在导师的指导下，独立进行研究工作所取得的成果，所有数据、图片资料真实可靠。尽我所知，除文中已经，，注明引用的内容外，本学位论文的研究成果不包含他人享有著作权的内容。对本论文所涉及的研究工作做出贡献的其他个人和集体，均已在文中以明确的方式标明。本课程论文的知识产权归属于培养单位。

团队成员签名： 李西洋 日期： 2022.12.20

## 摘要

.NET是微软开发的，一种用于构建多种应用的免费开源开发平台，可以使用多种语言，编辑器和库开发Web应用、Web API和微服务、云中的无服务器函数、云原生应用、移动应用、桌面应用、Windows WPF、Windows窗体、通用 Windows平台 (UWP)、游戏、物联网 (IoT)、机器学习、控制台应用、Windows服务。

本文从.NET的定义出发，阐述其发展历程，历史内容、现状、发展趋势、贡献以及在计算机领域解决复杂系统及工程问题过程中现实的和未来的应用等内容。

关键词**：** .NET；.NET Framework；.NET Core；跨平台;开源

**目录**

[.NET平台介绍 1](#_Toc32405)

[摘要 3](#_Toc30594)

[1. .NET平台相关定义 5](#_Toc14157)

[1.1、什么是 .NET？ 5](#_Toc16423)

[1.2、什么是.NET implementation? 5](#_Toc13588)

[1.3、.NET 体系结构的组件是什么？ 5](#_Toc12108)

[1.4、什么是 .NET 编程语言？ 5](#_Toc24409)

[1.5、什么是 .NET runtime？ 6](#_Toc13831)

[1.6、什么是 .NET 应用程序模型框架(application model frameworks)？ 6](#_Toc29873)

[2. .NET平台发展历程 7](#_Toc19208)

[2.1、微软 .NET 战略初衷: 7](#_Toc16821)

[2.2、2000年至2014年： 7](#_Toc7812)

[2.3、2015年至2019年： 8](#_Toc31682)

[2.4、2020年至今 8](#_Toc9854)

[3. .NET平台历史版本具体介绍 8](#_Toc18070)

[3.1、.NET 1.0（2002年） 8](#_Toc3669)

[3.2、.NET 1.0（2005年） 8](#_Toc25651)

[3.3、.NET 2.0（2007年） 9](#_Toc11196)

[3.4、.NET 2.5（2008年） 9](#_Toc18533)

[3.5、.NET 3.0（2010年） 9](#_Toc27790)

[3.6、.NET 3.5（2012年） 9](#_Toc21977)

[3.7、.NET Core（2016） 10](#_Toc3094)

[3.8、.NET4.0（2020年） 11](#_Toc30900)

[3.9、.NET 6（2021年） 11](#_Toc5061)

[3.10、.NET 7（2022年） 11](#_Toc13059)

[4. .NET平台现状 12](#_Toc2668)

[4.1、工具 12](#_Toc4471)

[4.2、跨平台 12](#_Toc3699)

[4.3、多语言 13](#_Toc29522)

[4.4、强大的社区和开发者 13](#_Toc15019)

[4.5、开源 13](#_Toc9841)

[4.6、受企业信任 13](#_Toc29342)

[5. .NET平台发展趋势 13](#_Toc8395)

[6. .NET平台的贡献 16](#_Toc23190)

[6.1、对于用户 16](#_Toc10988)

[6.2、对于开发人员 16](#_Toc8373)

[6.3、对于行业标准 17](#_Toc11051)

[7. .NET平台现在及未来应用 18](#_Toc10183)

[8. 结语 19](#_Toc32685)

[9. 参考文献 20](#_Toc27068)

## .NET平台相关定义

### 1.1、什么是 .NET？

.NET 是一个开源平台，用于构建可在任何操作系统上本机运行的桌面、Web 和移动应用程序。.NET 系统包括支持现代、可缩放和高性能软件开发的工具、库和语言。

* 将 .NET 编程语言代码转换为计算设备可以处理的指令。
* 为高效的软件开发提供实用程序。例如，它可以查找当前时间或在屏幕上打印文本。
* 定义一组数据类型以在计算机上存储文本、数字和日期等信息。

### 1.2、什么是.NET implementation?

.NET的各种实现允许.NET代码在不同的操作系统上执行，如Linux，macOS，Windows，iOS，Android等。

**.NET Framework ：**

.NET Framework 是最初的 .NET 实现。它支持在 Windows 上运行网站、服务、桌面应用程序等。微软在1990年代初发布了.NET Framework。

**.NET Core：**

微软在2014年底推出了.NET Core，为.NET开发人员提供跨平台支持。该公司于 2020 年 11 月发布了最新版本的 .NET Core .NET 4.0，并将其重命名为 .NET。本文中的术语 .NET 是指 .NET 4.0。.NET Core 在 GitHub 上是开源的。

**.NET Standard ：**

.NET 标准是不同函数（称为 API）的正式规范。不同的 .NET 实现可以重用相同的代码和库。每个实现都使用 .NET 标准 API 和特定于其运行的操作系统的唯一 API。

### 1.3、.NET 体系结构的组件是什么？

.NET 具有模块化和优化的体系结构。用户可以选择不同的组件来满足他们的软件开发要求。

以下是三个主要的 .NET 组件：

* .NET languages
* Application model frameworks
* .NET runtime

开发人员使用 .NET languages和Application model frameworks来创建其 .NET 应用程序。然后，.NET runtime执行并运行它们。

### 1.4、什么是 .NET 编程语言？

C#，F#和Visual Basic是Microsoft支持的三种NET开发语言。不同的公司和开发人员还创建了与 .NET 平台配合使用的其他语言。

* C# 是一种简单、现代且面向对象的编程语言。C# 的语法类似于 C 语言家族，C#对于 C、C++、Java 和 JavaScript 程序员来说都很熟悉。
* F# 具有轻量级语法，只需很少的代码即可生成软件。它是一种开源语言，可以轻松编写简洁、健壮且高性能的代码。它还具有强大的编程规则系统和方便的标准库，用于构建关键任务、正确、快速和可靠的软件。
* Visual Basic是由Microsoft开发的一种面向对象的编程语言。使用 Visual Basic 可以快速轻松地创建类型安全的 .NET 应用。类型安全是编程语言阻止或防止逻辑编码错误的程度。
* 公共语言基础结构（CLI）语言,像ClojureCLR，Eiffel，IronPython，PowerBuilder和许多其他语言也可以在.NET平台上工作。这是因为 .NET 实现了公共语言基础结构（CLI）。可以将 CLI 视为与. NET兼容的语言。

### 1.5、什么是 .NET runtime？

.NET 运行时（也称为公共语言运行时（CLR））在不同的操作系统上编译和执行 .NET 程序。

**即时编译:**

CLR 在开发人员编写代码时编译代码。在编译过程中，CLR 将代码转换为公共中间语言 （CIL）。例如，用 C# 编写的代码具有类似英语的语法和单词。.NET 将此代码编译或转换为 CIL。CIL 代码看起来不同，因为它是一种较低级别的机器代码语言。

