时刻关注企业软件开发领域的变化与创新

架构师

7月 ARCHITECT

特别专题

Visual Studio 2010之美

我眼中的Visual Studio 2010架构工具

基于VS2010进行敏捷/Scrum模式开发

应用VS2010辅助敏捷测试

基于Azure的云计算应用设计

REST能够解决系统集成难题吗？

FLEX技术在企业级开发中的应用

RAILS中精彩的图表

解读NOSQL技术代表之作DYNAMO

又拍网架构中的分库设计

InfoQueue
Enterprise Software Development Community

每月8号出版

篇首语 | Editor's letter

你我的世界杯

盼望着，盼望着，四年一度的世界杯足球比赛又一次把盛夏的温度推上了顶点。

朋友们时刻关注着比赛的进展，对各个球队品头论足，对比赛过程津津乐道，真是不亦乐乎，犹如身临其境。

细细想来，短暂的世界杯似乎就是生活的一个缩影，值得我们在软件开发领域借鉴。

比如，球队内讧导致一个个强队过早地出局，而弱队则通过集体力量爆出冷门，向人们昭示着团结和协作对于一个团队的 关键影响。对于软件开发也是同样的道理，无论是多么优秀的人才、多么出色的架构、多么阔气的资源，如果项目管理层不引导团队以协作的方法进行软件开发，那 么最终会陷入混乱甚至失败。

比如，分数落后的球队一次次追平比分甚至反败为胜，告诉我们毅力和信念是如何驱动人们朝着预想 的目标前进。君不见多少家如日中天的 IT 公司当年创业时的艰难，如果不是"再坚持一下"的信念，可能它们早就夭折在汹涌的互联网大潮中。

比如，巨星闪耀让我们感叹中流砥柱的作用，而新人辈出则印证着时势造英雄。软件开发领域不乏牛人，我们总是期待他们能够指点江山，不过也应该为新人提供一些崭露头角的机会，只有开发社区的整体质量提高了，中国的 IT 品质才能进一步提升。

比如，裁判的误判改变了比赛结果的方向， 表明机遇和外部因素对事物发展的重要影响。所谓万事俱备只欠东风，在软件开发领域，对外部因素的风险评估和控制依然是项目管理重要的一环，希望少一些因为外部因素导致的产品发布延期、延期、延期！

比如，华丽足球败给了实用足球，让我们在失望的同时，也感叹功利主义的价值和现实的残酷。是的，理想情况是鱼和熊掌兼得。不过现实世界中，我们时不时地看到崇尚技术、备受开发者爱戴的公司被其他善于商业运营的企业收购，令人扼腕痛惜，如何在技术与盈利之间实现最优化发展？值得我们深思。

比如.....

或许，我们不是世界杯的看客，而是一名参与者，不论是教练、还是球员，不论是鼎鼎大名、而是默默无闻，大家都在踢着属于 IT 界的世界杯.....

本期主编：崔康

目　　录

[篇首语]

你我的世界杯 I

[人物专访]

DON SYME 谈 F# 2.0 1

[热点新闻]

REST 能够解决系统集成难题吗？ 15

FLEX SDK 与 FLASH BUILDER 4：深度对话 ANDREW SHORTEN 18

2010 年 SOA 状态调查：SOA 在当今业界的地位得到巩固 22

GOOGLE、MOZILLA 和 OPERA 质疑微软的 HTML 5 兼容性测试结果架构？ 25

向作者支付的新方式： FLATTR 小额支付服务 29

SOA 的未解之谜 32

[特别专题]

我眼中的 VISUAL STUDIO 2010 架构工具 36

基于 VISUAL STUDIO 2010 进行敏捷/SCRUM 模式开发 46

应用 VISUAL STUDIO 2010 辅助敏捷测试 65

基于 WINDOWS AZURE 的云计算应用设计 81

[推荐文章]

FLEX 技术在企业级开发中的应用 94

RAILS 中精彩的图表 104

REST 和 SOAP：谁更好，或者都好？ 114

又拍网架构中的分库设计 117

增强的恶意软件检测 128

解读 NOSQL 技术代表之作 DYNAMO 137

[新品推荐]

INTERNET EXPLORER 9 PREVIEW 3 提供了更好的 HTML 5 支持 144

AZUL 发布 ZING：面向 X86、基于软件的 JVM 虚拟化/弹性运行时 144

ECLIPSE HELIOS 发布了 144

JAVA LAMBDA 语法的首个版本引发众多争论 145

HYPERSQL 2.0：期待 5 年，新版本终于发布 145

代码质量管理工具 SONAR 提供了设计与架构度量 145

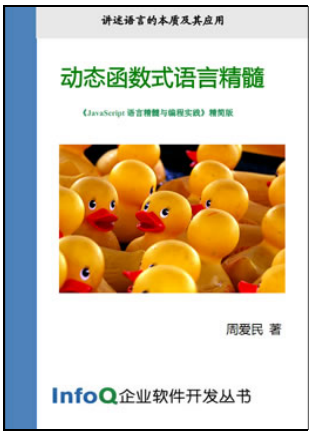
GOOGLE 向开发者提供云存储 146

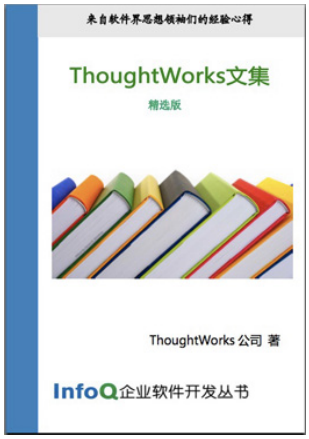
[封面植物]

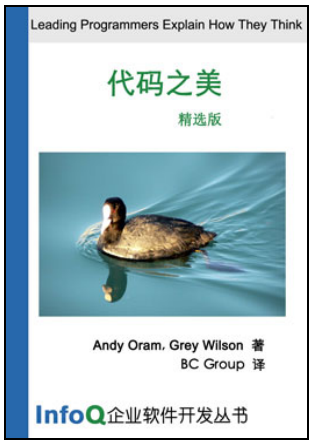
四药门花 148

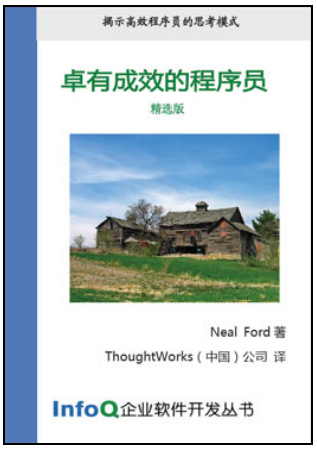
InfoQ企业软件开发丛书

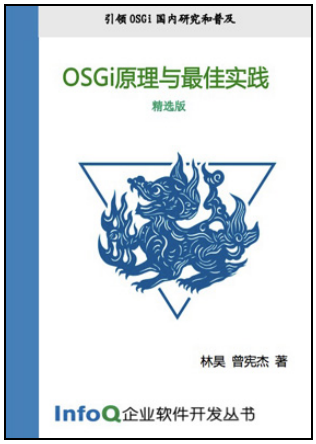
欢迎免费下载











商务合作：sales@cn.infoq.com　　读者反馈/内容提供：editors@cn.infoq.com

人物专访 | Interview

Don Syme 谈 F# 2.0

——VS2010 中的新晋头等公民

Don Syme 谈论了 F# 2.0，它的应用领域，和 Visual Studio 2010 的集成，以及 F#的开源 Power Pack 函数库。Don 也讨论了 F#的起源，和 OCaml 的关系，以及它的特点。他解释了 OOP 和 FP 如何混合到一种语言中，并提到设计此语言的一些决策和必须做出的妥协。

Don Syme 是 F#的主要设计者。他也参与了 C#泛型和.NET CLR 的设计。他于 1998 年加入微软研究院，1999 年从剑桥大学计算机实验室获得博士学位。

InfoQ：我是 Sadek Drobi，我在 QCon 现场，同 Don Syme 再次会面。那么 Don，将近有 2 年或 2 年多了吧。自从上次采访，你在 F#和其他方面有什么进展？

Don：我觉得我们是在埋头于 Visual Studio 2010 方面的工作，也快完工了，目前技术工作已经完成，最后还有一些小细节需要完善，接着就可以发布了。在 Visual Studio 2010 中，你可以用到 F#，当然在 Visual Studio 2008 中也可以，你还可以在 Linux 和 Mac 上通过 Mono 来使用它。我们总是提醒人们，它不仅仅在 Visual Studio 2010 才能用到。

InfoQ：有意思。你们实际上维护了一个 2008 的版本，一个 2010 的版本，那么说起来人们就不必为了使用 F#而一定要运行.NET 4.0。

Don：在设计过程中，我们有一些有趣的地方，例如关于是否依赖.NET 4.0 这个问题，我们认为对于 F#而言，能够完全跨平台是非常强大的一个特点，这不仅仅限于.NET 2.0、3.5、4.0，也包括 Mono 和 Silverlight 以及 Moonlight。基本上，在我们考虑跨平台问题的时候，倾向于认为(当我们使用跨平台这个词的时候，你也立刻就应该想到)是跨 Linux、Mac 和 Windows。不过现在世上的平台也发生了变化，云我们也要考虑进来，甚至也不得不考虑客户端的东西，

比如在 Silverlight 中，我们还要考虑手机平台。

虽然现在的 F#，你还不能看作是一个重要的移动电话编程语言；不过还是有所关联的，或许在某一天我们要在所有平台上都运行它，让这个语言可以做更多可行的开发工作，比我脑海中想到的还更多。当然，如果你可以在 Visual Studio 2008 中使用 F#，在没有 Visual Studio 的 Windows 系统上使用，也可以通过 Visual Studio 2008 Shell 来使用，可以通过 Visual Studio 2010 来使用，你也能在 Linux 系统上通过 Mono 来使用。在 Linux 和 Mac 系统上，我们特别测试了在 Mono 2.4 到 2.6 的运行情况。

InfoQ：我也注意到，你们后来还添加了一些功能到 F# Power Pack 中——我的意思是和 LINQ 的集成以及其他方面。

Don：最近这两年是很有趣的时间。当然，我们做了大量的设计调整和错误修正，并同大量的 F#用户和采用者一起工作，不过 F#的很大一块工作是研究项目，所以它实际上被分成了两个部分。一部分是我们认为的这个语言的一些核心功能，我们需要悉心维护；另外还有大量的工具方面的事情在项目启动的时候就已经开始在做，这些功能显然也很有价值，不过我们不打算花太多精力去维护。当然，我们也希望这样的功能也能最终得以发布。

我们花了很长时间来考虑，如何保证这些功能可被社区所用。最后，我们在 CodePlex 创建了 Power Pack，一个基于微软公共许可协议（这是一个获准的的微软开源协议，即 OSS 协议）的开源函数库。它首先包含了一个兼容函数库，以及一个数学运算库，有 Lex 和 Yacc 函数库，也有一些异步编程辅助函数，随着时间推移还会有很多功能。确实是，这种方式就是我们一直寻找的。

InfoQ：你能和我们说说 LINQ 的功能不？

Don：嗯，当然，这个功能就是查询集成。这个东西我们确实希望能放到核心函数库中。这也是其中一个先在 power pack 中发布，经过一段时间后会放到核心函数库中的功能。这是个非常好的特性，基本上可以让你在 F#中嵌入对数据库的查询来完成 C#那样风格的 LINQ 编程。LINQ 已经被广泛运用，我们很高兴能在 power pack 为大家也提供这样的功能。大家不妨一试。我已经撰写了一篇博文，讲述如何在 F#中使用 LINQ 查询的基本知识。

InfoQ：在 F#的包【译者注：即类似 C#的命名空间】中有 Quotations，那么说你们有函数库来处理引用语句，对吗？

Don：对，就是在核心 F#函数库中。核心 F#函数库就是用来构建额外丰富功能的一切基础。Quotation 就是其中一个。让你在 F#代码中可以处理表达式树。这个功能也揭示了一个问题，

就是 F#的下一步要实现什么，以及一个重要的现象就是 F#的功能特性实际上被做了分支，这也就是我们称之为版本 2 的原因，这个版本也即现在大家拿到手的版本。我们把研究中的 F#称之为 F# 1.0，而现在它已经成为 F# 2.0 了。

Quotations 真的是一个令人非常激动的特性，会带来很多潜在的应用。例如，一个应用之处就是 LINQ 查询支持，可以在 power pack 找到；另外一个应用就是把 F#代码转化到 GPU 指令，在 GPU 中以数据并行的方式运行，这确认让人振奋。还有一个应用领域是，把 F#代码转化为 JavaScript，它被使用这项技术的 Intel 公司里的人称之为一个伟大的开端（实际上，我甚至都不认为它是个开端），我有一个出色的 F# 2.0 JavaScript 框架能让你在客户端编程中使用 F#。

InfoQ：实际上你可以转换得到所有代码？

Don：所有的，而且如果你愿意的话，能用 F#来选择处理 HTML 的任意部分，事实上正如 DSL 所做的那样。你实际上也可以进行异步编程。在客户端，你当然需要进行大量的对服务器的异步请求。让我吃惊的一点就是，对于 Web 编程，你需要知道如此多的语言。

你必须知道 HTML，必须知道 CSS，必须知道 SQL，必须知道类似 C#这样的编程语言，大概还有 1 到 2 种语言标记语言也要知道，例如 ASP.NET 标记语言。那么多东西需要知道，让人望而生畏。而现在，我能使用 F#引用功能，一般而言它能以一种很妙的方式把这些功能进行简化。

我想这就是很多人在寻找的解决当今编程过程中遇到的一些挑战的功能，在进行 Web 编程过程中，我们急切地需要对其简化和强类型化。

InfoQ：当我提到 F#的时候，通常每个人都认为它就是一种主流语言或主程序，只不过语法和 F#的属性更适合表示数据而已。

Don：是的，假如那样的话，我们就能在 HTML 中表示数据，由于 F#的部分代码能被转换为 JavaScript 代码，这样也能表示 JavaScript。就像你正在编写运行在 JavaScript 中的 F#一样，这确实是一个非常让人兴奋的开发框架。

InfoQ：我能把 F#用于保存配置信息的配置文件，或任何不算是真正代码的东西里？

Don：对于保存配置信息，我们确实把 F#用作一种更丰富信息表示方式，这样将变得常见。通常，数据都是从外部的日志文件读进 F#，你不太会考虑用 F#来表示数据。在 F#编程中，你能用 F#表达式作为数据表示方式，而不用引入某种新语言或技术。你也不用去采用一种

新的文件格式，对于一个或两个数据结构或数据表，你能轻易地在 F#中实现它，且可以得到强类型的功能，并立马就能对这些数据进行操纵。

InfoQ：另外，F#有一种 FSX 扩展，它是一个脚本文件。能用它来在 Windows 或其他操作系统中，做一些构建脚本或任务执行脚本吗？

Don：当然。F#可以作为脚本语言的，我们把 fsx 看作是编程脚本。你可以以脚本的方式来进行编程。我们并不鼓励用 F#来开发所有类型的脚本工作。也许，有些人已经编写了一些脚本文件，或已经有他们喜爱的脚本语言，比如在 Windows 上的 PowerShell。或者在 Apache 用的一些脚本，或在 Unix 系统上某些脚本。

不过，也许在进行日志文件分析或编程的过程中，如处理被混淆过的文本文件，处理正则表达式匹配，或处理类似于跨文件搜索来做一些文件内容分析的时候，会遇到一些限制。这样的限制都比较棘手。某些限制，你或许能使用一些外部的工具能进行解决，不过在某些时候，用 FSX 文件将让你十分轻松的解决问题，只需在 Visual Studio 中通过 Control-N 来新建 FSX 文件，或在 emacs 或其他编辑器中也可以编写 FSX 文件。

InfoQ：你能使用 F#的强大异步功能来完成这些工作。

Don：例如，你能利用变换来使他们更适合使用并行功能。接着，你可能做的就是。把数据源进行某种变换，例如对于日志文件，接着开始构建一段分析代码，然后在完整的数据集上实际运行，可能会有几 G 的数据，这样你认为“好，这个程序已经写好了，它是有用的，它是用 F#编写的，将要花费 2 周的时间来处理这些不同的数据集。我们可以不用对这些数据进行分区。”你可能会手动进行分区，或最好是进行并行处理。

InfoQ：每次，在提到诸如多核和并发编程的时候也会提及 F#。那么 F#在这方面的强大之处是什么？为什么每个人都把 F#和异步编程放到一起讨论？

Don：实际上，这就是我今天要在 QCon 上讨论的话题。他们要我不仅仅谈论 F#编程和并行编程，也要更多谈一些关于函数式编程和并行编程的常见话题。在这两方面，存在有趣的重叠，因为我不仅仅在谈论函数式编程，也在让大家听到除了并行编程本身的东西外还有函数式编程的基本原理。

函数式编程，是一种很奇妙的编程范式，用于面向大量数据的工作和引擎，或一般用途的编程，不管你用不用并行编程。不过这有一定的重叠，我尽量用各种方式来描述这些重叠的特征。一个基本事实是，所有的函数式语言都鼓励你在应用程序中最小化和减少状态数，根本来看，就是要在所有代码中都使用不可变数据结构。

F#代码中贯穿了很多不可变模式。F#的一个特别强大的应用领域就是任务并行处理。你可以回顾一下.NET 4.0 任务并行库。F#也存在一个特别的函数库称之为异步库（Async Library）。它很有意思，因为能让你混合 IO 的并行处理，比如在你并行地抓取一组网页的时候，或并行地伺服一组 TCP 的服务器端请求的时候，你可以把这些处理过程和某些 CPU 计算过程混合在一起。

你有大量的请求进来，你实际上是获得了一个独立的小任务，比如响应任务。它不是线程，因为无论何时需要从后端数据库获取数据的时候，它都会以一种轻量级睡眠模式进入睡眠。在你获得进入的请求的时候，我的意思是很多请求，如 50000 个同时进入的请求或类似的通过 TCP 通道进入的东西，都需要对这些请求作出响应，接着去和后端数据库打交道，并异步地返回的这些请求。这种程序在 F#中非常容易编写，因为它实际给了你一种轻量级任务模型，尤其适合处理那种不大的 IO 处理和 CPU 运算。

InfoQ：对于“轻量级”，你的意思是不是没有线程那样昂贵？

Don：是的，你能在单独的进程中创建 50000 个任务。某些情况下，你不需要这样处理。比如，你要同时伺服 10 个请求，这样你完全有资源为这些请求每个创建一个线程。有趣的是，虽然如此，在你设计一个服务器端系统的时候，需求可能会不断改变。你要考虑到，在某些服务器上可能需要处理计算非常昂贵的视频解码工作，而过一会，你把这部分工作移交给其他后端服务的时候，前端服务器实际上突然变得非常类似电话交换机那样。

“电话交换机”忙于收到呼叫，传递呼叫，共享信息，需求变化的非常快。你所认为的伺服 10 个请求的情况，会突然变为要伺服成千上万的请求，因为负载均衡被改变了。这类重量级的服务器端异步编程，在 F#中可被轻易地完成，最近这个话题已经成为我特别感兴趣的一个。以商务的口吻来说，“我们有非常优秀的特性集”。

在 F#中的这种处理轻量级响应代理或任务的机制，显然是一种构建很多不同应用的好机制，不管在客户端还是服务器端，我们会提供相应的功能分支给你来构建所需的应用，在什么时候使用何种机制也依赖于准确的性能权衡。我们确实一直都很高兴，能发布这些特性集，这种不同的编程模式，能够解决非常复杂的任务。

InfoQ：在 F#中异步编程是相当常见的目的。我也看到一些使用异步编程来轻易实现的 Actor 模型例子。

Don：构建轻量级代理有一个基本原理，就是你启动一个 Actor，启动一个消息通道（cues），

所有东西都用消息通道来处理。那样做很好，也是 Erlang 所用的模型，不过我实际上感兴趣的仅仅是更底层的东西，就是整个轻量级异步机制。

Actor 只是你可以构建的一个东西而已，你还可以构建 TCP 请求处理程序，它可不是 Actor。你也不想为每个请求都创建一个消息通道，因为那样过于昂贵。我非常高兴，异步机制似乎在多得惊人的应用中都能起到作用，而 Actor 只是其中一种应用。

InfoQ：F#提供了这些类型的任务，对吗？类似处理请求的任务？

Don：在服务器端，我们正在计划放这样的东西到 Power Pack 中替代辅助类，来帮助你更好更容易地构建 TCP 服务器。在 F#中，你也可以在 C#中使用的标准 TCP 代码，也可把相应的诀窍用到 F#中，也可以看一下最近 F#提供的一些示例。作为一个构架师，我的大部分工作是轻易让你构建能处理 5000 或 10000 并发请求的服务器端系统。

有时，如果你的 Web 2.0 公司获得成功，拥有 20000 人，你就需要另外的服务器了。那样，你可能打算把应用转换为云计算。我现在非常感兴趣，如何把一个正在运行中的伺服所有这些请求的 F#异步应用程序，立刻地迁移到微软的 Azure 或 Amazon 的 EC2 或能找到的各种云计算平台中去运行的转换点。

要研究那个转换点，我们常常通过一些代码来实践，并构建一些重要的示例，来辅助大家把应用程序的生命周期过渡到下一阶段，我真的喜欢做这样的事情，因为你可以通过这种方式学到关于云计算的很多东西。既然如此，我当然利用 Windows Azure 来开展这个工作，我可以说，很高兴我能做到非常平滑地进行转换。

你有一个服务器端应用程序，它使用了特别的 TCP 通道，你只用把它部署到 Azure 中，并为自动负载均衡进行一点调整，那么云计算服务器就能做和你在 F#中完成的同样事情。第一次使用 Windows Azure，只工作了一个晚上，我们就开发出了一个可伸缩的应用程序。我们将能任意伸缩，只要你购买了足够多的云计算资源，你就能伸缩到支持数以亿计的同时请求。

我真的很高兴，这样的转换能正常工作。我认为，使用 web 作为数据源是一种绝妙的方式，作为一个数据分析者，同样还是可以使用 F#作为编程工具来访问这样的数据源。例如，你可以连接到一个金融数据源，通过这种新的编程访问方式来获取金融数据流，来进行相关分析工作，这会是一个很棒的方法。

在服务器端，绝妙之处在于可以作为这些数据流的实现语言，这些数据流的提供器，或许更好地从其他数据流中进行匹配。你获取某些数据流，构建某个分析引擎，并基于 Web 依次

提供。我想这非常有意思，为 F#这样的语言而振奋，因为它把 F#的强大体现到了数据分析的应用当中，也同时能得到很好的封装边界。

我们知道，Web 服务器或 Web 的 REST API 要让外部访问，当然你可以使用任何语言来开发，不过不希望强制你的客户端也必须用同样的语言来开发。对于这样的函数库，当然是件好事，不过对于类似提供数据服务的这样的情况，你显然不得不通过一些更通用的 API 来进行部署，以便让任何其他语言都可以访问。

InfoQ：这是 F#的下一步要做的，对吗？

Don：我想这是 F#的一个精彩应用领域，有时候，我认为 F#是进行 Web 2.0 开发的完美语言，可以处理哪些除了 UI 以外的数据分析工作。我们也看到某些使用 F#开发的机器学习应用程序，F#也用于广告引擎，也用于微软系统中的庞大预测引擎。我想这是一个很奇妙的应用领域。函数式编程世界也有这样一股趋势，更多关注于老式编程方面的东西。你姑且认为是 Fibonacci 运算这一类的例子，以及由此引发的一些东西。

InfoQ：你提供了一些根据实践性编程的 F#例子没有？

Don：是的，这也是我正在讲述的工作的很大一部分。我们努力研究，并展示如何在这类工作中使用 F#异步编程。

InfoQ：人们现在开始学习 F#——他们大部分都是 C#、Java 或其他的程序员，他们看到这个是一个基于 ML 的编程语言。这对于他们是非常陌生的，所以他们开始提出一些问题，比如“我应该在什么时候使用函数，我应该在什么使用使用方法，应该在什么时候使用类、对象，或我应该在什么时候使用 Union Type？”，对此你有何看法？

Don：我们在最近一年中大部分工作都是，确保大家在遇到这些迷惑的时候都能尽可能顺利理解。在某种意义上说，不管你使用静态方法还是模块中的函数，问题都不大。从编程的角度看，两者的区别也非常少。在某个特别的例子当中，静态成员确实允许你进行重载。这可能需要更多的设计，而使用模块和函数，就需要不同名称或更好的推断类型，这样才能相对简单明了。不过在一定的运用程度，你如何使用都无所谓。

对于实现代码这点而言，这样的讨论是无所谓的，实际上，我们确实听到一些有面向对象编程背景的人们在用 F#进行面向对象的编程。你会注意到，由于使用函数来代替之前不得不创建的其他类，或者使用 F#对象表达式来避免仅仅为了实现抽象方法而创建的子类，你会突然发现大量的类就会随之消失。

某些面向对象设计模式逐渐趋同为这种简化版本的更佳的 OO 编程，接着，使用函数式编程的人们也倾向于放弃原有的编程结构。也许，如果他们来自于同样的 ML 背景，甚至都不会使用任何面向对象编程的思想，不会封装可变状态，接着，他们会突然发现在 F#中封装可变状态是很不错的。

那么，通过这样的转变，基本上你可以非常恰当地运用面向对象编程——封装固定类型，某些很好封装的接口类型，并把这些隐藏到实现背后。实际上，我们可以在 F#中看到两个编程流派，并可以在中间找到一个很好的平衡点。

InfoQ：在 F#中的函数库里，有时你会看到一个 Option 类型返回，有时又是异常返回。有什么规律或技术可以遵循吗？

Don：我们很愿意收集来自 F#社区中为 F#设计函数库的人们的诸多问题。我们倾向于在函数库设计（那种鼓励人们非常直率地作出决定的东西）区别对待，我们称之为香草（Vanilla）.NET 函数库设计，那种你绝对会用到任何其他.NET 语言中的函数库。

F#函数库有着 F#风格的设计，那种能运用在应用程序的内部构架中，或者开发的一系列函数库，或构建一个 F#的特定框架。当然，在后者中，你能更自由地使用 F#的特定结构。例如，如果你完全在进行一个香草.NET 函数库设计，那么你应该遵循常规的.NET 函数库设计准则，有很多非常了不起的准则的。比如，有准则会讲到什么时候使用结构体——你只需去阅读相关的.NET 设计准则就行。

在使用异常方面，它也会给你很好的建议，给你很好的准则。在用于 F#的 F#函数库中，如果真的要把这个函数库（如 F#调用函数库）作为一个长期维护的代码，那么 F#语言的特定特性实际上在这这方面的有所实现，或只是给出关于函数库设计的提示。F#在某种意义上影响着函数库的设计，比某些函数式编程拥护者所认为的要少，不过在某些地方确实还是对用于 F#的 F#函数库设计有所影响。让我们来举个例。

如果你编写了一个函数，它本身是异步方法，那么你可以调用它，并异步地给你返回结果。接着，F#代码中所蕴含的模式理所当然会以异步类型或异步 String 返回，可以包含各种内容。你可以运用到用于 F#的 F#代码中的一些惯用法。我们来看另外一个例子——表达式树。在 API 边界上，你会遇到某种差异并集（discriminated union）。

其中有很多充分的理由，为了软件工程把差异并集的表现掩盖在了某些基本的用于创建差异并集元素的 API 背后，而且也不必分析它们。比方说，Java 视图 API 就是一种差异并集，表

示了命题（propositional）对象或底层的某些东西。你大概不需要在外部显示它，从软件工程的角度看这也不是一个好主意。

就算 F#具备这些特性，也不意味着在 API 边界，就因为你打算把自己开发的函数库进行长期维护，就要掩藏它们废除它们。例如，进入你的函数库的实际数据可能来自 XML，也可能来自 JSON 格式。

不一定要 F#程序员来编写那个函数库。我们为这两个例子中提供一系列准则——香草.NET 函数库设计，以及用于 F#的 F#函数库设计。我们将通过 F#社区来推进这些准则，通过与 F#社区和其他 F#用户合作，来了解这些准则是否适合，所以我们期盼大家的反馈。

InfoQ：在 F#的设计过程中，你尝试尽量少用特殊的功能。甚至对于我们能用到的首要功能，文本机制也是如此？

Don：我们之前聊过这个话题的。我想，你指的是经常用到的语言元素，让我们以异步编程或数字文本来说明。你知道，要把某个功能放到语言中，总有一个或两个特殊的原因。你有一个大整数，想编写一个为 3 亿的整数，那么我们就提供了大整数类型，不过 10%到 5%的用例中，人们实际上需要改变这些文本的意思。

例如，有各种不同的大整数函数包，总有一个充分的理由会突然决定转换到不同的函数包上。在语言中，你不能把机制仅仅绑定到一个底层类型上，应该尽力让它表示不同类型，如果我们会以各种方式来使用它那么也会映射到不同类型上。在 F#异步编程中，我们也同样地这样处理。你可以定义自己的计算表达式（Computation Expression）构建器，让其能够在异步编程上实现不同的变化。

InfoQ：计算表达式不是专用于异步编程的。是一个通用的功能？

Don：它是一个通用功能，在 F#中它实际上也是很有意思的方面，人们有时看到 F#计算表达式，他们会说“这是 Haskell 的单元（monads），那么他就是用于单元的某种机制。”实际上，它不仅仅是用于单元的机制。它也是用于类群（monoids）的机制，基本上说来，它类似于 Haskell 的列表表达式或 Python 中所理解的那样。

有趣的是，你能使用这个人们都知道的东西来实现它，比如 F#中的序列表达式，当然也有类似异步序列这样的东西可用。你希望获得返回结果的流，比如说 Twitter 数据。比方说，你想订阅某个特定用户名下的所有消息，那么 Twitter 将给你一些 API 来完成这项工作，这样查看这些数据流的方式自然可以多种多样。

查看它的方法是将要提到的异步序列。基本上，每次你想查看下一条信息，程序就会进入一个轻量级的睡眠状态，直到下一条信息到达，才“醒”过来。这让你，可以启动很多同时的响应器来处理进入的事件字符串。F#计算表达式语法让你能够轻易地定义异步序列，由于这个非常通用的机制让你既可以完成单元处理也可以完成类群处理。这很了不起。

InfoQ：F#是函数式编程语言，也是面向对象编程语言。你如何进行设计，以便面向对象的特性不会影响函数式编程的特性，或反过来说，如何保证函数式编程的优势而不损害面向对象的特点的？

Don：某种意义上，这是由于 F#的历史所带来的。熟悉 F#的人们应该知道它是作为核心 ML 的实现，也即第一个 OCaml 语言的核心而开始的，其本身非常类似于我所使用的第一个 ML 版本，就是 Edinburgh ML。它尽力去保证这种编程范式的基本优势。在他的核心中，总是有那样的远见，我们从来不用去维护它。

不过，也不意味着我不用去做一些折衷，或在某些领域的设计改变，比如我们要额外支持重载。比如，我们也使用 F#的额外底层类型添加了两个 64 位的整数类型，两个 32 位的整数类型，这实际上和 OCaml 语言的设计是有所不同的。但是，也说明了，由于我们添加了面向对象的概念，就要花更多的精力去保证它们能容易的在函数式编程的上下文中使用。

回到几年前，大约 4 年前，当我们做出有关面向对象编程的决定的时候，这个关键的设计决策不仅要涉及命名类型（nominal types）的使用，也要在软件工程和编程过程中接受命名类型的角色。函数式编程世界的传统总是使用同样的构造类型（structural types）。如果你打算进行函数式的面向对象编程，那么总是会用到构造类型和兼容于构造类型的东西，其中涉及很多重要的理论，有数个优秀的实现可以在实践中展示它能很好的运转，包括类型推断。

比如，OCaml 对象系统就是其中一个。当我们刚开始涉及.NET 函数库的时候，我们只是作为对函数式编程具有一些特别见解的语言研究者，而当我们提出相关的函数库设计原则的时候，在早先的几年中已经积累了一些经验了。我们可以把 C#看作是面向函数库的语言。这是一个伟大的编程语言，不过他们把它的第一个版本称之为面向组件的语言，一个为编写.NET 函数库和.NET 组件的语言。这就是为什么在这个语言的第一个版本中就具有属性，具有事件。

我们说“没问题”，接着就由函数库设计来驱动我们的工作，他们也遇到了一些难处理的规格，就是我们希望构建的这个函数库，需要语言来完成。他们没有接纳构造类型或类似的方式。构造类型不是编写优秀函数库的必要元素。它有一定用处，可以有用武之地，不过不是必要

的，同时这真的为我们提供了学习实践——实际上命名类型和名称在函数库设计中更重要，因此，在 F#面向对象世界中，你会看到很多数量超乎你想象的命名类型。

有趣的是，这也有很多优点。让你的类型推断问题简化了，帮助你更多地使用智能感知，比如实际上你可以把在程序中从头到尾的很多类型都声明为一种特别的命名类型。我想在某种意义上而言，这是一个关键的设计决策，放弃了函数式编程理论思想的一些东西，而接受了更多软件工程的现实。我的老板，Luca Cardelli 在面向对象问题上，具有的非常清晰的见解就是关于软件工程的，函数式编程更多地是为了在编程中更少代价地实现代码和功能融合。这真的有助于指导我们做出某些设计决策。

InfoQ：你刚刚提到智能感知。我猜，Visual Studio 如今可以支持智能感知和 F#的调试了。对于一个函数式编程语言来说，为 F#实现智能感知是否更加困难？

Don：因为它是一种类型推断语言，在类型推断语言中确实做了一些折衷。C#智能感知的实现绝对一流，我们也很高兴在 F#中能提供这样的功能，虽然它在某些地方还有点粗糙，你可能需要在代码中放置一点点类型批注（annotation）。固定的类型信息越多，智能感知的体验变得越好。我们也非常高兴能让你的 F#代码获得自动背景类型检查的功能，不用说，我想这是在 IDE 中一个绝对重要的功能。

从推断类型中获得信息也非常有用。在编写代码的时候你就可以检查大部分代码，而在鼠标悬停到代码之上的时候还可以看到推断出的类型。调试体验——我们同样很高兴能为大家提供这个功能。它能起到一定作用，当然在 F#中对代码的执行有着严格的顺序，我们关心的主要事情是，很好的断点体验和很好的逐步调试体验。

我们主要关注开发代码（grade code）的调试，而较少关注优化代码（optimized code）的调试。在调试优化代码的时候做出了一些折衷，你依旧能用它，不过我们做出了一些有趣的折衷。在调试开发代码的时候尾调用在默认情况下是关闭的，因为它们会影响调试体验。当然，对于用于直接目的的立即循环，尾调用还是能发生，不过如果尾调用是互相递归且用于非直接尾调用，或如果你编写的代码非常依赖于尾调用（在 F#中，要绝对依赖尾调用的情况是少之又少的），而且你还希望运行在调试模式下，那么你可能不得不显式地打开尾调用。这也很容易就办到。无论如何，我们很高兴为大家提供了调试功能。

InfoQ：我们谈到了 F#文件的两种格式，还有第三种，就是 F#接口文件。你能简单谈一下 F#接口文件？

Don：是签名文件。当然，这也有意思，因为我非常熟悉那些使用过 OCamel 的人们。在 Java

中，每个类都是一个单独的文件。在 F#可以不这样做。你通常会有一个相当大的实现文件。就算只有一个文件实际上也会代表你的构架实现中的整个层，或整个组件，比如在 F#中的类型检测器就是一个大文件，又或名称解析引擎，在 Java 中它可能对应 10 或 15 个实际的类，如果可能的话会更多。

名称解析也是一个单独的文件。因为它封装了相当多的实现逻辑，在一个大的程序集中，你实际上可以在文件级别来封装它，由于你额外的封装边界，你可以在签名文件级别掩藏大量的细节。这是签名文件的一个用处。基本上，在这种文件中你会看到一个列表，或许是你的文档和某些特性标记，类型，和任何在签名文件掩藏起来没提及的任何东西。

签名文件的另外一个用处是，在你编写框架或函数库的时候，它们能非常好的用来进行函数库的外部设计，只需把这些外部设计代码写到签名文件中，在实现文件中进行函数库的实现。例如，注释，或许有所不同。在签名文件中，你能编写 XML 的辅助注释，它在函数库和实现文件中是用于构造器的正式帮助，你的注释将说明实现的逻辑。

虽然在两者之间有一点重复，比如公共方法的名称就会重复，但这是机制本身上的基本折衷，它还是可以遵守函数库设计和函数库实现方面的优良方法学。签名文件是这样一个特性，人们经常会想"我为什么非要用它？我不想在另外一个文件重复任何事情！"，而接着，他们会突然就用起它并意识到，在某些特定的应用领域，它实际上可以简化函数库的。

你在很多情况下大概都用到了签名文件，使用的频率介于接口签名和逻辑实现之间，大概有 1/3 或 1/5。有些情况下，根本上几乎没有实现。就像类型定义或其他的东西，在这样的情况下你完全不应该使用签名文件。

InfoQ：最后的阶段，或者说稳定 F#这个语言的阶段，你们准备好了吗，你们在这样的阶段吗？

Don：这就是最后的设计决策阶段。所有的都完成了。对于我而言，真的不可思议。能发布这个语言，我感觉好极了，我们依旧还有大量的工作要做，要完成所有的规格文档，还要在上面多花时间。在下面几个月内，将最终完成这些，以达到产品化的阶段。

我们还谈论了 Power Pack，不过对于我，重要的事情是，这个语言现在已经完工，我们能把这个语言用到所有有趣的事情上，比如正如我刚刚和你说到的服务器端编程工作。当然，重要的事情还有，你曾经编写（大约 6 个月前）的东西要能和现在的 F#代码保持源代码级的兼容。

