从Visual Studio 2017谈起，解析微软技术生态进化之道

原创 2017-03-28 何恺铎 [InfoQ](http://mp.weixin.qq.com/s/gupn6MXsTnjFo3UTDrYC0A##)

作者｜何恺铎编辑｜尾尾曾经被业界取笑「闭关锁国」的微软如今也走向了「改革开放」的道路，Visual Studio 2017的发布，不仅是VS二十周年的大事件，更是微软技术生态焕然一新的直观体验。以前只支持Windows及自家产品的微软，现在iOS、Android、Mac都支持了。写在前面

北京时间2017年3月8日凌晨，Visual Studio 2017如期发布。今年恰逢Visual Studio二十周年，Visual Studio团队可谓诚意满满、不负众望——VS2017不仅拥有全新的模块化设计和更强的性能，功能上也是具有颇多看点和干货，绝非不痛不痒的“年货式”更新。更重要的意义在于，它折射出了微软对于业界技术趋势的最新思考，也展现出了微软拥抱开发者社区的全新姿态。

大家可能还记得鲍尔默在众人前高喊"Developers, developers, developers!"的场景，这说明微软本有着非常重视开发者社区和开发者体验的传统。然而前些年，也许是失之于封闭和捆绑，微软在开发者群体中的影响力走势低迷。将各技术产品线进行排他性整合的思维最终被证明是不受欢迎的——在这个崇尚开源的互联网时代，它反而削弱了微软各大技术平台的竞争力。

好在新任CEO纳德拉执掌公司以来，给微软带来了巨大的思维方式的转变，开放与协作迅速成为了主旋律。 崭新的公司文化帮助微软的技术生态跳出了过去的窠臼和局限，并逐渐进入了一种生机盎然的绝佳状态。本文试图通过梳理 VS 2017 的诸多特性，从编程语言及其工具体系、移动和云、容器化和DevOps等不同视角来探讨Visual Studio乃至整个微软技术栈的基因变化，解析其进化之道。

编程语言及其工具体系

Visual Studio 2017一如既往地为多种编程语言提供了原生支持，如C#、TypeScript/JavaScript、C++、F#、Visual Basic等，也集成了这些语言的最新版本。

C#语言稳步发展，开始支持模式匹配

众所周知，VS的当家语言是C#。C#现已是一门多范式的主流编程语言，历久弥新，LINQ、dynamic、async/await等标志性特性使其在世界范围内拥有众多拥趸。新版Visual Studio按照惯例带来了全新的C#语言版本C#7，新增了Out变量（Out variables）、元组值类型(Tuples)、引用返回(Ref locals and returns)、本地函数(local functions)等特性。C#7的演化思路很清晰，是在C#6的基础上带来稳健的增量更新，使代码更为简练，可读性和可维护性更强，进一步巩固C#主流编程语言的地位。

最值得一提的新特性，恐怕要数**模式匹配**（pattern matching）了。这一标志性的函数式编程范式引入C#后，巧妙地融入了原有的is关键词和switch语句，可在诸多场合大幅简化繁琐的变量声明、if语句和类型转换，凸显业务逻辑本身。

与此同时，我们还能够看到C#语言团队的谨慎和克制——因为模式匹配原本计划做得更加深入，如加入match关键词和类型分解（destructuring）等高级特性，但C#团队最终选择了暂时避免激进变化，采用了更温和更稳健的策略。这样的一种克制，对于C#这样具有较重历史负担和大量遗留代码的语言来说，是一种对庞大的用户群体负责任的态度，也降低了众多C#开发者学习和升级的门槛。

Roslyn编译器为C#发展推波助澜

C#众多新特性的背后，离不开Roslyn编译平台的坚定支撑。历经数年孵化的Roslyn使用C#实现了一个现代的C#编译器，现已能完全替代原有的C++实现，真正走向了成熟。任何语言的生命力源于一个开放的、架构良好的、可维护的编译器code base，开源的Roslyn做到了这一点。像前面提到的模式匹配的类型分解，其实在Roslyn平台上实现并不困难，甚至已经有了初步的代码实现。因此，全新出发的Roslyn平台，是C#后续良好快速发展的保证，我们对C#的未来应当充满信心。