**执行:**

.NET runtime管理 CIL 代码的执行。CIL是跨平台兼容的，任何操作系统都可以处理它。跨平台兼容性是指应用程序以最少的修改在多个不同的操作系统上运行的能力。例如，C# 中的应用程序可以在 Windows、Linux 或 macOS 上运行，而无需修改任何代码。此类应用程序称为跨平台应用程序。

### 1.6、什么是 .NET 应用程序模型框架(application model frameworks)？

应用程序模型框架是支持快速高效的 .NET 项目开发的开发人员工具和库的集合。不同类型的应用程序存在不同的框架，如下所示。

* **网络应用程序(Web applications)**

ASP.NET 框架扩展了专门用于构建基于 Web 的应用程序的 .NET 开发人员平台。它支持Web技术，如REST API，HTML，CSS和JavaScript。它提供了一个具有多因素和外部身份验证的内置用户数据库。ASP.NET 支持具有内置安全机制的行业标准身份验证协议，以保护 .NET 应用免受网络攻击。

* **移动应用(Mobile applications)**

您可以使用 Xamarin/Mono 在所有主要的移动操作系统（包括 iOS 和 Android）上运行 .NET 应用。Xamarin包括Xamarin.Forms，一个开源移动用户界面框架。.NET 开发人员使用 Xamarin.forms 跨移动平台创建一致的用户体验。所有 .NET 应用的外观都相同，即使在不同的移动设备上也是如此。

* **桌面应用程序(Desktop applications)**

可以使用 Xamarin 进行桌面应用程序开发。此外，通用Windows平台扩展了.NET Windows 10应用程序开发。Windows Presentation Foundation 和 Windows Forms 是 Windows 上用户界面设计的其他框架。

* **其他应用(Other applications)**

使用 ML.NET，可以开发自定义机器学习模型并将其集成到 .NET 应用程序中。可以使用 .NET IoT 库在传感器和其他智能设备上开发应用程序。对于框架中不可用的任何解决方案，可以在公共 NuGet 存储库上找到许多特定的函数库。可以使用 Nuget 创建、共享和使用许多 .NET 库，几乎可以用于任何目的。

## .NET平台发展历程

### 2.1、微软 .NET 战略初衷:

　　20 世纪 90 年代，微软只有 Visual Basic 和 Visual C++ 语言。同时期 Sun 公司在 1995 年推出了 Java 语言，在业界得到了广泛的支持。

　　短短几年内，Java 迅速建立了庞大的生态系统，微软也不得不考虑加以应对。即使在这样的竞争压力下，当时的微软 CEO 比尔·盖茨也毫不畏怯。他雄心勃勃地说，微软将“为新一代的互联网奠定基础”。这就是那个时期的比尔·盖茨，曾经连续 13 年蝉联世界首富的人(从 1995 年开始)。

　　于是，微软开始制定 .NET 战略，并为之而努力。这个战略背后的主要想法是要将 .NET 涵盖进所有的微软产品中，并在现有的 Windows 产品名称中添加 .NET 结尾，即 Windows.NET。在 2000 年，微软把下一代 Windows 服务(简称 NGWS)的名字取名为 .NET Framework，从此 .NET 这个名字家喻户晓。也是 2000 年，比尔·盖茨卸任微软 CEO，史蒂芬·鲍尔默上任。

### 2.2、2000年至2014年：

2000 年至 2014 年这 14 年间，微软的主要开发产品线是 .NET Framework。

2000 年末，微软发布第一个 .NET Framework Beta 版本，紧接着在 2001 年初发布第一个 .NET Framework 正式版 v1.0。在 2003 年升级到 v1.1，完善了桌面应用开发和 Web 开发，推出了 ASP.NET。2006 年左右，微软相继推出了 .NET Framework 的 1.0 和 2.0 版本。在 2.0 版本中，微软推出了 WPC、WCF、WF 等框架。随着 2.0 的问世，微软进入黄金时代，称霸了一大半开发市场。不过好景不长，在短短几年后的 2009 年左右，开发市场风起云涌，Web 开发大行其道，Java、PHP 崛起。微软先前打下的江山开始迅速被开源的 Java、PHP、Ruby 等瓜分。为了夺回失去的 Web 开发市场，微软在 2009 年发布了 Web 框架 ASP.NET MVC 1.0 并完全开源，并在随后几年不断完善。

2010 年至 2014 年，.NET Framework 从 3.0 升级到 3.5。ASP.NET MVC 框架也在不断升级，期间不仅没有夺回 Web 开发市场，反而还在继续失守。2014 年，微软最终向市场妥协，开始热情拥抱开源和跨平台。微软先后开源了 Roslyn、ASP.NET 等众多项目。也是在领取这一年，萨提亚·纳德拉作为微软第三任 CEO 上任。

### 2.3、2015年至2019年：

2015 年，微软决定开启一条全新的跨平台产品线：.NET Core，并完全开源。同时，这一年微软把 .NET Framework 升级到了 3.6 版本。

2016 年，.NET Core 问世，微软推正式出 .NET Core 1.0。同时也推出了新的跨平台 Web 框架 ASP.NET Core 和开发 Win10 应用的 UWP 框架。之后几年，.NET Core 以几乎每年一个版本的速度迅速迭代。

2017 年，.NET Core 1.0 发布，同时传统的 .NET Framework 升级到了 3.7 版本。

2019 年，.NET Core 2.0 发布，这个版本代表着 .NET Core 已经彻底稳定下来，且它的生态圈和社区也发展地非常成熟。这一年 .NET Framework 升级到了 3.8，不过此时的 .NET Framework 新版本已经无人问津，这也是 .NET Framework 的最后一个版本。

### 2.4、2020年至今

2020 年，为了避免产品线的混乱，微软开始统一产品线，关闭了 .NET Framework 这条产品线，并将 .NET Core 产品线更名为 .NET。为了避免与 .NET Framework 3.x 版本号冲突，.NET 产品线直接从版本号 5 开始，即 .NET 4.0，并于这一年 11 月正式发布。

2021 年11月，微软推出统一的 .NET 长期支持版本 6.0。.NET 6 是一个 LTS 版本，它支持 3 年，直到 2024 年 11 月。

2022 年11月，微软推出.NET 6.0。.NET 7 是 STS 版本，它支持 18 个月，直到 2024 年 5 月。

2023 年，微软计划推出.NET 8.0。

## .NET平台历史版本具体介绍

### 3.1、.NET 1.0（2002年）

在 .NET 出现之前，编写 Windows 上可以运行的程序就要直接和 COM（微软的一套软件的接口标准）打交道。.NET 框架试图通过 CLR 代替 COM，并提供更多功能，令用户可以更轻松的使用 Windows 的资源进行软件开发。相应地 Visual Studio 也升级到 7，称为 Visual Studio 2002（上一个版本叫做 Visual Studio 6）。该框架包括 C# 和 VB.NET（ Visual Basic 的继任者）。同时，第一个版本的 ASP.NET 也在 .NET Framework 1.0 亮相，它作为网站的解决方案，一直是 .NET 框架最重要的产品线之一。