在开发最新版的 F#和调整代码或其他事情的过程，没有做出多大的改变。你现在编写的代码将能永远运行——我们已经在用这个语言了。当然，我们也开始思考下一版本的 F#，我们在心中已经想到了一些很好的特性，要最终实现这些特性还需要获得更多的建议，比如更好的错误消息，提高总体体验的方法，以及我们能做到的一些增强。

我们也在开发一些重要特性。现在我还不能完全告诉你。一个重要的事情是，强类型语言是一旦获得数据，就要遵循相应的类型规则，那么编程体验才会顺畅。不过，在类型规则下还是可以有所作为，在 F#中进行编码如今也有一些极好的技术。如果你看过《Expert F#》这本书，它马上就要出第二版了，实际上名称会改变。在第二版中，它真正的名称叫做《Art of F# Programming》。我们的出版社在为第二版取名的时候，真的很敏锐。

我们有一个称之为数据构架阅读器设计模式的东西。它实际上是一种在语言的类型规则之下能够获取额外数据，并把数据转为数据构架的一种好方法。它已经被一些人所使用。如果你是 F#构架师，或你有志于成为这个语言的牛人，我们推荐你去购买这本书，可以看到这些内容和主题。我们认为，有一些有意思的方法真的可以开启从 F#这样的语言的静态类型规则之下获得数据的过程，我们希望能消除这种约束——以静态地获取类型。

InfoQ：F#使用一些.NET 函数库。F#的某些层是否能在.NET 的外部来使用？或者说只是用于.NET 的 F#？

Don：没有，我们没有这样做。F#的核心是提供给 F#使用的，它没有设计为在.NET 之外的语言中使用。不是特别要阻止你这样做，它的设计不是随意的。现存的函数库主要用于支持 F#代码。你购买的一些第三方产品也有用 F#实现的，你还是可以用在其他.NET 语言当中，不过它比那些通用的.NET 组件要特别的多。

InfoQ：就是说，常见的场景就是在 F#中首先进行一些应用，接着在从其他.NET 语言中使用它。

Don：是的。我们看到很多进行应用的人们，一个组件用 F#编写，而整个用户界面会用 Visual Basic 或 C#来编写。这是非常常见的方式。我们希望这种方式真正成为一种通用做法，以便实际发布的产品是一个 F#组件和几种其他语言开发的组件构成的。我们在 F#本身的实现当中也这样做，它包含了大量的 F#代码。大部分逻辑是用 F#开发，不过在 Visual Studio 中的属性页是用 Visual Basic 开发的，也有一个称之为托管打包框架（Managed Package Framework）的 C#组件，它整体代码中的一部分也是我们工作的基础。我们就是以这种模式来使用 F#。

原文链接：

http://www.infoq.com/cn/interviews/don-syme-fsharp-2_0-cn

相关内容：

- 利用F#在Silverlight中实现并行编程
- 微软Visual Studio 2010 正式发布，增加多项新功能
- 给VS2010 的UI进行重大升级
- 在多核GPU和CPU上使用Accelerator V2 执行并行编程
- 大型软件开发项目中的功能小组模型

InfoQ中文站

www.infoq.com/cn

我们的**使命**：成为关注软件开发领域变化和创新的专业网站

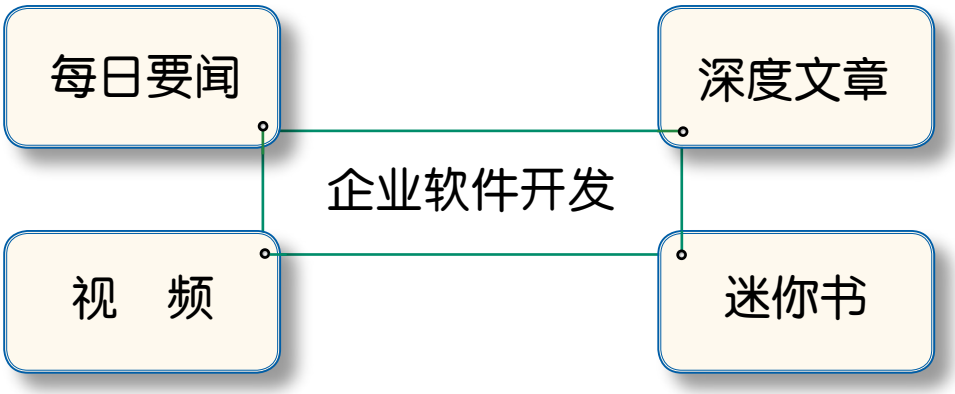
我们的**定位**：关注高级决策人员和大中型企业

我们的**社区**：Java、.NET、Ruby、SOA、Agile、Architecture

我们的**特质**：个性化RSS定制、国际化内容同步更新

我们的**团队**：超过30位领域专家担当社区编辑

……



讨 论 组：groups.google.com/group/infoqchina
编辑电邮：editors@cn.infoq.com
广告合作：sales@cn.infoq.com

热点新闻 | News

REST 能够解决系统集成难题吗？

作者 Boris Lublinsky 译者 吴宇

Steve Jones 在其最新一篇博文中写到，IT 并未成为企业的驱动力，而往往适得其反：

> .....现代世界大都由计算机来驱动，正是计算机使经济体正常运行，而计算机之间的集成在网络世界中很重要。IT 在解决企业方案方面的不成熟以及以技术为中心的方式使得 IT 完全没有成为引领业务需求的助推器，相反往往恰恰是 IT 在企业发展过程中拖了后腿。固执地沉醉于为根本不存在的问题寻找最纯粹的解决方案无异于在意淫，而这会对发达经济体造成很大的伤害。

在 Jones 看来，目前 IT 所处的境地比五年前更糟糕，主要基于以下事实：即关注于那些新鲜的"酷的"工具和技术，并始终在企业支持中错失良机，使 IT 越发变成自证预言。

Jones 认为在企业架构中最困难的事情之一就是系统集成，这也是有待解决的最复杂的任务之一。尽管如此，解决集成问题仍然是艺术多于科学：

> .....REST 未以任何显著的方式改进企业的这种现状。自 REST 被炒作以来已经 5 年有余了，而关于它与 WS-*标准给企业带来的益处的比较也有 5 年了，期间它终究是海市蜃楼子虚乌有。

尽管 REST 被吹捧为应对所有集成难题的灵丹妙药，但是：

- 对大型集成，进行大规模编程还是和 5 年前一样困难。
- 厂商还得使用他们自己的设备，这就意味着创新更少、挑战更小，而成本更高，因为开源社区的人们花费时间在其中意的项目上，而这些项目往往并不能削减企业预算。

Jones 描述的问题是指包括 SAP，Oracle 和 IBM 在内的厂商都仍然十分支持 Web Services，而主流的开发者却正埋头尝试切换到 REST。结果是：

> 没人把他们真当回事儿。随着所有那些有头脑又有型的玩家们远离 REST……我们面临着企业 IT 内部的知识真空，IT 厂商所填补其中的是那些只有他们自己才懂的玩意儿。也就是不断增加私有扩展和软件的数量，各自为阵推行自身的计划。

Jones 给出的关于这种情况的一些例子包括：

> ……Siebel 的 Web Service 技术堆栈包括 OASIS 规范之前的 WS-Security 标准，该标准最后一次更新是在 2006 年……而 IBM 仍然把 MQSI 老马车作为"高级 ESB"来推动。

他继续指出，除了一些规范和一些较小的改进外，实际上在 Web Services 领域过去的 5 年中没有任何创新。反观 REST 也并未取得多大的成就。他认为问题是两方面的：

1. 大部分 IT 人员似乎对其根本一无所知。现在我仍然惊讶于如此之少的 IT 从业人员明白它到底怎么回事儿，但真正让我吃惊的是 REST 拥有的推进力如此之少。
2. "所有事情"都是人工完成的，而且仍然没有标准的方式了解你到底在团队间搞了些啥。

即便如此，REST 社区仍在努力改进这种情况，但成效不大，Jones 是这样说的：

> 那些热衷于 REST 的粉丝们会跳出来高谈阔论他们所做的有关 WEB 的那些东西有多棒。不管那真的有多棒多精彩，那都不是我的世界，我的世界是面对 300 个系统，从 20 多年前就存在的到最近新建的都有，并且我需要让它们一起协同工作起来。即便 REST 是正确的"架构"方案，其"程序"方案仍旧是错误的，因其会带我们走向深渊。尽管我的世界相对来说有较大的预算支持，但是……你知道吗，如果想赚取真金白银，针对 IT 市场最富有的部分不是很好吗？

Jones 总结道：WS，而不是 REST，才是系统集成的方案：

> ……过去 5 年是企业 IT 惨淡的 5 年。WS-*标准是唯一可行的针对大规模编程的系统到系统的集成机制，但是它正处于停滞状态。REST 在前端有一些很酷的玩意儿可以提供给那些只投资最高端卓越人才的人，但是对一般企业环境而言那不是什么好玩意儿。

Jones 的这篇博文引发了相当强烈的反响。Joel Potisch 认为不应拿 WS 的方式来解读 REST：

> 我所注意到的唯一 REST 可能对企业 IT 所造成的伤害在于人们原本是想开发、部署那些正在运行的、可维护的、分布式系统的，结果却恰恰远离了企业 IT 并盲从于 WS-*标准听命于厂商……用 WS*-标准的方式来使用 REST 必然会使你倒退 5 年。REST 拥趸

> 们并不会盲从于新鲜酷炫的玩意儿，而是认可 REST，恰当地使用它，来消除分布式系统中 90%的共通性，使各种各样的问题顿时烟消云散。

Christopher Atkins 相信虽然 REST 是系统集成一个极佳的方法，但是在大多数情况下使用它需要仔细的规划：

> .....对我而言在企业 IT 系统集成中运用 REST 架构形式的主要挑战在于没有任何一个需要集成的系统是事先以 REST 的理念构建的.....至少你需要.....提供 REST 语义，例如，通过超文本定义内部状态机的明确的方法.....通过同时定义简单协议以及暴露内部状态机的超媒体来消除集成接口的歧义，这对于网络来说绝对是扭转局面的事。我就是无法相信 REST 可以有效地应用于大多数 IT 商店的集成场景中；也不相信无论是 TPS 系统无 FSM 可暴露（例如，直接的 OLTP/RDBMS 应用）还是 IT“天才”无法意识到 REST 的益处。

Dan Creswell 认为主要的问题不是技术或者标准的使用，而是人员：

> 九成以上的 IT 从业人员并不那么聪明，大部分的甚至算不上好.....这才是症结所在，而不在于他们所使用的什么.....REST、WS-*标准、数据库、Java 等，无论他们使用什么最终产生的结果都是废物。

尽管存在很多吹嘘空谈，我们很难去反对 Jones 的说法，不可否认的是 IT 行业目前正处于停滞不前的状态，而集成问题的难度不亚于从前或者被解决的速度并不比 5-6 年前快。时间能证明我们到底是回过头去使用 WS-*标准，还是 REST 最终追赶上企业计算的步伐。

原文链接：http://www.infoq.com/cn/news/2010/06/RESTForward

相关内容：

- REST和SOAP：谁更好，或者都好？
- 创建RESTful服务，有GET和POST足矣？
- REST及其版本控制
- 基于资源的架构：资源元数据
- 面向资源的架构：REST的另一面

热点新闻 | News

Flex SDK 与 Flash Builder 4：深度对话 Andrew Shorten

作者 Ryan Slobojan **译者** 张龙

最近，Adobe 发布了 Flex 4 SDK 与 Flash Builder 4（此前叫做 Flex Builder 3）。InfoQ 有幸采访到了 Adobe 高级产品经理 Andrew Shorten 以深入了解此次发布的相关信息以及他们将如何适应整个 Flash 平台和未来的期望。

InfoQ：随着从 Flex 3 到 Flex 4 的迁移，整个 Flash 平台似乎都有一些变动，包括将 Flex Builder 的名称改为 Flash Builder。能否详细谈谈这一点呢？

Andrew Shorten：在 Flex Builder 3 发布后，我们发现不仅是那些需要代码编辑、调试、分析等特性的 Flex 开发者在使用这个 IDE，那些只开发 ActionScript 项目和使用 Flash Professional 的开发者（也会花不少时间编写代码）也都纷纷转向了 Flex Builder。

Flex Builder 对日益增长的非 Flex 开发者的吸引力与日俱增，Flash 平台的势头也越来越猛，我们觉得无论是否加进 Flex 框架，开发者都还在开发着 Flash 应用，基于以上考虑，我们觉得改名字很有意义。现在，随着 Flash Builder 4 的发布，我为我们的决定感到骄傲：Flash Builder 依然为 Flex 开发者们提供最具效率的开发环境，此外还为那些使用 Flash Catalyst 的人们提供了工作流，包括对 Flash Professional 项目的新支持，根据 ActionScript 开发者们的需要提供了健壮的代码编辑功能。

本质上，我们承认 Flash Builder 是个专业的开发环境，面向那些使用 Flash 平台的开发者们，也就是说，如果你在编写 MXML 或 ActionScript 代码，那么 Flash Builder 将是你的不二之选。现在，Flex 的定位更加清晰了：Flex 只用来连接开源的 SDK 来构建 RIA。

除了名字上的变化，我们在 Flash Builder 4 中增加了 70 多个新特性：很多新特性都与代码编

辑、调试和分析相关，这会极大提升 ActionScript 或 Flex 应用开发者的生产率。甚至对于 Flash Builder 4 中所增加的特定于 Flex 的特性来说，那些使用 Flex 3 SDK 和 Flex 4 的人们也会从中受益无穷。

我写了一篇博客，完整地介绍了 Flash Builder 4 中的新特性并给出了对应的文档链接，感兴趣的读者可以自己去研究这些新特性。

InfoQ：Flex 3 开发者如何迁移到 Flex 4 上呢？现有的项目如何迁移到 Flash Builder 4 上呢？

Andrew Shorten：好消息是你可以将使用 Flex Builder3 开发的 Flex 3 应用迁移到 Flash Builder 4 上而无需使用 Flex 4 SDK：这样就可以在现有的项目中直接感受 Flash Builder 4 所带来的众多新特性。

我曾写过一篇文章，介绍了如何将项目导入到 Flash Builder 4 中，在迁移项目时要记住的关键一点是：在导入过程中，需要指定继续使用现有的 Flex 3 SDK 进行编译。

在将 Flex 3 应用迁移到 Flex 4 上时，工作量并不大。除了一些 bug 修复和默认主题的一点变化外，应用的开发方式与 Flex 3 相差无几，可能还要好一些。

如果想使用 Flex 4 的新特性则需要升级代码以使用新的 Flex 命名空间、将非可视化属性迁移到 declarations 标签中并更新状态代码以使用新的内联语法，此外，如果想使用全新的 Spark 组件、基于 Spark 的可视化效果或是 Flash Player 10 新增的 Text 功能，那么你需要做一些额外的迁移工作。

Adobe 发布了一个迁移指南（PDF）来帮助开发者升级现有的应用，Adobe Developer Connection 站点上也有关于该主题的一些资源供大家参考。

InfoQ：Flex 4 和 Flash Builder 4 的新特性主要有哪些呢？

Andrew Shorten：在 Flex 4 SDK 中，我们实现了一个全新的组件和皮肤架构（Spark），这为 RIA 增加了新的表现力。借助于新的 Spark 组件和皮肤架构，组件逻辑从组件可视化代码中“剥离”开来，可以更加直接地定制组件的行为和感官。此外，在 Flex 4 SDK 中，Flex 编译器的性能也得到了改善，同时增强了大量语言和基础设施等特性，对新的 Flash Player 10 运行时提供了完美的支持。

在 Flash Builder 4 中，我们可以通过一套完整的以数据为中心的特性、借助于 Flash Catalyst 和 Flash Professional 所提供的新的设计与开发工作流轻松实现与后端服务的连接；此外，我们还增强了核心的代码开发特性，以达到提升开发者效率的目的。

我们还实现了新的重构功能、改进了调试器以支持条件断点、观察点和表达式赋值，增加了代码生成功能，借助于新的网络监控和 FlexUnit 支持，应用的测试变得轻而易举。如前所述，无论使用 Flex 3 还是 Flex 4 SDK，开发者都可以充分利用这些改进，这样 Flash Builder 4 对整个开发过程都会起到帮助作用，即便没有使用新的 Spark 组件也没关系。

InfoQ：Flex 3 和 Flex 4 的组件模型有何不同？

Andrew Shorten：Flex 4 新增的一个主要特性是全新的皮肤与组件架构，叫做 Spark。

借助于 Spark，设计师和开发者可以更加直接和高效的方式无缝定制 Flex 组件的可视化元素与行为。Spark 架构的主要思想由一个皮肤模型支撑，它将组件的可视化元素与逻辑分离开来。这种分离赋予设计师和开发者更大的自由，可以随心所欲地设计 Flex 组件的可视化元素而不必顾虑组件的逻辑实现了。在了解了 Spark 皮肤模型后就能在很短的时间内构建富于表现力和健壮的 Flex 组件了。

InfoQ：最近很多人都在议论更大的 Flash 平台，比如几个月前收购了 Omniture，能否谈谈这个 Flash 平台呢？

Andrew Shorten：这个 Flash 平台包含了客户端运行时、框架、工具、服务器以及服务，设计师与开发者可以使用他们构建应用并部署到日益增长的联网设备上。

Flash Builder 是 Flash 平台不可分割的一部分，开发者可以使用它和 Flex 框架构建跨平台的 RIA 与内容，然后使用 Adobe Flash Player 部署到 Web 上或是使用 Adobe AIR 部署为桌面应用。将来还可以使用 Flash Builder 开发面向移动电话、平板电脑和其他设备的 Flex 应用，只要这些设备上有 Flash Player 或 Adobe AIR 就行。

从集成的角度来看，我们把重点放在了与 Adobe 服务器（比如 ColdFusion 和 LiveCycle）和服务供给（比如最近收购的 Omniture 所提供的服务）相关的工作流上，当然了，开发者还是可以使用众多的非 Adobe 技术来连接并交换数据。

在 Flash Builder 4 中，我们增加了一套以数据为中心的开发特性，可以探析到服务器端服务，Flash Builder 也可以生成服务器与相关数据模型的客户端表示，开发者可以将服务操作拖拽到可视化的组件上以绑定返回的数据，这样就能很快地将数据集成到 Flex 应用上，也能够与 XML、PHP、ColdFusion、Java、Web Services（通过第三方适配器）和.NET 协同工作。Adobe LiveCycle Data Services 通过向 Flash Builder 增加应用建模工具进一步升华了这种能力，它还支持高性能的数据消息和高质量的服务交付，同时还能实现离线的数据管理。

关于 Omniture，我们打算发布一个 Flash Builder 扩展，可以让开发者在 RIA 开发过程中以可

视化的方式追踪 SiteCatalyst，这么做就无需手工标记内容，进而减少实现时间，还可以映射自定义变量以实现追踪与分析的功能。

InfoQ：在 Flex 4 中，Adobe 为大多数现有的 Flex 3 组件都提供了新版本，但并非全部。比如说，Halo（Flex 3 组件模型）有 DataGrid，但目前并没有对应的 Gumbo（Flex 4 组件模型）版本。是否打算解决这个问题呢？如果是，大概什么时候呢？

Andrew Shorten：是的，但遗憾的是，目前有些组件我们还不能迁移到 Flex 4 组件模型上。也就是说，我们一开始就知道这是个问题，因此我们的做法是在同一个应用中，让 MX 和 Spark 组件的和谐共处变得简单，这样，即便是缺少 Spark DataGrid 也不会妨碍你迁移到 Flex 4 上。

当然了，我们打算发布剩余关键组件的 Spark 版本（比如 DataGrid、Tree 和 Form），就在即将发布的代号为“Hero”的 Flex SDK 中。虽然我们还无法预计该版本 SDK 的确切发布日期，但开发者可以通过 Adobe Open Source 站点上的每夜构建了解我们的开发进度。

InfoQ：虽然 Flex 4 改进了编译器性能，但 Flex 编译器在编译时间上还是远远落后于 Java 编译器。编译 Flex 的时间为何要比 Java 慢那么多？Flex 编译器的编译时间何时能达到 Java 编译器的水平？

Andrew Shorten：改进 Flex 编译器的速度始终是每个 Flex 版本的目标。在 Flex 4 中，编译期的编译速度和内存使用率上已经好了很多。接下来的几个版本中，我们会加倍努力提升 Flash 的编译速度，以期能达到其他技术的水平。值得注意的是，人们已经投入了多年的努力改进 Java 等技术的编译性能。与之类似，Flash 和 Flex 的每个版本都会向编译性能的改进这个目标大踏步前进。

原文链接：http://www.infoq.com/cn/news/2010/06/shorten-flex-flash-builder

相关内容：

- 企业级RIA应用的发展——专访Adobe技术专家董龙飞
- Adobe Flash平台开发者峰会谈RIA开发现状
- 高效率的超大规模Flex开发
- FlexMonkey将单元测试引入Flex用户界面开发
- Flex 4 的新体系

热点新闻 | News

2010 年 SOA 状态调查：SOA 在当今业界的地位得到巩固

作者 Boris Lublinsky **译者** 马国耀

最近，TechTarget 与 Forrester Research 就 2010 年 SOA 的状态合作完成了一项调查报告，该报告的内容有：

- SOA 发展的统计数据。
- IT 从业者在实施 SOA 时遵循的方法论。
- SOA 应在何时以及何处使用。
- SOA 的未来。

据调查报告的工作总结：

> 自从 2002 年被当作治理激增且不和谐的 Web 服务的一剂良药受到业界的广泛关注以来，面向服务的架构（SOA）的发展已经历了很长一段时间。作为一个行业热词，'SOA' 不再像过去飘得那么高了。与此同时，作为架构或实践方法的 SOA 在企业的技术领域中已经打下了扎实的根基。在今天的 IT 中，架构师和开发团队不断从开发及整合项目中抽象服务，以期待在将来进行重用和交互……调查结果显示 47.4%的回卷者所在的公司正在实施 SOA，同时开展若干 SOA 项目的公司占 30.9%。从项目的角度看，62.6%的项目本身就是"企业级的"……SOA 的使用没有出现滑坡，特别是在"云计算"以一个取代者的形象出现并且受到人们广泛推崇的背景下。

报告还显示，为了克服过去几年中的预算限制的问题，SOA 的方向发生了改变——现在它与 BPM 项目更加靠拢，而且往往通过应用程序升级项目的方式被引入。据调查，70.5%的回卷者正在使用或计划使用 SOA 进行 Web 应用程序的实施，而 51.3%和 48.7%的人分别将它用在数据服务和遗留系统整合的项目中。

> SOA 起初是作为驯服桀骜不驯的 Web 服务的手段而出现的，不过现在同样是这些服务又继续为 Web 化企业软件包及 SOA 运动提供了动力。

报告得到的另一明显的结果是，SOA 项目的成功率为 65.58%（其中 23.19 认为非常成功，42.39 认为某种程度的成功）：

> SOA 项目显现了明显的好兆头，过去人们认为它的投机性很大，失败的可能性不小，而现在却成了主流。这也表明了成功-失败与否取决于特定的 IT 项目。

报告还显示，SOA 使用的背后驱动力也在改变，2010 年的 SOA 的主要驱动力是：

> 改良的数据整合（29.2%）、遗留系统整合（26.7%）、应用开发的灵活性（27.2%），而且部门级应用整合（21.9%）已经超越了成本的节约（20.5%）与开发效率（19.2%）。

就此报告，Forrester 的 Randy Heffner 评论道，最有意思的结果是：

> ……目前为止，SOA 项目面临的最大挑战已经超越了 SOA 本身。27%的人认为，最大的担心是"如何通过多个项目（如 BPM，事件，BI，规则等）整合的方式设计 SOA？"。只有 13%的回卷者选择了第二大担忧，即"[为 SOA]评估及选择合适的工具"。换言之，工业界已经认识到，这是一个多元技术的世界，单一技术战略可能会错失目标。SOA 固然重要，但是业务技术解决方案需要的不仅仅是 SOA，还需要 SOA 与其他方法以及设计域结合起来才能满足需求。

Heffner 认为，为了建立一个新的技术战略的基础，SOA 执行者们正向商业能力架构前进：

- **始于业务输出**……最能衡量企业成功与否的尺度（业务输出）可归结为资产负载表、运维状态、资金流状态，以及指示企业在市场上的实力、形象以及影响力的软尺度。
- **引导业务能力的发展**。商业计划很难作为技术战略的很好的基础……更好的基础是企业的核心能力，产品设计和客户服务等。我们建设与发展这些能力才能获得更好的产出……为了实现这些能力，我们需要使用整合的、可操作的、可度量的方法将人员、流程、技术以及物理资源整合起来……
- **提供面向商业变更的设计实施模型**。有好的商业能力图固然很好，但是（还需为它配备）整合的、一致的、整体的商业能力实施的设计原则和实现模型。这里面有一个商业能力平台的概念，它是一个内聚的、整合的、多技术的以商业为中心的平台，它超越了我们

目前的以单技术为中心的 BPM 应用，事件驱动应等。

该报告再次证明了，尽管受到许多阻力，SOA 的地位却在 IT 及商业中打下了扎实的基础。

原文链接：http://www.infoq.com/cn/news/2010/06/soa2010

相关内容：

- 2010 年大规模技术架构的思路
- mySOA：敏捷的、治理的并且可持续的
- REST及其版本控制
- SOA的未解之谜
- Charles Nutter：JRuby性能再获提升，使用JRuby实现RESTful服务

热点新闻 | News

Google、Mozilla 和 Opera 质疑微软的 HTML 5 兼容性测试结果架构？

作者 Abel Avram 译者 侯伯薇

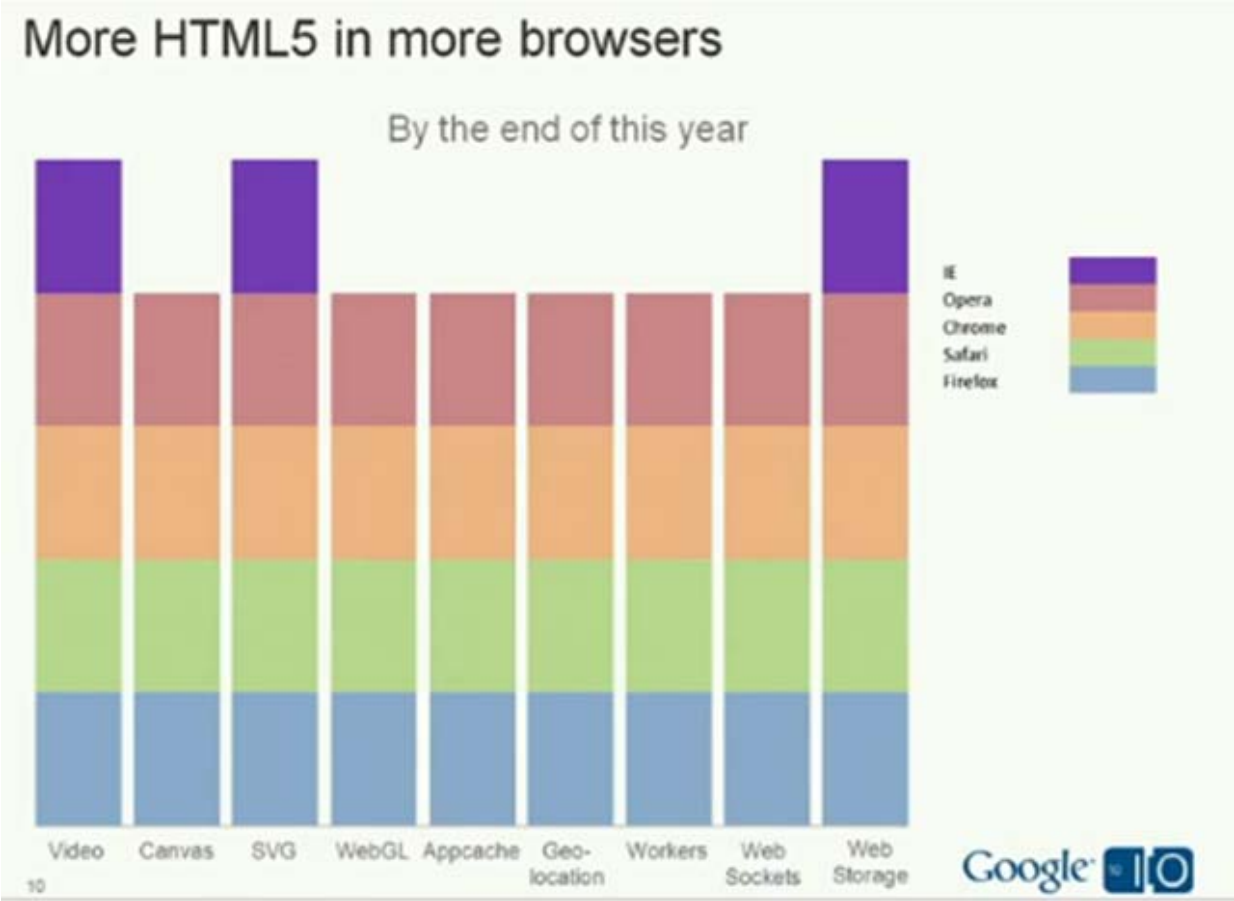
微软近日发布了 192 项测试的结果，这些测试分为 HTML5、SVG 1.1、CSS3 以及二级、三级 DOM 等八类，结果显示 IE9 成功通过了所有测试，而 Chrome、Firefox、Opera 以及 Safari 在各个类别中都或多或少有未通过的情况。由此得出的结论是，与 Google、Mozilla、Opera 相比，IE9 对于 W3C 标准有着最好的兼容性。

微软与万维网联盟（World Wide Web Consortium W3C）合作，创建了 192 个测试页面，各种主要浏览器在这些页面中的测试结果如下图所示：

W3C Web Standards	Number of Submitted Tests	Internet Explorer 9 Platform Preview	Mozilla Firefox 3.6.3	Opera 10.52	Apple Safari 4.05	Google Chrome 4.1
New! HTML5	40	100%	58%	45%	38%	38%
SVG 1.1 2nd edition	31	100%	84%	94%	90%	90%
New! CSS3 Media Queries	19	100%	74%	68%	42%	53%
CSS3 Borders & Backgrounds	33	100%	27%	88%	27%	91%
CSS3 Selectors	16	100%	81%	63%	50%	50%
New! DOM Level 3 Core	18	100%	78%	78%	94%	94%
New! DOM Level 3 Events	30	100%	37%	33%	20%	27%
DOM Level 2 Style	5	100%	100%	80%	20%	0%

结果的总结之后是详细的各项结果，并带有包含各种测试页面的链接，任何人都可以执行这些测试。由此得出的结论是：IE 9 的预览版完全兼容这些测试中所包括的 W3C 标准，而其他浏览器无法达到这一点。在其他浏览器中，成绩最好的是 Firefox，它完全通过了 DOM 2 的测试，但是大多数浏览器在很多类别中的测试结果都是黄色或者红色。InfoQ 采访了 Google、Mozilla 和 Opera，想要了解他们对于此次测试、结果以及结论的看法。

Google 提到了 Google I/O 大会第一天中的内容，在其中他们讲述了浏览器中有关 HTML5 的问题，并且期望在 2010 年底，除 IE 之外的所有浏览器都能够完全实现 HTML5 以及其他一些标准。



Mozilla 认为微软的测试覆盖范围有限，因此不够准确，并会产生误导：

> 我们很高兴微软热衷于使用这套测试来衡量浏览器对标准的兼容性，但是 MSDN 上这些特定的测试涉及到的范围很有限，并且关于浏览器兼容性的说法很显然会产生误导。
>
> 大家都知道，MSDN 提到的 SVG-in-HTML 的测试是不正确的[1][2]，它不符合 HTML5 解析规则和 DOM 规范，事实上利用的是 IE 的缺陷。微软的工程师都了解该缺陷，并且知道该项测试本身就是错误的，因此很让人奇怪的是，这项测试竟然被作为浏览器之间更进一步的互操作性的重要部分来介绍。另一方面，Firefox 的 beta 版本确实已经具备了 HTML5 解析器[3]，并且每夜构建[4]将会通过 SVG-in-HTML 的正规测试。
>
> 此外，MSDN 上列出的 CSS 测试很不精确，事实上，完全通过了这些测试的浏览器并不与标准兼容。有一份官方的 CSS3 选择器测试，我们已经通过了所有测试[5]，并且，我觉得对于 web 开发者来说，当讨论浏览器对 HTML5、CSS3 和 SVG 的支持的时候，像 caniuse.com[6]这样的站点会更有用。Mozilla 会继续致力于为处于考虑之中的规范[7]创建开发和公平的测试套件，也希望其他浏览器厂商能够参与进来。

更新：Mozilla 还更新了他们的声明，增加了下面这段简短但很有意义的信息：

> 我们得到微软的反馈，他们正在试图修正那些测试。

Opera 知道微软的测试和结果，但是他们使用了不同的测试套件，并得到了完全不同的结果：

> 我们没有执行内部的专门测试，而是一直在做大范围的测试，以确保我们在自己的浏览器产品中实现了所有 HTML5 中我们所制定的部分。Opera 对 HTML5 标准作出了很大的贡献，正是在 Opera 提出了 Video 标签之后，它才被加入到标准之中。我们一直在努力改善产品，以确保最大可能地遵守规范。
>
> 这项测试是非常有选择性的，它慎重地选择了所有要测试的元素，使结果看起来是"独赢"的效果。当评估 HTML5 测试的时候，标准应该与其他所有类型测试相同。测试的权威性应该是独立的，并且测试应该反应出规范的真实情况。我们很容易就能指出 Opera 比 IE 表现得好的 HTML5 相关的测试，比如下面这个：http://www.codedread.com/svg-support.php....
>
> 我们一直在测试，以了解我们的浏览器的兼容性如何，但是对于 HTML5 的兼容性工作还在进行之中。因此根据微软的网页，他们看起来只测试了 HTML5 中特定的一部分，而没有测试更多的内容，像 Canvas 元素等。

很有趣的是，Opera 指出测试结果显示，Opera 10.53 浏览器对 SVG 1.1 测试的通过率是 94.89%，而 IE 9 Preview 2 的通过率只是 30.55%。Opera 10.52 在微软的测试中有同样的表现，但是 IE9 在该项测试中的通过率是 100%，这与 30.55%相去甚远。

一位软件开发者 Jeff Schiller 执行了 Opera 所提到的测试，并对微软的测试结果做出了评论：

> 尽管我对 IE9 的发展方向感到很激动，并且一直对于你们参与 SVG 工作组的程度有很深的印象，但是我感觉"IE 测试中心"有很大的问题，并且会误导他人。即使是微软也承认他们只包含了自己所编写的测试（声称 IE9 可以通过）而不是完全的测试套件。
>
> 例如：SVG 测试套件包括 275 项测试。而微软在测试页面中只包含了 31 项。例如，IE9 甚至不支持渐变色。
>
> 另外一个例子，他们的 HTML5 的测试中关于一行中的 SVG 标记的测试实际上是错误的。它们之所以在 IE9 中能够通过，是因为浏览器在处理子节点的时候存在缺陷。如果我修正该项测试，它们就只会在 Firefox 每夜构建版本中通过，而在 IE9 中会失败。

Apple 没有对我们的调查作出响应，但是他们发布了 HTML5 Showcase，其中提到了他们自己的观点。

原文链接：http://www.infoq.com/cn/news/2010/06/HTML5-Compliance-Dispute

相关内容：

- 微软声明将支持HTML5 和H.264 视频标准
- Internet Explorer 9 预览：新特性与分析
- Chrome 4 现已支持HTML 5 Web SQL Database API
- Google将不再支持老式浏览器
- IE和火狐将使用DirectX进行呈现

热点新闻 | News

向作者支付的新方式： Flattr 小额支付服务

作者 Michael Hunger **译者** 侯伯薇

今年早些时候，出现了一种小额支付的服务，叫做 flattr——这是按时间收费（flatrate）和使其满意（flatter）两个词的双关语）。服务的原理很简单，但是它会改变我们在网上对有质量的文章支付的方式。Flattr 最初是由 The Pirate Bay 的创立者之一 Peter Sunde 创建的，也是由他在像 re:publica 这样的社交媒体会议上发布的。

在美国，另一种已经存在的服务叫做 Kachingle，它是在 2009 年发布的，关注于通过（再次）访问网站来分发收入。在 OpenFlattr Wiki，我们会看到关于类似的开源方法的想法。

小额支付系统的工作原理：

- 在 beta 测试阶段你需要取得邀请码
- 首先你需要注册一个账号
- 然后你需要通过 paypal 或者 moneybookers 向服务中存些钱
- 你需要定义每月的额度，它会被分发给所有你喜欢的内容，比方说，分发给你点击了 flattr 按钮的人。
- 作为创建者，你可以直接或者通过插件向你的内容中添加 flattr 按钮
- 在每月结束时，你的月度金额最终会被发放给你喜欢的内容的所有者，并且你会收到来自于喜欢你的内容的人的钱。

当前，Flattr 会从转入该服务的钱中抽取 10%的手续费。另一部分会被转账服务商（像 paypal）抽取。

这个服务现在已经上线了一个月左右，并有了第一手数据——他们拥有了大约 16000 个 beta 测试用户。最成功的作者在第一个月里面赚到了大约 200 欧元，其中每次点击 0.14 欧元。