Roslyn带来的另一个好处是强大的编译器API和元数据暴露能力，也在VS2017中得以体现，提高了生产效率。在使用VS2017时不难发现，最常用的查找所有引用（“Find All References”）的结果得到了良好的格式化和语法高亮，还支持更多的展现选项；“Ctrl + .” 的智能提示组合键也变得更加强大，覆盖了更多场景，还能帮助开发者迁移代码到C#7——这些都离不开Roslyn的加持。之前类似的功能可能需要通过第三方商业扩展（如JetBrains的Resharper）才能拥有，而现在相当部分已经在Visual Studio中直接内置了。

颇受好评的TypeScript持续快速迭代，同时支撑了新版Visual Studio中的JavaScript工具体系

TypeScript是微软近年来的另一个重大投入，由C#之父 Anders Hejlsberg亲自掌舵，可以理解为JavaScript语言的一个超集。TypeScript设计和发展的思路紧紧围绕两点：**一是引入可选的强类型机制**，以提供强大的编译期检查和智能提示；**二是全面拥抱乃至提前实现ECMAScript标准**，使得众多开发者可以第一时间尝鲜，不必等待浏览器的内置支持，而对标准的遵循也使得开发者的学习成本得以摊薄。

这两个正确的设计决定让TypeScript具备了鲜明的强类型生产力特征，同时又能与JavaScript和Node.js的已有社区成果深度兼容。因此，TypeScript一经推出就广受欢迎，在近年不断获得关注和采用。如Angular框架团队于2015年宣布Angular2放弃自有的AtScript转而使用TypeScript，堪称微软与谷歌开源技术合作的破冰之举。

毫无疑问，VS2017自然内置了最新的TypeScript 2.2版本。除去在Visual Studio中开发和编辑TypeScript代码的丰富体验外，基于TypeScript的核心能力和强类型生态，Visual Studio 2017还全新打造了JavaScript Language Service（原开发代号"Salsa"）来支撑在新版VS中的JavaScript tooling体系。

换句话说，微软基于TypeScript的能力打造了为JavaScript开发服务的语言基础设施，类似Roslyn对于C#的基石作用，使得VS中JavaScript相关的智能提示、代码导航、代码重构等体验取得了大幅改进。之所以相关能力可以跃升，基于TypeScript的静态代码分析引擎功不可没，它替换了老旧的主要依赖动态执行获取信息的Intellisense引擎 。另外，JavaScript编辑时上下文中的静态类型信息既可以来自TypeScript生态的大量强类型定义（主要由DefinitelyTyped 项目提供），也支持解析社区常见的JSDoc，这是又一个亮点。

TypeScript近期版本还不断地加强了对JSX语法的原生语言级别支持（JSX自TypeScript 1.6版本开始引入）。从面向React.js进行JSX特性的设计，一直到最近为React Native进行适配和优化——倾听社区声音的TypeScript非常接地气 ，体现了对流行框架的支持和融合。

微软技术栈中的其他语言，也在Visual Studio 2017中得到了更新，如C++、F#和Visual Basic。其中，C++进一步实现了C++14和C++17标准中的特性，并开始支持跨平台开发；F#发布4.1版本，提出要成为具备最佳效率工具支持的现代函数式编程语言；而Visual Basic则将一定程度停止对C#的简单克隆，着重发展自己的特色。因篇幅所限，此处就不再展开。

以Visual Studio 2017领军的VS家族已经完成布局，形成了一个开发生态的矩阵

伴随着新生的Visual Studio 2017一起陆续走到前台的同门兄弟，还包括开源、跨平台、多语言支持的轻量级开发环境Visual Studio Code以及由Xamarin Studio发展而来的Visual Studio for Mac（仍处Preview状态），它们都已是Visual Studio大家庭里的重要成员。