### 3.2、.NET 1.0（2005年）

2005 年 11 月，.NET Framework 1.0 发布。该框架包括 C# 2，相比第一代添加了新的语法特性，其中最重要的就是泛型。与此同时，CLR 的版本也相应更新为 1.0，Visual Studio 也升级到 Visual Studio 2005。.NET Framework 1.0 中，Web 应用解决方案仍然是 ASP.NET WebForm， 但新增了多种多样Web Service ，包括 .NET Remoting（TCP/HTTP/Pipeline communication）以及基础的 Winsock 等。Windows Form 则作为 Windows下的应用程序解决方案。ADO.NET则作为数据库连接方案（支持 SQL Server 和 Oracle）

相比于.net 1.0，.NET1.0存在大量的API变更。新的API让需要管理.NET运行库实例的非.NET的应用程序可以做到这点。这个新的API对.NET运行库的各种功能，包括：多线程、存储器分配、代码加载等，提供了很好的控制。它最初是为Microsoft SQL Server能够有效率的使用.NET运行库而设计的，因为Microsoft SQL Server拥有它自己的日程管理器和存储器管理器。

### 3.3、.NET 2.0（2007年）

.NET 2.0 标志着 Windows 平台开始全面转向 .NET 时代（以后所有版本的 Windows 都预装 .NET）。而在这之前，只有服务器版本的 Windows 会预装 .NET。当时的 .NET 只能在 Windows 上运行，通过 CLR 调用 Windows 的 API，从而控制电脑硬件。.NET 2.0 的 CLR 和 .NET 1.0 相同。

.NET Framework 2.0主要由三部分组成：

WPF（Windows Presentation Foundation），内部代码“Avalon”，是一套基于XAML（eXtensible Application Markup Language）、.NET和矢量图技术的全新的图形界面系统和API，它充分利用了计算机的3D显卡硬件功能和Direct3D技术。

WCF（Windows Communication Foundation），内部代码“Indigo”，是一个面向服务的通讯框架，利用它可以实现本地和远程的程序之间的交互。

WF（Workflow Foundation）于2005年8月公开，微软提供的工作流引擎，通过它可以实现任务的自动化和事务的集成。InfoCard是一个软件组件的代码，用来安全的保存和使用用户的数字身份，并提供统一的界面来针对特定的事务选择身份，例如登陆一个网站。

### 3.4、.NET 2.5（2008年）

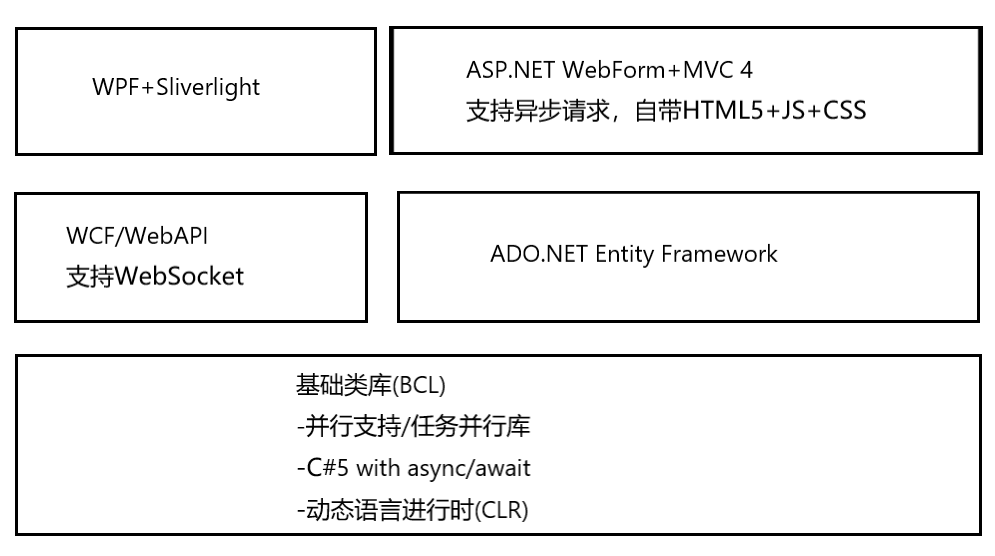
.NET 2.5 集成了 C# 3，包括 C# 3 最重要的一个功能，即 LINQ。相应地，为了支持 LINQ，此版本也加入了扩展方法、Lambda 表达式等新功能。从 C# 3开始，C# 不再是 Java 的一个跟班小弟，某种意义上实现了对 Java 的超越（Java 直到 2014 年才在 Java 8 加入了 Lambda 表达式）。另外，.NET 2.5 SP1 新增了 ADO.NET Entity Framework 取代 ADO.NET，作为 ORM 的解决方案。在 ADO.NET Entity Framework 发表之前，NHibemate 是在 .NET Framework 上经常使用的 ORM 实现，顾名思义，它是 Hibernate 在 .NET Framework 上的实现。相应地，Visual Studio 也升级到了 Visual Studio 2008。

### 3.5、.NET 3.0（2010年）

.NET 3.0 集成了 C#4，主要增加了动态语言运行时（DLR）和任务并行库（TPL）,它包括 PLINQ、任务等，优化了多线程的编程方式。CLR 由 2 直接升级到4。相应地，Visual Studio 也升级到了 Visual Studio 2010。

### 3.6、.NET 3.5（2012年）

.NET 3.5 和 C#5 —起发布，.NET 3.5 框架的生态系统如下图所示。.NET 3.5框架的生态系统C# 5 基本上全部都是围绕 async/await 关键字的。ASP.NET MVC 作为 ASP.NET 的一组类库，于 2009 年初次发布，最新的版本是 ASP.NET MVC 4.2。Web Service 也在不断进化，从基于 XML 的 WCF 发展到 RESTful 的 WebAPI。



从图中可以看到，最上面的应用层，ASP.NET 茁壮成长，继续和 PHP/JSP 等技术分庭抗礼，用户有 WebForm 和 MVC 可供选择。在 JS 方面，选择了 jQuery 作为官方 JS。值得注意的是，MVC 中的 M 由 ADO.NET Entity Framework 负责。Windows 下的解决方案仍然是WPF。中间的服务和数据层中，Web 服务拥抱 RESTful，数据库方面也是由 ADO.NET Entity Framework 唱主角。最下面当然就是基础类库（BCL）。在 C# 5 中，增加了对异步编程的简化。相应地，Visual Studio 也升级到了 Visual Studio 2012。在这之后，.NET 还有更高的版本 3.6 和 3.7，对应 Visual Studio 2015 和 Visual Studio 2017。