Kent Beck 曾经写过一些文章，其中讲述了他在作为程序员和作者，在软件开发工作中得到回报是如何的困难。可能小额支付会是一种解决方案。

网络上的文化与传统的出版文化是不同的，主要在于大出版商会向读者出售大量出版物，读者需要先支付才能阅读。而在网络上，很多作者都会免费分发他们的作品，他们不会要求读者预先支付或者订阅。他们乐于创建并分享，因为他们也从别人那里得到了很多。这不是金钱的市场，而是关注度或者名声的市场。但是如果我真的喜欢一篇网志或者视频，因为它帮我节省了大量时间，那么如何才能够给作者报酬呢？一种方式是撰写并发布自己的文章。还有其它的方式吗？

通过 Paypal 或者其它服务支付报酬是不可行的，特别是对于很小的金额，因为都不够麻烦的，而且也不值得为其付出手续费。而易于设置和使用、并且会有大量社区中的消费者和生产者使用的小额支付解决方案会改变这种现状。

不仅仅文章、播客和视频的创建者能够通过这个平台获利。而且想要免费提供代码，但是也愿意让用户为其支付的程序作者也可以使用该项服务。

Tim Pritlove 在关于 flattr 的反馈文章(德语)中，提出了一个有趣的想法，flattr 的数量比 Facebook 中的“我喜欢”链接更能够说明文章内容的质量。"人们用钱来代替称赞。“如果喜欢的东西不用花费任何金钱，那么它到底价值几何呢？而如果你可以通过 flattr 服务来为他们支付金钱，那么就会表明你认为他的内容或者工作的价值有多大。

一些文章中谈到了礼物经济（德语），指的是你以文章的形式接受礼物，并且能够在稍后回赠对方，但那并不是必须的。

一些为软件开发社区——像 software engineering radio、chaosradio express（德语）以及 scrum coach Mark Löffler ——提供高质量文章的作者已经在他们的站点上添加了 flattr 服务。

一些程序作者也这样做了。

- netfilter/iptables - Linux 的包过滤器
- three.js——Javascript 的 3D 渲染程序
- Dia DIagram 编辑器
- Mercurial 网站

现在已经有了针对 WordPress 的插件，而针对 CMS 系统和音频播放软件的插件还在开发中。

你的想法如何？你会在 InfoQ 提供了这种服务的时候，通过 flattr 支付金钱吗？

原文链接：http://www.infoq.com/cn/news/2010/06/flattr

相关内容：

- 使用Apache Shindig为客户应用程序加入社区内容
- 社交BPM，你听说过吗?
- Facebook推出Graph API：语义网的未来？
- HTML5 Web Sockets与代理服务器交互
- Spring 3.0.3 与Spring Web Flow 2.1 相继发布

热点新闻 | News

SOA 的未解之谜

作者 Boris Lublinsky **译者** 马国耀

虽然围绕着 SOA 有数以千计的出版物、提供商和分析师的吹嘘，以及 SOA 曾被宣布死亡然后又在 SOA 宣言中重生的事实，但是该话题周围仍然存在许多疑团。McKendrick 在他最新的一篇博文对此进行了讨论。

SOA 与云计算之间的区别？David Linthicum 就二者之间的关系做了很好的定义：

> SOA 关注的是定义 IT 解决方案和架构的过程，而云计算是另一架构选择。所以 SOA 可以被云计算替换。事实上，大多数云计算解决方案都是通过 SOA 定义的。它们不是互相排斥，而是互相补充。

McKendrick 对此做了进一步补充：

> 一旦你有了彻底的了解，云计算其实是跨越企业防火墙获取或供应可重用的服务。类似地，Enterprise 2.0 就是通过访问服务更好地协作，利用终端用户的信息进行混搭。它们是面向服务的架构，并且依赖 SOA 的原理工作。

在人们还没有真正完全地实施 SOA 之前何来 SOA 的失败？我们参照最简单的 SOA 定义：

> ……面向服务的架构（SOA）是一组用在系统开发和整合阶段的灵活的设计原则。

这意味着 SOA 是对系统进行架构的方法——即，它关心的是“怎么做”而非“是什么”。McKendrick 认为：

> SOA 是一个演化的方法，而且还没有人真正完全地实施过 SOA……大多数企业仍处在计划和考虑他们的第一个 SOA 项目的阶段。事实上，这段时间我不断听到的 SOA 的主要挑战是它过于成功，在开展 SOA 的企业中，不是新建了太多的服务，就是服务

> 被无端（或者按需要）地开启了。

人们如何度量 SOA 项目的成功或失败？这里的问题是，对通用的企业架构，特别是 SOA 的成功与否的评判标准未被很好地被定义。Todd Biske 认为：

> .....企业评判成功与失败之间的主要差异可归结为期望和目标。如果期望和目标是清晰的，那么对成功与失败的评判也是清晰的.....这应是你的试金石。如果你采纳 SOA，你能回答这个问题吗"如何判断是否成功呢？"如果你不能回答此问题，你猜会怎么着，你可能会臆测自己是失败了。

Ugo Corda 对此做了补充：

> .....合理地检验 SOA 在其特定的优势领域里是否成功需要很长时间（譬如若干年），而且那些成功的故事应与成功的验证相隔很远。

McKendrick 认为：

> 这对开始实施 SOA 提出了一个刻薄的挑战——成功是长期积累的，它表现在跨业务单元的服务被共享，使得服务开发时间明显缩减，或者，业务可以方便地进行服务的重配置从而让产品和服务更快地进入市场，这些都要归功于 IT 基础设施的灵活性.....在市场上衡量长期成功的唯一正确的方法不是利润的增加就是股价的增长，而除 SOA 之外，还有许多其他因素作出了贡献。

有多少功能完备的，真正的 SOA 实现，确切的数字是多少？同样，问题是如何度量该数字？通过服务的数量和粒度？通过服务的消费者？借用 McKendrick 的话：

> 一组 Web 服务在何时能转变成 SOA 呢？有没有这样的阈值，它定义了当 Web 服务得到更好的关注和维护、治理、注册、管理及其它好的事物时就更像是 SOA 了？

Herbjörn Wilhelmsen 做了进一步解释并提出，功能完备的 SOA 需要：

- 清晰的战略领导力
- 区分业务价值的优先顺序
- 企业文化
- 合理的动机

- 服务发现
- 互操作性
- 重用的机会
- 促进服务的发展
- 服务级别协议（SLA）
- 测试面向服务的架构
- 监控服务

如果 SOA“与技术无关”，那为何我们这些技术人员要驱动它呢？McKendrick 认为：

> 虽然在每次技术大会、每个分析师的标注中、每篇文章中，你都会不断听到该说法，但是，SOA 并非绝对地、确定地、无疑地“与技术无关”。它由技术提供商推行，而且通常划归到 IT 部门的支持范围之内。

McKendrick 指出，SOA 是一个不断发展的架构方法，而且不管人们怎么说，许多关于它言论更多是出于情感，而非出于实际行动。

原文链接：http://www.infoq.com/cn/news/2010/06/SOAMysteries

相关内容：

- SOA设计关乎契约还是服务实现？
- 开放的云让业务更“闪亮”
- 将虚拟化技术作为云计算的平台
- OpenSocial的分析与实现
- Stu Charlton论云计算

特别专题 策划：InfoQ 中文站； 执行：霍泰稳 刘申

自 Visual Studio 2010 于今年 4 月份发布以来的，得到.NET 社区的诸多好评。本专题特邀四位.NET 技术社区的专家，结合当前的技术热点，如架构设计、Scrum、敏捷测试和云计算等，对 Visual Studio 2010 和 Windows Azure 做深入剖析，领略其技术之美。

我眼中的 Visual Studio 2010 架构工具 - 基于 Visual Studio 2010 进行敏捷/Scrum 模式开发

应用 Visual Studio 2010 辅助敏捷测试 - 基于 Windows Azure 的云计算应用设计

特别专题

我眼中的 Visual Studio 2010 架构工具

作者 张逸

影响架构质量的是构建体系架构的思想、原则、实践与架构师的经验，绝不是工具。即使是最优秀的架构工具，也不可能像倚天宝剑一般——倚天一出，谁与争锋——似乎谁握住了这把利刃，就能够成为武林盟主。架构工具可以改善架构师的工作，却不能替换架构的过程。软件开发过程中，最重要的依旧是人。

我在尝鲜Visual Studio 2010 架构工具[1]时，偶然看到一篇文章，用夸张的语言吹捧VS 2010架构工具，认为它是架构师最怕程序员知道的新工具。这让我有感而发，我想起数十年前甚嚣尘上的一个理论，那就是CASE工具在未来可以取代编码工作的论断。随着DSL的逐渐流行，这个论断似乎有了能够实现的希望。我们已经看到了很多优秀的代码生成工具，通过建模，可以花很少的时间完成编码实现。然而，现实是CASE工具至今仍然无法完全取代编码工作；而若要完全取代架构与设计的工作，则近乎不可能，因为工具不可能取代人类的思想与经验。我们可以不断地丰富最佳实践，在知识库中总结出架构的模式，利用工具来展现这些规律与原则[2]。这意味着我们可以从变化中寻找到不变，利用共性分析提高复用的可能（包括架构的复用），但变化始终存在，针对的问题域会因项目而异，即使是抽象，它也不可能无所不包，代表一切具体的实现。

理想架构工具的特点

“工欲善其事，必先利其器”。诚然，工具对我们的帮助不可低估，否则，就是拒绝革新的保守思想；然而，盲目夸大工具的效力，忽视人的作用，带来的负面影响并不亚于故步自封

[1]只在 Visual Studio 2010 Ultimate 版本中提供

[2]就好似重构工具对重构原则的支持

对前进过程的阻挠。用正确的态度对待工具，工具才能为我所用，这是我一贯坚持的态度。敏捷宣言认为：个体与交付重于过程和工具。客户满意才是硬道理，架构与设计所使用的工具，与客户满意度没有任何关系。所以，我们评价工具，是要看它能否提高我们的工作效率，改善我们的工作质量。那么，VS 2010 架构工具能够做到这一点吗？

我们需要先思考一下，作为架构师，最希望看到的架构工具是什么？我认为，理想的工具应该具备如下特点：

1、 易用性：可以非常容易和快速地构建与设计模型；

2、 可验证性：构建的模型是可以验证的；

3、 标准化：利于设计人员和开发人员的沟通；

4、 工程化：支持正向工程与逆向工程；

5、 可文档化：能够较好的支持文档化；

6、 可度量性：有助于对架构模型进行分析与度量；

7、 模板化：支持通用的架构标准与原则，便于快速生成模型；

8、 方法学支持：支持通用的软件方法学（尤其是架构方法学）；

9、 可集成性：能够结合在软件开发生命周期过程中；

VS 2010 的易用性

首先来看易用性。构建软件系统的架构，一部分工作是与图形在作战。无论是物理架构还是逻辑架构，用图形表达整个系统错综复杂的关系，最为直观。我希望在绘制与架构相关的模型图时，并不会因为绘图的不便对我的设计思路产生影响与阻挠，不会因为工具的无法表达或难以表达而影响我的工作质量，更不会因为构建出来的架构模型图被设计人员与开发人员所误解。这意味着模型图的绘制要尽可能简单与快捷，图形的表达能力尽可能丰富，同时还要符合通用的标准，不会因为一套全新的标记而产生沟通上的分歧。以对 UML 的支持为例，我们在构建架构的过程中，常用的 UML 模型涉及到包图、组件图、部署图、活动图以及用例图。VS 2010 克服了之前版本对 UML 支持不够的弱点，充实了 UML 建模的能力，提供了

包括组件图、类图、活动图、时序图、用例图五种 UML 模型。例如，我们可以绘制如下的组件图：

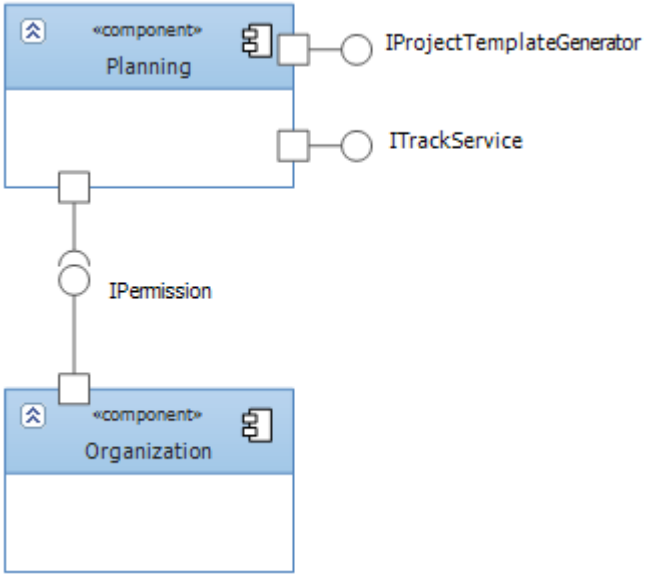


图 1：组件图示例

又或对时序图的支持：

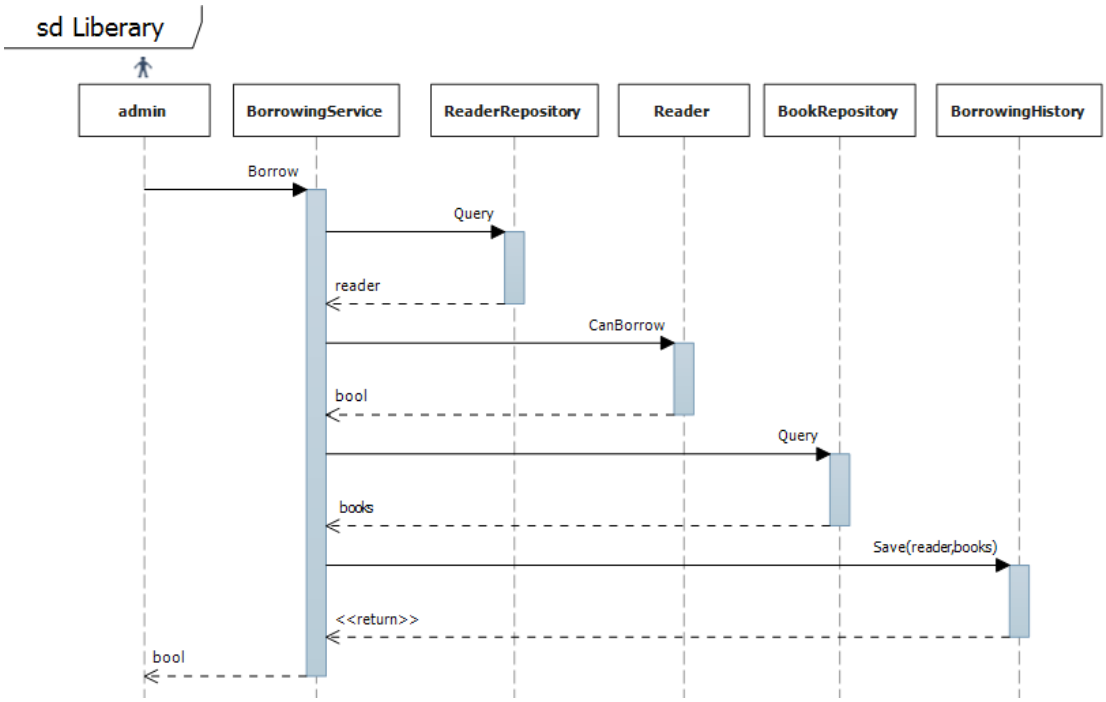


图 2：时序图示例

整体来看，它对 UML 模型的支持差强人意，虽然可以绘制出异常漂亮的 UML 图，但操作并不方便，无法提高建模的效率。例如：在组件图中无法轻易地绘制接口对组件的实现，对提

供的接口与要求的接口之间的对应关系也较复杂；在时序图中，Actor 的图形既不符合 UML 通用标识，使用也不方便——它没有将 Actor 作为一级公民，而是作为 Lifeline 的一项属性提供。

作为架构师，如果希望完成UML建模，我不会选择VS 2010，因为它没有提供包图[3]和部署图，不过它所提供的层模型却值得尝试。如果我们需要对大型生态系统的建模，或者绘制网络拓扑图，VS 2010 更加难以胜任。从这一点来看，VS 2010 若想成为架构师的首选，还有很长的路要走。

VS 2010 对双向工程的支持

根据我对 VS 2010 的分析，我认为，微软的野心并没有这么大，它并没有期待 VS 2010 提供的架构工具完全涵盖架构与设计的各个领域。它的杰出表现，尤其对于 UML 建模而言，还是在于双向工程（主要是逆向工程）的完美支持。

事实上，微软对UML双向工程的支持可以追溯到 1997 年与Rational合作开发的Visual Modeler。在Rational被IBM收购之后，这一工具就销声匿迹了。从VS 2005 开始，提供了对逆向工程的支持，但这种支持非常有效，仅限于对类图的支持。VS 2010 完善了对UML主要模型的支持，因此双向工程更具有普遍意义。VS 2010 对正向工程的支持并不够，它需要根据 T4 模板来创建，并充分利用了Visual Studio的扩展功能。我不明白微软为何不直接在菜单项中提供对正向工程的支持，因为它本身可以非常完美地做到[4]。

VS 2010 对逆向工程的支持极为强大，除了支持之前版本已经提供的类图外，还支持生成依赖图、时序图以及层模型。依赖图可以帮助我们了解程序的结构以及类之间的关系，时序图则有助于理解对象之间协作的方式，至于层模型则在更高的层面上帮助我们了解分层架构。

VS 2010 可以按照程序集、命名空间、类等建立依赖图。以.NET 提供的示例程序 StockTrader 为例，我希望了解服务层中相关类之间的倚赖关系，就可以通过 Architecture 菜单的菜单项“Generate Dependency Graph”。我按照自定义的方式生成依赖图，如图 3 所示：

[3]在类图中提供了 Package，但没有专门的包图功能强大

[4]正向工程的做法请参见 MSDN 文档：http://msdn.microsoft.com/en-us/library/ee329480.aspx#What

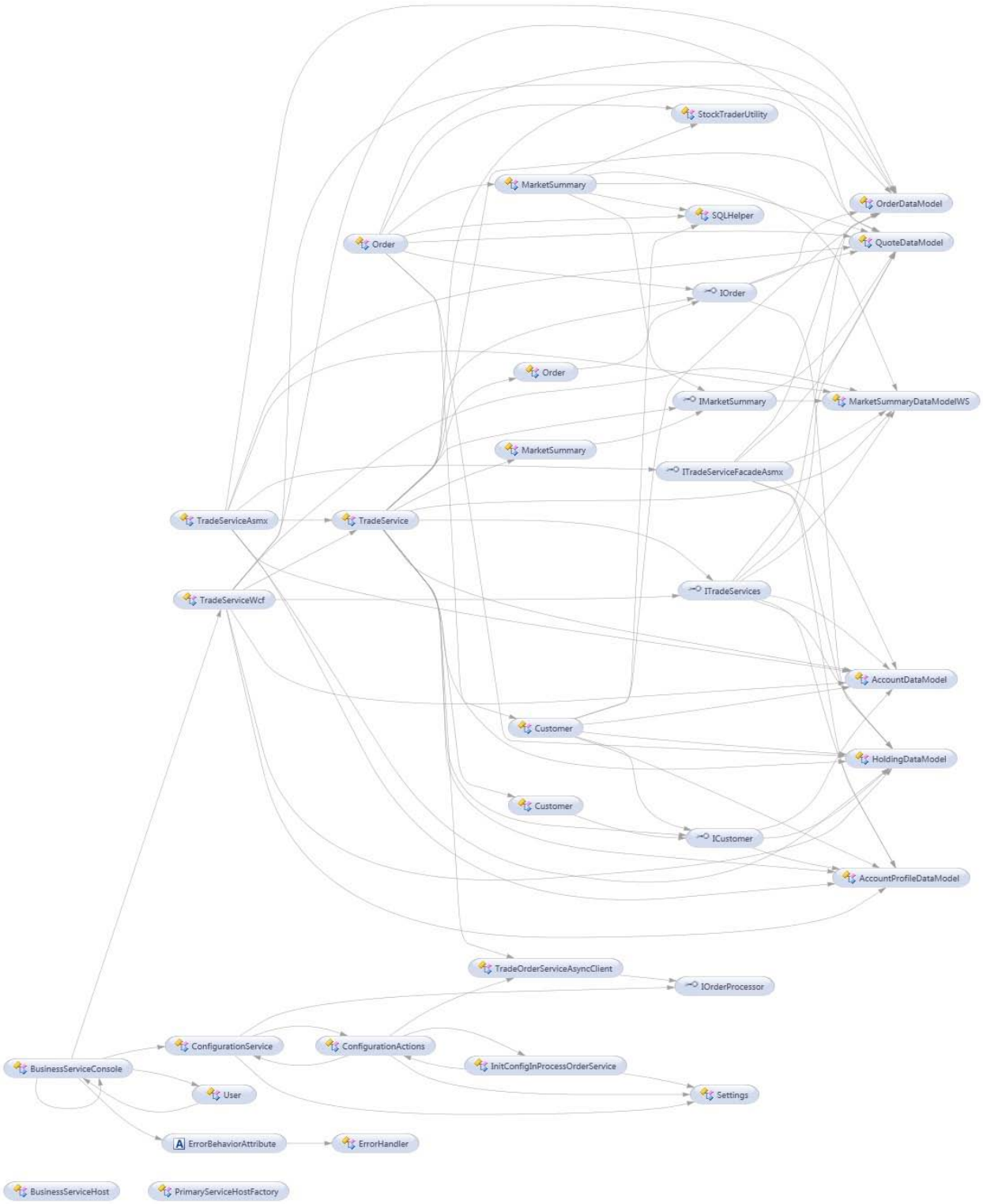


图 3 根据自定义方式（公开的类）生成的依赖图

图 4 为该依赖图的局部：

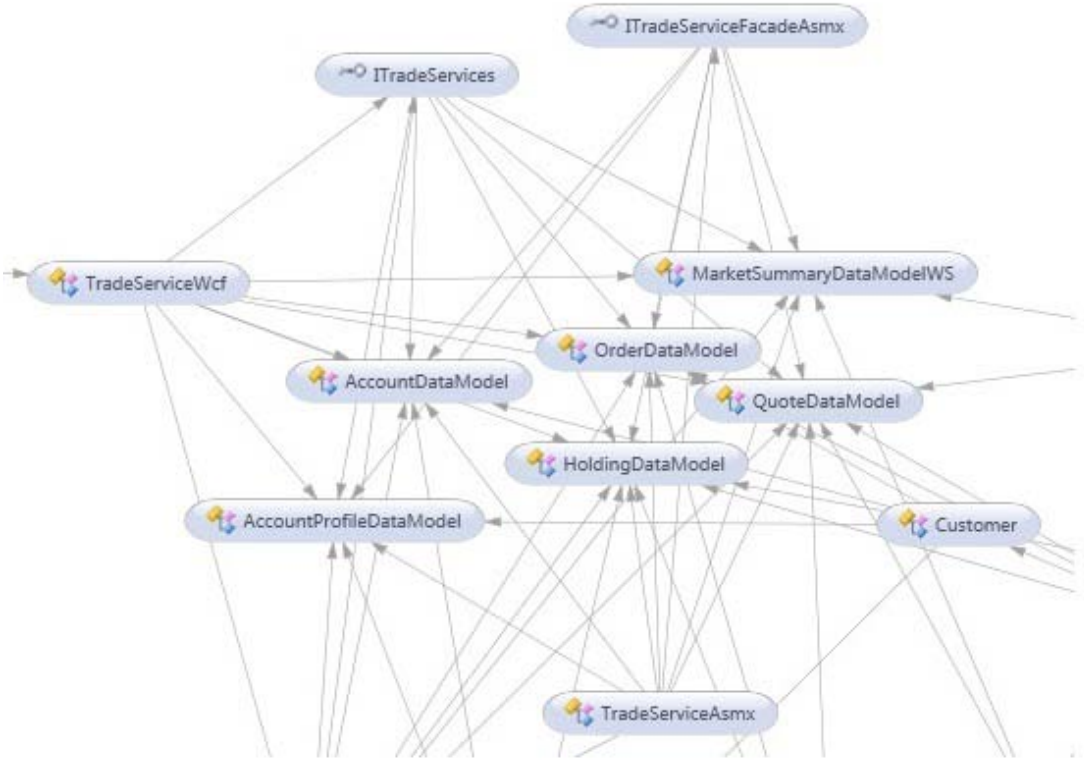


图 4：依赖图的局部

我们可以看到 Model 对象被依赖的强度是最高的，其中 ITradeSerivces 接口几乎依赖所有的模型对象，这是因为该接口是服务层的主要服务，相当于外观服务，负责协调和操作订单、报价、账户信息等模型对象的消息处理。

就我的感受来看，这样的依赖图显得有些华而不实，因为这样一张蜘蛛网般的图形，实在让人有些茫然，我们可能会在一张超级大型的依赖图中迷路。不过，如果希望对依赖关系来一次鸟瞰，或者需要初步了解各个对象的依赖强度，该依赖图还是有一定的参考价值。此外，倘若系统规模不够大，则可以选择类型级的依赖图；如果系统规模太大，则可以根据程序集或命名空间生成依赖图，这可以在一定程度减小依赖图的复杂程度。

时序图的逆向工程实在太精彩了。静态的类图虽然有助于我们了解类的结构，但类之间的协作却根本无法跟踪。我们在做设计的时候，常常会借助时序图来帮助我们了解某个功能项的执行过程，追踪它的消息传递方式，清晰把握类的协作方式，并以此为基础寻找到类的行为，以及不存在于真实世界的类对象。在阅读并理解代码时，如果能够有时序图的帮助，会更容易理清设计的思路，明白设计的目的。例如，同样在 StockTrader 项目下，我需要了解服务层中 TradeService 类的 login()方法实现，就可以为其生成一个时序图。在 VS 2010 中，生成

时序图只需要在方法上点击右键，选择“Generate Sequence Diagram...”项即可。图 5 是为 login()方法生成的时序图，我们可以清晰地看到 TradeService 与 Icustomer 和 ConfigUtility 之间的交互情况。

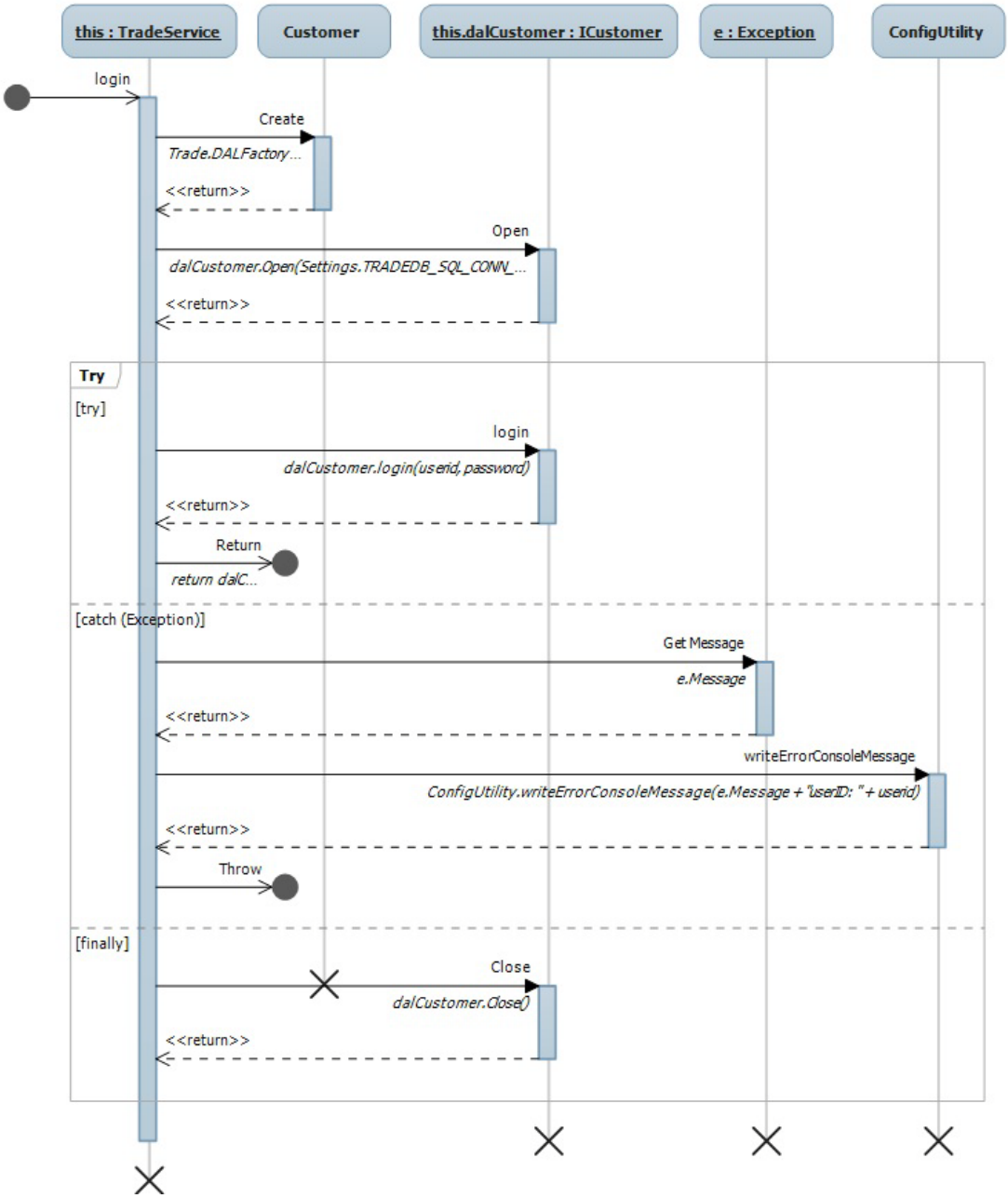


图 5：为 login()方法生成的时序图

在 VS 2010 中生成层模型和验证架构

层模型对架构师的帮助更大。VS 2010 可以根据现有的解决方案生成层模型。在打开现有解决方案后，添加一个新的 Modeling Project，并创建一个 Layer Diagram，然后将解决方案的

相关程序集拖拽到 Layer Diagram 的设计器中。通过执行快捷菜单中的“Generate Dependencies”命令，可以检查并获得各个程序集之间的依赖关系，并以图形方式展现出来，如图 6 所示。

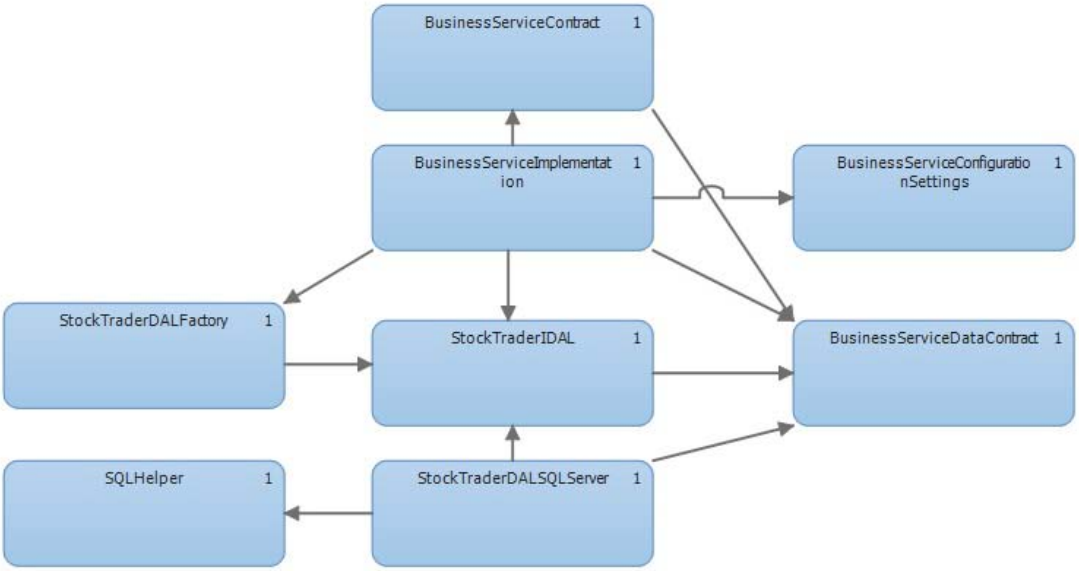


图 6：为服务层生成的层模型

通过图6，我们可以看到BusinessServiceImplementation依赖了StockTraderDALFactory、StockTraderIDAL、BusinessServiceDataContract以及BusinessServiceConfigurationSettings，同时它作为BusinessServiceContract服务契约的实现，还要依赖BusinessServiceContract。很明显，BusinessServiceImplementation与具体的数据访问层实现StockTraderDALSQLServer之间没有任何依赖关系，这就意味着服务层与数据访问层的具体实现是解耦的，这符合一般的架构原则。利用Layer Diagram，我们就可以很好地了解各个模块之间的依赖关系，帮助我们分析架构的合理性，是否存在双向依赖、循环依赖，或者模块设计是否很好地遵循高内聚、松耦合原则。

VS 2010提供的“Validate Architecture”菜单项还可以对架构进行验证。我们可以直接对上面生成的Layer Diagram执行验证。但这样的验证并没有价值，因为验证的规则就是通过现有实现生成的。微软推荐的做法是架构师根据对层与模块的理解，绘制一份符合架构原则的Layer Diagram，然后将已经实现的程序集拖拽到相应层上，再执行验证。如果实现违背了Layer Diagram要求的原则，就会提示错误[5]。这样的验证功能可以帮助架构师快速地验证团队的开

[5]具体做法参见微软的官方博客 http://blogs.technet.com/b/teamarchchina/archive/2009/08/31/vsts-2010.aspx

发人员在实现过程中是否遵循了自己的设计。

从上述分析可以看出，VS 2010架构工具充分利用了它与IDE集成的优势，为设计人员与开发人员提供了便利的工具，完成模型的转换与输出。这既有利于设计人员对架构的验证，帮助维护人员理清程序结构之间的关系，通过对依赖关系的度量检验模块和类之间的耦合关系，更有利于团队在项目后期生成设计文档。可以说，VS 2010在可验证性、标准化、工程化、可度量性方面都有闪光之处。遗憾的是，VS 2010似乎并没有提供自动将这些模型图转换到Word的功能[6]，这不能不说是一种遗憾。

VS 2010 的不足之处

VS 2010似乎并没有足够的功能支持我们快速生成架构，例如经典的三层架构、管道-过滤器或MVC架构。这也正是我想表达的工具不能替代架构师的最大问题。即使可以构造一些模板，就像提供项目计划模板一般，支持这些经典架构的快速生成，但始终无法替代架构师对领域知识以及质量属性的分析与设计。

从方法学支持的角度来看，VS 2010仍然支持不够。我很喜欢Enterprise Architect对ICONIX方法的支持方式，在VS 2010中没有看到对相关方法学的支持[7]。即使是对UML的表达，VS 2010都显得过于简单（支持模型少）与复杂（操作不方便），虽然它的图形真的非常炫。当我需要构建一个架构时，VS 2010绝对不会是我的首选；但当我需要为架构的实现进行验证或者提供设计文档时，只要我工作在.NET平台下，我绝对会毫不犹豫地选择它，它真的是一件具有超强战斗力的利器。

结语

现在的Visual Studio 2010已经不是单纯的IDE这么简单，它是一个全方位作战的快速工作平台，通过它可以完成设计[8]、开发、测试、重构以及团队的管理与协作。这种涵盖软件开发

[6]只能将模型图粘贴到 Word，而不提供直接导出 Word 文档功能

[7]当然，VS 2010 支持微软解决方案框架，即 MSF，可以实现概念设计、逻辑设计与物理设计，但就现有的 VS 架构功能来看，对这些设计的支持显然不够。

[8]对 Modeling Project 还支持版本控制功能

生命周期各个阶段的综合工具[9]，是开发人员和管理者梦寐以求，因为我们不用再考虑各种不同工具之间的集成与部署，何况VS 2010还提供了非常强大的扩展功能，使得我们能够因项目或技术而异，实现自己的定制。

不过，这种大一统的强权模式，很容易限制开发人员的自由与创新，抹杀人的个性。“知其然而不知其所以然”，开发人员慢慢会产生一种惰性。由工具来完成许多繁杂的工作，固然可以提高工作效率，却失去了探究背后原理的机会，正如当年的ASP.NET，造就了一帮不明HTTP工作机制的程序员一般。这就需要我们进行取舍，回到开篇的话题上，那就是我们必须要明确自己对工具的态度，让工具为我所用，却不会被其所制。

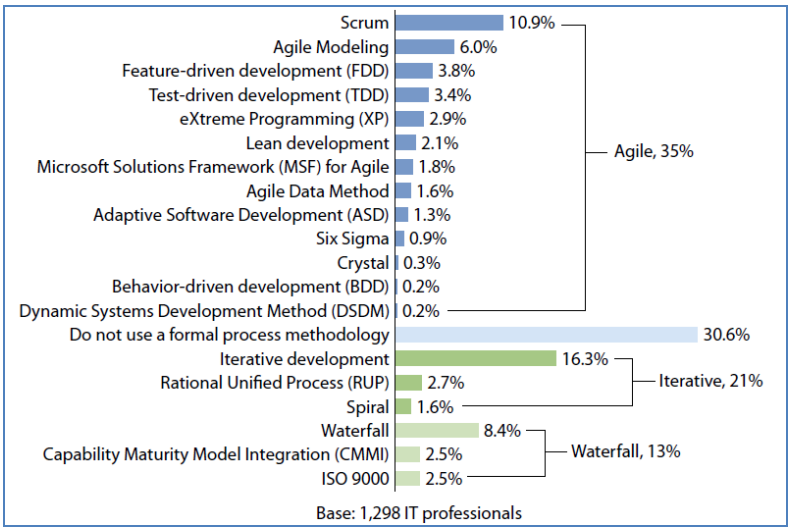
[9] VS 2010 加大了对敏捷的支持，并将 Scrum 作为基本的敏捷开发模型