前面文中提到的C#、TypeScript等语言的新版本和新特性，得益于相应语言基础设施的开放性和模块化，也都可以在VSCode和VS for Mac中找到并使用。例如Visual Studio Code中完善的C#支持，是通过一个巧妙的被称为Omnisharp Server的后台Http服务进程实现了对代码结构的实时解析和智能提示，而在Omnisharp背后进行支撑的正是Roslyn。

移动和云

“移动为先，云为先”是微软的全新战略，因此Visual Studio 2017对这两方向的开发支持都进行了巨大的投入，达到了新的高度。

移动开发方面微软采用了迂回策略，通过C#/Xamarin和JavaScript/Cordova吸引两大阵营开发者，进军Android/iOS应用开发

移动方面，**去年微软成功地完成了对Xamarin的收购，终于将Xamarin赖以成名的利用C#编写Android、iOS、Mac等平台原生应用的能力收入囊中**，并与微软产品线进行整合——此举受到了.Net社区的广泛欢迎。今天我们看到的Visual Studio 2017已经体现了整合的成果：用户们不必再额外付费购买昂贵的Xamarin商业套件，新版Visual Studio中已经直接纳入了使用C#语言和类库生态编写Android/iOS应用的能力，大幅降低了使用Xamarin技术的门槛。

并入微软官方体系之后，Xamarin的生态也许将逐渐迎来真正意义上的繁荣时期。不过对于中国的广大开发者来说，国内目前Xamarin的声量还比较小，对国内移动互联网生态（如认证、分享、支付等）尚无太多的官方支持和兼容考虑，这可能会成为国内公司使用Xamarin进行移动开发的主要障碍。如果微软和社区在这方面进行积极的合作和推动，解决这样的“水土不服”问题，相信国内的Xamarin成功案例也将开始涌现。

除Xamarin外，微软在移动开发还有另一套基于Apache Cordova的解决方案，以吸引JavaScript和Web技术背景的移动开发者。Visual Studio团队其实多年前就开始构建Visual Studio Tools for Apache Cordova（简称TACO），试图为开发者提供一站式的Cordova开发体验，帮助解决繁杂的安装、依赖梳理、环境配置、调试支持等工具链问题——这正是Visual Studio所擅长的。Visual Studio 2017带来了新版的TACO，在相关体验上再次上了一个新的台阶，还支持最新版本的Ionic2框架，有兴趣的朋友不妨一试。

跨平台.Net Core走向成熟，成为适合云端的应用构建选择

**云计算方面，日前微软Azure在公有云领域稳居世界第二，且势头不断攀升，已是微软整体业绩的重要增长引擎**。毫无疑问，Visual Studio 2017不遗余力地为开发云端应用程序准备了最新最强的支持。

为云助力的第一要务，是帮助开发者构建适应在云端环境运行的应用程序。**.Net Core和Asp.Net Core在此处扮演了重要角色**，因为它们具有良好的跨平台支持，并且能够比较轻量地连同整个runtime一起打包进行自包含部署（Self-Contained Deployment）。之前.Net Core/Asp.Net Core曾在Visual Studio 2015中试水，并随着技术思路的调整在VS2015的多次更新中不断变化。终于，Visual Studio 2017正式将蜕变完全的.Net Core和Asp.Net Core项目作为一等公民。之前因为.Net Core生态处于预览状态而担心稳定性和breaking change的朋友们，现在则可以放心大胆地使用了。

.Net Core和Asp.Net Core的历史和关系说起来颇为有趣：多年前Asp.Net团队在微软内部就以开放和进取著称，他们在设计下一代的时称Asp.Net 5的框架时（早期代号为Project K ），雄心勃勃地希望加入自包含部署、即时编译、跨平台运行等特性，并在当时.Net的工具链和运行时尚不能支持相关需求的情况下，竟自底向上地开发出了一套相对独立的生态和工具链（DNX），事实上领先于.Net Core的孵化。Asp.Net 5的预览版发布后，在设计和性能上都取得了相当好的反响，声势上反倒盖过了标准.Net Core Tooling，一度造成了略显尴尬、令人费解的局面。