### 3.7、.NET Core（2016）

.NET Core 是适用于 Windows、Linux 和 macOS 的免费、开源托管的计算机软件框架，是微软开发的第一个官方版本，具有跨平台能力的应用程序开发框架 (Application Framework)，未来也将会支持 FreeBSD 与 Alpine 平台，也是微软在一开始发展时就开源的软件平台，它经常也会拿来和现有的 开源 .NET 平台 Mono 比较。

由于 .NET Core 的开发目标是跨平台的 .NET 平台，因此 .NET Core 会包含 .NET Framework 的类库，但与 .NET Framework不同的是 .NET Core采用包化 (Packages) 的管理方式，应用程序只需要获取需要的组件即可，与 .NET Framework 打包式安装的做法截然不同，同时各包亦有独立的版本线 (Version line)，不再硬性要求应用程序跟随主线版本。

.NET Core 项目的主要目标有：

•支持或可以移转 (port) 到更多的操作系统平台与芯片架构 (也就是未来项目会跨出 x86 平台)。

•具有引人注目的性能与高可靠度。

•开发人员能快速与直接的获取 .NET Core 开发环境。

### 3.8、.NET4.0（2020年）

.NET 5 => .NET Core vNext，NET 5 是 .NET Core 的下一版本。该项目旨在通过以下几个关键方式改进 .NET：

•构建一个可在任何地方（Any where）使用的 .NET 运行时和框架，并具有统一的运行时行为和开发人员体验。

•通过充分利用 .NET Core、.NET Framework、Xamarin 和 Mono 来扩展 .NET 的功能。

•从单个代码库构建该产品，开发人员（ Microsoft 和 社区）可以一起工作并一起扩展，从而改进所有方案。

该项目和方向是 .NET 的一个重要转折。使用 .NET 5，无论您正在构建哪种类型的应用程序，您的代码和项目文件都将是相同的。每个应用都可以访问相同的运行时、API 和语言功能。也包括几乎每天都在进行的 corefx 的性能改进。

### 3.9、.NET 6（2021年）

2021 年 11 月 8 日，微软正式发布了 .NET 6 及其一系列内容，推出了 C# 10、F# 6 和 PowerShell 6.2，.NET 6 还原生支持了苹果 M1 芯片，将得到三年支持。.NET 6 是从 .NET 5 开始的 .NET 统一计划的最后部分，在 .NET 历程上具有里程碑意义。这种统一体现在跨平台、桌面、IoT和云应用的统一的 SDK、基本库 和 运行时（Runtime）。除了这种统一，.NET 6 生态系统还提供：

•简化的开发：入门很简单。C# 10 中的新语言功能减少了需要编写的代码量。通过 Web 堆栈 和 最小 API(minAPI) 方面的投资，可以轻松快速编写更小、速度更快的微服务（Microservice）。

•更好的性能：.NET 6 是最快的完整堆栈 Web 框架，如果是在云中运行，则它可以降低计算成本。

•终极工作效率：.NET 6 和 Visual Studio 2022 提供热重载（Hot Reload）、新的 git 工具、智能代码编辑、可靠的诊断和测试工具，以及更好的团队协作。

•重要点：.NET 6 将支持三年（2021年11月8日到2024年11月12日） ，作为 LTS (长期) 支持。

2021 年 11 月 8 日发布的 .NET 6 是 .NET 团队和社区一年多努力的结果。其中 C# 10 和 F# 6 提供语言改进，使代码更简单、更好。性能有了巨大的提升。.NET 6 首次发布了对本地化 Apple Silicon（Arm64） 的支持，并且还改进了 Windows Arm64 的相关性能。.NET 6构建了一个新的动态配置文件导向优化 （PGO） 系统，该系统可提供仅在运行时才可能实现的深度优化。云诊断已改进与 dotnet monitor 和 Open Telemetry。WebAssembly 支持更有能力、更具有性能。新的 API 已经添加支持 HTTP/3，处理 JSON 数据，并直接操纵内存。

### 3.10、.NET 7（2022年）

微软在 2022年 11 月 8 日至 10 日的 .NET Conf 2022 上发布 .NET 7 正式版。.NET 7 RC 1 包含多项改进：

•.NET MAUI：.NET 多平台应用程序 UI (MAUI) 将 Android、iOS、macOS 和 Windows API 统一到一个 API 中，开发者可以编写一个在多平台上本机运行的应用。作为 .NET 7 的一部分，.NET MAUI 提供了一个项目来处理跨设备及其平台的多目标。

•云原生（Cloud Native）：云原生是一组最佳实践，用于在云中构建应用，以利用弹性、可扩展性、效率和速度。

•ARM64：.NET 可帮助开发者构建在 ARM 设备上运行的应用，.NET 7 将迎来多项改进。

•现代化：为了使升级体验尽可能无缝，.NET 升级助手为开发者提供分步指导体验，通过分析和改进项目文件、代码文件和依赖项来现代化 .NET 应用。

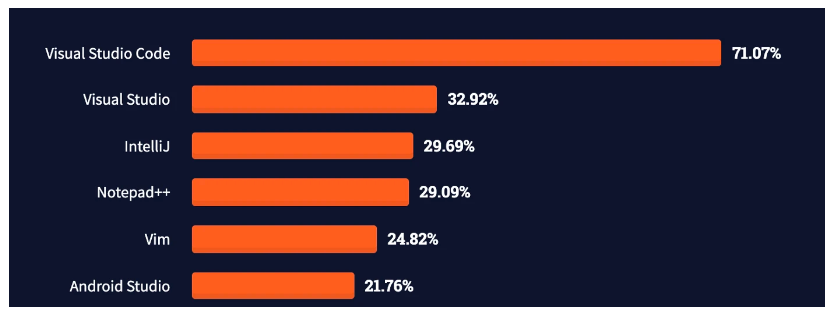
•性能：.NET 7 是目前最快的 .NET。.NET 7 对反射、堆栈替换 (OSR)、启动时间、本机 AOT、循环优化和许多其他领域进行了超过一千项影响性能的改进。

•支持：.NET 7 不是长期支持 (LTS) 版本，因此在发布之日起仅能获得 18 个月的免费支持和补丁。

## .NET平台现状

### 4.1、工具

Windows 和 Mac 的 Visual Studio 为.NET 提供了一流的支持，VS Code 也为其提供了有限的支持（微软的一个疏忽，我们将在下面讨论）。它们代表了世界上使用率最高的 IDE，事实上，VS Code 是 IDE 的绝对统治者，使用人数是 Visual Studio（位居第二）的两倍多。JetBrains Rider 被许多人认为是现有最好的 IDE，它不仅与 Visual Studio 一样为.NET 提供支持，而且在此基础上进行了相当大的改进，带来了可能是所有工具中最好的.NET 工具体验。