特别专题

基于 Visual Studio 2010 进行敏捷/Scrum 模式开发

作者 徐磊

根据 Forrester Research 今年第二季度的一份研究报告，在超过 1000 名专业开发人员中，采用敏捷模式进行软件开发的已经有 10.9%采用了 Scrum 模式，在所有的敏捷开发模式中名列首位，而在所有的软件项目管理模式中，敏捷模式更是被 35%的开发人员所采用。当然，研究报告为我们呈现的仅仅是一个统计学的观点，到底你的开发团队应该采用什么样的开发模式，这还是要根据各自不同的开发环境，人员构成，公司架构以及文化背景来决定。



图：Forrester 关于敏捷模式的调查报告

Visual Studio 2010 是微软在 2010 年 4 月发布的全新一代的集成开发环境，配合同时发布的 Team Foundation Server 2010（TFS——团队服务器），为开发团队提供了全面的应用程序生命周期管理（ALM）工具和平台。在 2010 这个版本中，对于敏捷，或者说 Scrum 模式的支持是前所未有的。虽然微软的 Visual Studio Team System 从 2005 年开始发布的时候就提供了敏捷流程模板（也就是 MSF Agile）模板，但是 2008 版之前的这个敏捷流程模板都是基于 MSF（微软解决方案框架）的；这个框架是微软针对自己的研发团队的最佳实践进行抽取总结出来的，与广大敏捷开发社区里面所流行的很多敏捷方法并不是很契合，造成了开发团队在实

施的时候有很多不适用的地方。因此，微软在开发 2010 版本的过程中，大量的听取了敏捷开发社区中的声音，在自己的 MSF Agile 5.0 的模板中进行很多针对敏捷，更确切的说是 Scrum 开发模式的改进，使得 2010 版本中所集成的 MSF Agile 5.0 的模板非常适合我们来进行 Scrum 模式的开发组织。当然，微软的产品为了追求通用性，在 MSF Agile 5.0 的模板中并没有完全采用 Scrum 模式通行的名称和流程；同时，微软在两周前又发布了一个纯粹的 Scrum 流程模板以供那些希望完全使用 Scrum 模式的开发团队使用，当然这个模板现在仍然是 Beta 版。

我个人认为，开发团队采用哪一个模板并不是最重要的，重要的是我们需要在开发过程中不断地改进过程，并对这个模板进行定制，以便适合我们自己的开发流程。这也是为什么 TFS 所提供的是一个模板，因为它的目的就是希望我们在这个模板的基础上不断的改进，最终找到适合

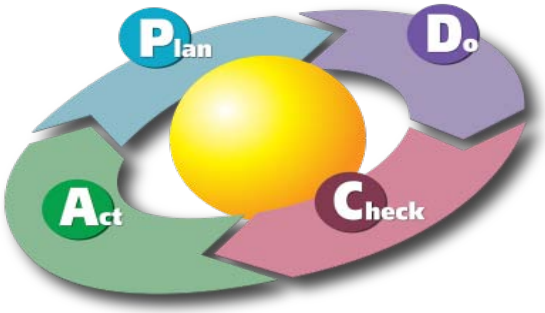
自己开发团队的流程。其实这也很符合 Scrum 模式的理念；简单一点来说，**Scrum 模式是一种针对复杂项目的流程组织方式的框架**，其目标是为了让我们开发出更高质量的软件产品。围绕的这个目标，Scrum 模式为我们提供一个团队模型，一系列工具和一个简单的流程。在这样一个框架之下，Scrum 模式要求我们不断地改进流程以达到适合团队的最佳状态，这种对改进的要求也是 Scrum 模式区别于其他开发流程的重要特点之一。

为什么 Scrum 模式适合软件开发？

软件行业至今已经有超过 40 年的历史，很多在软件工程中的管理方法都是在不断摸索中改进而来的。早期的软件行业由于规模有限，绝大多数属于作坊型，几个人在一起靠着自己的聪明才智创造出软件产品；但是当团队规模不断扩大的时候，开发人员开始需要一种模型来组织越来越庞大的团队，满足越来越复杂的需求。因为没有经验可循，软件开发团队将很多传统工业工程的方法借鉴到软件行业，因而出现像“瀑布式”的模型。“瀑布式”模型要求我们在实际的开发工作开始之前进行很多非常细致的设计和计划，力图将不可控的开发过程细化成可以控制的颗粒，以达到对复杂项目的总体控制目的。但是“瀑布式”模型忽视了软件项目的一个本质特点，那就是需求的不确定性；我们不可能像造汽车一样在上生产线之前把所有的零件都设计好，所有的流程都规定好，再进行装配；因为任何软件在实际进行编码之前都没有人知道这些代码应该如何实现，而且每一个开发人员的水平不同，习惯不同，写出的代码也是不同的；再加上客户对于软件的需求也是在不断变化的，一年之前的业务流程很可能在一年之后就产生的变化，如果还按照之前的需求进行开发，那么交付的时候肯定是无法满足要求的；更重要的事，在客户没有看到或者实际操作软件产品之前，他们永远也不能明确地告诉你他们要的到底是什么。因为这种种原因，造成了软件开发不可能采用传统的

工程方法进行组织，因为其本身是一种需要依赖于开发人员智慧的探索性行为，也造成了我们的软件项目中有很大一部分是失败的。

Scrum 模式的出现正是基于对于软件开发行为本质的认识，提供了一种松散的框架，让我们使用一种探索性的流程方法来组织本来就是探索性的开发过程；从根本上满足了软件开发本身对于流程的需求。这种方法论实际上是基于爱德华·戴明所提出的戴明环的管理方法；戴明环理论提出：人类在进行任何复杂活动时，获得成功的最有效过程要经过：Plan 计划– Do 执行 – Check 检查– Act 改进，四个子过程，并不停的迭代以便找到最佳的方法来解决问题。这个理论不是针对软件开发提出的，但是软件开发本身其实就是最典型的复杂活动。



这里我们再回头看看 Scrum 的流程，Scrum 的流程主要包含以下内容：

- (P) Release/Sprint Planning：发布/迭代计划
- (C&P) Daily Scrum：每日回顾
- (C&A) Sprint Review：迭代产品检查
- (A) Sprint Retrospective ：迭代流程检查

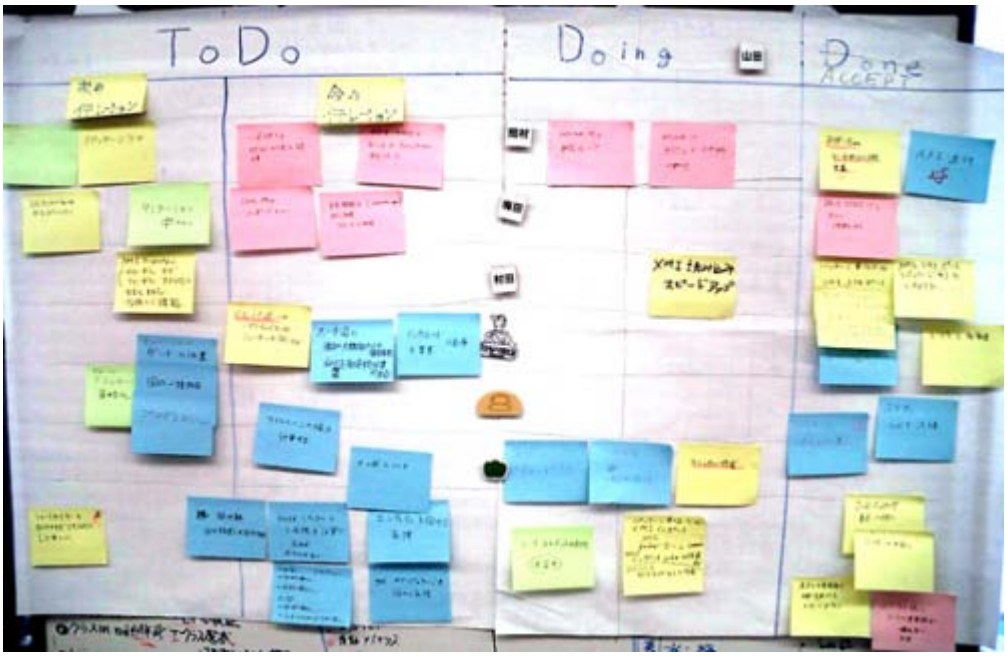
我们可以看到，Scrum 模式的流程与戴明环仅仅相扣。有很多认为敏捷模式会弱化计划的作用，其实不然，敏捷模式更加强调计划，而且强调更加频繁的计划，比如：每日回顾这个流程就要求我们的团队每个成员每天早上用 15 分钟的时间来回答 3 个问题：

1. 你昨天做了什么？
2. 你今天计划做什么？
3. 有什么问题阻碍你的开发进程？

其实这正是对于之前开发内容的检查，同时也是对后续开发内容的计划过程。

Scrum 模式需要怎样的工具来实现？

对于使用什么样的工具来实现 Scrum 模式，现在也有很多不同的观点。其实有很多人认为白板和即时贴就是最好的工具，其实对于小型团队来说这的确是最有效而且最经济的方法。但是如果考虑到软件公司的管理需求（工作量统计等），远程团队，开发工具集成，代码质量控制，发布后期支持等等；我们还是需要一个高度集成的平台和一整套工具来支持我们的开发团队。



Visual Studio 2010 所提供的集成开发环境可以满足我们以上的一系列需求，帮助我们的开发团队更好组织开发，帮助我们的管理层更好地掌控开发过程，帮助软件公司开发出更高质量的产品。

Scrum 模式对于工具的要求，主要集中在以下一个方面：

1. **团队组织：**满足 PO （产品经理），Scrum Master （流程经理）和开发团队管理，以不同的权限访问团队项目并对不同角色提供个性化的信息支持的能力。
2. **产品需求记录和跟踪：**对于 Product Backlog Item （PBI 产品需求列表）的添加，编辑，优先级排序以及交付开发团队以后进行跟踪的能力。

3. **流程管理：**满足 Sprint Planning, Daily Scrum, Sprint Review 和 Sprint Retrospective 这些流程中对于信息共享，信息转移和跟踪的能力。
4. **产品质量：**在整个开发过程中，配合 Scrum 模式达到产出高质量代码和产品的能力。

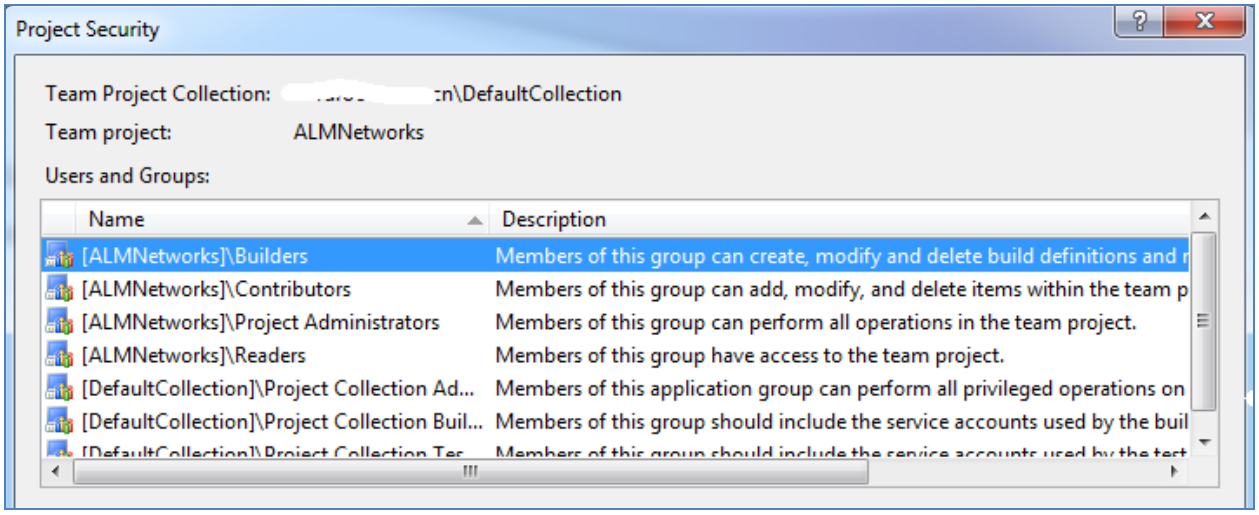
下面我们就看看 Visual Studio 2010 系统在这 4 个方面如何满足 Scrum 模式的需求，并协助我们开发出高质量的产品。

Visual Studio 2010 上的 Scrum 团队组织

一个完整的 Scrum 开发团队主要由以下角色组成：

1. **Product Owner （PO 产品经理）：**我喜欢把 PO 翻译为产品经理，因为 PO 的工作职责就是向客户和干系人收集产品需求，进行排序并保证开发团队按照干系人对需求优先级的要求进行交付。
2. **Scrum Master （SM 流程经理）：**对于 Scrum Master 我一直没有更好的翻译，将其译成为流程经理是因为这一角色要保证团队按照 Scrum 的方式来组织开发，并协助团队和 PO 进行有效的沟通，解决团队所遇到的问题。Scrum Master 和项目经理的区别在于，他更加倾向于保证开发流程的完整性而不是倾向于满足客户/干系人的需求。
3. **开发团队：**开发团队在 Scrum 模式中是作为一个整体出现的，一般来说团队的大小控制在 3-7 个人的规模；团队作为一个整体向 PO 负责，而不是每个人对于自己的任务负责。

在 Visual Studio 2010 系统中，使用 TFS 服务器基于角色的权限控制，我们可以很方便地定义出不同的权限范围。当然，最简单的方法是把 Scrum 团队的角色和 TFS 的默认角色之间进行映射。



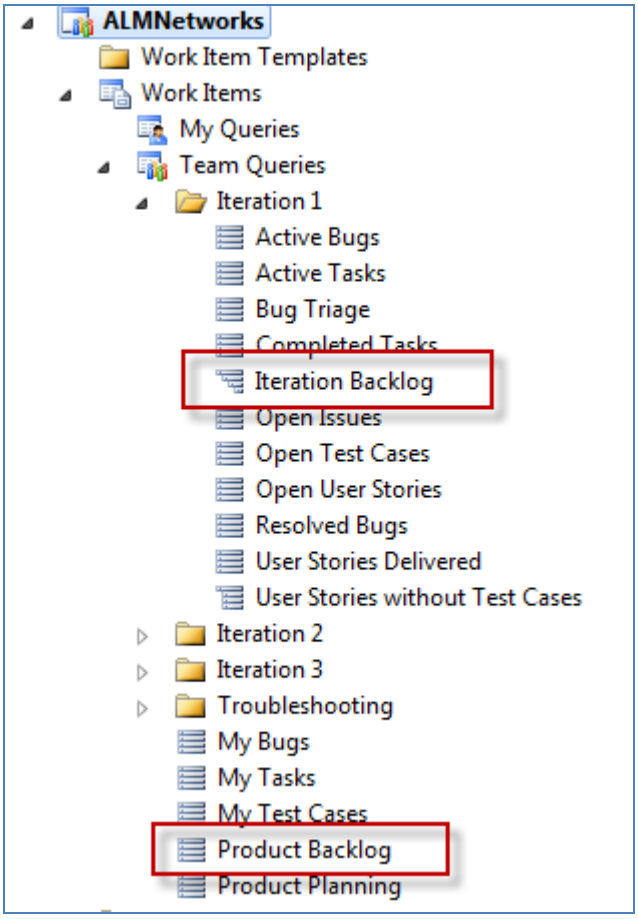
图：TFS 团队项目的默认角色

Scrum 团队角色	TFS 团队角色	
Product Owner	Contributor	
Scrum Master	Project Administrator	
开发团队	Contributor Builders Project Administrator	根据团队不同人员的职责具体分配
项目干系人	Readers	如果客户愿意更直接的参与项目，可以允许他们直接访问 TFS。

表：Scrum 团队和 TFS 团队角色映射

Visual Studio 2010 系统中对需求记录和跟踪的支持

Scrum 模式中的需求主要是采用 Product Backlog Item（PBI 产品需求列表）和 Sprint Backlog Item （SBI 迭代需求列表）来进行管理的，在 Visual Studio 2010 系统中，直接提供了针对这两个列表的工作项查询，并且还提供了 Agile Workbook （敏捷工作簿）帮助我们更好对工作量和任务分配进行调控。



图：使用 MSF Agile 5.0 模板创建的 TFS 团队项目集成了对 PBI 和 SBI 的管理功能

Product Backlog [Results]

Save Results | Refresh | Open in Microsoft Office | Edit Query | Column Options

Query Results: 24 items found (1 currently selected).

ID	Stack Rank	Story Points	Title	State	Iteration Path
26	1	5	As a customer I should be able to create a new account.	Resolved	TailspinToys\Iteration 1
27	2	3	As a customer I should be able to add items to my shopping cart.	Resolved	TailspinToys\Iteration 1
28	3	2	As a customer I should be able to log in to the site.	Resolved	TailspinToys\Iteration 1
29	4	1	As a customer I should be able to log out of the site.	Resolved	TailspinToys\Iteration 1
30	5	3	As a customer I should be able to view my shopping cart.	Resolved	TailspinToys\Iteration 1
31	6	6	As a customer I should be able to check out and pay for my items.	Resolved	TailspinToys\Iteration 1
32	7	2	As a customer I should be able to remove items from my shopping car cart.	Active	TailspinToys\Iteration 2
35	8	3	As a customer I should have to enter a strong password when creating a new acco...	Resolved	TailspinToys\Iteration 2
38	9	9	As a customer I should be able to see images for all items.	Active	TailspinToys\Iteration 2
42	10	6	As a store administrator I should be able to track all open orders.	Active	TailspinToys\Backlog
46	11	2	As a store administrator I should be able to examine contact details for a customer.	Active	TailspinToys\Backlog
49	12	2	As a store administrator I should be able to cancel or modify an open order.	Active	TailspinToys\Backlog
53	13	9	As a store administrator I should be able to review reports of open orders.	Active	TailspinToys\Backlog
54	14	7	As a store administrator I should be able to review reports of rebates owed.	Active	TailspinToys\Backlog
55	15	4	As a store administrator I should be able to help a customer reset their passwords.	Active	TailspinToys\Backlog
56	16	3	As a customer I should be able to track my rebates online.	Active	TailspinToys\Backlog

图：Product Backlog 查询结果

上图中就是使用 TFS 内置的 Product Backlog 查询获取的产品需求列表，这个列表是 PO 使用的主要工具，我们可以注意到这个列表已经根据 Stack Rank 列进行了排序，这也反映了产品需求列表的特性：需要根据客户/干系人对需求项的优先级向团队交付任务；而 PO 的除了需要不断完善这个列表，还需要不断和客户干系人进行沟通，一边确定这个优先级。

在 Scrum 模式中，对于优先级的定义决定于两个因素：需求的商业价值和紧迫程度；另外一个重要的指标就是 Story Point，这个指标标志着当前需求项的相对大小，注意这里说的相对大小，很多人将这个值理解为人天或者人时，其实是不准确的，因为在 PO 准备产品需求列表的过程中，仅凭 PO 的经验是很难准确的判断出以时间为度量的工作量的，但是相对的大小是比较容易判断的。

另外，从 State 和 Iteration Path 两个列的值我们可以看到，已经有一些需求在迭代 1-2 中已经解决。根据这些信息，PO 可以很容易的对工作进度和剩余需求进行管理。

另外一个重要的查询就是 Iteration Backlog 查询：

Iteration Backlog [Results]

Save Results | Refresh | Open in Microsoft Office | Edit Query | Column Options

Query Results: 10 items found (3 top level, 7 linked items, 1 currently selected).

ID	Work Ite...	Title	State	Assigned ...	Remainin...	Complete...	Story Poi...	Stack Rank
32	User Story	As a customer I should be able to remove items from my shop...	Active	Michael Af...			2	7
33	Task	Design UI for item removal	Active	April Stewa...	8			
34	Task	Bind cart removal function to back end	Active	Doris Krieg...	5			
35	User Story	As a customer I should have to enter a strong password when c...	Resolved	Michael Af...			3	8
36	Task	Write regular expression for strong password	Closed	Abu Obeid...	10			
37	Task	Add RegEx to account creation Web page	Closed	Abu Obeid...	6			
38	User Story	As a customer I should be able to see images for all items.	Active	Michael Af...			9	9
39	Task	Update database to point to actual images	Active	Doris Krieg...	12			
40	Task	Upload images for all products	Active	April Stewa...	6			
41	Task	Update web site to include new product images.	Active	April Stewa...	4			

图：Iteration Backlog 查询结果

Iteration Backlog 中包含了团队在某个迭代中需要完成的需求以及针对这些需求细化出来的具体开发/架构/测试等任务。在 Visual Studio 2010 中，微软终于开始支持树形结构的工作项关联，从上图可以看出，每一个 User Story 的下面都挂接着相应 Tasks，这些任务是在 Sprint Planning Meeting 中由团队成员自己根据 PO 对需求的阐述进行的细化，同时团队成员还需要根据经验对这些 Tasks 进行估算，给出基线估值（Original Estimate）。在开发过程中，团队成员在每天的 Daily Scrum 之前需要对前一天的任务更新状态（State），已完成工作量（Completed Work）和剩余工作量（Remaining Work）字段的内容；通过这些信息我们就可以使用 TFS 自带的燃尽图报表对进度进行查询和预测了。

实际上，纯粹的 Scrum 模式并不关心已完成工作量（Completed Work）也就是以完成工作量的值，但是对于使用人天/人时等信息来衡量团队工作量，甚至依赖这些数据想客户收取开发费用的咨询类公司来说，这些信息是非常重要的。

Visual Studio 2010 对 Scrum 流程中重要事件的支持

Scrum 模式中的几个重要的会议包括：

1. Sprint Planning Meeting
2. Daily Scrum Meeting
3. Sprint Review Meeting
4. Sprint Retrospective Meeting

这一系列的会议是真正体现 Scrum 模式对于开发流程控制的核心内容，在 Scrum 模式中另外一个非常重要的概念是：时间箱（Time Box），它要求我们对于流程中的事件进行非常严格的时间控制。很多人在开始进行 Scrum 模式开发的时候的一个普遍问题是：一个迭代（Sprint）的长度应该是多少？对于这个问题其实也没有标准答案，而必须根据团队的大小来进行判断。对于之前我所建议的 3-7 人大小的团队，我会建议采用 2 周的迭代长度。原因在于 1 周太短，团队还无法完成真正有商业价值并可以进行交付的需求；而 3 周的时间则太长，需求的变化所造成的风险会变得比较大。

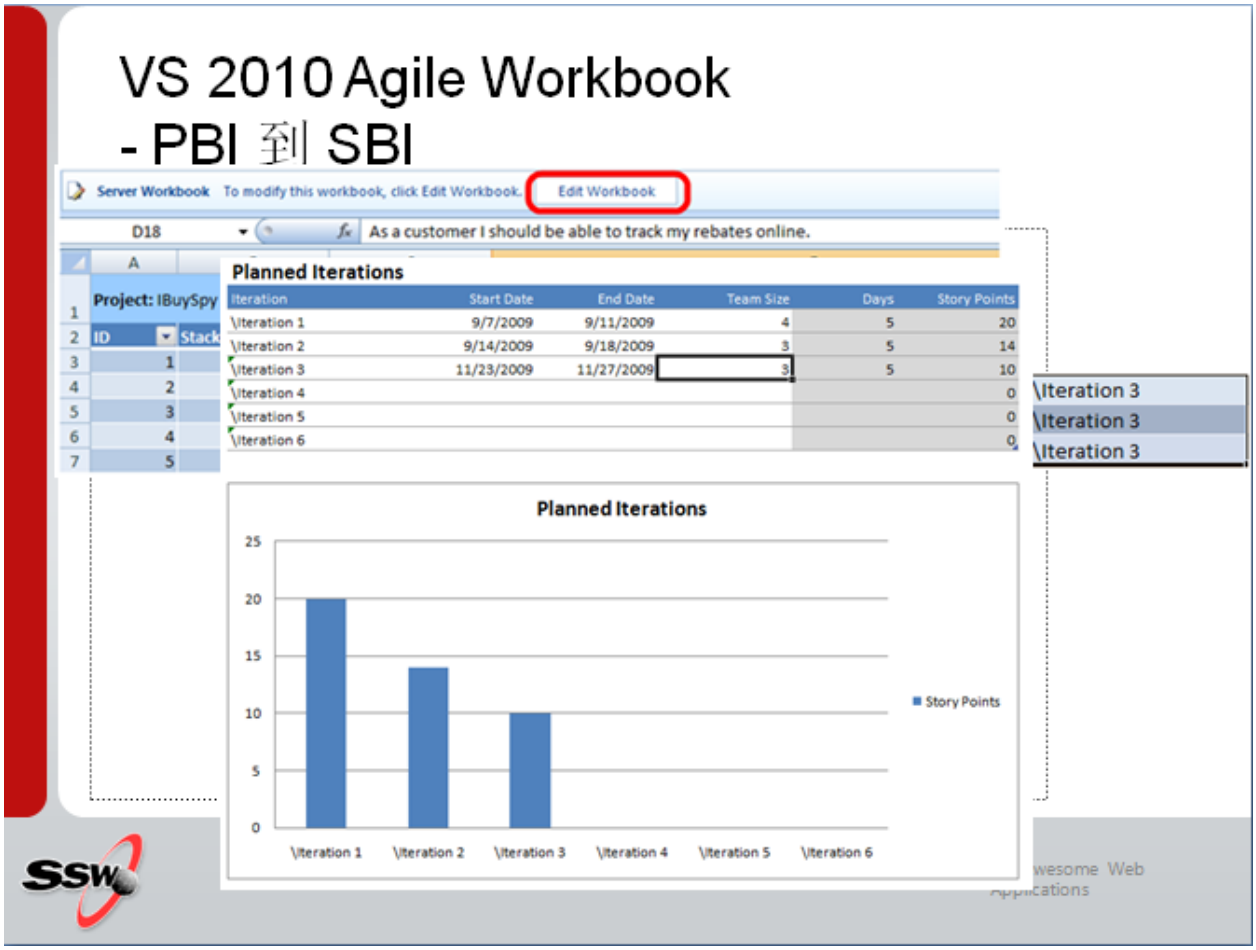
采用迭代式开发的时候其实长度是越短越好，我们总是尽可能的缩短迭代以便可以通过给客户的交付获得更有价值的反馈以便对后续的开发进行调整，因此这个长度应该是团队刚刚可以完成可交付需求的最短时间。我们需要严格控制的是，迭代的长度应该是一个时间概念儿不是工作量的概念，也就是说如果 2 周的时间已经耗尽但是团队还没有完成当前迭代中的所有需求，那么也必须结束迭代进行交付，而不能选择延长迭代来完成未尽需求。这样做的结果有两个：1）当前的迭代会以失败告终；2）通过对已经完成需求的交付，我们可以获取客户的反馈。很明显，失败的迭代是我们不愿意看到的，但是客户对于已经完成需求的反馈比保持常胜将军的名誉更加重要，因为后者是保证我们软件质量（符合需求）的重要手段。

当然，这里隐藏着另外一个很重要的问题，在团队无法完全完成需求的情况下如何还能提供可交付的成果，这就要依靠我们对于需求定义方式的变化和 Visual Studio 2010 中对持续集成和更加高效的测试支持来实现了。在需求定义上，我们需要采用业务导向的需求定义，保证每一个需求的完成都可以交付一定的商业价值。以往的需求往往是功能导向的，但是功能导向的需求对于用户来说不一定具备商业价值，但是业务导向的需求则可以保证这一点，比如：我们可以这样定义一个 User Story，作为市场经理，我希望对客户数据进行查询以便可以找到本市的客户并和他们进行联系。使用这样的需求定义意味着只要我们完成这一需求对客户就是有价值的，因为它不是一个功能碎片，而是一个用户交互用例。如果在一个迭代中我们无法完成所有的需求，只要完成其中一个，那么都是可以向客户交付的。另外，借助

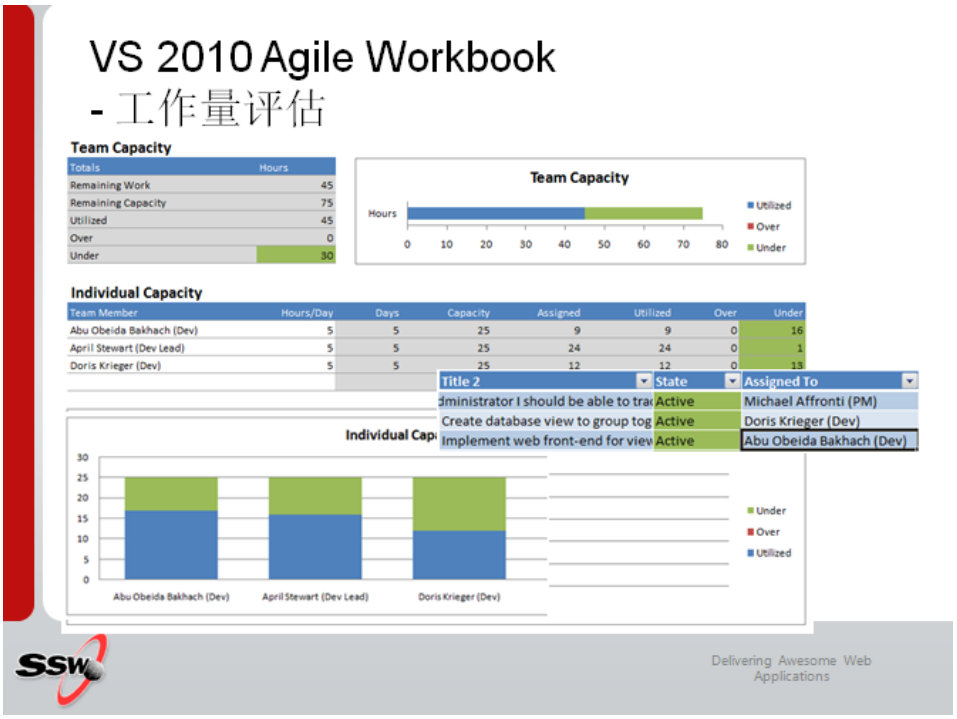
Visual Studio 2010 对持续集成和测试的支持，我们可以采用每日构建的方式保证所有完成的代码都符合质量要求，也就避免了在迭代后期进行集中测试而拖延交付的可能性。

Sprint Planning Meeting 的支持

在 Visual Studio 2010 中提供了一个叫 Agile Workbook 的 Excel 模板，可以帮助我们很好地完成 Sprint Planning Meeting。在这个会议中，最重要的任务就是将 PBI 转化成 SBI，并且由团队给出完成这些 SBI 的承诺；团队要做出这样的承诺最重要的依据就是这些需求所涉及的工作量是否可以承受。Agile Workbook 正是帮助我们回答这一问题的强大工具。从下图我们可以看到，当我们制定了迭代上的人员配备并将 Task 分配给每个开发人员以后，模板会给出非常直观的柱状图，帮助团队判断工作量是否可行。



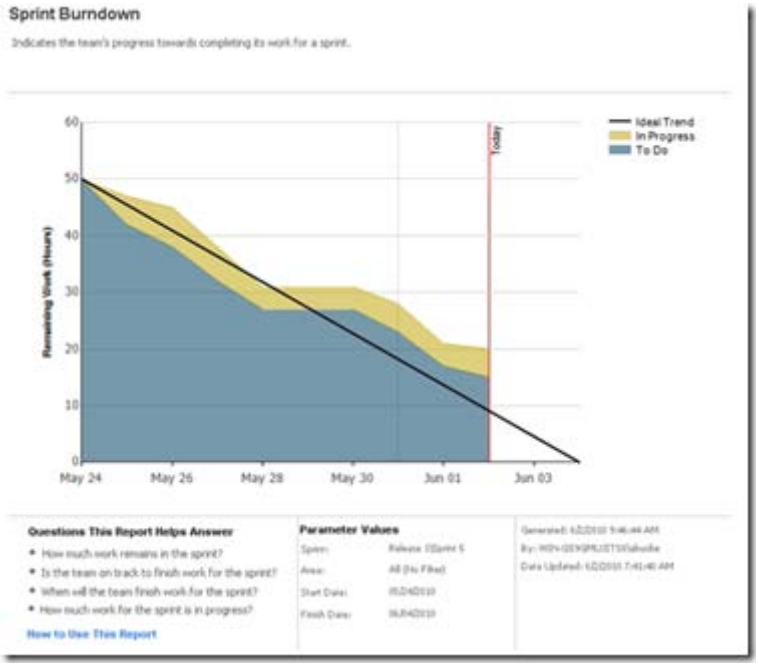
图：对迭代 1-3 上的工作量进行横向比较，根据历史数据判断后续迭代是否可行



图：在当前迭代上对每个开发人员的工作量分配进行比较

Daily Scrum Meeting 支持

这个会议非常简短，所以我们更加需要非常直观的图表以帮助团队对进度进行审核，在 TFS 中提供了燃尽图为团队提供这些信息。



图：迭代燃尽图

根据每个开发人员对于工作量的更新，从上图我们可以很容易对完成时间进行预测，图中黑色实线和横轴的焦点就是当前迭代的可能完成时间。

Sprint Review & Retrospective Meeting 的支持

Sprint Review 的支持更多地体现于 Visual Studio 2010 的持续集成能力，因为这个会议是对于需求完成情况的审核，如果我们能够保证需求是业务导向的并充分利用 Visual Studio 2010 的自动化构建和测试集成能力。那么我们就可以保证在这个会议上交付一定的商业价值。具体如何使用 Visual Studio 2010 来实现在后面做详细介绍。

Retrospective 会议其实非常简单，需要我们团队成员对当前迭代的运作进行总结，但为了使这些信息可以完整的保存以便后续使用，我们可以利用 TFS 提供的门户站点，定制一个 SharePoint 的列表分类的记录这些反馈以便团队查询。

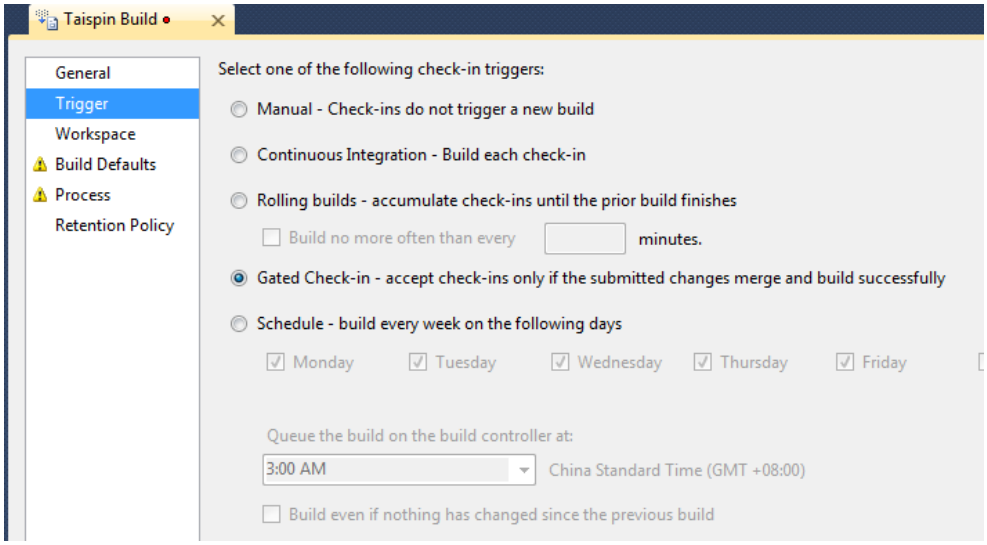
Visual Studio 2010 对于产品质量的保证

提高产品质量是 Visual Studio 2010 在设计阶段就确定的重要目标，在 2010 版本所添加的新特性中，已经想着这个目标形成了一套完整的解决方案。对于 Scrum 模式来说，交付高质量的产品也同样是其终极目标，而且我们需要在迭代时间很短的情况下仍然保证质量，这就更加需要依赖工具的支持。