令人高兴的是，经过了一段相当长的过渡时期后，两个团队逐渐走向了协调和融合，期间Asp.Net 5重命名为了Asp.Net Core，并最终完全基于.Net Core的CLI进行管理和运行。直到最近Visual Studio 2017发布时，DNX工具链的最后标志project.json描述文件也随着.Net Core Tools 1.0的正式交付而退出历史舞台，进化后的csproj/msbuild体系则再次统一项目模型。这一段轶事其实也折射出了微软自我革新的历程，可谓好事多磨，终成善果。

Azure上的PaaS能力不断推陈出新，Visual Studio帮助大幅降低使用门槛

为云助力的另一角度，是需要为Azure上琳琅满目的PaaS特性提供支持和无缝集成，这部分一直由Azure SDK进行支撑。在Visual Studio 2017的Azure套件中，自然包含各种基于PaaS的云端项目模板，为用户采用Azure的特性提供了一站式解决方案，典型的如WebJob和Service Fabric。而对于需要消费PaaS和各种标准API的应用程序，Visual Studio的Connected Service也能够帮助开发者迅速地连接和配置这些平台或服务。此外，Azure SDK所包含的Azure模拟器可以帮助开发者在本机虚拟云端环境，极大地方便了云端应用程序在本地的开发和调试。

针对大数据处理和计算的PaaS能力是当前公有云厂商争夺的重点。Azure近来在此领域内动作频频，一方面不断深化与HortonWorks的合作，高频率地强化和更新HDInsight（基于HDP）；另一方面PaaS化程度更高的Azure Data Lake也已结束了预览进入GA状态。Azure Data Lake上的数据分析颇具特色，可使用微软自研的U-SQL语言结合SQL和C#的特点进行混合编程，给大规模数据处理和查询带来了让人耳目一新的方式，受到.Net生态开发者的欢迎。在Visual Studio 2017中，开发者可以很方便地连接和管理云端的大数据资产，并创建相应生态的应用程序，如U-SQL、Storm、Hive、Pig等。

总结移动和云方面，微软可谓兼收并蓄，与时俱进。同时，通过充分利用Visual Studio这样的生产力工具，微软成功地对各种平台和不同层面运行环境下的开发单元进行了良好的抽象，给予了开发者多样化的选择。

容器化和DevOps

微软同Docker进行深度合作，融入容器生态圈

容器化和DevOps已成行业热词，微软自然不会错过相关生态的布局，之前就成功而及时地与Docker达成了深度合作。在这个领域，微软做了三件看起来非常具有前瞻性的事情，**一是与Docker合作催生Windows容器**，包括Windows Server Core和Windows Nano Server两种形式，以方便传统Windows应用程序向Docker容器迁移；**二是着力优化Windows作为Docker宿主的能力和体验**，使得Linux容器能够通过Docker for Windows稳定轻快地运行于开发人员Windows上，Docker的原生支持也添加进了Windows Server 2016中；**三是不失时机地推出了Azure容器服务**（Azure Container Service），使得基于容器的大规模应用程序能够顺利地在微软云中部署生根。

受益于上述策略的不断落地，在Visual Studio 2017中进行流畅的容器化应用开发可谓水到渠成。我们可以轻松地为相关Project添加Dockerfile打包应用进容器，并在Solution层面使用Docker Compose进行容器编排。笔者曾尝试在较短时间内建立起若干简单的Asp.Net Core微服务，并互相组合形成了基于微服务架构的应用程序，然后通过Visual Studio在本机的多个Linux容器上一键部署运行。更方便的是，Visual Studio 2017还帮助生成了.Net Core调试相关的Docker配置，使得调试模式下运行在Linux容器内的代码轻松地命中了VS内的预设断点！这样的单机多容器“远程”调试确乎是一种奇妙的体验。