工具本身是基于底层的平台功能（如健壮的一键式构建和部署系统）而构建的，提供了轻松的首次运行体验，开发者能够在几分钟内完成从平台安装到应用程序部署的过程。事实上，无需纠缠于复杂的工具链、构建文件或平台版本控制就能从零开始使用.NET 来部署应用程序，这种能力已经为其他平台树立了标准，并提升了开发者对平台的忠诚度。

### 4.2、跨平台

就像 20 世纪 90 年代的 Java 梦一样，.NET 也运行在所有的主流平台上，并拥有定制的集成工具来充分利用每一个平台。虽然它最初被锁定在 Windows 上，但在今天，它已经可以运行在 Windows、macOS、iOS、Android、Linux、大型机甚至微控制器上。它也运行在云端，Azure、AWS 和谷歌云都提供了内置的.NET 应用程序支持。

### 4.3、多语言

公共语言运行时(CLR)是.NET 的一个核心特性，开发者可以使用超过 25 种语言编写应用程序，包括 C#和 F#——世界上最好的两种编程语言，以及 VB.NET——虽然人们喜欢调侃它，但它本身确实非常强大。

### 4.4、强大的社区和开发者

根据具体的计算方式，活跃的开发者数量在 200 万到 250 万之间。而根据我们自己的研究，世界上将近一半的活跃开发者知道或者正在用.NET。自.NET 诞生以来，开发者生态系统每年都在增长。因此，在难以招到开发人员的时候，你可以放心，.NET 是最大的开发人才库之一。并且，它还在快速增长。最新的 TIOBE 编程语言指数显示，在过去的一年中，C#的受欢迎程度增长最快，有望进入前三，取代 C 语言。事实上，如果与 VB.NET 一起，它将轻松位居榜首。

### 4.5、开源

.NET 的所有东西都是开源的，可以免费使用，包括类库、运行时、编译器、编程语言和应用程序框架。你可以在 GitHub 上的[DotNet](https://github.com/dotnet?accessToken=eyJhbGciOiJIUzI1NiIsImtpZCI6ImRlZmF1bHQiLCJ0eXAiOiJKV1QifQ.eyJhdWQiOiJhY2Nlc3NfcmVzb3VyY2UiLCJleHAiOjE2NTc1OTI3NTcsImZpbGVHVUlEIjoiZnl1REI1UGlKN1F4a1pyNyIsImlhdCI6MTY1NzU5MjQ1NywidXNlcklkIjoyNDM2MDc5MH0.tK2xbjYpDDnb4guyXsPdgdxRLmHEt5PZK6nWKJ75RWg" \o "" \t "https://www.infoq.cn/article/_blank)存储库中找到所有的源代码。

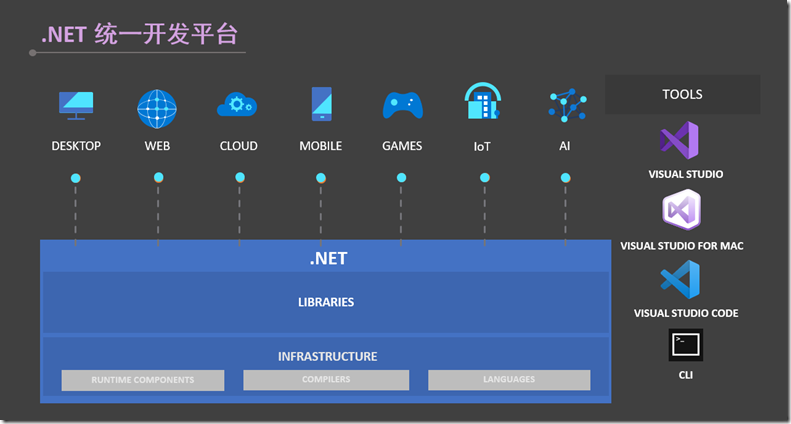
### 4.6、受企业信任

众所周知，.NET 是企业的中流砥竿，虽然微软没有公开发布.NET 的采用情况，但有 95%的财富 500 强企业都在使用 Azure。而且，我私下里也被告知，至少 80%的财富 500 强企业在使用.NET，但实际数字可能更高，因为遥测技术经常会被关闭。.NET 尤其适用于关键任务型应用程序，在这些应用程序中，.NET 的可靠性和久经考验的运行时让人心情舒畅。事实上，美国国防部在广泛使用.NET。

## .NET平台发展趋势

从2000年到2014年微软让.NET Framework作为Windows的一部分，微软也站在开源的对立面。2014年之后微软换了CEO，全面拥抱Linux，.NET Core开源，收购Xamarin，构建起了全新的生态，微软的重心转移到了云。从2014年开始的演变，支持多平台，并提供给各种操作系统，如Linux，Mac，iOS和Android等。它涵盖广泛的应用程序，从桌面、Web 应用程序、云、移动、游戏、IoT 到 AI。

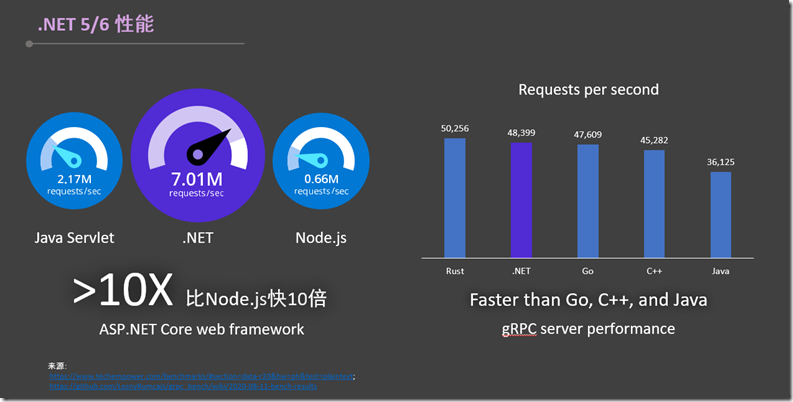
Windows 这个操作系统， 主要是以桌面为中心的用途，从某种意义上说， 是为了提高特定环境下的开发效率而开始的，在.NET第 20 年终于达到了"所有应用的统一平台"的高度，随着.NET 6的发布，One.NET 愿景已经实现。



其实在 NET Framework发布之前，微软从 COM 时代起就一直梦想着多平台。微软可以算是一家坚韧不拔的公司， 经过20年的不懈努力，反复试验和试错，这其中也有不少是错误的，甚至现在可以感受到社会上的很多的软黑曾经是软粉，也许包容不下微软的坚韧不拔。 经过各种反思和20年的时光， 现在最重要的是开发者， 以开放为目标，经过深思熟虑，回归服务好开发者。