自动化构建

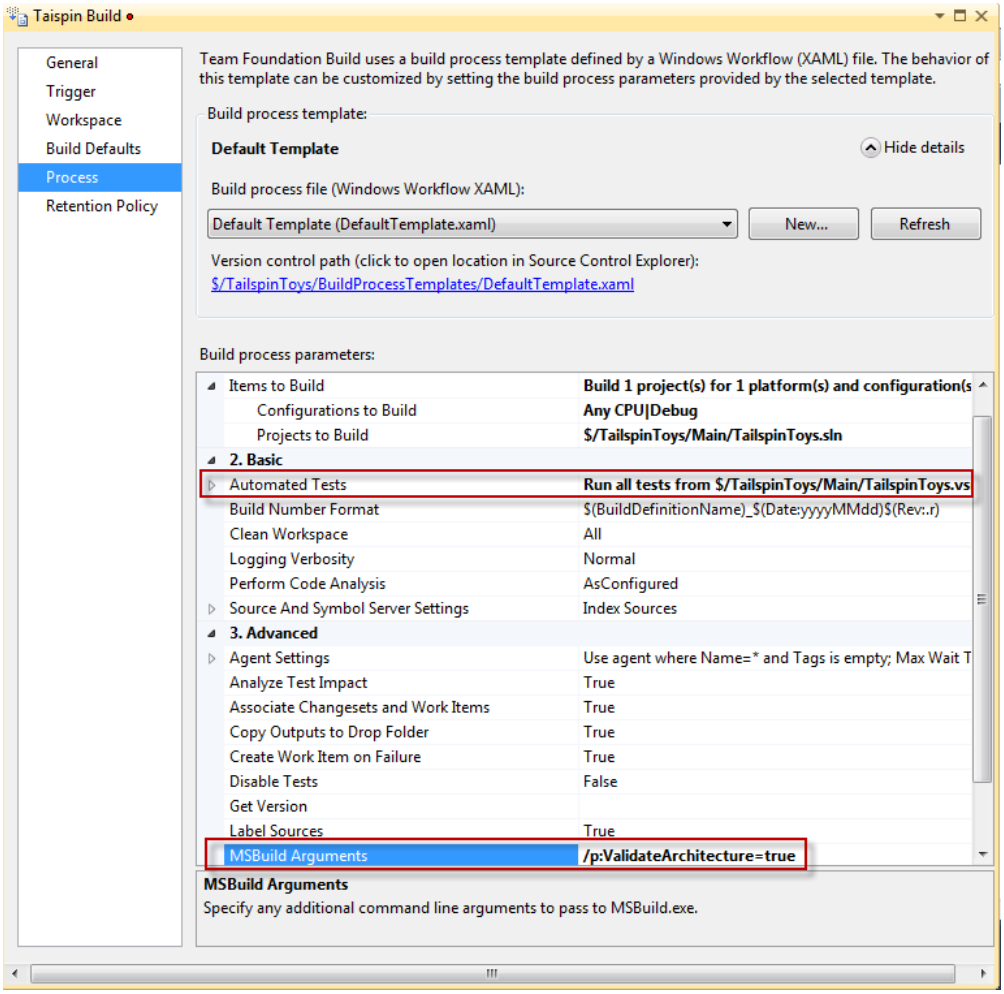
之所以把自动化构建列在首位，是因为软件工程发展到今天，自动化构建已经是任何一个想要实现高质量的软件开发团队都必须采用的工程方法；另外，对于 Visual Studio 2010 系统来说，自动化构建也起着承上启下，贯穿全局的重要地位。当开发软件进入第一个迭代的开发时，所要进行的第一项工作并不是开始实际的编码，而是创建出符合团队需求的构建模板。这样做的目的在于团队在后期的实际开发中可以更加专注于需求的开发，而不必花费额外的时间和精力来集成开发人员的代码；开始阶段的代码量很少，团队可以有更加清晰的思路将迁入策略，架构验证，自动化测试列表设置好并保证构建可以正常运行；如果把这个工作放到迭代后期进行，往往会因为代码中的缺陷和不同开发习惯造成构建模板不能正常运行。

在 Visual Studio 2010 中，提供了更加便捷的模板创建工具，特别是添加了 Gated Check-in 构建的触发方式，可以保证所有嵌入源代码库的代码都是经过验证的。



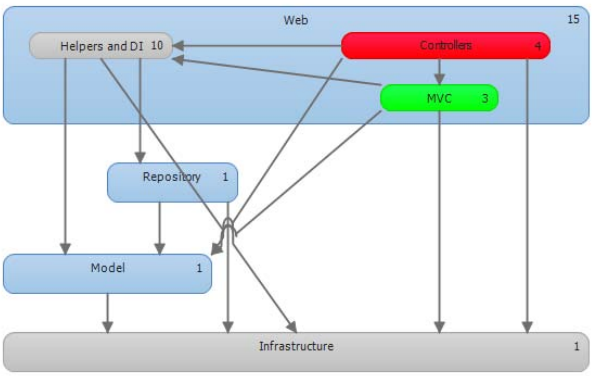
图：Gated Check-in 构建触发器

Gated Check-in 触发方式和以往的触发方式所不同之处在于，开发人员执行迁入操作的时候代码并不会直接进入源代码库，而必须先经过构建的验证：保证编译成功和定义好的迁入验证测试可以成功运行，然后 TFS 才会把代码真正嵌入服务器。之前的持续集成（Continuous Integration） 方式也会在迁入的时候进行构建，但是这种构建是将代码先迁入，然后再运行构建，如果代码中已经存在了缺陷，那么在服务器上就会留下缺陷代码；Gated Check-in 借助 TFS 源代码管理中的“搁置”功能，先把代码搁置到服务器上临时存储中，在构建成功后才会正式迁入，所以缺陷代码不会进入服务器。



图：构建参数配置

TFS 的自动化构建可以集成测试列表，图中的上方的红色区域中就是要求构建从项目文件中的测试列表文件中提取单元测试并自动运行；另外一个在 Visual Studio 2010 种的重要改进就是下方红色区域中的架构验证参数。如果我们的项目文件中包含了架构层次图（Layer Diagram）的话，那么我们就是添加这个参数让构建自动的验证项目的代码是否符合架构设计的要求。



图：Visual Studio 2010 的层次架构图 Layer Diagram

Scrum 模式开发中的架构设计给我们提出了非常大的挑战，由于我们采取业务导向的需求定义，开发人员必须从数据层一直实现到表现层；在这个过程中如何保证项目的架构仍然符合需求非常困难，而 Visual Studio 2010 的架构验证功能则可以帮助我们在每次迁入代码的时候都进行验证，保证违反架构规范的代码不会进入最终的交付产品。

消除无法重现的 Bug

无法重现的 Bug 一直都是困扰开发人员的问题，开发环境，测试环境，生产环境的不同；开发人员，测试人员和最终用户的不同都是造成 Bug 无法被重现的客观因素。在 Visual Studio 2010 中，提供了很多强大的调试和测试工具来帮助我们解决这个问题。

- IntelliTrace（历史数据调试）
- 协作调试
- 测试管理器和手工测试（Test Manager）
- 实验室管理（Lab Manager）

IntelliTrace——历史数据调试器

IntelliTrace 在开发过程中的名称就叫 Historical Debugger （历史数据调试器），后来这个用来进行市场宣传的名称反而不能反映它的实质。IntelliTrace 可以把程序运行过程中的所有历史数据都记录下来，使得程序员可以回滚到任何的历史点来查看程序状态，这对于开发人员调试复杂逻辑非常有用；之前我们在做同样工作的时候必须反复运行程序，以便找到问题，而现在则可以让程序反向运行。

图：IntelliTrace 调试器重所记录的程序历史数据

另外，IntelliTrace 还可以把这些调试数据另存为 tdlog 文件；当开发人员 A 发现了 B 的一个问题的时候，他可以把自己调试环境中的 tdlog 发送给 B，开发人员 B 就可以使用这个文件让 Visual Studio 恢复到开发人员 A 的调试状态，从而保证 B 可以有效的重现 A 所看到的问题。

协作调试

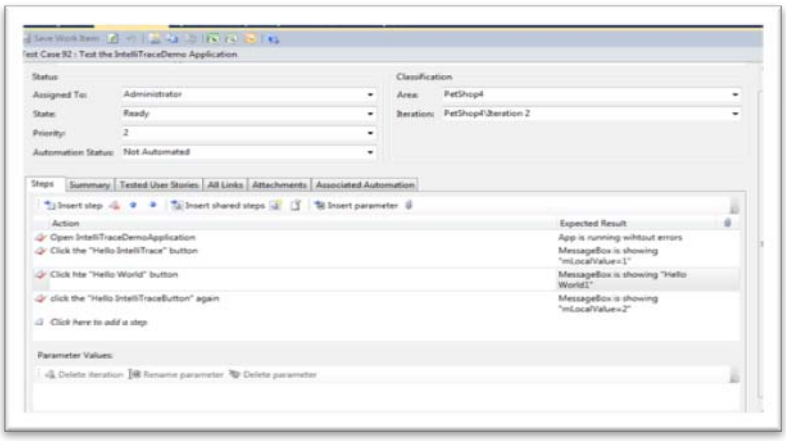
协作调试实际上解决多个开发人员在调试过程中的另外一些信息共享问题的方法，上面的 IntelliTrace 可以共享调试历史数据；但是用过 Visual Studio 的开发人员都知道，像“断点”是不能保存到调试数据中，也不会被保存到项目文件中；所以协作调试就提供了开发人员共享断点信息，并且还可以让开发人员在断点信息上添加一些说明，以便帮助其他的开发人员理解问题。

测试管理器和手工测试（Test Manager）

测试管理器是 Visual Studio 2010 系统中为测试人员特意开发的可以独立运行的测试环境，它完全独立，不依赖于 Visual Studio IDE，提供非常强大的测试录制等功能。在前面介绍构建的时候我曾经将单元测试集成到构建中去自动运行，但是单元测试只能针对后台逻辑进行，不能解决 UI 测试，或者叫黑盒测试问题。微软的测试管理器的出现，就是为解决 UI 测试的

问题。

TFS 2010 中专门提供测试用例（Test Case）工作项类型，这个工作项允许测试人员对具体的测试步骤进行设计，并且给出预测的结果；同时，借助测试管理器的录制功能，还可以把测试人员换的操作全部都录制下来，一边后来自动播放；或者生成 Coded UI 测试，一旦有了 Coded UI 测试，我们就可以把这些针对 UI 的测试也集成到自动化构建中去。



图：测试用例（Test Case）工作项

实际上，真正可以使用单元测试覆盖的测试仅占所有的测试的 30%都不到，另外这 70%的测试以往都是依赖于测试人员手工的进行；现在借助微软测试管理器的功能，我们可以将这些测试集成到高度自动化的开发流程中。可以帮助我们更加快捷的完成测试，为开发人员提供反馈。

在 Scrum 模式中，业务导向的需求也要求我们的测试团队可以更加快捷的给出测试结果，前一天完成的需求最好可以在第二天就将测试结果反馈给团队；依赖于每日构建，我们可以在每天晚上将前一天的代码生成一个新版本，共测试团队使用；测试团队在第二天就可以把测试结果反馈给开发团队，同时将可以自动化运行的测试继承到每日构建中；在第三天的时候我们的团队就可以利用这些已经自动化的测试来验证我们的程序了。

由于每天都进行测试，那么新增的代码量就非常有限，也就使得 Bug 的数量可以得到有效的控制，从这个方面上说，测试管理器所提供的手工测试，自动化测试录制和回放，并且和构建的继承为我们提供了一个非常高效的高质量的开发平台，从流程和工程技术上为质量提供了保证。

实验室管理（Lab Manager）

实验室管理是我在 Visual Studio 2010 系统中见过的最酷的功能，也是微软继承了自己的多项产品为开发团队提供的最完整的测试解决方案。在测试中一个非常难实现的问题，就是对于

不同环境的创建，还原和状态的保存。如果同一个用例在不同的环境中运行，结果往往是不同的，而且我们客户的使用环境也往往很复杂，所以就要求我们的测试人员可以搭建很多不同配置的测试环境，以便验证应用程序可以适应他们要求。

微软借助自己的 Hyper-V 虚拟化平台，为测试团队搭建这样的测试环境提供了非常好的支持，比如：我们可以使用 SCVMM 和 TFS 协同工作，当 TFS 需要测试环境的时候，通过 SCVMM 部署一台符合要求的虚拟机，并把需要测试应用自动的部署到这个虚拟机中，最终在这个环境中运行指定的测试。这样的测试环境避免了测试人员自己的机器不干净而导致的结果偏差，而且还可以通过环境快照的方式吧虚拟机的某个状态直接交付给开发人员进行检查。

在上面所介绍的这些功能中我们可以看到，实际上我们解决了 3 个不同测试的不可重现问题：

1. 开发人员本机上的不可重现：IntelliTrace
2. 开发人员和开发人员之间的不可重现：IntelliTrace, tdlog 和协作调试
3. 开发和测试环境之间的不可重现：微软测试和实验室管理器，Hyper-V

这些功能在工程技术上为团队保证了高质量，同时配合 Scrum 模式所推行的时间箱管理，业务导向的需求定义以及流程上的保证，Visual Studio 2010 系统和 Scrum 一起帮助我们创建更好的产品和更好的团队。

结束

我使用 Visual Studio Team System 是从 2005 年开始的，最初的目的只是为了满足远程迁入代码的需要；但随着 2008 和 2010 版本的发布，对于流程定制和整体性的质量解决方案的需求越高。幸运的是，这个时候公司为我提供了到澳大利亚接受 Scrum Master 培训的机会，使我可以体系化的了解了 Scrum 模式的精髓，回来之后就对我们的开发团队进行了一系列的优化。

同时，作为 Scrum Master 我也同时获得了提供 Professional Scrum Developer 培训的机会，PSD 课程是微软和 scrum.org 共同开发的一套基于实践的 scrum 开发人员培训课程，它使用 Visual Studio 2010 系统作为平台，将参训人员分为不同的团队，进行实际的开发工作，在开发的过程中让学员体会 Scrum 的妙处和 Visual studio 2010 的强大。目前我们已经在澳大利亚墨尔本和意大利米兰成功运作了这个课程。作为在亚洲去唯一向中国提供这一课程的提供商，我也希望能够和更多的开发人员分享这些内容。

更多内容请见：

我的博客：www.almnetworks.net

我的公司主页：www.ssw.com.au

个人介绍：http://sharepoint.ssw.com.au/AboutUs/Employees/Pages/Lei.aspx

PSD 课程中文介绍：http://www.ssw.com.au/ssw/Events/Scrum-Training-Course-China.aspx

特别专题

应用 Visual Studio 2010 辅助敏捷测试

作者 周京生

敏捷软件开发是近些年来比较热门的话题，《敏捷宣言》四条主要精神和十二条基本准则概括了敏捷开发的基本思想。围绕着这些基本概念和思想，产生了一系列的轻量级方法，如：极限编程、测试驱动开发、Scrum、特性驱动开发等。虽然具体名称、过程和侧重点不尽相同，但是相对于非敏捷的开发方法而言，它们都更强调面对面的沟通、团队不同角色之间的紧密协作、频繁交付新的可用的软件版本、紧凑而自我组织型的团队等。敏捷开发只是提供了一个思想和方法论,而要在实际的工程中正确运用它，并真正显现出它的优点和产生实际的效果，这对于每个团队而言一开始都是一个挑战，尤其是对那些那些习惯了传统瀑布模式的团队。

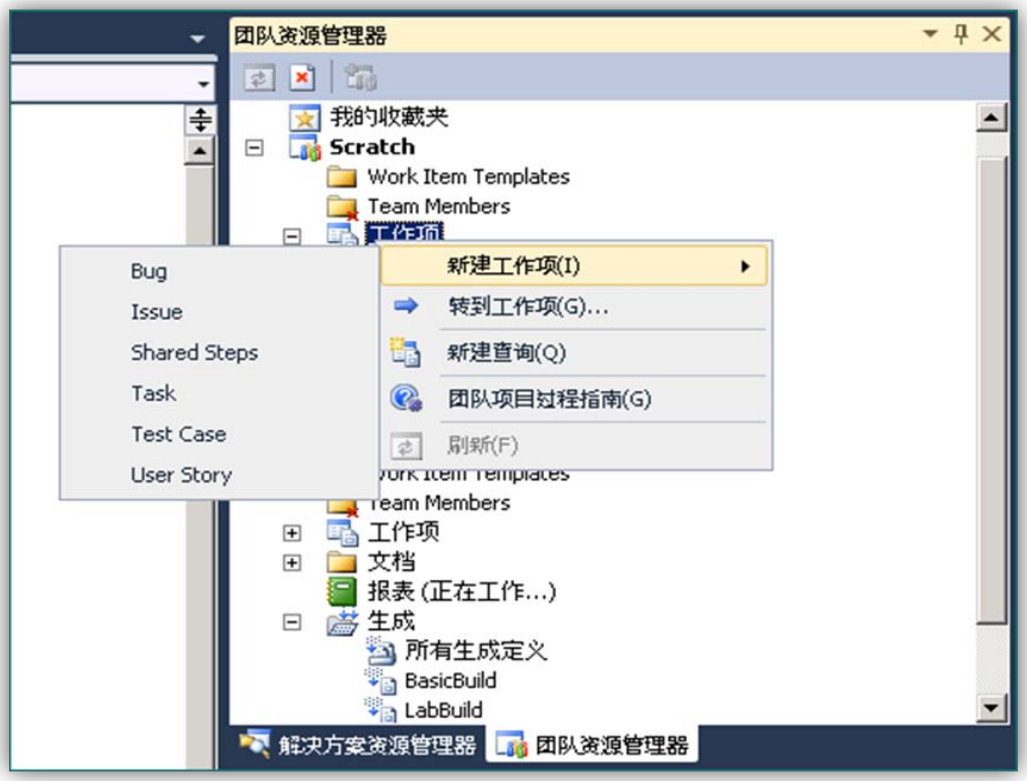
敏捷是整个团队的敏捷，不只是团队中某个角色或者某个阶段的敏捷，开发、测试和项目经理等所有角色都要敏捷起来。敏捷方法的采用对团队每个成员都提出了新的挑战，尤其是测试人员。之所以这样说，是因为相对于传统的瀑布模型，敏捷开发所要求的频繁交付，给测试所留出的时间更为紧迫，要求测试人员更早的介入和更及时地完成测试任务。如何在这么短的时间内完成测试的计划和实施呢？如何有效地避免回归问题的出现？手工测试人员如何能更好的融入到敏捷团队？等等问题接踵而至，这都需要需要测试人员不断的思考和尝试。

无论是哪种开发模式，软件的开发过程都可以归结为：人、工具和过程这三个因素，三者的有机结合才能更高效的完成任务。有人会说：《敏捷宣言》四条主旨精神的第一条就是“个体和交互重于过程和工具”，工具还有那么重要吗？回答是肯定的，工具很重要，这条主旨所提到的是“重于”而不是不要。为了支持敏捷开发，Visual Studio 2010（以下简称为 VS 2010）应用程序生命周期管理中引入了 MSF for Agile Software Development v5.0 过程模板，用于辅助敏捷团队在实际工程中进行敏捷实践，它支持 Scrum 敏捷开发过程框架。本文将从工具角度出发， 介绍 Visual Studio 2010 如何帮助测试人员更胜任敏捷项目中的测试工作。对于工具与人的关系而言，好的工具应该是将人从重复和机械的劳动中解脱出来，让人有更多的精力和时间花在有创造性地劳动上，而由工具去完成将繁琐和冗余的事务性操作；而对于工具

和过程的关系，工具是过程能够得到确实落实和准确执行的基石，很多时候我们总是依赖于人去执行某个过程或者流程要求，但人的执行往往带有一定不稳定性和主观性，而工具则可以帮助我们准确客观的执行。

团队有效协作的基石——Team Foundation Server

敏捷开发强调人与人之间的有效沟通和紧密的团队协作。对于测试团队和测试人员而言，首先应该需要考虑的是 :如何让测试工作更有效的集成整个敏捷开发的活动中去？而不是将测试工作仅作为一个“附件”或者可有可无的副产品。当然，这会受到团队组织形式和开发过程的限制，例如：采用功能小组模型的团队，所有角色成员（PM、开发人员、测试人员）隶属于同一个功能团队，客观上其沟通就更为方便；而对于采用纵向按职能划分团队的公司而言，测试和开发在隶属关系上是分开的，相对在沟通上障碍就会更多些。无论是哪种组织形式，好的工具能帮助促进和统一各个角色间的信息互通和共享，而不是要让他们彼此之间更为孤立、工作在各自的一亩三分地（Silo）中。Team Foundation Server 2010（以下简称为 TFS 2010）就是这样的工具，作为整个团队协作的核心，它统一了团队不同角色信息、实现了信息之间的有效互联互通、彼此之间的共享和关联，例如：TFS 2010 定义 6 种默认的工作项类型，如下图所示。



其中，Test Case 和 Shared Steps 是 2010 专门为测试新加入的。不要小看这些工作项，它们之间有着丰富的关联关系，这种关系背后所代表是角色之间的关系。对于测试而言，它将测试和团队紧密的结合在一起。例如：Test Case 工作项用来详细定义和管理测试用例，它还可以和 User Story 相关联，也就是将测试和用户需求进行了关联，用户可以从需求追溯到覆盖的它的测试用例，这背后体现的是测试人员和需求人员/PM 的协作；Test Case 还可以与 Bug 关联，通过这种关联可以挖掘出哪些 Bug 被测试用例覆盖，哪些还没有，这种关联体现了测试人员与开发人员的写作，如果是自动化测试用例，则体现了手工测试人员和自动化工程师的协作 ；Bug 还可以可以和签入集（Change-set）关联，可以找到为了修复 Bug，开发人员修改过哪些产品代码，这体现了测试人员和开发的关联。

敏捷开发频繁的迭代和较短的迭代周期，对项目管理的精确性、透明性和可见性都提出了更高的要求， 尤其对于那些项目复杂和人员较多的团队。Task 是另一个重要的工作项类型，它用于管理开发过过程中的所有任务项，包括：开发、测试以及需求等任务，统一管理开发中的所有任务，统一计算项目的开销和剩余工作量等。例如， 项目的燃尽图就是由它产生出来的。现在，人们虽然在理论和概念上已经非常认同软件测试的在工程中的重要地位，但在具体实际操作中，测试却仍然被看作是低于开发和需求分析等的“二等公民”。当然这是由于多方面的综合因素造成的，从管理技术角度讲，这是由于测试工作本身缺乏可度量性和可见性，从导致了测试工作的透明性的缺失，团队往往看不到测试工作的进度和所带来的成果，从而意识不到测试的真正作用。对于测试人员自身而言，缺乏可度量性也让自己无法对工作进度准确把握，进而失去了对自己工作的目标感和认同感。将测试工作同其他工作一样的用 Task 工作项管理起来，增加了它的可度量性和可见性。将测试工作和其它任务一起统筹，时刻确保测试被作为整体中的一部分进行考虑，所有的测试任务都被作为 Task 工作项记录下来，例如：编写测试计划、设计测试用例、自动化测试用例等等，每项任务都有三个默认时间估计数据需要填写，它们是：Original Estimate、Remaining 和 Completed，分别代表了任务的预估时间、剩余工作量和完成工作量。

为了增强敏捷过程的透明性和可见性，TFS 2010 定义了很多的报表和仪表板（Dashboard），它们会自动生成各种报表，以可见的方式描述敏捷项目的健康状况，这其中就有很多反映测试工作的报表，如下面所示。Stories Overview 展示了用户故事的进展情况，包括了每个用户故事的测试用例覆盖数量和执行结果，以及相应的 Bug 数量；Test Dashboard 显示了测试用例的状态，包括正在设计的用例以及设计完毕可以执行的用例数量，现实当前 Bug 的状况，包括未被修复和以修复 Bug 的数量。

Stories Overview

Helps you track how far each user story has been implemented. Shows each story's actual number of hours of work remaining and completed, its acceptance test results, and the number of bugs that are linked to each story.

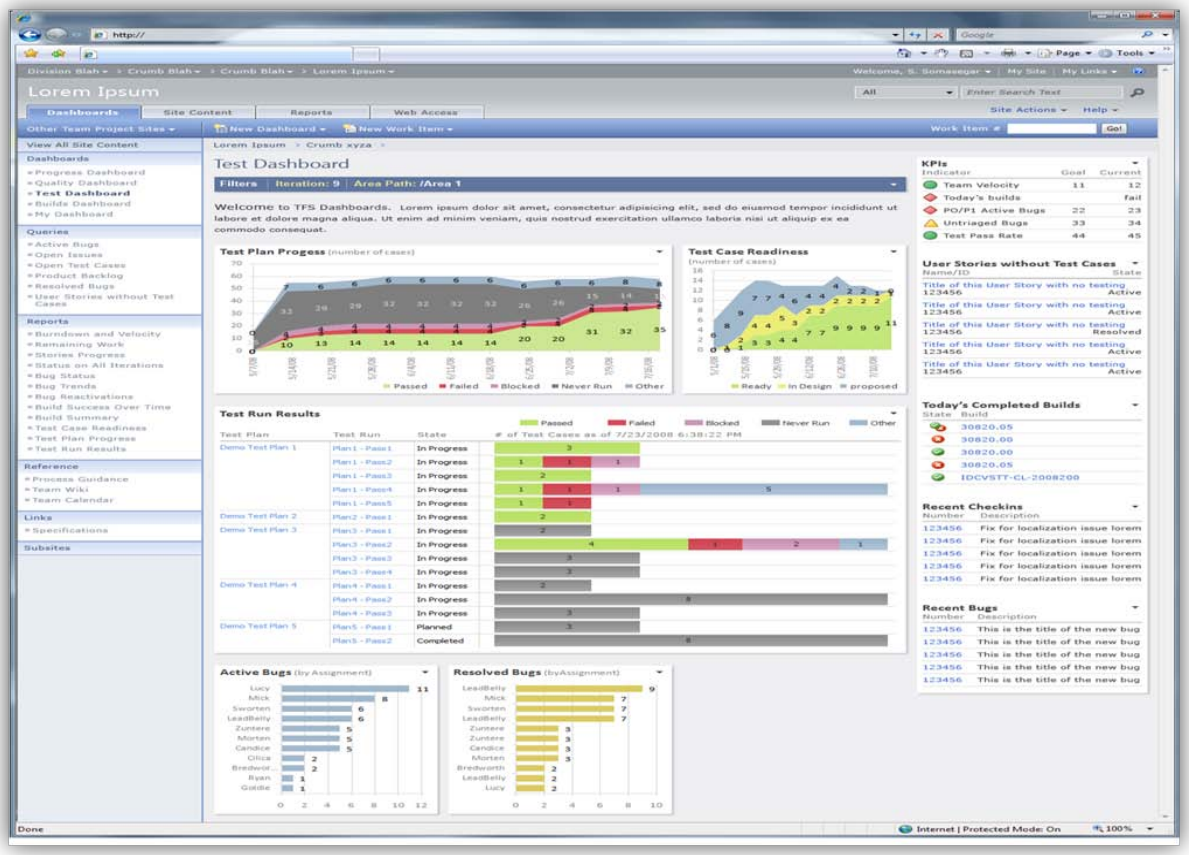
Related Reports

- Bug Status
- Status on All Iterations
- Stories Progress
- Test Case Readiness
- Test Plan Progress

Completed / Remaining

Passed / Failed / Not Run / Active / Resolved

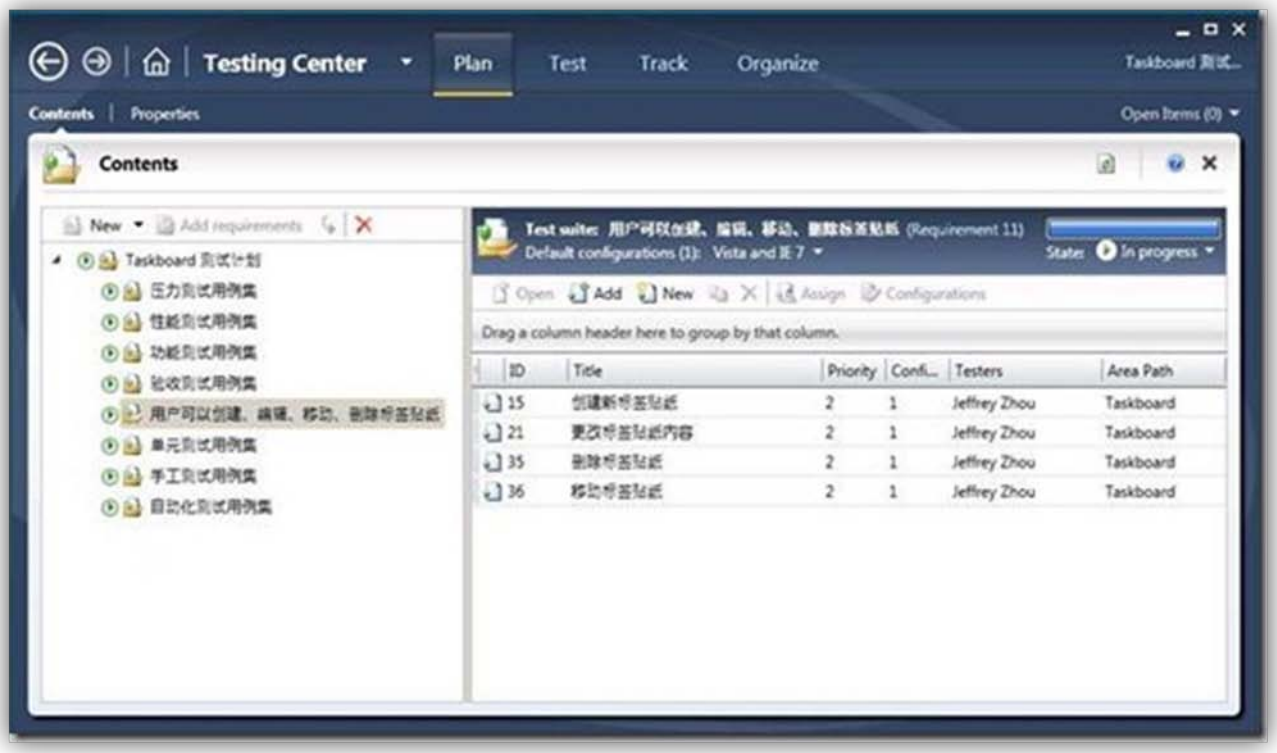
Title	Work Progress: % Hours Completed	Work Progress: Hours Remaining	Test Status: Tests	Test Status: Test Results	Test Status: Bugs
As a new customer, I want to order a meal.	80 %	6634	3	33 % 55 %	1 2
As a customer, I want to track my order history.	79 %	14053	0		6
Enable selection based on strength, intelligence, etc	19 %	144	2	48 % 52 %	1 2
As a returning customer, I want to order one of the meals that I've recently ordered.	78 %	17	2	48 % 52 %	
As a new customer, I want to choose a meal from a specific provider.	80 %	42	0		
As a customer, I want to save orders.	80 %	9432	0		
As an event planner, I want to let participants in my event choose meals from DinnerNow.	17 %	298	2	33 % 53 %	
As an event planner, I want to filter the menu to meet my constraints so that I can control the cost of the meals or so that I can offer only meals that are appropriate for the event.	40 %	126	0		
Gold member can search for villians	79 %	724	2	33 % 53 %	1
As a returning customer, I want to be able to override my default location so that I can order from DinnerNow when I'm on the road.	35 %	110	0		
As a delivery provider, I want orders to be submitted to my business at least 45 minutes before we pick the order up from the provider so that we can optimize the delivery.		28	0		
As a delivery provider, I want to provide a premium just-in-time service so that customers can decide at the last minute to order from DinnerNow.		0	0		
As a delivery provider, I want DinnerNow orders submitted to my dispatch system so that the cost of handling DinnerNow orders is minimized.		0	0		



集成测试环境——Microsoft Test Manager

在过去的十几年中，为了适应了软件项目的复杂度和规模的不断膨胀，软件开发工具和框架得到了长足的发展，而测试工具则始终是块短板 ，特别是对于那些需要手工完成的测试任务而言，进展就更为缓慢 ，例如：现在很多团队仍然使用 Word 或者 Excel 这样“原始”工具来管理测试用例 。通过对业界的调查和分析，我们发现 70%的软件测试工作仍然是通过手工或者简单的脚本来完成的 ,在测试团队中不具备编程能力和仅有基本脚本编写能力的测试人员仍然是测试的主力。

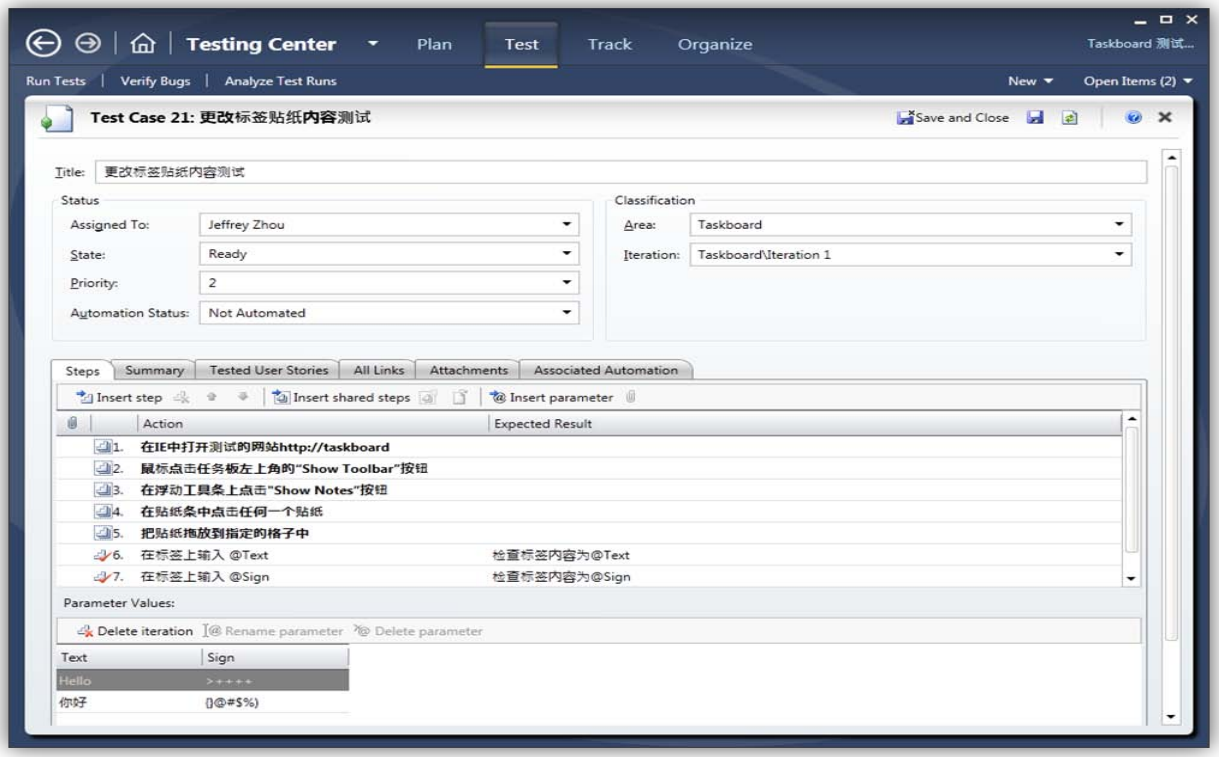
要让你的项目敏捷起来，对于那些仍以手工测试为主的团队而言是一个非常大的挑战，如何提高手工测试工作的效率将是实现敏捷的成败关键。在 VS 2010 中，微软首次为测试人员设计了一款专用的集成测试环境，称为微软测试管理器 2010（Microsoft Test Manager 2010，以下简称为 MTM）。之所以称之为集成测试环境，是因为 MTM 的功能涵盖了测试计划、测试用例、手动测试用例的执行和录制回放、自动测试用例执行、创建信息丰富的 Bug、验证 Bug、以及与测试实验室管理相关的对策是自动化相关的功能等。下图展示的是 MTM 测试计划的操作界面，它以树形的层次结构来组织测试用例。



《敏捷序言》强调：“可工作的软件重于完备的文档”，那么是不是意味着敏捷测试也不需要测试计划呢？当然不是。敏捷的本质是要去除软件过程所有造成时间浪费地方，不需要的是

那些动辄就几十或上百页的文档。敏捷对文档要求是要简明扼要，一两页列出测试要点计划还是必须的，较短迭代周期（1-4 周）也不可能要求文档面面俱到。敏捷需要更快的对功能进行验证，是不是不需要编写测试用例直接根据用户故事或者功能需求进行探索性测试就可以了？当然也不是。功能需求和用户故事勾画出的是一棵大树躯干和主要枝杈，而那测试用例则不但要准确描述出躯干和主枝，还要描述出细小的枝杈和绿叶的正确位置。从某种意义上讲，测试用例在敏捷中的作用和地位应该更为加强，它扮演着详细功能文档的角色。功能需求和设计文档可以简单，但测试用例可不是这样，相反我认为敏捷对测试用例的设计和管理要求更高。

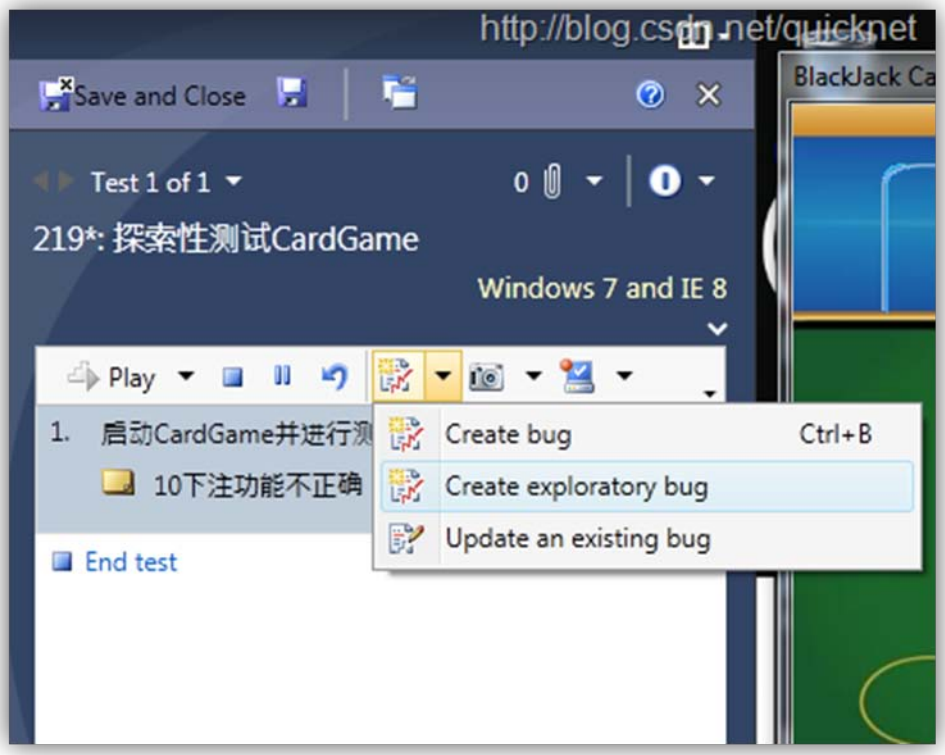
每个迭代周期，团队都会专注于实现不同的产品功能，用户故事虽然描述了功能的内容，但并不足以覆盖所有相关的内容。很多由用户故事展开和关联的功能一般在文档中会体现出来，需要测试人员在早期围绕着用户故事测试展开需求文档测试（需求评审），已明确那些未严格定义出来的内容，以测试用例的形式明确和记录下来。由 1 个简单用户故事就有可能扩展为 1+N 用户可能执行的执行片段，也就我们测试用例。当你有 M 个用故事，需要M个迭代周期来完成产品，那么就会有（M + N1 + N2 … + NM）个测试用例，不把它们落实到笔头上，很容易就会丢失一些重要的测试细节。此外，在敏捷方法中需求变化比较快，随着多个迭代的深入，文档的变化往往赶不上产品功能的变化，这时唯一能够赶上这个变化的只有测试用例，应为只有它准确地反映了产品的变化，否则测试用例就是无法通过的。