Azure容器服务是微软云的容器解决方案，支持多种编排引擎

前面提到的Azure容器服务，值得再略着一些笔墨。有不少人认为，容器生态的不断发展对于传统PaaS形成了强而有力的挑战，因为细粒度的容器一定程度上可以替代PaaS，同时提供更大的灵活度以及可移植性——很多情况下的确如此。在这一问题上，Azure选择了两边下注，在继续不断发展PaaS的同时（比如积极投入Azure Functions这样的Serverless计算框架），也拥抱容器生态，推出并不断改进Azure Container Service 。保留不同粒度的抽象，把选择权留给客户，的确是最佳方案。目前Azure容器服务已支持Docker Swarm、Mesosphere DC/OS和Kubernetes三大编排引擎，同时底层利用Azure VM Scale Set特性优美地实现了弹性扩容。

VSTS/TFS提供了现代的持续集成能力

真正的DevOps离不开持续集成pipeline的搭建和管理。在大家的印象中，Visual Studio一直有比较好的手动发布支持，可以方便地一键部署应用程序至本地或云端。Visual Studio 2017已更进一步，通过与云端Visual Studio Team Service(VSTS)或本地Team Foundation Server(TFS)的深度配合，可彻底将CI/CD的流程串联并打通，实现自动编译、测试、发布和部署。

最新版本的VSTS/TFS不但让完整的DevOps流程开箱即用，并且从设计理念上充分考虑了开放性，可以在CI/CD pipeline的不同环节使用不同的技术：如使用GitHub的仓库进行源代码获取和编译触发，又如支持以Docker为构建和发布的基本单位，再如使用Octopus Deploy进行部署等。此间，Azure也扮演重要角色，既可以提供动态伸缩的Build Agent，又能够作为应用最终部署的目的地。

DevOps流程中容易忽视的一个痛点在于关系型数据库schema的版本控制和升级，实践中往往需要依赖手动编写的迁移脚本进行人工操作——这与DevOps的精神是相违背的。为了尝试解决这一问题，微软与第三方工具提供商Redgate进行了合作，首次将价值不菲的Redgate Data Tools内置于Visual Studio 2017企业版中。Redgate相关工具的妙处在于，不仅能够将数据库schema进行完善的源代码管理和编译检查，更能记录不同版本的演化细节，并在VSTS的配合下自动地增量更新到CI pipeline的各个数据库中，包括在审批后到达最终的生产环境。

展望与期待

不知不觉，Visual Studio已经走过20个年头。 Visual Studio 2017是一个重要的里程碑，同时也是一个新的开始。**我们有理由相信VS2017还将不断升级，把握新的机遇**。事实上，微软已经开始为VS2017准备后续的更新，并紧锣密鼓地发布了Visual Studio Preview，使得开发者们能够在正式的Update发布之前就可以率先尝试即将到来的新特性。

通过Visual Studio Preview我们可以知道，在未来的VS2017 Update1中，全新的Python开发环境将粉墨登场，还会带来支撑Windows 10 Creator Update新特性的UWP套件更新。之后，还会有针对数据科学和分析类应用程序（Data science and analytical applications）开发的专属支持。相信这些将来的更新都会广受欢迎。

**针对中国的开发者群体和市场环境，Visual Studio团队想必也会越来越重视**。近期微软与华为的合作就是一个很棒的成功案例：华为通过使用VS2017进行面向Linux的C++开发，大幅提升了生产效率。 在此，我们也期待Visual Studio未来能够做得更多更好。其实除了解决部分组件水土不服的问题外，Visual Studio还大可以尝试与国内的公司、平台和生态进行更深入的合作。让我们大胆试想，如果VS能够支持微信小程序的开发，或是在VS中能够方便地对接使用阿里云的基础设施，是不是听上去就更有诱惑力了呢？

写在最后

**"Any Developer, Any App, Any Platform"**，Visual Studio 2017在这样的新口号下酝酿和诞生。**开放而非封闭，合作而非拒绝，兼容而非分裂，这是属于微软的改变，这是属于微软的进化之道**。众多微软技术生态爱好者们期待已久的春天，似乎真的要到来了。也许有一天，我们不再需要强调“微软技术栈”，因为它本就是开放技术世界的组成部分，互相融合，难分彼此。

注：本文仅代表作者个人观点。作者个人与文中所提及商业公司均无利益关系。