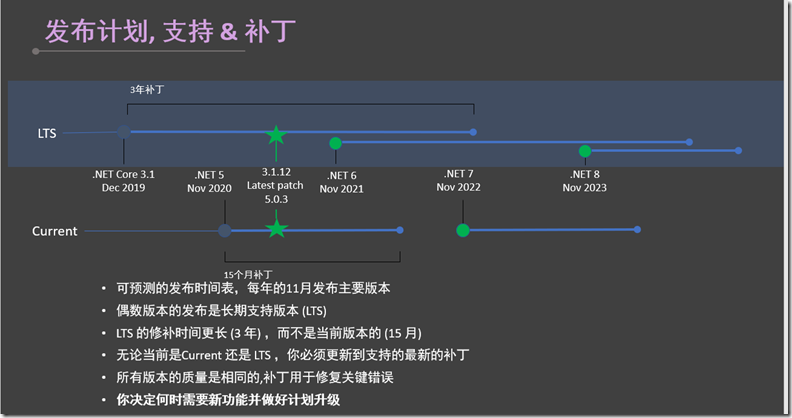
.NET Framework、. NET Core 和 Xamarin 的统一，.NET 6完成了统一 .NET愿景，统一了 SDK、类库和工具链，这些工具链以前是不同的，支持跨平台本机和 Web UI。愿景还包括云原生，Microsoft 计划为所有现代工作负载提供最佳解决方案。.NET 为所有开发人员提供了更多的选择。开发者使用.NET 上的工具和平台，无论他们选择什么，都很容易实现他们想要的。.NET 6 统一了类库和 SDK。 除了发布新的 C# 10 和 F#6 外，它还支持 Apple M1 处理器。 此外，与 NET 5 不同，它还提供长期支持（LTS），提供三年支持。

NET在每次升级时，性能都会成为焦点。比如 NET 6的 ASP.NET Core 网络应用程序比Node.js快10倍， gRpc 的.NET实现是性能最好的，超过了C++的实现。在 .NET 6 中，还专注于加快 Web 应用程序性能，降低资源的占用，换句话说就是对云原生的投资。



现在.NET 也是 OSS，并且更新像任何其他 OSS 一样频繁发布。 然而，在实际操作中，了解版本升级周期更容易使用，因此时间表变得清晰, 作为一家企业公司，如果.NET 的未来计划被披露，那么就可以放心地投资.NET技术。从使用长期支持版本的同一版本的想法出发，很容易转向敏捷流程和思维方式，这些流程和思维将尽可能短的周期更新到新版本，而从安全应用程序开发的角度来看，现在最需要考虑的就是解决漏洞。

开源社区也有很多缺陷是因为工程运作上的问题，微软具有商业公司血统的管理科学更多融合到.NET 开源项目中去，将开源、商业、学术将结合得更加完美。



.NET 程序员的现状是："在被迫开发各种应用的情况下，我们必须选择针对性工具和框架，当前系统正在向云转移，基础设施和服务也在发生变化，开发现场变得越来越复杂。. NET 开发人员 可以使用称为宇宙第一的 Visual Studio 集成开发环境，可在各种目标上部署您喜爱的应用，从而保持高工作效率。

NET 6 还改进了对 "ASP.NET Core Blazor"的支持，以方便 C# 开发人员进行 Web 前端开发。Web 应用程序开发在[服务器](https://cloud.tencent.com/product/cvm?from=10680" \t "https://cloud.tencent.com/developer/article/_blank)端和前端使用不同的语言和框架。 前端 Web UI 基于基本的 JavaScript，通常使用框架（如Angular、React 和 Vue）进行开发。 ASP.NET Core Blazor 的职责是使能够以 C# 等语言开发前端，而无需了解 JavaScript 框架 。

ASP.NET Core Blazor以大多数现代网络浏览器支持的WebAssembly 为技术基础。 因此，在 C# 中编写的代码以本机性能作为 WebAssembly 运行。 根据创建方法，您可以使其脱机运行，也可以将其作为 PWA（主动 Web 应用）安装在计算机上，并像本机应用程序一样运行。 这非常像以前的 Silverlight 或 Flash 那样在插件上运行，不一样的地方是Blazor Wasm使用 Web 标准 WebAssembly。

ASP.NET Core Blazor可以将 DLL 下载到 Web 浏览器，并在 WebAssembly上运行。 由于它完全在客户端运行，因此无需服务器组件即可脱机运行，但下载大小较大。 相反，它提供了一种在 Blazor 服务器上运行程序并使用称为 SignalR 的技术进行渲染的方法。 这需要持久连接，UI 延迟高，但启动速度快，并且具有在较旧的 Web 浏览器和瘦客户端上可用的优点。

随着.NET平台繁杂组件的统一与发展，在未来，.NET在改善.NET各方面效率的同时，应该会致力于完善生态环境。并且随着大数据与机器学习的发展，人工智能领域也将会是.NET的发展方向，微软肯定也不会放弃这块大蛋糕，势必会在这方面投入更多的心力去发展和完善.NET。

## .NET平台的贡献

### 6.1、对于用户

.NET 可使用户轻松进行互联网连接，并轻松完成那些在当今看来十分费时而且费力的事务，它们往往要求用户进行数据重输入并需运行几个小时才能完成。通过将多项安全数据流合并到单一的用户界面（或者甚至是可编程决策引擎），.NET 架构将用户从充斥于当今 Web 的数据竖井的束缚中解脱出来。用户可以自由访问、自由查看、自由使用他们的数据。.NET 将保证完全消除当今计算技术中的所有缺陷。.NET 定能实现确保用户从任何地点、任何设备都可访问其个人数据和应用程序的宏伟蓝图。除此之外，.NET 技术还可实现多个应用程序在逻辑上的松散耦合链接和紧密耦合链接。

用户可以通过手写、语音和图象技术与其个人数据进行交互。这些数据将安全地存放在互联网上，用户通过办公室（或家庭）PC，还可以通过移动电话或寻呼机、PDA、甚至是新发明的寻呼机——移动电话——PDA——PC 联合设备访问这些数据。应用程序可进行灵活的功能调整，以适应用户所用设备的功能状况。应用程序可根据用户预定义的选项集和指令集，完全代替用户自动执行相应的操作。

根据设计，.NET 使得用户无需在如何与计算机进行交互上劳神，从而全身心地投入到使计算机自动执行任务、实现最终目标的工作中。通过使用 XML 行业标准，可将用户数据进行跨站点和应用程序的链接，从而轻松实现当前很难实现的操作。比如：对用户在数家不同银行、信用卡公司以及计费代理商那里的数据进行集中处理；这样，用户便可依据处理后的数据支付帐单，将费用明细报告归档。

### 6.2、对于开发人员

.NET 的策略是将互联网本身作为构建新一代操作系统的基础，对互联网和操作系统的设计思想进行合理延伸。这样，开发人员必将创建出摆脱设备硬件束缚的应用程序，以便轻松实现互联网连接。.NET 无疑是当今计算机技术通向计算时代的一个非常重要的里程碑。