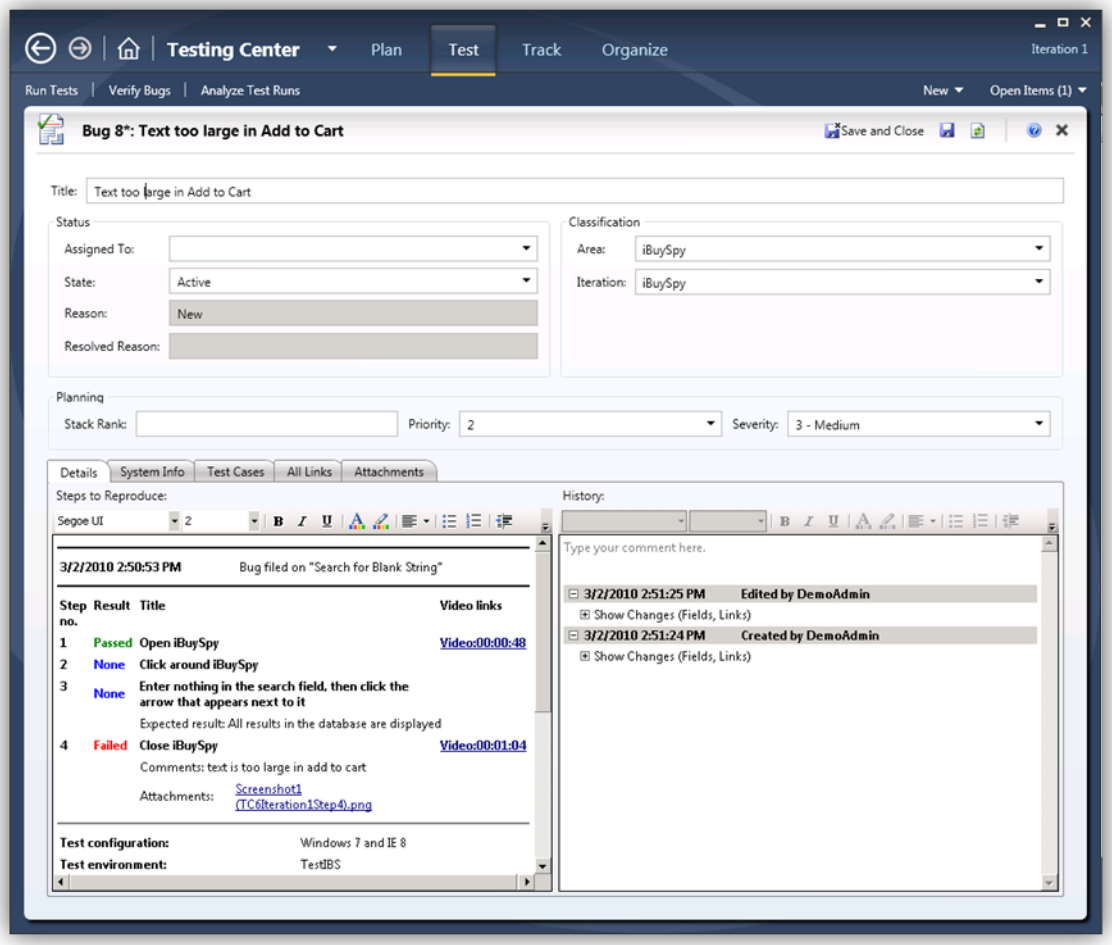
在 MTM 中，测试用例被分类至各个测试用例集，结构十分清晰。测试用例只是逻辑上从属于某个测试用例集，并没有物理从属关系，即一个测试用例可以同时被分在多个测试用例集内，比如某个测试用例性质上是一个性能测试，但是由于该用户故事的诉求就是性能改进，我们也就很自然得可以将其作为该用户故事的验收测试，此时我们就可以将此测试用例添加到验收测试和性能测试两个测试用例集中；另一个例子是给每个用户故事都定义了不少测试，这些测试用例都应该能在用户故事测试用例集下找到，但是这些测试既可能是手动测试也可能是自动化测试用例，所以它们又会被本别归类至这两个测试用例集。在这种逻辑分类的支持下，我们可以很容易的根据需要指定运行测试集中一部分测试用例。比如，我们可以定义一个签入测试的测试用例集，挑选最基本的若干个测试置入其中，这样在每次签入前通过运行这个测试用例集就能帮助我们确保签入的代码不至于破坏最基本的功能，即保证了版本随时可运行可测试，这无疑为测试带来了更多的方便。具体如何创建测试用例集的结构，团队可以根据自己项目的特点，灵活运用此功能，制定分类规则以提高工作的效率。

很多测试团队仍然在使用 Word 或者是 Excel 管理测试用例，有些是使用专门的测试用例管理工具，使用独立的数据库来存储测试用例信息。MTM 相对于这些工具的优点在于，它的所有数据都是存储在 TFS 上，测试用例使用的是 Test Case 工作项。由于同存储在 TFS 上，所以可以轻松的实现与其它数据项的关联，例如：在上一部分我们介绍的不同类型工作项之间关联，此外还可以把 Test Case 与代码关联，即将测试用例与自动化测试代码关联。这样在 MTM 中，也可以直接管理和运行自动化测试用例，使 MTM 兼具了管理手工测试用例和自动化测试用例的能力。

探索性测试（Exploratory Testing）是测试人员在对被测试系统的功能进行不断了解和学习的过程中进行测试，包括：设计测试用例、执行测试、以及汇报测试结果。与传统的测试相比，它不需要事先定义好的齐备的测试文档，更强调测试人员在对系统不断地学习中，边了解边测试，它在很大程度上给测试人员更多地自由和想象空间，充分发挥他们的创造力，在不断地学习中找到测试的灵感和快乐。这种测试的灵感和快乐对于组建和培养一支热爱测试的团队是非常非常重要，它会让测试人员觉得自己不是执行重复测试劳动机器，而是一个有着创造力和灵光的团队成员。MTM 也支持探索测试功能，用户可以使用 MTM 创建一个仅有一个测试步骤的测试用例，然后执行它，Test Runner 工具会辅助执行手动测试。它会记录下所有用户的操作，一旦发现有 Bug 时候，可以直接选择‘Create exploratory bug’直接创建一个 Bug。



Bug 是测试工作最重要的产出之一，也是测试和开发人员之间重要接触点。每个提交的 Bug 都应该详细记录下如何重现（reproduce）的步骤，这是衡量 Bug 质量的重要因素之一。因为不可重现的 Bug 是没有意义的，只会耽误开发人员和项目经理的时间。偶尔出现不可重现的 Bug 还是可以理解的，但如果经常出现，那就会引来开发人员的抱怨和不满，久而久之会造成开发和测试之间的不信任。 好的 Bug 应该是有清晰和详细的重现步骤，期望的结果和实际得到结果，并提供尽可能多的信息，例如：出现问题的产品版本编号、语言、操作系统的版本以及日志信息等。大多数情况下，用文字进行描述的内容就可以了，但如果能配上一张问题现场截图，或者对于更为复杂的 Bug，配上一段小的录像，这样的 Bug 会给开发人员快速诊断和修复产品问题带来很大帮助，大大提升测试和开发人员之间的协作效率，避免了不可重现 Bug 在开发和测试之间推来推去的“Bug 乒乓”现象。然而要收工创建这样一个信息丰富的 Bug，是需要很多时间的。MTM 提供了这样的功能为帮助测试人员创建这样高质量 Bug，它实现了多种诊断数据适配器（Diagnostic Data Adapters），在测试确认 Bug 的过程中，这些适配器会在后台运行收集大量的数据，包括：执行操作、系统配置、IntelliTrace 已经操作过程的录像等，当测试人员要创建一个 Bug 时，这些信息会被自动添加的 Bug 中，如下图所示，测试仅需填写很少的内容就可以创建好一个信息丰富的 Bug。



实现自动化测试用例——自动化测试用例框架

随着需求的不断变化和迭代的深入，代码库不可避免的会有频繁的签入和签出，此时测试人员一项很重要的任务就是要预防回归问题发生。执行手工测试用例可以帮助我们预防及和发现回归问题，但是它的执行效率太低，无法胜任频繁执行的要求。这时就我们需要考虑采用自动化测试用例完成这项工作。决定是否采用自动化测试是有很多因素决定，其中很重要的一条就是自动测试的收益，下面的公式从概念上解释了如何来计算这个收益，当收益值大于 1 的时候，实施自动化测试就是合算的；否则，就是不合算的。

$$\text{自动测试的收益} = \frac{\text{执行次数} \times \text{执行效益}}{\text{自动测试开发成本} + \text{自动测试维护成本}}$$

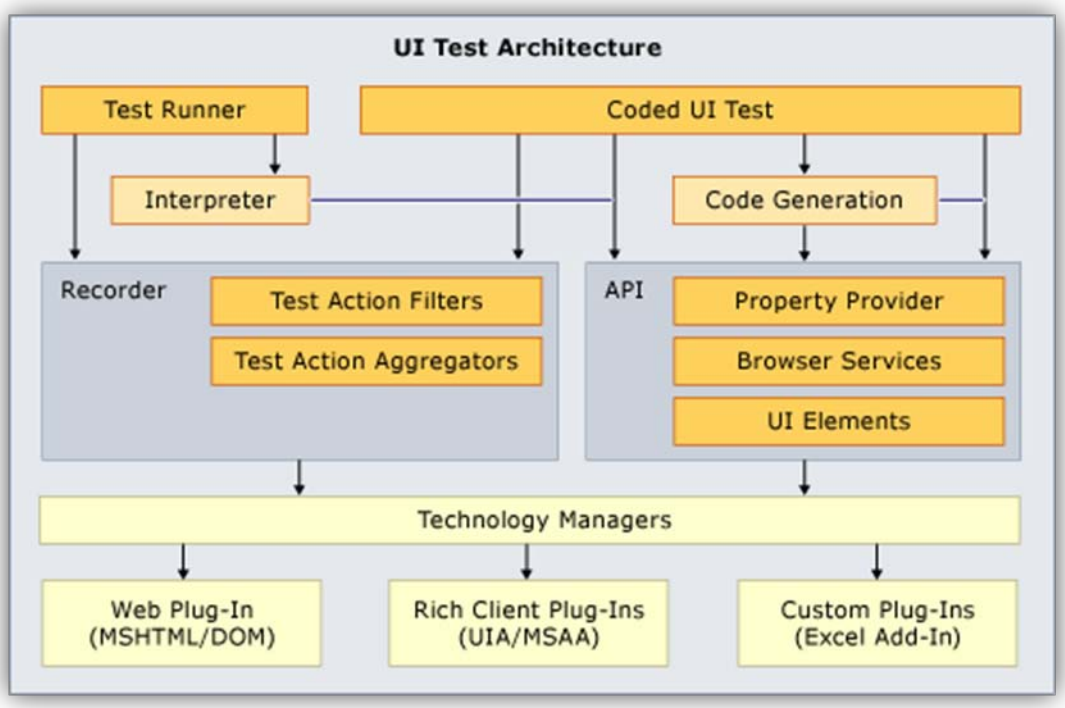
这其中，开发和维护自动测试的成本是影响这个收益的重要因素，为此 VS 2010 提供了一整套的解决方案，帮助测试团队减少这部分成本，这包括前面我们所提到的测试计划和用力管理工具，以及后面将会要介绍的生成和实验室管理。此外，Visual Studio 提供了多种测试工

程模板，帮助测试人员开发自动化的测试用例，如下图所示。

这些测试工程模板可以帮助测试自动化工程师，在 Visual Studio 集成开发环境中创建和管理单元测试、功能测试、Web 性能测试、负载测试等等一系列的自动化测试用例。这其中，编码的 UI 测试（Coded UI Test，以下简称为 CUIT）是首次出现，是 VS 2010 测试部分一大亮点。测试人员可以通过它使用 C#或者 VB.NET 语言编写自动化测试用例，从用户界面层驱动 Web、Winform 或者是 WPF 的应用。CUIT 为测试用例的自动化提供了一个框架、API 和可扩展的接口，测试人员可以很轻松地开发出所要的自动化测试用例。其实 CUIT 背后的测试自动化实现技术对大家并不陌生，下面列出针对 Web、WinForm 和 WPF 应用的测试技术基础。对每种技术的支持采用的是插件的形式实现的，VS 2010 包括了如下的三种插件：

- Document Object Model（DOM）插件：IE 7/8 HTML/AJAX
- User Interface Automation（UIA）插件：WPF
- Microsoft Active Accessibility（MSAA）插件：Winform，Win32 和 MFC 。MSAA 插件是默认选项，用来支持出其它两者之外的任何应用。

CUIT 现在支持主要的微软平台，详细的内容可以参见 MSDN : Supported Configuration and Platforms for Coded UI Tests and Action Recordings。对于那些尚不支持的平台或者 UI 控件，CUIT 提供了很好的扩展机制，允许大家针对自己的特殊应用进行扩充，下图就是 CUIT 框架的体系结构图。

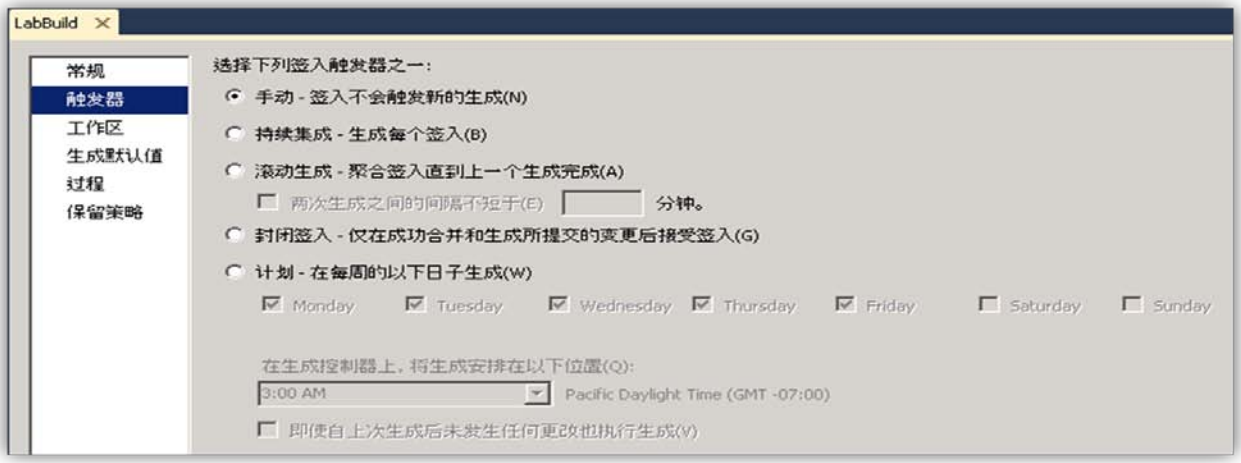


开发自动化测试用例对于有效预防回归问题的出现时非常必要，在实际应用中应该特别注意它的合理比例关系和灵活的策略，包括：自动化用例和手工用例的比例、UI 和非 UI 测试用例的比例关系。自动化测试用例、执行、分析和维护它们都是需要一定投入的，对于敏捷项目而言时间资源的紧缺尤为突出，所以在任何时候团队都要根据自身的资源，有选择性进行测试用例的自动化，通常情况下应该优先自动化那些高优先级的测试用例。

对于 UI 和非 UI 的自动化测试用例而言，应该是以非 UI 的单元测试和功能测试为主，UI 测试未必要的补充。基于 UI 自动化测试用例有它独特优点，例如：它从真实用户角度出发进行测试，即涵盖了界面层、逻辑层和数据层，自动化人员不需要了解被测试应用的代码实现细节等；但是相对于非 UI 测试它也有着先天的不足，包括：执行速度相对比较慢、易受干扰不稳定等。所以在自动化过程中，能用非 UI 测试覆盖的功能尽可能采用非 UI 的测试覆盖，如：API 单元测试等，UI 测试用例只用来实现最基本用户故事的验收测试（Acceptance Test）。

早测试和经常测试——封闭签入和滚动生成

敏捷开发中最可怕的事情莫过于在迭代最后一两天进行测试，结果发现了严重功能缺陷或者回归缺陷，导致不能按计划发布给用户试用。要想彻底解决这样的问题，一方面要在迭代开发阶段测试人员就要参与进来，从客户的角度出发对功能需求和设计文档进行文档测试，即文档评审。测试人员和开发还有项目经理一起从源头上保障将要实现的功能是用户想要的。另一方面就是要在迭代的早期就开始就开始测试，特别前几个迭代已经实现好的自动化测试用例，尽早的执行它们可以有效地避免回归问题的出现。在 TFS 2010 上专门提供封闭签入（Gated Check-in）、滚动生成（Rolling Builds）和持续集成（Continuous Integration）等功能，帮助敏捷团队实现早测试和经常测试。这其中封闭签入和滚动生成是对敏捷团队比较实用的功能。



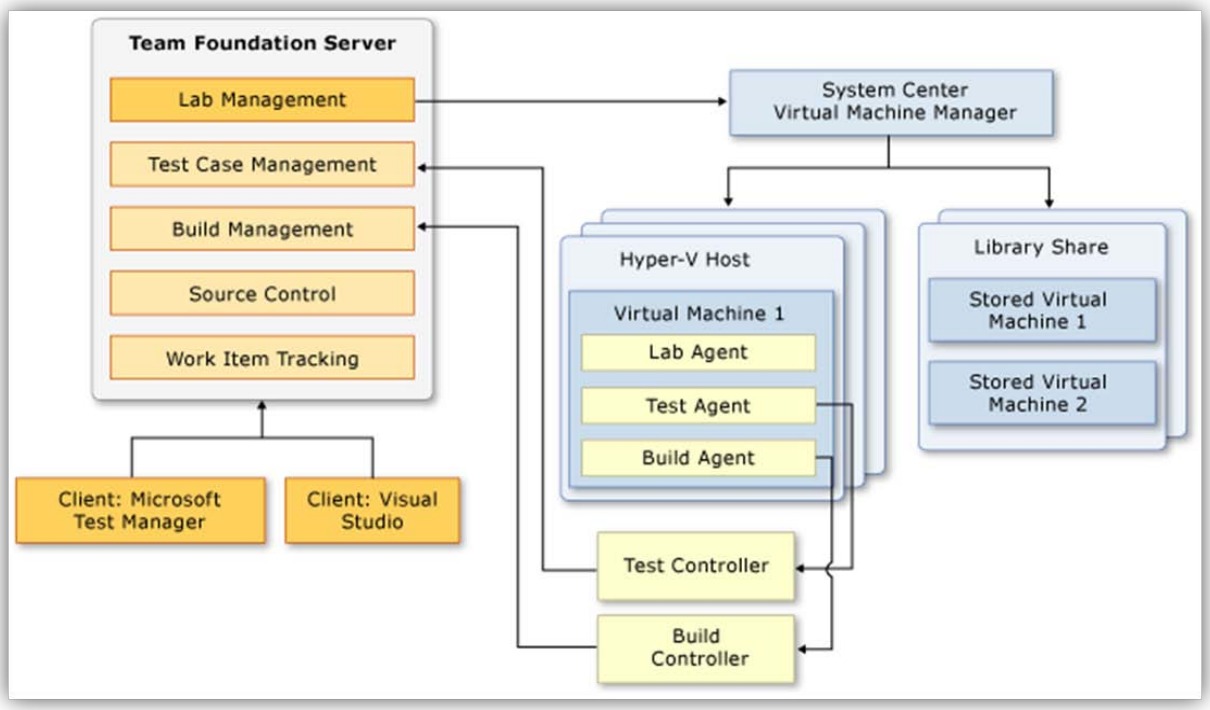
封闭签入是 TFS 2010 提供的一种新的代码签入方式，在配置这项功能后，当用户要签入任何代码时，系统会先将用户本地要签入的代码打包成一个搁置集（shelve-set），然后提交到服务器端，TFS 生成（Build）服务先从 TFS 源代码控制器中同步项目的最新代码，再将提交的代码与之进行自动合并，然后进行编译，如果编译成功，则执行配置的自动化测试用例，如果测试用例全部通过则代码会被自动签入到代码库中，否则返回错误信息给用户，代码是不会进入到代码库。表面上看是与产品测试没有直接关系，但实际上它和测试以及最终产品质量的密不可分。因为代码签入是整个开发过程中发生最为频繁的操作，每次签入代码的质量直接影响着日常的开发活动。对于绝大多数的开发团队来说，check in 代码前不仅要保证编译通过，同时还要最大限度的保证新代码不会破坏已有的功能，也就是要执行测试用例去验证。Gated Check-in 中提到的“Build 成功”，实际上包含两部分内容：编译成功和测试用例执行成功，有了它作为保护代码库的第一道屏障，就可以保证它在任何适合都是可编译，并且达到一定质量标准的。

滚动生成是在 VS 2008 种就有的功能，当 TFS 检测到在它所监控的范围内有任何新的代码变化被签入后，它就启动对最新的代码库进行生成验证，该验证包括编译和运行指定的自动化测试用例。之所以称之为“滚动”，因为它是在一个生成验证操作完成后再去探测有没有新的签入发生，对这期间发生的所有新签入进行新的生成验证。这里需要再强调一下滚动生成的重要意义：它看似只是一个自动生成代码的功能，但实际上起着协调整个开发团队、时刻监控代码库质量、以及尽早暴露产品问题的作用。因为滚动生成时刻都在不停的运转着，对于任何代码签入它都保持着警觉，会去自动验证编译是否成功，自动化测试用例是否都能通过。它就像一个不知疲倦的“代码守护者”一样监控着代码库，第一时间发现其中的任何问题，将问题通知给整个团队，从而避免了问题的积累和拖延。这非常符合敏捷开发中“今日问题今日解决，不要拖到以后”的原则，它帮你最早的发现问题、报告问题，开发团队则应该建立制度要及时响应滚动生成所报告的问题，把它作为 Priority 为 0 或 1 的高优先级问题去对待和解决。

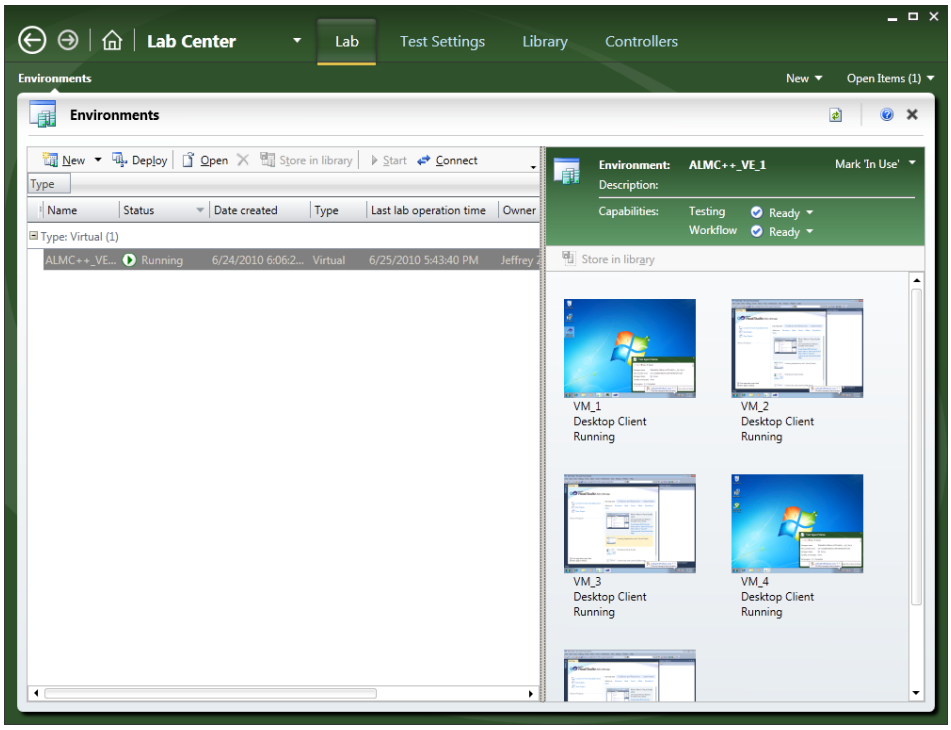
封闭签入和滚动生成都是来保护代码库的正确性和产品质量，它们是否在功能上重复反而让我们不敏捷了呢？其实两者并不重复，只是各有侧重，将它们搭配使用才会发挥其最大效能。封闭签入是在代码进入代码库之前进行验证，签入提交者一般希望竟快知道结果，以便决定下一步的工作，所以封闭签入的时间（编译和运行测试用例）不要太长（10-20 分钟）。这也就决定了我们加入的测试用例不能太多，只添加那些高优先级的测试用例，保证主要的用户故事不被破坏。滚动生成是在代码签入后在后台执行的，由于不存在着与用户的交互等待，所以它执行时间可以更长（几个小时），可以为它加入更多的测试用例，从而全面验证代码库的质量，一旦有任何问题它可以及时通知团队进行修复，这种验证是在几个小时或者每天都在发生的，保证了敏捷对频繁测试的。

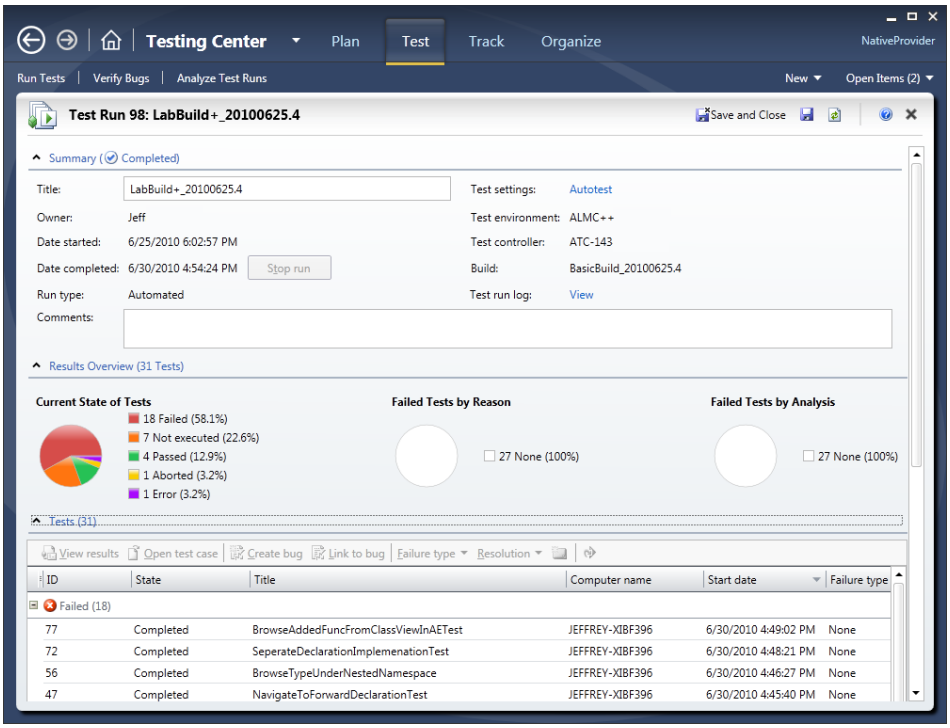
完整的自动化测试解决方案——实验室管理

在谈到软件自动化测试的时候，很多人会误以为实现了自动化测试用例就是自动化测试，其实不然，自动化测试仅是测试自动化的一个重要步骤，绝对不等同于自动化测试。一个完整的自动化测试应该包括：构建、部署、执行测试用例、分析测试结果并作出结论。在前面的自动测试的收益公式中，我们可以看到减少自动测试的维护成本，是提高自动测试收益的重要因素之一。VS 2010 的实验室管理（Lab Management）与测试用例管理、生成管理、源代码控制、工作项管理等功能相结合，为自动化测试提供了这样一个完整的解决方案，目标就是要降低了自动测试的运营和非维护成本，下面这张图展示了实验室环境的系统构架图。



实验室管理功能充分利用了微软的虚拟化技术，包括：Hyper-V 和 System Center Virtual Machine Manager （SCVMM），快速创建干净的虚拟测试环境并进行产品生成和部署，然后执行指定的测试用例集，将结果以报表的形式呈现出来，方便对此产品质量进行分析，如下图所示：





同时，利用虚拟技术的环境快照功能，对于那些难于复现或者环境相关的 Bug，利用虚拟环境的快照技术，可以为开发人员准确的复现 Bug 出现的环境，从而能够快速的进行诊断和及时修复。

总结

Visual Studio 2010 作为 Visual Studio 系列中一个非常重要的版本，为测试人员和团队提供了一整套解决方案，包括：测试计划和用例管理、创建自动化测试用例、测试用例的自动执行、以及实验室管理等。这些功能强调了测试作为整个软件过程的重要角色的作用，促进了测试人员与其它角色的有效沟通与协作，非常适合于敏捷团队使用来完成测试工作。

工具不是万能的，但没有合适的工具辅助也是万万不能的。对于工具在敏捷开发的作用，应该用辩证的观点来看待。不能片面唯工具论，毕竟软件开发过程是人、工具和过程三者共同作用的结果，工具影响着人和过程，同时人和过程也影响着工具所能发挥的效力。所以这决定了工具的引入和部署应该是一个渐进的和逐步适应的过程，特别是对 Visual Studio 2010 这样比较大型和综合性的工具。下面是三个向大家推荐的与 Visual Studio 测试相关的微软论坛，希望能够对大家有所帮助。

- Testing with Visual Studio Test Manager (MTM)
- Visual Studio UI Automation Testing (includes CodedUI)
- Visual Studio Team System - Testing

参考文献

- A different way of presenting the Visual Studio 2010 testing tools
- Content Index for Coded UI Test
- Series on Coded UI Test Extensibility
- Microsoft Test Runner series – part 7 – Exploratory Testing
- Testing the Application

作者简介

周京生，微软亚太研发集团服务器与开发工具事业部软件开发测试工程师，目前主要负责 Visual Studio 生命周期管理工具对 C/C++支持工具的测试工作。自 2006 年加入事业部以来，一直致力于架构工具的设计开发以及如何使用架构工具促进软件开发生命周期管理。周京生先后参与了 Visual Studio 2005 SDK 和 Visual Studio 2008 的测试工作。在刚刚发布的 Visual Studio 2010 旗舰版中，周京生和团队共同完成了多种 UML 建模工具的开发和测试工作。

特别专题

基于 Windows Azure 的云计算应用设计

作者 方国伟

本文从云计算应用的特点出发，分析了在云计算环境下应用程序开发设计的一些变化。根据这些新的特点，本文提出一个“自我感知应用”（Self-Sensing Application）的新概念。接着文章从架构设计的角度阐述了 Windows Azure 平台的构建方式和架构特点，然后结合 Windows Azure 平台的一些功能特点，讲述几个在云计算环境下典型的应用架构设计方式，期望抛砖引玉。最后，文章展望了应用程序未来的发展趋势。

云计算环境下的应用特点

多年来应用程序开发者和架构师们都在努力设计一种既能够在功能上满足当前业务需求，另外又能够在用户需求发生变化或者能够在可预见的将来适应环境变化的应用。尤其是在互联网领域，架构师都在努力让自己设计的应用具有比较强的扩展能力，能够跟得上用户不断增长或者出现突发请求的一些情况。在传统的 Web 应用设计中，我们在架构上一般采用基于多层架构的设计，在 Web 层中大量使用了负载均衡等技术。一般我们的处理方式都是在应用程序设计好之后，在应用部署的过程中事先把环境配置好，应用程序在运行过程配置都是不发生变化的。但是，随着云计算时代的到来，我们面对一些新的挑战，相应的应用程序设计方式随之发生了一些变化。我们首先从云计算的技术特点开始讨论应用的变化。

从技术角度看云计算的特点

毫无疑问，云计算是目前信息产业中讨论得最多的话题。虽然大家对于云计算还没有一致定义，但是对于云计算的一些特点，相关的服务模型等内容日渐趋于统一。在讨论云计算应用架构特点之前，我们先从技术的角度来讨论一下云计算本身的一些特点。

- 按需服务

云计算是一个把信息技术作为服务（IT as a Service）提供的一种方式。这种服务的概念都是从消费方（用户）角度出发，而不是从服务提供方出发考虑问题，因此，一个基本特点是云计算要求按需服务，即用户可以根据需求即时得到服务。从这个角度讲，云计算就像我们公

共服务中的自来水、电和煤气一样，集中供应并按需服务和计费。

- 资源池

云计算的一个好处是提高资源的利用率，而这个一般需要通过共享的方式来达到这个目的。这里可以类比一下我们日常吃饭中的自助餐和桌餐的差别。如果需要共享就需要先把资源集中到一个公共的资源池中。在云计算当中，根据这个资源池中资源的类别，我们把云计算的服务模型分为三种，即所谓的 SPI 模型，如下表所示：

资源类别	**云计算服务模型(SPI)**
应用程序	Software as a Service (SaaS)
系统平台	Platform as a Service (PaaS)
基础设施	Infrastructure as a Service (IaaS)

- 高可扩展性

云计算平台的资源池相对于单个用户的需求而言是比较大的，因此考虑到会有大量不同用户共用一个资源池，他们之间的资源使用模式一般存在一定的互补性，所以对于某个用户的需求而言，云计算具有很高的扩展性。另外，云计算平台在做架构设计的时候，都会考虑到如何让用户可以平滑扩展他们的资源需求，比如计算资源，存储资源等。

- 弹性服务

弹性服务指的是云计算的资源分配可以根据应用访问具体情况进行动态地调整。也正是因为如此，云计算对于非恒定需求的应用，比如需求波动很大、阶段性需求等，具有非常好的应用效果。在云计算的环境中，资源的扩展方式可以分为两大类，一种是事先可以预测的，比如一些季节性的需求。另一种是完全基于某种规则实时动态调整的。无论是哪一种，都要求云计算平台提供弹性的服务。

- 自服务和自动化

对于自服务和自动化概念本身都比较好理解，但是我把这两个放在一起是因为它们之间的内在联系。自服务是云计算中降低服务成本，提高服务便捷性的一种途径，因此它是一个服务的提供方式。但是，对于云计算服务提供方来说，自服务就要求尽量简化用户操作，降低用户使用服务的难度，提升服务响应速度，而这个只能通过后台自动化的方式才能实现。因此从这个意义上来讲，自服务是目的，而自动化则是手段。

- 服务可度量

管理学中有一句名言：没有考核，就没有管理。云计算作为服务提供的方式，需要对服务进行度量。一般服务提供方和用户之间需要有一个服务水平协议（SLA）。这样对于私有云来说，可以根据服务情况进行内部费用核算。而对于公有云来说，服务可度量就是计费的前提，然后根据实际使用量来进行计费。

云计算应用的特点

从前面的描述我们可以看到云计算给应用程序带来的一些挑战，那就是应用程序如何在云计算环境下充分利用云计算平台的一些特点来更好的满足用户需求。云计算应用要能够利用云计算环境中可动态扩展的资源，构建一个具有弹性的高可用应用程序。下面我们分别讨论一下云计算环境下的应用特点和要求。

- 自动化要求

自动化是人类的梦想，而计算机对自动化领域的发展有着巨大的影响，它极大的提高了工作和生产效率。在云计算环境下，自动化要求实际上是对计算工作本身的一个自动化改变。云计算的自动化可以赋予用户对平台基础架构的资源配置任务进行全面统筹的能力，并实现对资源的动态分配以提高管理效率、减少人为错误并加快用户对资源请求的响应速度。应用程序在设计的时候要能充分利用云计算环境的自动化特性，从而使得应用程序可以在很少或没有人工干预的情况下，自动适应需求的变化。

- 分布式计算

大部分云计算平台都是用廉价和标准的计算机硬件构成，然后通过云计算软件的方式在计算能力、可靠性等方面来达到传统的大型计算机的水准。也就是说在云计算环境下，资源池通常是通过分布式软硬件方式来实现。因此云计算应用程序的运行往往涉及到多个计算资源。无论是计算还是存储需求，应用程序一般都会涉及到多个节点，这样在设计的时候要考虑并行设计的思想或采用分布计算的方式。比如，有些云计算应用可以根据计算的要求，采用类似 MapReduce 的编程模型。

- 松耦合

无论是功能上还是性能上云计算对应用的灵活性提出了更高的要求。这就要求应用程序在设计的时候要考虑松耦合的架构。耦合度与灵活性一般是相反的，也即耦合度越高灵活性越低，而耦合度越低灵活性越高。因此，在做云计算应用架构设计的时候，一般要追求松耦合的设计。比如，在做 Web 应用设计的时候，对于用户状态的保持就需要尽量采用无状态的方式

来设计，这样应用程序的水平扩展能力比较好。

- 数据存储方式

在传统的应用设计中，我们一般采用关系型数据库来存储数据。但是在云计算环境下，尤其是对于互联网应用，存在两个需要面对的问题。一是云计算环境下的数据量都比较大，传统的关系型数据库面临数据扩展能力的挑战。另一个是许多应用对于数据存储的要求更多体现在非结构化数据或者是半结构化数据的存储上面。因此，大多数云计算平台都会提供针对非结构化和半结构化的数据存储方式。这样应用程序的架构需要针对新的数据存储方式作出调整。

上面描述的一些应用特点对我们开发和设计应用程序会带来许多影响。一个是应用程序在设计的过程中不仅仅需要考虑操作系统平台或中间件级别的编程接口，还要针对其运行的云平台的接口来对应用程序进行设计。另外一个比较大的影响是有一些工作量从平台产品转移到了应用程序的开发设计人员。比如说，在采用半结构化数据存储的时候，开发设计人员需要处理数据的一致性问题。还有，在云计算平台上如果想要得到比较好的性能，开发设计人员往往还需要对数据的分区进行特别设计，或需要采用一些并行设计的算法等。

具有自我感知能力的应用

传统的基础架构或系统平台中的资源都不能动态配置，因此应用程序在设计的时候主要考虑自身的业务逻辑的实现。应用程序本身的监控和管理都是通过其他系统管理软件如 System Center，Tivoli 等来实现。有一些管理得比较好应用程序，可以通过这些系统管理软件实现部分资源动态调整。但是，这些传统的应用程序本身对底层平台的运行情况是没有任何感知的。随着云计算的出现，应用程序本身的自动化逐渐成为可能。我把这些具有运行环境感知功能的应用程序称之为“**自我感知应用**”（Self-Sensing Application）。

自我感知应用的出现是应用程序发展的一种趋势，是自动化在应用程序运行过程中的一种体现。在做传统应用程序设计的时候，我们把主要的精力都放在功能性的需求方面，但是对于一些非功能性的需求往往采用手工配置的方式来实现。云计算平台的出现使得计算平台的资源具有可编程的特性，因此我们在应用程序的架构设计中可以通过基础架构平台的一些接口来感知应用程序的实际运行情况，并可以结合访问情况对应用程序的运行资源进行动态调整，以实现完善的自动化程序运行。

云计算平台具有可编程的资源分配，因此我们还可以设定自动化的部署过程，也就是让应用程序的可以自动化的完成应用程序的部署，升级等工作。自动化部署也是应用程序动态扩展的一个前提。这样当需要更多计算实例来处理更多的用户请求的时候，新的计算实例可以动

态的生成出来并自动启用。

自我感知应用是应用程序朝着成为具有独立、自治单元的方式发展的一种体现。应用架构师在做设计的时候，面对的是底层抽象的、几乎无限的计算资源，而不是传统意义上的物理资源。从另外一个角度看这种设计方式也是应用程序与底层计算平台松耦合的一种体现，从而使得应用程序不绑定具体物理硬件。云计算平台从底层提供几乎无限的计算、存储和网络资源，其上的应用程序就像一个个具有人工智能的独立单元，他们在完成自身业务工作之外，还能够具有自我管理和自我修复的功能。

Windows Azure 的架构设计

Windows Azure 的概况

Windows Azure 平台目前包含 Windows Azure ,SQL Azure 和 Windows Azure Platform AppFabric 三大部分，而其中 Windows Azure 是平台最为核心的组成部分，它被称为云计算操作系统，也是本文讨论的重点。不过这个操作系统目前只运行在微软的数据中心当中，微软是通过销售云计算服务作为她的业务模式。

实际上为了便于理解，我们也可以把 Windows Azure 看成是数据中心的操作系统。当然，这里我们称之为操作系统实际上是一种类比，因为 Windows Azure 不是传统意义上的操作系统。但是它履行了资源管理的职责，只不过它管理的资源更为宏观，数据中心中的所有服务器、存储、交换机和负载均衡器等都接受它的管理。因为未来的数据中心会越来越像一台超级计算机，因此 Windows Azure 也会越来越像一个超级操作系统。Windows Azure 的设计团队中就有许多微软技术重量级人物，其中包括 Dave Cutler，他被称为是 Windows NT 和 VMS 之父。

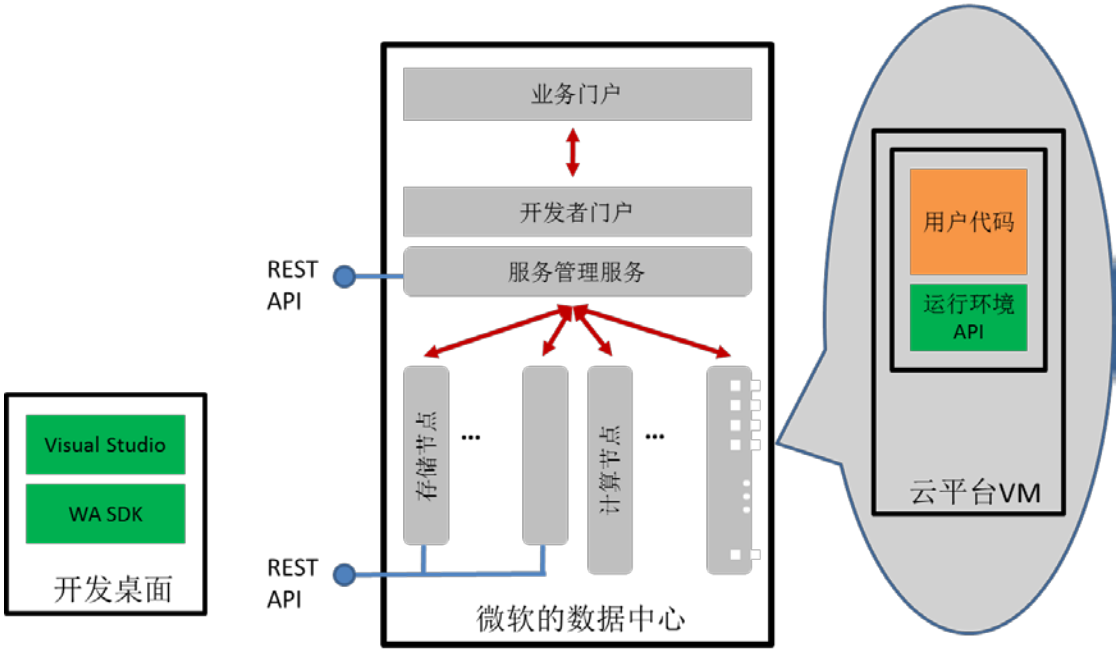
Windows Azure 为开发者提供了托管的、可扩展的、按需应用的计算和存储资源，还为开发者提供了云平台管理和动态分配资源的控制手段。Windows Azure 是一个开放的平台，支持

微软和非微软的语言和环境。开发人员在构建 Windows Azure 应用程序和服务时，不仅可以使用熟悉的 Microsoft Visual Studio，Eclipse 等开发工具，同时 Windows Azure 还支持各种流行的标准与协议，包括 SOAP、REST、XML、和 HTTPS 等。

Windows Azure 的组成和架构

从云计算 SPI 模型来看，Windows Azure 主要处于平台即服务的层次。按照微软目前公开的发展计划，其将在年内提供基础设施即服务层次的一些服务，比如 VM Role。Windows Azure 作为基础平台的调度和管理软件，它是构建高效、可靠、可动态扩展应用的重要平台。主要由四大部分组成：

- 计算服务
- 存储服务
- 管理服务
- 开发环境

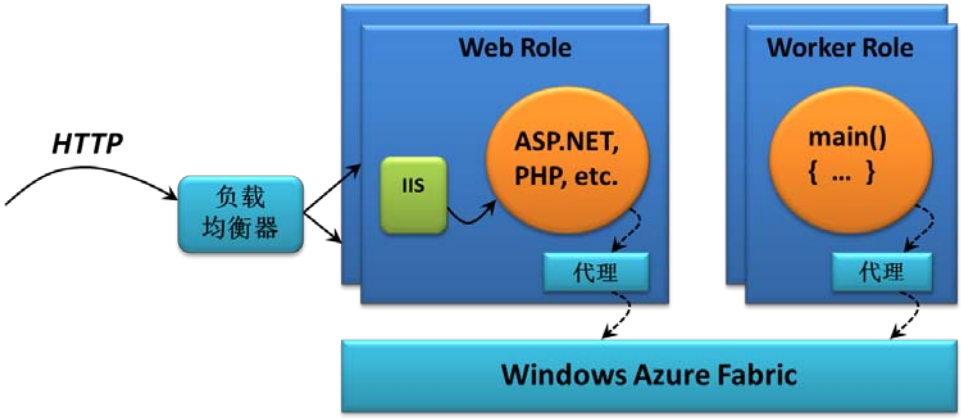


上述四个组成部分中只有开发环境是安装在用户的计算机上的，用于用户开发和测试 Windows Azure 的应用程序，其余三部分都是 Windows Azure 服务平台的一部分而安装在微软数据中心。

Windows Azure Fabric 控制器被比作是 Windows Azure 的大脑，它负责平台中各种资源的统一管理和调配。而 Windows Azure Fabric 则由其管理的大量 IT 设备组成。开发人员通过

Windows Azure 开发工具（比如 Visual Studio 2010 和相应的 SDK）开发的应用程序一般分为两大部分。一部分是应用程序代码，也称服务代码，另一部分是应用的配置文件也称服务模型。每个应用包括两个配置文件：服务定义文件和服务配置文件。这两个配置文件中会包含应用程序在 Windows Azure 上运行和发布的一些信息，如认证信息、服务端口、服务角色、需要的实例数和自定义变量等。当用户通过开发者门户把应用程序上传到 Windows Azure 平台的时候，其中的配置文件则由 Windows Azure Fabric 控制器来读取，然后由其根据配置文件中指定的方式进行服务部署。

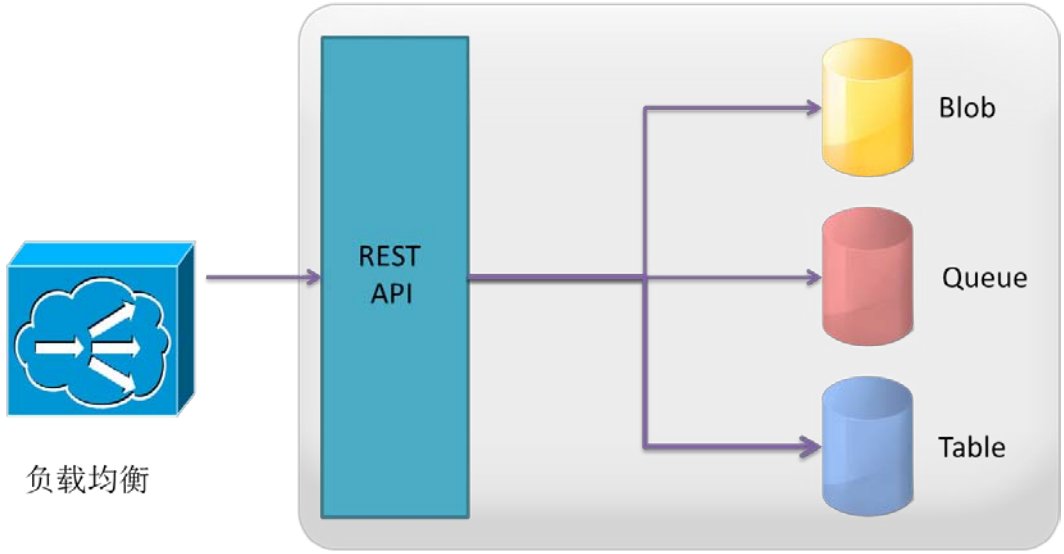
Windows Azure 的计算资源目前是通过成为 Web Role 和 Work Role 的方式来分配的。为了便于理解，我们可以认为 Web Role 和 Worker Role 是两种不同的虚拟机模版。其中 Web Role 是为了方便运行 Web 应用程序而设计的，而 Work Role 是为了其他应用类型，比如批处理而设计的。一种比较常见的架构设计方式是使用 Web Role 来处理展示逻辑，而通过 Worker Role 来进行业务逻辑处理。Web Role 负责客户端的 HTTP 请求，为了支持应用的扩展，Web Role 上的应用一般会设计为无状态的，从而使得系统可以方便的增加 Web Role 实例数量，提高应用的并发处理能力。



当应用程序部署完后，Windows Azure Fabric 控制器便开始监控应用的状态，以保证应用程序的正常运行。为了使控制器能够实时获取应用和运行实例的状态，所有 Role 实例，也即虚拟机实例中都预先安装了代理程序，Fabric 控制器就是通过这些代理来实时获取相应的状态信息。当检测到实例故障的时候，Fabric 控制器就会启动新的包含同样服务的实例并添加到服务组中去。按照微软的计划，Windows Azure 将在今年晚些时候提供另外一种称为 VM Role 的计算服务，它将让用户对底层计算平台有更多的控制权，可以通过远程桌面服务（RDS）方式连接过去。其主要目的是要已有的 Windows 应用程序可以相对平滑地迁移到 Windows Azure 上去。

Windows Azure 提供的存储不是一个关系型数据系统，并且它的查询语言也不是 SQL，它主要被设计用来支持建于 Windows Azure 上的应用，它提供更简单容易扩展的存储。当然，对

于在 Windows Azure 之外的应用程序也可以通过标准的 REST API 来访问和使用这个存储服务。Windows Azure 主要提供了三种数据存储方式以满足应用程序的不同需求，这三种存储方式分别为 Blob、Table 和 Queue。Windows Azure 为了提升兼容性而提供的 Drive 其底层实现实际上就是 Blob 的一种。所有这三种存储服务都可以通过标准的 REST API 来访问。



Windows Azure 提供的存储服务具有以下特点：

- 可以存放大量数据
- 大范围分布的
- 可以无限扩展的
- 所有数据都会复制多份
- 可以选择数据存储地点

Blob 非常便于存储二进制数据，比如 JPEG 图片或 MP3 文档等多媒体数据。Blob 适用于部分应用，但它存储的数据缺乏结构化，为了让应用能够以更易获取的方式来使用数据，Windows Azure 存储服务提供了 Table。它最大的不同之处是提供半结构化数据的存储，并且是一种可扩展存储，它通过多个节点对分布式数据进行扩展和收缩，这比使用一个标准的关系型数据库更为有效。与 Blob 和 Table 都是用于长期存储数据不同，Queue 的主要功能是提供一种 Web Role 实例和 Worker Role 实例之间通信的方式。而 Drive 的主要作用是为 Windows Azure 应用程序提供一个 NTFS 文件卷，这样应用程序可以通过 NTFS API 来访问存储的数据。提供这种 Drive 存储方式使得迁移已有应用程序到 Windows Azure 的过程变得更为平滑。无论数据以 Blob、Table、Queue 或 Drive 任何方式存储，Windows Azure 存储服务都会将所有数据

复制三次，如果有一份拷贝出现问题，存储服务能自动恢复出一份新的拷贝，保证应用能够准确及时读取原始数据信息。

Windows Azure 应用程序的自我感知

对于一个自我感知应用而言，一方面它要能够感知底层运行平台的一些环境信息，另一方面它还需要一种机制能够把自身运行要求转递给底层运行环境。我们知道 Windows Azure 上应用程序由代码和基于 XML 的配置文件两部分组成。应用程序可以通过配置文件把自身运行要求传递给 Windows Azure，确切的讲是提交给 Fabric 控制器。但是，应用程序如何才能感知环境的一些变化呢？加入配置文件更新之后应用程序如何才能得到通知并作出响应呢？这里就要用到 Windows Azure 提供的服务运行时编程接口（Service Runtime API）。

Windows Azure 的服务运行时编程接口最常用的使用方式就是帮助应用程序了解应用服务和应用所在的 Role 实例的信息，包括：

- 它能够让应用程序访问在服务配置文件和服务定义文件中的最新服务配置信息。当配置文件的信息被更新的时候，服务运行时编程接口能够保证返回最新的配置信息。
- 它能够应用得到最新的服务拓扑结构，比如哪些 Role 实例在运行，每种 Role 类型有多少实例等。
- 由于 Worker Role 中的代码运行周期有点类似于有限状态自动机的处理方式，服务运行时编程接口能够帮助应用得到 Worker Role 实例的生命周期信息。

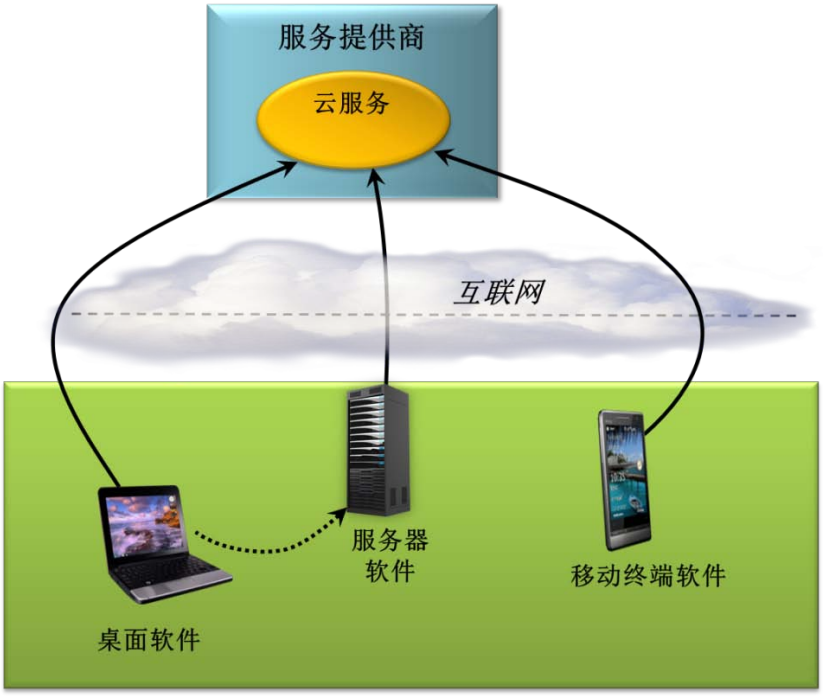
服务运行时编程接口可以通过两种方式使用。对于.NET 托管代码，Windows Azure 的 SDK 中包含一个名为 Microsoft.WindowsAzure.ServiceRuntime.dll，当用 Visual Studio 新生成一个云服务项目时它会被自动引用。而对于本地代码，可以通过使用 SDK 中的头文件和库文件就可以用 C 来调用这个编程接口了。

服务运行时编程接口是应用程序本身用来得到自身及其运行环境信息用的，但是如果应用程序之外，比如说一个管理工具要得到指定应用程序的信息，那么一般需要利用另外一个称为服务管理的编程接口（Service Management API）。这两个编程接口在功能有重合的地方，它们之间的最大区别在于服务运行时编程接口在 Windows Azure 中运行，而服务管理的编程接口一般在 Windows Azure 之外运行，它更多的用在那些针对 Windows Azure 的管理工具开发当中。

常见的云计算应用设计方式

软件+服务的方式

当微软在 2005 年底提出软件+服务的概念的时候，许多人都认为这只是一个市场宣传口号。但是，在过去几年我们发现有越来越多的应用通过软件+服务的模式取得成功。从应用架构角度来看，软件+服务的方式结合了云服务和设备端两方面的优势。从传统的应用如杀毒软件，Live Messenger/QQ 到帮助苹果取得成功的 iTunes，应用商店等，都是结合了终端软件和云计算服务的各自特点。



软件+服务对于架构师的意义在于在设计应用架构的时候，可以打开思路，充分利用云和端的各自优势。由于 Windows Azure 提供的服务如计算、存储都是可以独立使用，所以对于像一个做在线远程教学的应用，在设计的时候就可以考虑自己构建运行应用程序的网站，但是把课件视频等大数据量存放到 Windows Azure 的存储服务上，大大降低项目对存储设备的投资，从而降低项目的进入门槛。

运行在各种端设备上的应用程序可以直接与构建在 Windows Azure 平台上的服务进行交互，但是对于一些需要透过防火墙才能与互联网上的云计算服务进行集成的企业应用来说，需要一个 Internet Service Bus（ISB）来进行应用集成。Windows Azure AppFabric 就提供了 ISB 的功能，让企业内部应用可以不受防火墙和 NAT 等限制进行服务集成。

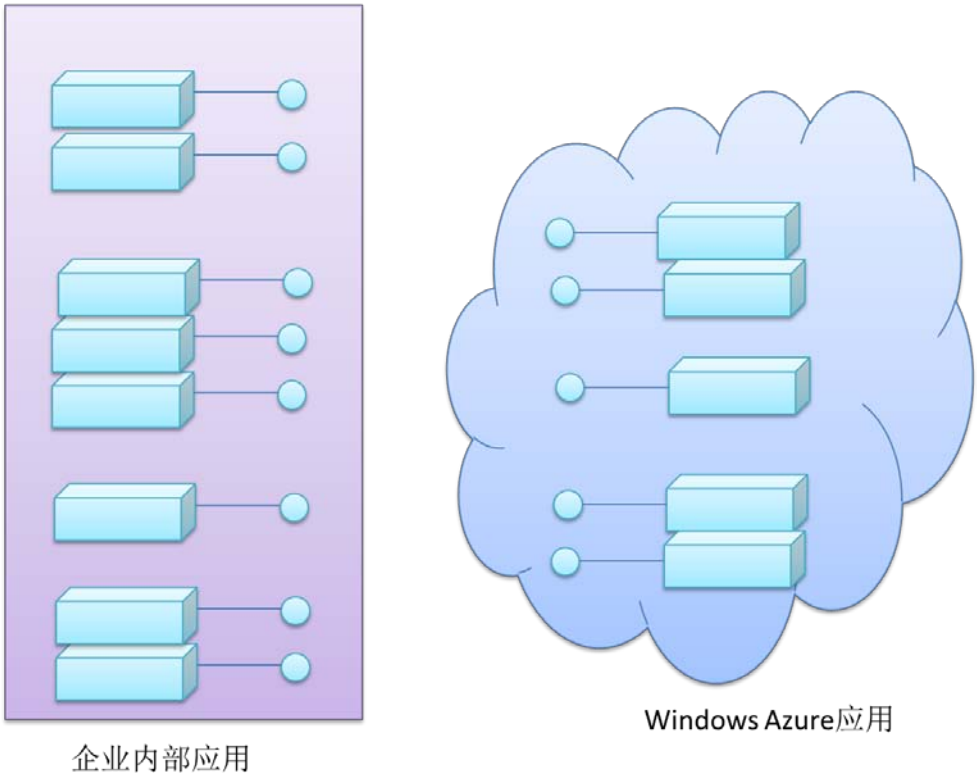
我们在通过软件+服务的方式进行应用架构设计的时候，由于是基于互联网的分布式计算，

因此要特别考虑到网络延时对应用的设计和架构的影响。我们可以采用不同的方式来降低分布式方式带来的影响。比如尽量采用粗粒度的调用方式，降低软件与云服务的交互次数。另外，在架构设计的时候，尽量让动态数据与计算工作靠近，让静态数据与使用用户距离更近。Windows Azure 的存储服务在创建账号的时候，用户可以选择数据存储的数据中心位置。除此之外，Windows Azure 还提供了 CDN 服务，从而使得应用数据可以尽可能靠近访问用户。

采用 SOA 设计方式

云计算平台能够从资源使用的角度保证应用程序的灵活性。但是要让应用的构建和功能上保持灵活性，我们需要采用面向服务的方式来构建我们的应用，也就是说在云计算的环境下我们同样需要 SOA 的思想来设计我们的应用。我们一般认为服务就是通过某种标准的接口来访问应用的功能和数据。如果我们事先构建了一个服务池，那么就可以通过某种服务编排的方式来创建一个新的服务或一个新的组合应用。这种构建方式可以给应用的架构带来很大的灵活性。

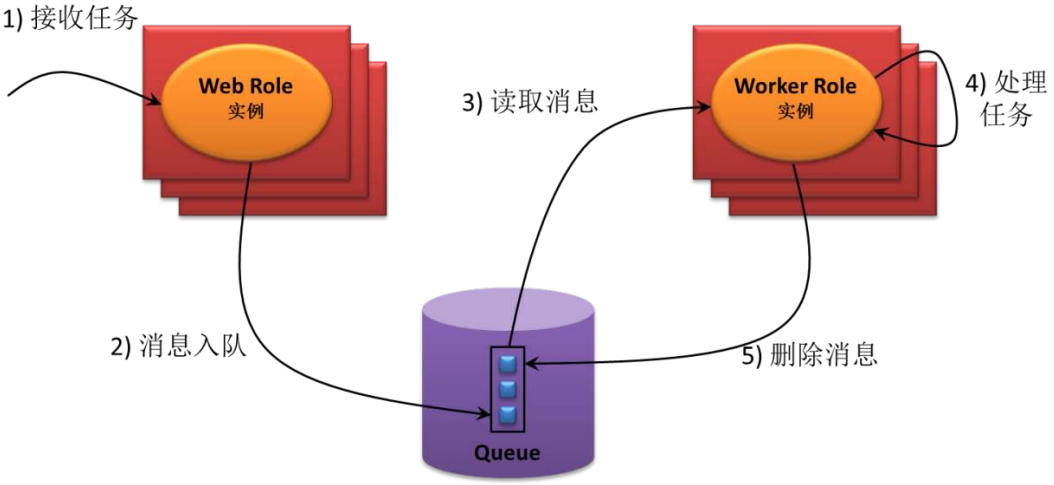
在云计算的环境下，我们可以融合软件+服务和 SOA 的设计思想，从问题本身需求出发设计和组合应用的服务，可以利用不同平台上的服务而不在局限于企业内部。在设计服务的时候，一个重要的设计目标就是让服务变成平台或者位置无关。新的服务或应用可以组合在企业内部的应用服务以及部署在像 Windows Azure 云平台上的应用服务，如下图所示：



松耦合的架构设计

在云计算环境下，应用程序的运行环境等变得不像传统环境那样可控。对于大规模分布的应用而言，要充分考虑到底层环境的单个节点出问题的可能性。为了保证应用程序的可靠运行，一般会采用松耦合的架构设计，并在架构设计的时候考虑系统如何快速从错误中恢复过来，比如状态的保持，自动化数据备份和恢复等。另外，松耦合架构设计不但可以增强系统的整体健壮性，还能增加系统的可扩展性。

一种的常见的设计方式就是采用异步消息机制来实现松耦合的架构设计。通过异步消息机制可以使得应用各个模块之间通过事先定义好的接口进行异步通信，相互之间就像一个不透明的黑盒子。比如对于 Web 应用程序，我们可以通过这种方式把 Web 服务器、应用服务器和数据库服务器之间进行隔离。在云计算环境中，这种异步机制对于应用程序的水平扩展显得更为重要。这也就是为什么无论是亚马逊的 AWS 还是微软的 Windows Azure 都会提供基于消息队列的通信机制。下面是一个典型的基于 Windows Azure 的应用架构。



在上面的架构中，我们可以看到 Web Role 实例与 Worker Role 实例不再一一绑定，而是通过 Windows Azure 的存储队列进行通信。因此，Web Role 的实例数和 Worker Role 的实例数可以根据业务具体情况进行分配。这种通过消息队列的方式同时也很好的解决了数据的并发处理，并使整个系统具有良好的扩展性。而且，某个实例暂时停止工作都不会对整个系统造成大的影响，从而增强了应用程序的可用性。

总结

云计算无疑已受到大家极大的关注，但是它还是一个相对较新的概念，相关的技术正处于快速发展的过程中。我们已经可以看到云计算对 IT 行业的硬件模型，应用模型和用户体验等

方面带来了革命性的影响。从应用模型的角度来看，云计算平台的出现使得开发人员可以快速的构建高可用的，可以几乎无限扩展的应用。Windows Azure 平台为广大开发人员提供了一个开发、设计和运行应用的云计算环境。随着云计算相关技术的进一步发展，我们也将在 Windows Azure 上看到越来越多的新特性，从而使我们能够进一步简化在其上开发自我感知应用的工作，另外非功能性的一些要求也将更多通过配置文件而不是代码来实现。

人们对于信息技术需求的发展始终没有改变，那就是追求可靠，便捷，易用的信息应用。当我们把一个冰箱连接到电源插座的时候，我们从来不会去考虑这个电力是从哪个发电站来的，我们也不需要成为一个电力专家来使用这个冰箱。云计算的出现让信息技术往这个方向更进了一步。我们认为应用程序将逐渐演变成具有自我感知能力的应用，成为一个能够根据平台环境和用户请求进行自我调整和自我修复的自治单元。也许不远的将来，整个互联网就像一台巨大的计算机，其上提供无限的计算资源和服务，人们使用其上的应用程序就像我们现在使用自来水，电力那样方便，自然。

作者简介：方国伟，资深系统架构顾问，目前在微软开发工具及平台事业部负责微软新技术和产品的技术推广工作。之前曾是微软企业和合作伙伴部担任制造业客户技术经理。在加入微软之前，它是 IBM（中国）有限公司软件部的资深软件工程师。方国伟于 2000 年从清华大学计算机系研究生毕业。

Microsoft Visual Studio

Microsoft

/*CODING完美世界*/

Code，创造了万事万物。几乎任何地方，均有它的存在，正如作为开发人员的你，眼中所见的无限可能。利用Visual Studio 2010的新工具，你设计、开发或是部署的工作方式，都将发生意想不到的变化；你心中的伟大构想，终将以前所未有的方式，付诸实现。

/*CODE，无处不在*/
用Visual Studio 2010打造你的奇迹

立即登录VisualStudio.com
或拨打咨询电话: 800-820-3800 / 021-96081318

请用手机拍摄

拍摄此标签，获取Visual Studio 2010最新消息
如手机未安装二维码应用程序，请至http://gettag.mobi免费获取 *信息及数据资费按相关标准收取

©2010 Microsoft Corporation保留所有权利。Microsoft, Visual Studio及Visual Studio徽标是Microsoft Corporation在美国和/或其他国家（或地区）的注册商标或商标。

推荐文章 | Articles

Flex 技术在企业级开发中的应用

作者 池建强

从我个人的从业经历来看，在长达十几年的软件研发过程中，无论是研发的产品或实施的项目，大部分是在为企业客户提供服务。当然，期间我还从事过两年左右的互联网应用的开发。早期的互联网应用开发和企业级应用开发的区别还是很明显的，无论是技术、架构、业务和用户体验，都有很大的不同。举个简单的例子，比如开发语言，最早在 2000 年左右，大家都用 Perl 和 Asp 做网站，后续陆续开始使用 PHP、Ruby 和 Python 这样的动态语言来构建丰富多彩的互联网应用，当然这其中也少不了 Flex 技术。而为企业客户构建的应用，则更多地倾向于静态语言，比如 Java 和 C#等。当然随着技术的发展，这两者之间的交集越来越多，大家会越来越多的发现，很多大规模的互联网站点是基于 Java 或 C#构建的，也有一些企业应用开始使用动态语言。这一点也很明显地展示了企业级应用与互联网的融合。

什么是企业级应用？

说了这么多，需要为企业级应用系统做一个定位。事实上这个概念在业界并不是十分清晰，没有一个明确的定义，什么是企业级，为什么叫企业级呢？有的观点是从系统规模上划分，有的是从团队规模上划分，有的是从开发周期上划分。我个人对企业级应用系统的定义比较简单，主要是用来区分互联网应用和个人软件。什么是互联网应用呢，四大门户（如新浪、网易等）、百度和淘宝、各种 SNS 网站、博客系统和微博系统等等；个人软件呢，就是指安装在个人 PC 上的客户端软件，例如编辑器、绘图软件、开发工具等。这两种类型的应用和软件受众都是普通大众，而企业级应用系统的受众是企业客户，是为企业服务的，企业级应用系统的使用者是企业内外部客户以及与企业业务关联的人员。

2009 年，在技术层面可以说是一个风起云涌的年份，互联网像一条巨大而充满吸力的纽带，把各种 IT 服务相关的技术、应用和实现都吸引过来，形成了一个完整而庞大的互联网生态圈。那身处其中的我们认识到了什么呢？随着我们持续的通过技术、平台、产品和项目为企业客户提供服务，我们发现企业应用不再局限在 Intranet 内部，企业应用系统的互联网化趋

势越来越明显，主要体现在以下三个方面：

- **Intranet 到 Internet 的转变**：企业应用系统由局域网转到互联网，企业应用开始要求多浏览器支持，国际化的支持，全球业务的互联互通。同时企业应用不再满足简单的表单和表格界面，富互联网应用（RIA）的需求应运而生，企业客户越来越倡导用户体验，RIA 也是我们后续要重点讨论的话题。
- **企业应用的内容转变**：除了企业的核心业务系统，这样一些需求渐渐浮出水面：交互性门户系统、电子商务平台、企业级 2.0（博客、Wiki、RSS 和微博等）、企业级 SNS（社区平台）和无线企业应用等。
- **需求的转变**：除了功能需求，客户对于安全、性能、大容量和大并发等特性愈发关注，在可预见的未来，企业应用一定是构建在互联网而非局域网，可能是在云端，也可能在其他的新技术上实现

作为现阶段的 IT 服务提供商，必须从技术层面和业务层面去适应和支撑这样的趋势变化，否则我们会变得步履艰难。
好了，谈了这么多，主要讲了一个趋势的变化。下面我们来看一下在互联网和企业应用中都能发挥巨大作用的 RIA 技术。

RIA 简介和选择 Flex 的原因

RIA 技术的全称是富互联网应用（Rich Internet Application），RIA 首先应该是一个网络应用程序，其次它还要具有桌面应用程序的特征和功能。可以这样理解，如果你的桌面程序能在网络上（目前主要是基于浏览器）运行，并且能保持其原来的功能和特征，那么我们就可以称它们为 RIA 应用（富互联网客户端应用）。

目前 RIA 的主流技术主要包括 Adobe 公司的 Flex，微软公司的 Silverlight 和 Java 阵营的 JavaFX。Flash 由于 Flex SDK 的支撑，很早就从单纯的动画展示转入 RIA 领域，而且由于 Flash 的普及，Flex 目前应该是三大技术体系中市场份额最大、应用最广泛的技术；Silverlight 是微软推出的跨浏览器和跨平台的插件，能在微软的.NET 上交付炫目的多媒体体验和有丰富交互功能的 Web 应用，已经对 Flex 有了很大的冲击；JavaFX 是未被收购前的 Sun 公司在 2007 年推出的用来对抗 Flex 和 Silverlight 的桌面应用，但由于起步较晚，目前应用并不广泛，但其 Java 的原生性和开源性质对 Java 社区的开发人员还是有很大的吸引力。

基于以上三种技术，我们最终选了 Flex 做企业级的富客户端应用开发，虽然苹果公司的 CEO 乔布斯老师已经开始公开表示不在苹果的移动设备上支持 Flash，尽管 HTML5 和 CSS3 来势

汹汹，但是在企业应用开发这样一个不是非常激进的领域，考虑到 Flash 广泛的群众基础，最终我们还是选择了 Flex。

事实上在互联网应用中，RIA 技术早已散发出夺目的光辉和迷人的魅力，无论是电子商务中的产品展示，还是 SNS 网站上的交友游戏，亦或是游戏和教育领域里的交互性设计，已经为广大互联网用户带来了无以伦比的客户体验。那么在企业应用系统中，企业客户还在满足于呆板的树形结构、简单的表格和文字性质的描述吗？就我们的经验来说，2008 年开始，企业客户就开始向我们提出这样的需求了，例如操作复杂的表单、图形化内容展示、动态报表绘图、图形化流程配置、流媒体视频播放和文档播放等，这一切都是在浏览器上进行的。对于大部分这样的需求，我们都是笨手笨脚的使用了 Javascript、Extjs、Jquery 和 Activex 等前端技术勉强实现了，对于不能实现的需求，我们只能腼腆的告诉客户，这些功能我们还实现不了，或者说浏览器不应该有这样的操作等等，当然这种话事实上也很难说服我们自己。

直到我们决定采用 Flex 技术来实现富客户端操作之后，我们才发现很多问题在 Flex 面前迎刃而解了。在 Full-Stack 系统中，如果 Ajax 技术和 Flex 技术配合形成前端组件体系，将大大提高开发效率、系统性能和改进客户体验。

Flex 是 Adobe 公司开发的可以输出成基于 Flash Player 来运行的互联网应用程序。Flex 基于标准的语言，与各种可扩展用户界面及数据访问组件结合起来，使开发人员能够快速构建具有丰富数据演示、强大客户端逻辑和集成多媒体的应用程序。Flex 目前最新版本是 4，一个 Flex 应用程序应该有两种语言代码完成，那就是 ActionScript 和 MXML。ActionScript 是一种面向对象的脚本语言，MXML 则是一种标记语言，非常类似于大家所熟悉的超文本标记语言(HTML)，扩展标记语言(XML)。简单来说 MXML 用来描述界面，ActionScript 用来处理业务逻辑。

以下是 Flex 的一些基本特点，也是我们采用 Flex 的重要原因之一：

1. 可视化开发，通过拖拽方式开发界面
2. 对于有 XML 和脚本开发经验的人员，很容易上手
3. 可实现表现层与后台的真正分离
4. 丰富的媒体支持和动画效果，良好的用户体验
5. 支持多种通讯方式和数据格式
6. 同时支持客户端和浏览器模式

7. 跨平台，支持各种操作系统和浏览器

基于 Flex 构建企业级应用开发平台

与普通开发者使用 Flex 技术不同的是，我们采用了一种组件化的方式引入 Flex，这是因为我们对这部分技术的引入并不是从零开始，为了应对企业级应用开发的需求，我们很早就构建了一个企业级应用开发平台——GAP（Global Application Platform）平台，这是一个 Full-Stack 的应用开发平台，除了底层框架、组织权限、工作流引擎、数据字典等等，还包括界面框架、通用 Web 控件，Ajax 控件等，Flex 的引入是对现在平台的补充和完善。