.NET 对开发人员来说十分重要，因为它不但会改变开发人员的开发应用程序的方式，而且使得开发人员能创建出全新的各种应用程序。新型开发范例的核心是 Web服务这个概念的引入。Web 服务是一种通过简单对象访问协议(SOAP)，在互联网上展露其功能性的、极为公开的服务。SOAP 是一种基于可扩展标记语言(XML)制定的协议。

在过去，开发人员通过集成本地系统服务来构建应用程序。在这种模型下，开发人员可以访问丰富的开发资源并能严格控制应用程序的行为。 如今，开发人员已在很大程度上挣脱了这种模型的束缚，致力于构建具有复杂结构的n层化系统，这种系统能将网络上众多的应用程序一并进行集成，大大提升了应用程序的价值。这样，开发人员便可把精力集中在充分挖掘软件独特的商业价值，而不是构建基本结构上。可喜的局面将应运而生：软件投放市场的时间大大缩短、开发人员的编程效率明显提高，最终把质量上乘的软件呈现给用户。

### 6.3、对于行业标准

#### 标准化与著作权

2000年8月，[微软](https://zh.wikipedia.org/wiki/%E5%BE%AE%E8%BB%9F" \o "微软)、[惠普](https://zh.wikipedia.org/wiki/%E6%83%A0%E6%99%AE%E5%85%AC%E5%8F%B8" \o "惠普公司)和[英特尔](https://zh.wikipedia.org/wiki/%E8%8B%B1%E4%BB%A3%E7%88%BE" \o "英代尔)共同制定[CLI](https://zh.wikipedia.org/wiki/%E9%80%9A%E7%94%A8%E4%B8%AD%E9%97%B4%E8%AF%AD%E8%A8%80" \o "通用中间语言)和[C#](https://zh.wikipedia.org/wiki/C%EF%BC%83" \o "C＃)编程语言的标准规格。2001年12月，以上两者通过并成为[ECMA](https://zh.wikipedia.org/wiki/Ecma_International" \o "Ecma International)标准规格。虽然这会有一定的风险，但它可以鼓励标准联合的实现，以提供一座桥梁来把非Windows的软件转变为微软.NET。将此推广为标准的一个团体是[国际.NET联合会](https://zh.wikipedia.org/w/index.php?title=INETA&action=edit&redlink=1" \o "INETA（页面不存在）)（INETA）。

尽管微软及其他合作伙伴持有CLI及C#的专利，ECMA和ISO要求所有专利都应该遵守“[合理与非歧视准则](https://zh.wikipedia.org/w/index.php?title=%E5%90%88%E7%90%86%E8%88%87%E9%9D%9E%E6%AD%A7%E8%A6%96%E6%BA%96%E5%89%87&action=edit&redlink=1" \o "合理与非歧视准则（页面不存在）)”。除此之外，这些公司亦同意让用户免费使用这两项专利。不过，这并不包括.NET框架内并不为ECMA/ISO标准涵盖的范围，这包括了：[Windows Forms](https://zh.wikipedia.org/wiki/Windows_Forms" \o "Windows Forms)、[ADO.NET](https://zh.wikipedia.org/wiki/ADO.NET" \o "ADO.NET)及[ASP.NET](https://zh.wikipedia.org/wiki/ASP.NET" \o "ASP.NET)。微软对上述部件所持有的专利，使开发者未必能够为非微软平台开发出一套完整的作品。

2014 年11月13日，微软新任首席执行官[萨帝亚·纳德拉](https://zh.wikipedia.org/wiki/%E8%96%A9%E5%B8%9D%E4%BA%9E%C2%B7%E7%B4%8D%E5%BE%B7%E6%8B%89" \o "萨帝亚·纳德拉)决心下，该公司发布的.NET源代码包含了服务器部分的核心，如.NET Framework Libraries、.NET Core Framework Libraries和RyuJit VM，正式对外[开放源代码](https://zh.wikipedia.org/wiki/%E9%96%8B%E6%94%BE%E5%8E%9F%E5%A7%8B%E7%A2%BC" \o "开放源代码)。

#### .NET Standard 介绍：

.NET Standard 是针对多个 .NET 实现推出的一套正式的 .NET API 规范。 推出 .NET Standard 的背后动机是要提高 .NET 生态系统中的一致性。 .NET 5 及更高版本采用不同的方法来建立一致性，这种方法在大多数情况下都不需要 .NET Standard。 但如果要在 .NET Framework 和其他任何 .NET 实现（例如 .NET Core）之间共享代码，则库必须面向 .NET Standard 1.0。 [不会发布新版本的 .NET Standard](https://devblogs.microsoft.com/dotnet/the-future-of-net-standard/)，但 .NET 5、.NET 6 以及所有将来的版本将继续支持 .NET Standard 1.1 及更早版本。

.NET Standard 版本控制规则：累加性：.NET Standard 版本在逻辑上形成同心圆。也就是说，较高的版本包含较低版本的所有 API。 版本之间没有重大更改；不可变：一旦发布，.NET Standard 版本就会冻结起来。

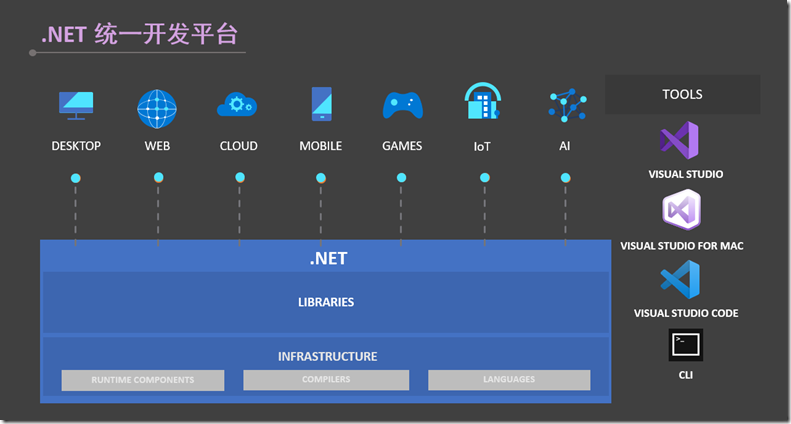
.NET Standard 的问题：

* **添加新 API 的速度缓慢：**.NET Standard 是作为所有 .NET 实现都必须支持的 API 集创建的，因此会对添加新 API 的建议进行审核。 目标是仅标准化可在所有当前和未来的 .NET 平台中实现的 API。 因此，如果某个功能错过了特定版本，则你可能需要等待几年，该功能才会被添加到 Standard 版本中。 然后，你需要等待更长的时间，新版本的 .NET Standard 才能受到广泛支持。
* **复杂的版本控制：**API 规范与其实现的分离导致 API 规范版本与实现版本之间出现复杂的映射。 这种复杂性在本文前面显示的表以及其解释方式说明中显而易见。
* **运行时出现不受平台支持的异常:** .NET Standard 公开了特定于平台的 API。 代码在编译时可能不会出错，并且看起来可以移植到任何平台（即该代码不可移植也是如此）。 当它在不具有给定 API 实现的平台上运行时，会出现运行时错误。

但是这些问题在.NET 4.0出世后得到了改善。

## .NET平台现在及未来应用

从2014年开始的演变，支持多平台，并提供给各种操作系统，如Linux，Mac，iOS和Android等。它涵盖广泛的应用程序，从桌面、Web 应用程序、云、移动、游戏、IoT 到 AI。在.NET第 20 年终于达到了"所有应用的统一平台"的高度，随着.NET 6的发布，One.NET 愿景已经实现。



如今的NET 支持 Web、桌面客户端、移动应用、微服务、云服务、机器学习、游戏、物联网(IoT)等开发，几乎涵盖了所有应用场景。但是其中部分生态目前相对还不是非常成熟，比如机器学习和大数据方面。而在未来几年的计算机行业中，毫无疑问机器学习是最为炙手可热的领域之一。

微软在正式推出.NET 7之后，现在更新其跨平台针对.NET应用程序的机器学习框架ML.NET ，现在发布最新版本2.0，这个版本更新重点包括文本分类和语句相似度API，并且改进自动化机器学习功能。ML.NET 2.0可以更好地支持文本分类场景，其提供的文本分类API可供用户训练模型，以分类文本资料。开发者通过模型构建器（Model Builder），就能使用微软的预训练模型，并以文本分类API及资料来微调模型。

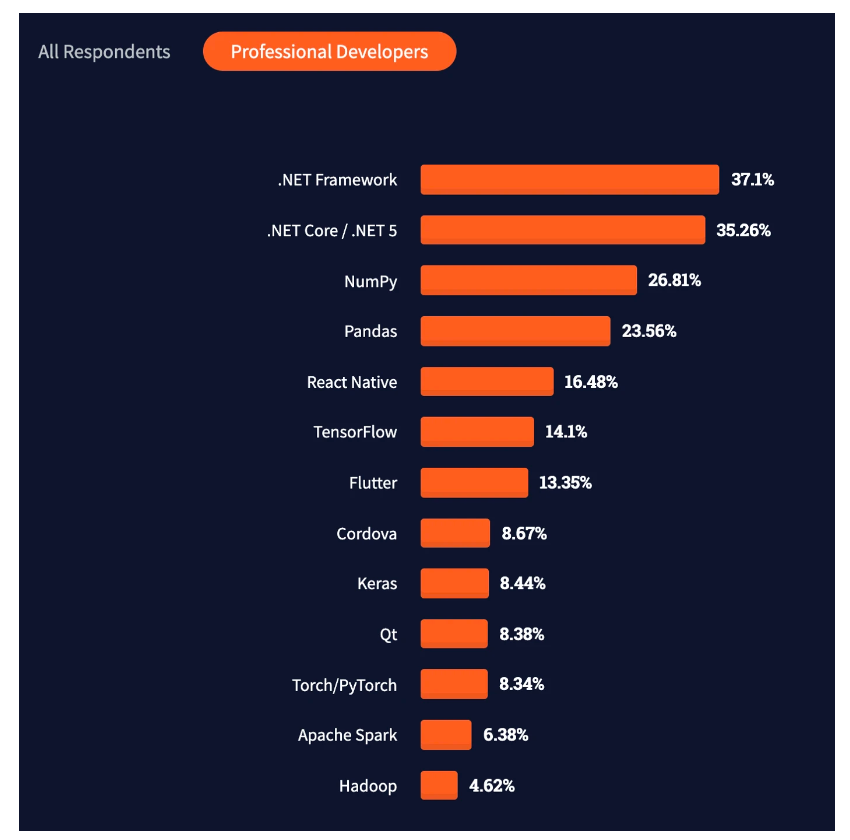
ML.NET 2.0还加入了语句相似度API，该API与文本分类API底层使用相同TorchSharp NAS-BERT模型，差异在于语句相似度API，提供两个语句的相似度数值，而非预测类别。另外，微软也改进自动化机器学习（AutoML）功能，AutoML能够自动化机器学习应用资料的过程，支持开发者在模型构建器和ML.NET CLI的开发体验。在ML.NET 2.0中，微软将AutoML实例加入dotnet/machinelearning存储库中，并且添加更多功能，使开发者能够更简单地开发机器学习应用，并且找出最佳参数。

未来的ML.NET，将会继续以机器学习模型为核心，着重发展分类/类别划分、回归/预测连续值、异常检测、建议、时序/顺序数据、图像分类等功能，力争在未来作为机器学习领域解决复杂工程问题的有力工具。

随着.NET平台繁杂组件的统一与发展，在未来，.NET在改善.NET各方面效率的同时，也势必会在机器学习领域绽放光彩。

## 结语

在一年一度的 StackOverflow 开发者调查中，.NET连续 3 年成为最受喜爱的平台。但它并不完美。其中，大部分直接归因于微软内部优先级的转变，以及领导团队一直在艰难地跟上一线的开发者文化。



MAUI 是微软的旗舰跨平台 UI 库，Xamarin 的继承者，同时也结合了之前的 Windows UI 框架 WPF，但长期以来资金不足，这使得 Flutter 成为当今最好的跨平台 UX 平台。在 Linux 平台上尤其如此，因为 Linux 目前还不支持 MAUI。同样是这些领导团队，他们有时候也错误地处理了与社区的关系，缺失方向感的.NET 基金会和 Hot-Reload 的惨败就足以证明。优先级转变也可能是限制为 VS Code 提供支持的原因，虽然有了基本的.NET 支持，但仍然缺少适当的插件来处理.NET 的多项目解决方案格式，而这对于支持更复杂的项目来说是必要的。这是由于微软一直以来优先将.NET 开发者引向 Visual Studio 的销售，以此来实现变现，而不是为他们提供免费的 VS Code。

话虽如此，很明显，微软内部的优先级正在转移到提供更好的跨平台工具支持上。然而，即使有一些小缺点，.NET 仍然可以轻松地成为世界上最高效、最值得信赖的开发者平台生态系统。没有什么比.NET 更完整、更开放、更高效的平台了。

所有迹象表明，在可预见的未来，.NET 将继续保持这种态势，因为它将继续发展，团队和生态系统将继续创新。因此，将未来押在.NET 上是可行的！

## 参考文献

1. 闫俊伢,安俊秀. .NET技术体系发展趋势的研究[J]. 科学之友,2008(11):127-129.
2. 微软官方文档
3. 百度百科.net词条
4. 维基百科.net词条