基于以上考虑，我们对 Flex 的应用分为三个阶段。

- 首先，进行 Flex 与 GAP 平台的整合，包括组件化集成、前后台通信机制的设计，在这一阶段我们主要采用了 ant 和 xdoclet 技术进行组件的打包、资源文件的合并，采用 Spring BlazeDS Integration 技术与 GAP 平台框架进行交互访问，通信方式采用了 Felx 提供的 RemoteObject。（Spring BlazeDS Integration 是 Adobe 与 Spring 共同联合开发一个开源项目，其目标是开发者可以利用 Spring 开发模型通过 Flex、BlazeDS、Spring 以及 Java 技术创建 RIA）。
- 其次，构建 Flex 图形组件框架，对 Flex 提供的控件进行封装、扩展，形成针对企业应用的个性化 RIA 控件库。在这一阶段我们主要是基于 Adobe 的开源项目 Cairngorm 进行构建的。Cairngorm 是一个基于 Flex 技术的微内核的 MVC 框架，设计简洁而易于扩展，非常适合构建自己的 RIA 控件库。
- 第三，使用 Flex 技术解决企业应用中的实际问题。

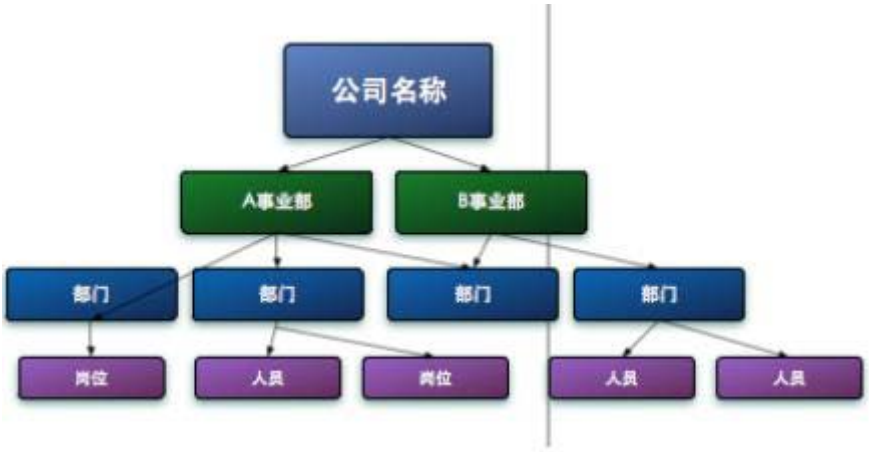
下面我们主要从实际应用的角度来看一下 Flex 在企业级系统开发中能够做什么。

一、企业组织结构的图形化展示

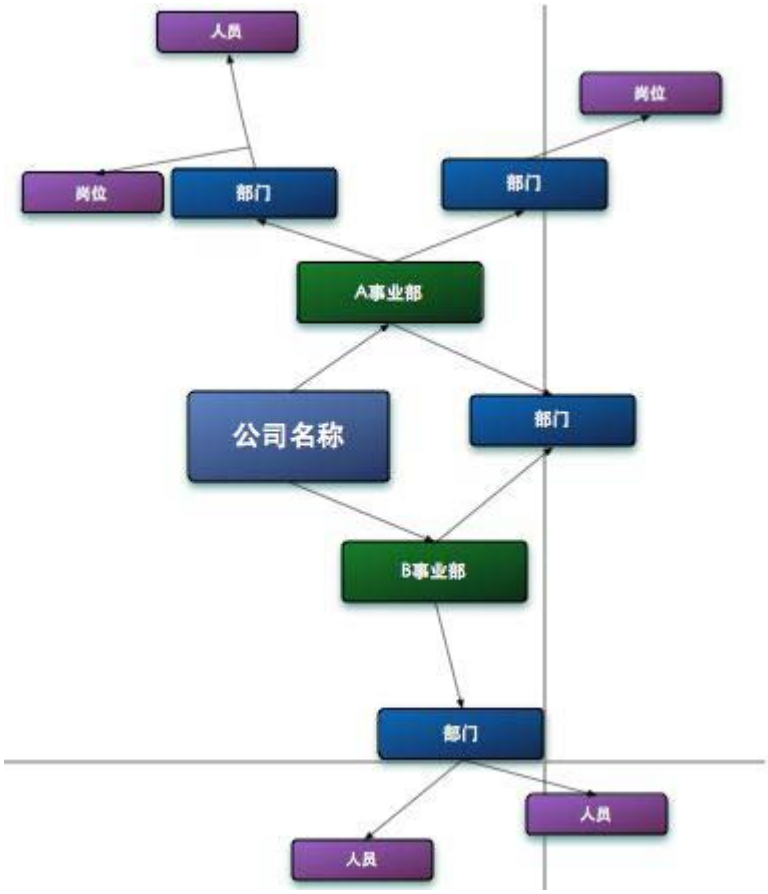
凡是为企业开发过系统的人都知道，企业的组织结构管理和权限管理几乎是每个项目或产品不可或缺的基础组件之一。从功能角度分析，GAP 平台的组织权限系统已经非常完善了，无论是多关系的组织结构、细粒度的权限控制，都可以非常好地满足客户对于组织管理和安全的需求。直到有一天一个客户提出，能不能把那棵呆板的组织机构树变成组织结构图，如果能支持图形化操作就更好了。听到项目组给我们反馈的这个需求，第一个反应就是拒绝，因为实现起来太麻烦了。

原有的组织结构树如下图所示：

客户需要的展示方式可能是：



也可能是：



同时企业客户还希望能够进行图形化操作管理，这种功能如果通过 Javascript 来实现无疑是困难的，而 Flex 技术在处理类似的功能时则具备先天的优势。首先 Flex 是一种可以运行在网络上的客户端技术，它提供了一套成熟的图形化控件和类库，可以很容易的实现图形和布局控制。同时，Flex 可以通过多种通讯方式（HttpService、RemoteObject、WebService）与 Server 端的服务进行数据交互，使得图形化操作变得非常简单，例如把人员拖到另一部门，双击显示该机构的详细信息等。最终我们也是通过 Flex 技术实现了客户的需求。

二、表单操作

Flex 同样可以构造出复杂的表单功能，操作便捷，响应迅速，适应企业不同场景的需求。例如这样一个基于 Flex 技术的表格，看似简单，实际上是包含了排序、过滤、表头拖拽、表头固定、合并等功能，类似的功能如果用 Ajax 的方式来实现代码量会很大，但是在 Flex 中，这些特性基本上是原生的，或经过简单开发即可实现，代码量非常小，而且性能远远超过普通列表控件和 Ajax 列表控件。经测试，在同一场景下，通过 Flex 列表控件加载 1000 条数据，平均响应时间是 0.1 秒，Ajax 控件 0.5 秒，普通刷新页面的方式最慢，从发出请求到返回并显示数据，大概需要 1 秒钟。

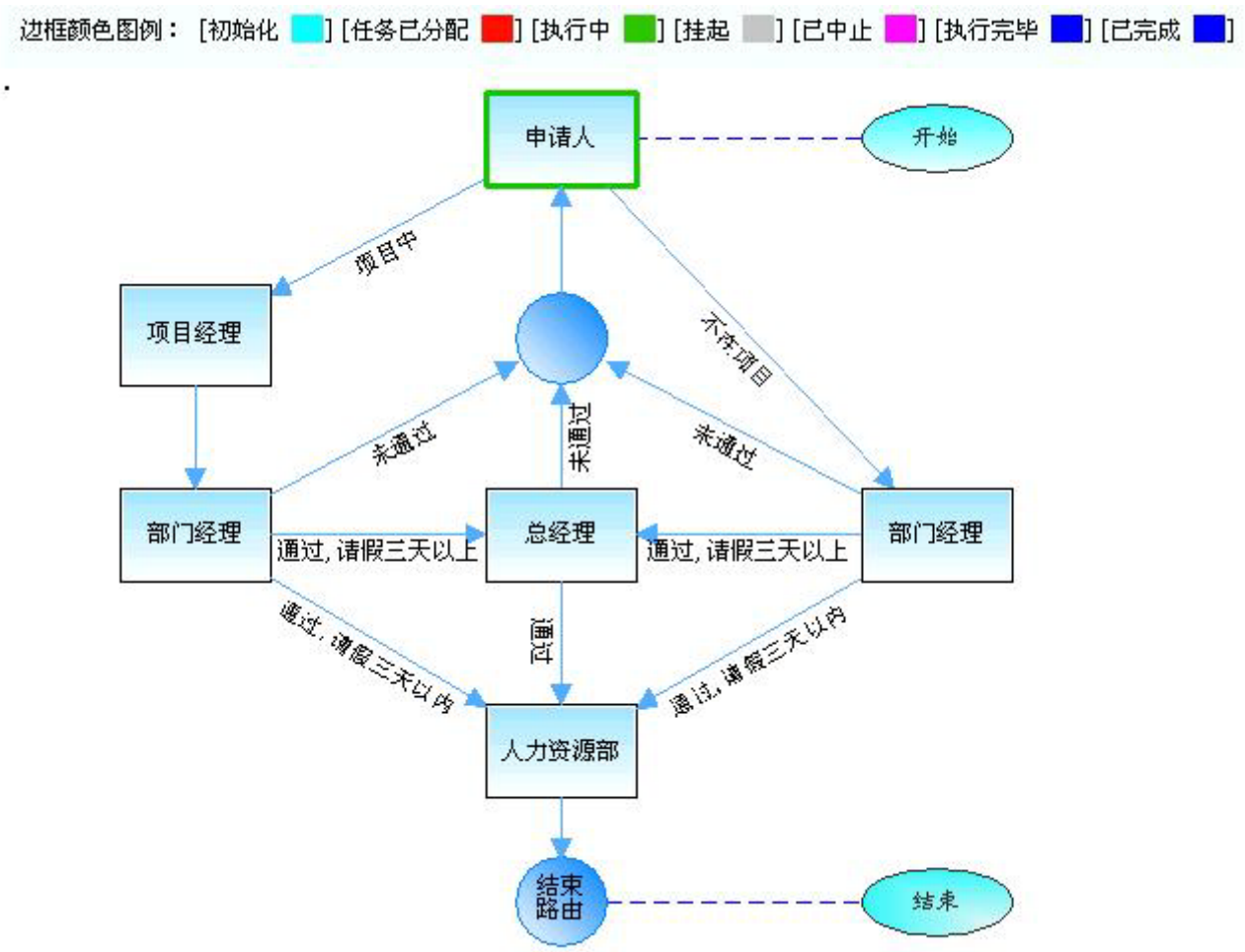
请输入要查询的内容 查询 重置

序	支持事由	支持日期 ▼	详细内容	支
1	Flex使用指导	2010-06-04	提供Flex开发IDE以及使用文档，并对开发人员进行使用指导，对其提出的问题进行	戚
2	培训组织权限和excel导出	2010-06-02	安排培训场所以及培训项目，对开发人员培训组织权限系统和讲理excel导出功能	戚
3	给加贸版ERP系统生成代码	2010-06-01	项目中文名称：加贸版ERP系统 项目英文名称：JM_ERP	马
4	flash播放器使用支持	2010-05-31	需求：Flash播放器支持定制窗口尺寸 解决：通过flex应用与js函数交互的方式实现窗口尺寸的定	戚
5	组织权限API使用支持以及机构树	2010-05-31	需求：根据机构树获得机构信息 解决：说明使用的AP接口，并提供参考实现的jsp代码	戚
6	Flash播放器支持	2010-05-28	需求：在页面中使用flash播放器，按指定尺寸播放视频文件 解决：使用一个开源的flash播放器的	戚
7	中电财项目	2010-05-28	项目中的业务需求转化为流程设计的问题	马
8	陕西联通项目问题	2010-05-28	陕西联通项目问题：业务表单中的数据部分提交，部分审批是否可以实现？表单设	马
9	组织权限admin使用支持,按钮授权	2010-05-28	需求：组织权限的管理员admin拥有最大权限，如何管理	戚

三、流程设计器

2004 年我们开始研发工作流平台，其核心功能是工作流引擎和流程设计器。为了开发出 Web-Based（基于 Web）的流程设计器，我们投入了极大的人力物力，最终采用 ActiveX 控件实现了复杂的流程设计、流程监控等功能。到目前为止基于浏览器的流程设计和监控仍然是我们的功能特色之一。但是随着技术的发展，基于 Activex 控件的流程设计器越来越显示出局限性，例如不支持多浏览器，不支持国际化，在各种 Windows 和 IE 版本中的自动安装经常会出现问题，最重要的是扩展起来比较复杂。

下图就是基于 ActiveX 技术的流程设计器。看上去很美，但的确存在着问题。事实上我们正在积极准备基于 Flex 技术对流程设计器进行改造，改造完成后，上面提到的问题即可迎刃而解。技术的进步带来应用的改进，所以我一直强调，技术创新才是软件企业的原动力。



四、动态图表

通过图表描述业务数据，加强数据的展现能力。每个图表都可以支持参数的动态变化，响应点击事件，实现动态效果，而这些几乎不需要额外编写代码。传统的报表工具或制图工具，例如 BIRT 或 JFreeChart 等，处理报表图片时都是通过流的方式输入静态图片，一旦生成，就是静态页面，用户无法与之交互。而使用 Flex 进行图表的开发，则可以轻易突破这一障碍，Flex 提供了大量内置的图表控件，用来进行图表展示，在浏览器上的表现方式为 Flash，可以在生成图表之后，继续实现前后台数据的交互和展示。例如使用 CandlestickChart 控件来实现蜡烛图的动态展示。从一下两张图可以很清楚的看出，通过选择不同的选项，可以显示不同的趋势变化，点击图元还可以显示该图元的相关数据。这些复杂的数据操作基本上是由 CandlestickChart 控件完成的，开发者只需要组装数据即可。事实上要实现这个功能样例，只需要 100 行左右的代码。

五、知识管理

企业信息化 10 年，积累了大量的数据、文件、视频需要进行展示，通过 Flex 可以构建通用的播放器，利用流媒体技术、全文检索技术实现企业内部的知识管理。类似的技术和功能在互联网上已经不是什么新鲜事了，例如 slideshare、youtube，国内的豆丁、优酷等，这些网站要么是基于文档的管理，要么是基于视频的处理，但它们的共同点是核心技术都采用了 Flex。对于互联网领域，领域的细分无疑是非常明智的，但是在企业级应用领域，企业客户更关注的是应用的整合，我们拟基于 Flex 技术开发通用的前台播放器，可以同事播放视频和文档，后台采用 Flash Midea Server 和文件服务器，结合内容管理、全文检索和标签云技术，实现企业信息数据的全流程管理。

六、动态商品展示

基于 Flex 技术的动态商品展示已经在很多中大型电子商务公司应用，通过 Flex 可以实现很多眩目的效果来进行商品展示，同时可以动态设置商品的属性，让客户更好的了解商品细节，增加客户粘度。这样的技术在企业级应用中同样适用，主要业务领域应该是企业的电子商务平台。

七、全键盘操作

某些特殊领域的客户，比如要快速录入大量数据，就会对全键盘操作有需求，要求在浏览器做的应用要能像 Excel 一样全键盘操作，除了支持 Tab 加 Enter 键之外，还需要能够支持四个方向键的操作支持，就像是 Excel 一样，当单元格中的文字处于全选状态的时候可以通过方向键进行导航。这样的功能用普通的 Javascript 实现一个是复杂，而是会降低网页性能。

但用 Flex 来实现这种特殊场景就完全没由这些问题，本质上来说 Flex 还是客户端技术，可以很好的提供键盘支持。

结束语

企业客户越来越认识到 RIA 技术对于企业应用的重要性，而 Flex 就是实现 RIA 重要的选择之一。但是，无论技术也好，创新也好，有用的才是好的，不能因为技术而使用技术，因为创新而创新。就 Flex 而言，从根本上说它还是一个客户端程序，所以一定会比普通的 Web 页面更多的占用更多的客户端资源，所以我不建议大家在构建企业应用时大量采用 Flex 技术，一定要用其所长。Flex 在互联网领域早已大放异彩，那么它是否能成功应用于企业应用的开发呢？不必拭目以待，我想这个答案无疑是肯定的。

原文链接：http://www.infoq.com/cn/articles/flex-application

相关内容：

- Felix Bachmann谈软件架构评估
- Flex 4 SDK与Flash Builder 4 发布
- Adobe Flex：最近的一系列事件是否会影响到其在企业中的应用？
- Flash开发者需要知道的 10 件事
- Flash vs. HTML5：RIA领域当前的混战

推荐文章 | Articles

Rails 中精彩的图表

作者 Srividya Sharma 译者 杨晨

当分析完数据库中的数据，需要将结果以某种形式展示给用户的时候，有些 Web 应用程序比较偏好采用报告的形式。不过，图表的优点是能够根据报告，给出直观的视觉展现，并且可以帮助开发者观察趋势以及简化数据的综合分析过程。有很多技术能够帮助开发者构建出这类应用程序，Ruby on Rails 就是其中之一。在本文中，我们将会详细介绍如何使用 Ruby on Rails 创建图表。

优秀的软件能够从数据库中的数据或者报告中快速高效地生成图表。Gruff、JFreeChart、XML/SWF charts 和 FusionCharts 是 Ruby on Rails 上为数不多能满足我们需求的软件。不过你得明白，并不是所有这些软件我们都试用和测试过，所以下面的信息大部分是取自产品文档。

	Gruff	JFreeChart	XML/SWF	FusionCharts 免费版	FusionCharts v3（商业版）
类型	Ruby 类	Java Class 文件	Flash 图表组件	Flash 图表组件	Flash 图表组件
操作系统平台	平台独立	平台独立	平台独立	平台独立	平台独立
是否需要 Adobe Flash 插件	不是	不是	是	是	是
是否可用于高级数据分析	**不可以**	**可以**	**可以**	**不可以**	**可以**

是否需要安装软件或者其他插件	**需要，需要安装 RMagick，ImageMagick 和 Gruff**	**需要，JDK 1.3 版或更高**	**不需要**	**不需要**	**不需要**
能否生成动态图表	**不能**	**能**	**能**	**能**	**能**
是否支持输出到图片	**支持**	**支持**	**支持**	**不支持**	**支持**
UTF8 支持	**文档未提及**	**是**	**是**	**是**	**是**
易用性	易用，但是需要写很多代码	并不易用。对用户较友好	很复杂，太多用于配置图表的 xml 标记	非常易用，开发者和用户都可以轻易上手	非常易用，开发者和用户都可以轻易上手
2D/3D 条形图支持	**是**	**是**	**是**	**是**	**是**
折线图支持	**是**	**是**	**是**	**是**	**是**
散点图支持	**是**	**是**	**是**	**是**	**是**
面状图支持	**是**	**是**	**是**	**是**	**是**
2D/3D 饼图支持	**是**	**是**	**是**	**是**	**是**
2D/3D 环状图支持	**不**	**是，以环形图形式**	**是**	**是**	**是**

甘特图支持	不	是	不	是	是（FusionWidgets v3 的一部分）
混合图支持	不	是	是	是	是
仪表图支持	不	是(XML/SWF 标准的一部分）	是	不	是（FusionWidgets v3 的一部分）
实时图支持	不	不	不	不	是（FusionWidgets v3 的一部分）
数据驱动地图	不	不	不	不	是（FusionWidgets v3 的一部分）

本文中，我们将会研究如何使用 FusionCharts 和 MySQL 在 Rails 中绘制图表。FusionCharts 是一个基于 Flash 的图表生成组件，它能够帮助开发者创建可交互的动态图表。同样，它也可以和任何一种 Web 脚本语言绑定，以最少的代码绘制出优秀的图表。不仅如此，FusionCharts 甚至可以封装成模块并在 RoR 中使用。易于使用、多种图表类型支持、优秀的文档以及良好的支持是本文我们介绍 FusionCharts 的原因。

这篇文章以一个应用程序为例，描述了在 RoR 下 FusionCharts 生成图表的机制。这个应用完整运行所需要的软件有：

- FusionCharts 免费版/v3：

  FusionCharts 免费版可以从 www.fusioncharts.com/free 下载。而功能更全面的商业版可以从 www.fusioncharts.com 上下载。在本文中，我们使用的是免费版。

  FusionCharts 的安装需要将安装包中的 SWF 和.rb 文件拷贝到合适的位置。.rb 文件是在 Download Package > Code > RoR > Libraries 目录下。

- Ruby 1.8.6 或更高版本：

  下载地址是 http://www.rubyonrails.org/down

- Rails gem 2.0.2 或更高版本：

  在 RubyGems 加载之后，如果要完整安装 Rails 以及其依赖包，你可以输入如下命令：

```
gem install rails
```

  Rails 也可以从此页面下载。

- MySQL 4.0 或者更高版本：

  MySQL 可以从此处下载。

这里我们以一个计时器为例。这个程序是为员工设计的，员工需要填写每周的上班时间，并且可以查看自己的工时表。本文将会从两个部分来讨论这个应用程序。第一个部分将处理开发上的问题，包括列举、编辑、展示和删除雇员以及工时表。如果你已经有一个需要集成图表的应用程序，你可以跳过这部分，然后直接进入下一个部分，下个部分将会阐述如何将图表和应用程序集成在一起。本文假设读者有基本的 Ruby on Rails 的知识。

创建基本的计时器程序

在脚手架框架（scaffolding framework）的帮助下，一些基本功能，例如列举、编辑、展示和删除雇员以及时间表，可以很简单地生成出来。生成基本的员工和时间表控制器的命令如下：（运行第一个命令后，在创建的 TimeTrackerApplication 目录下运行后面两个命令）

```
rails -d mysql TimeTrackerApplication
ruby script/generate scaffold Employee name:text
ruby script/generate scaffold Timesheet log_date:datetime hours_spent:integer
employee_id:integer
```

下一步，你需要修改 config 文件夹里面的 database.yml，让其指向“timetrackerdb”，并且配置好用户名和密码。运行如下命令创建数据库：

```
rake db:create
rake db:migrate
```

在 mysql 中运行如下命令，创建 employees 和 timetables 表的外键。

```
alter table timesheets add constraint fk_employee_id foreign key
fk_employee_id(employee_id) references employees(id) on delete cascade
```

当然，创建外键的方法有多种；不过如果要深入下去，那么就超出了本文的讨论范围。你需要在模型之间创建关联才能够在 Ruby on Rails 中表示外键。

在 Employee 模型中，在类声明语句后，加入如下语句：

```
has_many :timesheets
```

而在 Timesheet 模型你需要加入：

```
belongs_to :employee
```

运行 sql 脚本 db/sampledata.sql 将样例数据插入到数据库中。至此，基本的程序和数据已经就绪，你可以增加、查看、编辑和删除雇员。

在应用程序中集成图表

当程序就绪之后，下面我将介绍如何生成一名员工的工作时间图表：

- 将 FusionCharts 文件夹（Download Package > Code > FusionCharts）拷贝到 TimeTrackerApplication 的 public 目录下。
- 将 Download Package > Code > FusionCharts 文件夹中的 FusionCharts.js 拷贝到 public/javascripts 文件夹下。
- 将 Download Package > Code > RoR > Libraries 文件夹中的 fusioncharts_helper.rb 拷贝到 TimeTrackerApplication 的 lib 目录下。

至此，FusionCharts 的安装就已经完成了。

员工名单页中需要一个指向 Time Tracker Chart 的链接。这样，每个雇员都可以查看自己的工时图表。链接是在 app/views/employees/index.html.erb 文件中创建。列表 1 详细描述了链接创建的方法。这段代码是把这个链接加到此页面的其他链接之后。

列表 1

```
<td><%=link_to 'Time Chart',
{:action=>'view_timesheet_chart',:id=>employee.id} %></td>
```

图 1 展示的是员工名单页面。在设置完数据库之后，开发者使用 Ruby 脚手架框架，可以很容易地实现这个功能。我们可以访问“http://yourserver:port/employees”来查看此页面。图 1 是屏幕部分截图。

图 1

点击“Time Chart”链接将会调用 employees 控制器的 view_timesheet_chart 动作。列表 2 列出了它的代码。这个动作的作用是展示选中员工的工时表。它会向 Employee 模型查询指定员工在“2008-12-01”至“2008-12-07”这段时间的工时表，得到结果之后，渲染“view_timelog_chart.html.erb”文件。为了简单起见，在本文中，这个时间范围是固定的。当然，现实世界中，这个时间范围应该是可由员工指定。

列表 2

```
EmployeesController
def view_timesheet_chart
   start_date= "2008-12-01"
   end_date="2008-12-07"
   @employee_id = params[:id]
   employee_details_with_timesheets                                          =
Employee.find_with_timesheets_in_date_range(@employee_id,start_date,end_dat
e)
   if(!employee_details_with_timesheets.nil?)
      @employee_details_with_timesheets =employee_details_with_timesheets[0]
   else
      @employee_details_with_timesheets =nil;
   end
   headers["content-type"]="text/html"
end
```

这个动作以员工 id 为参数，它向 Employee 模型查询指定员工在特定时间范围内的工时表。我们将在 Employee 模型中加入如列表 3 所示的函数：

列表 3

Employee.rb

```
def self.find_with_timesheets_in_date_range(id, start_date, end_date)
   conditions="employees.id =? and log_date between ? and ?"

employee_details_with_timesheets=self.find(:all, :include=>'timesheets', :c
```

```
onditions=> [conditions,id,start_date,end_date], :order=>'log_date asc')
  return employee_details_with_timesheets
end
```

最后，这个动作会渲染“view_time_chart.html.erb”文件。 “view_time_chart.html.erb”模板使用的布局模板是“employee.html.erb”，所有 employees 控制器的视 图都会采用这个布局模板。

列表 4

view_timesheet_chart.html.erb （在 app/views/employees 文件夹中）

```
<%= javascript_include_tag "FusionCharts"%>
<%
#xml文件内容以字符串的形式从模板生成器中得到
str_xml = render "employees/timesheet_data"
#创建图表 - 3D柱状图，它的数据来自于strXML
render_chart '/FusionCharts/Column3D.swf' , '' , str_xml , 'TimeChart' ,  650 ,
400 ,  false ,  false do -%>
<%  end -%>
```

从上面的代码，我们知道了如何在视图页面中渲染图表：

- 引入 FusionCharts.js 文件。
- 从生成器中获取 xml 数据.（生成器的数据来源是控制器）
- 使用合适的参数调用 render_chart 函数渲染图表，

通常情况下，图表会以甘特图的形式表示，当然 FusionCharts 也支持这种形式。在此，简单起见，我们使用了柱状图。

第二步和 XML 的生成有关。FusionCharts 使用 XML（扩展标记语言）创建和操作图表。整个 FusionCharts 图表都是由 XML 控制。例如，你可以使用 XML 定义图表的外观，也可以定义图表的功能。每种图表都有大量特性可供选择。FusionCharts 也有自己的 XML 结构。现在讨论的图表是单系列图表（只有一个数据集），表示员工每周工作的天数和每天工作的时间。列表 5 是这个图表可以使用的一个 XML 例子。

列表 5

单系列图表 XML 样例

```
<graph   xAxisName="Day"   showValues="1"   caption="Time   Tracker   Chart"
numberSuffix=" hrs." subcaption="For Employee John Wilson" yAxisName="Hours
Spent">
```

```
   <set name="Monday" value="8" color="AFD8F8"/>
   <set name="Tuesday" value="6" color="F6BD0F"/>
   <set name="Wednesday" value="5" color="8BBA00"/>
   <set name="Thursday" value="9" color="A66EDD"/>
   <set name="Friday" value="9" color="F984A1"/>
   <set name="Saturday" value="8" color="CCCC00"/>
</graph>
```

要用现有的数据生成和列表 5 类似的 xml 文件，那么就需要用到生成器。

生成器是如何生成图表 XML 的呢？它会用到控制器动作中的 @employee_details_with_timesheets 变量来构建出图表 XML。列表 6 展示的是这个生成器的实现。

列表 6

timesheet_data.builder （在 app/views/employees 文件夹中）

```
xml = Builder::XmlMarkup.new(:indent=>0)
options = {
   :caption=>'Time Tracker Chart',
   :subcaption=>'For Employee '+ @employee_details_with_timesheets.name   ,
   :yAxisName=>'Hours Spent', :xAxisName=>'Day',
   :showValues=>'1',
   :formatNumberScale=>'0',
   :numberSuffix=>' hrs.'
}
xml.graph(options) do
   for timesheet in @employee_details_with_timesheets.timesheets do
      log_day = timesheet.log_date.strftime('%a')
      xml.set(
         :name=>log_day,
         :value=>timesheet.hours_spent,
         :color=>''+get_FC_color)
   end
end
```

在生成器中，你首先要创建一个 XMLMarkup 对象，将其“indent”属性赋值为 0，然后可以开始使用这个对象来构建 XML 了。

配置图表的所有选项都被放到了一个名为 options 的散列表中。如果希望了解更多 options 的信息，可以参考完整的 FusionCharts 文档。XML 文件的 root 元素是“graph”，其子节点是“set”，一个“set”节点表示一个特定员工的工时表。每一个“set”元素都有 name 和 value 属性。name 字段表示一周的第几天，而 value 字段表示的是在这一天中，当前员工的工作小时数。生成器使用控制器动作中的@employee_details_with_timesheets 变量来存储这些值。

最后，在视图页中，render_chart 函数将会渲染出图表。这个函数的参数是 swf 图表名称、url、width、height、debugMode 和 registerWithJS。它可以在 lib 文件夹的 fusioncharts_helper

模块中找到。正如列表 7 所示，引入 application_helper.rb 之后，就可以在所有的视图中使用这个函数。

列表 7

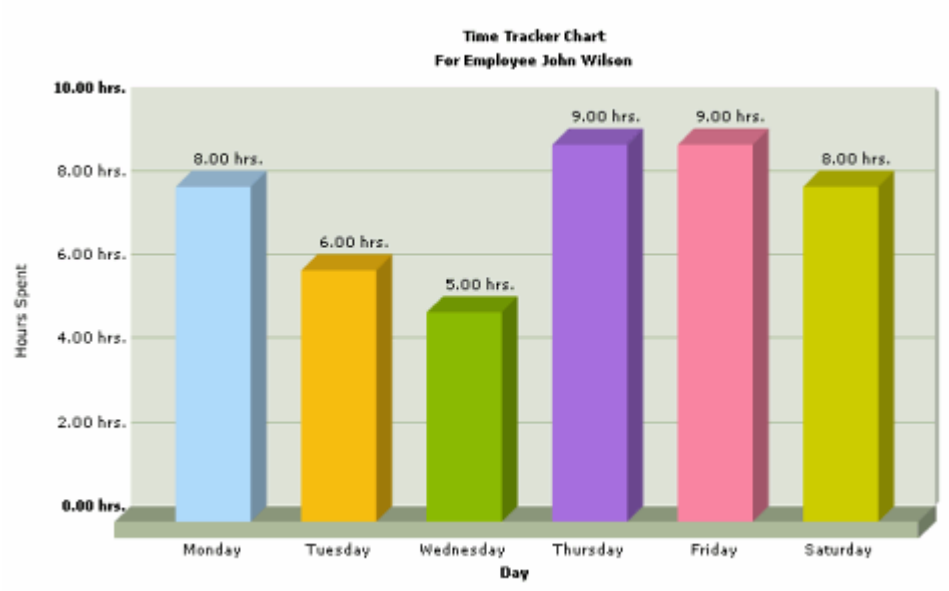
application_helper.rb

```
require 'fusioncharts_helper'include FusionChartsHelper
```

展示图表的 swf 文件在 public/FusionCharts 文件夹中。这里我们已经使用了 Column3D.swf 这个文件名，但是也可以生成任意名字的其他任何单系列的图表。而且，数据不仅仅可以从 dataXML 方法得到，也可以从 dataURL 中获取。这种用法在 FusionCharts 的文档中有举例。

现在，在员工列表页面中点击一个员工的“Time Chart”链接，从 01/12/2008 到 07/12/2008 这一周的工作时间将会以图表的形式展现出来。就像图 2 一样。

图 2



如何生成图表概述：

设置完 FusionCharts，并且引入了 application_helper.rb 中的 FusionChartsHelper 之后，按照如下步骤生成图表：

- 控制器动作负责从数据库中寻找所需数据。
- 根据当前数据编写生成器。
- 在第一步提到的控制器动作所绑定的视图中，开发者有两种方式从生成器中取得 XML，

一种是直接将数据传递给生成器，或者让生成器从控制器动作中获取数据。

- 下一步，使用合理的参数（包括从第三步中得到的 XML）调用 render_chart 函数。

通过一个计时器的讲解，你已经学到了如何在 Ruby on Rails 中使用 FusionCharts 创建简单的图表。而且，只要对这个程序进行简单的修改，你就能得到一个所有员工的工时图表。或者，你甚至可以深入下去，绘制出某个特定员工的详细工时表。当然，你也可以使用 Ajax 来获取图表。诸如此类的改进数不胜数。FusionCharts 的下载包中就包含了很多值得深入研究的例子。本文讲解的应用程序可以从这里下载。

原文链接：http://www.infoq.com/cn/articles/sharma-charts-in-rails

相关内容：

- 基于 Flash 的软件开发
- 构建于Rails之上的Hobo 1.0
- TorqueBox：JVM上的Rails企业级解决方案
- Rails最佳实践
- 张文钿谈 Rails 开发及台湾 Ruby 社区

推荐文章 | Articles

REST 和 SOAP：谁更好，或者都好？

作者 Mike Rozlog 译者 徐涵

如今，Web 开发者的可选技术相当之多；从简化的数据库访问技术，到易用的中间件服务包装技术，以及各种有趣的客户端软件等等，一应俱全。所有这些产品和工具，都是为了帮助 Web 开发者用最快的速度开发出最好的 Web 应用。

然而，拥有大量可选软件方案以及为 Web 应用的特定部分选用特定方案，都是具有挑战的事；而且，现在 Web 开发者必须持续跟踪各种不断变化着的标准与方法。

举个例子，Web 服务技术就有 SOAP（Simple Object Access Protocol，简单对象访问协议）和 REST（Representational State Transfer，表示性状态转移）这两种方案。它们都是有效的方案，但在具体场合下采用哪种方案好，这要取决于 Web 开发者。

目前，大部分 Web 开发者似乎都了解 REST——一种采用标准 URI 进行调用的方案。REST 很容易理解，而且只要是支持 HTTP/HTTPS 的客户端/服务器就支持它。你可以用 HTTP GET 方法来执行命令。所以，开发者们感受到的 REST 的优势是：开发简单、只需依托现有 Web 基础设施、以及学习成本低。

然而，SOAP 作为一种古老的 Web 服务技术，短期内还不会退出历史舞台。而且，随着 SOAP 1.2 的出现，SOAP 印象中的一些缺点已得到改进，采纳率和易用程度也都得到提高。另需注意的是，从 W3C SOAP 1.2 版开始，SOAP 这一缩写不再代表 Simple Object Access Protocol（简单对象访问协议），而是仅仅作为协议名称而已。

相对 REST 而言，SOAP 1.2 多出一些开销，不过这些开销也带来了一些好处。首先，SOAP 在三个方面离不开 XML（Extensible Markup Language，可扩展标记语言）：SOAP 信封（envelope）是基于 XML 的，它定义了消息里有什么以及如何处理它；一套用于数据类型的编码规则；过程调用和响应的规划。SOAP 信封由传输协议（HTTP/HTTPS）发出，RPC（Remote Procedure Call，远程过程调用）得到执行，然后一个 XML 文档随 SOAP 信封返回。

需要注意的是，基于“通用”传输协议是 SOAP 的一个优点。REST 目前基于 HTTP/HTTPS；而 SOAP 可支持任何传输协议，从 HTTP /HTTPS 到 SMTP（Simple Mail Transfer Protocol，简单邮件传送协议），甚至 JMS（Java Messaging Service，Java 消息传递服务）。不过，由于 XML 较为冗长且解析费时，因此采用 XML 也成为一个弊端。

不过，对 Web 开发者来说的好消息是，目前上述两种方案都是行之有效的方案。REST 和 SOAP 都能解决许多 Web 方面的问题与挑战，而且在许多情况下，它们各自都能满足开发者的要求，也就是说可互换使用。

但很多人不知道，这两种技术可以混合搭配使用。REST 很好理解，且极易上手；不过由于它缺乏标准，因此只被看作是一种架构方法。与之相比，SOAP 是一个工业标准，它具备良好定义的协议，以及一套良好确立的规则，在大型和小型系统中均有采用。

因此，REST 的适用场合包括：

- **有限的带宽和资源**：别忘了返回的结构可以采用（由开发者定义的）任何格式。另外，由于 REST 采用标准的 GET、PUT、POST 和 DELETE 动词，因此可被任何浏览器所支持。除此以外，REST 还可以使用为目前大多数浏览器支持的 XMLHttpRequest 对象，这为 AJAX 增色不少。
- **完全无状态的操作**：对于那些需多步执行的操作，REST 并非最佳选择，采用 SOAP 更合适。但是，如果你需要无状态的 CRUD（Create/Read/Update/Delete，创建/读取/更新/删除）操作，那么应采用 REST。
- **缓存考虑**：若要利用无状态操作的特性，使得信息可被缓存，那么 REST 是很好的选择。

以上已经覆盖很多方案了，那么，为什么还要考虑 SOAP 呢？正如前述，SOAP 比较成熟而且是经过良好定义的，具有完整的规范。而 REST 只不过是一种方法，对开发未作任何规约；因此，假如你遇到以下场合，那么 SOAP 是最佳选择：

- **异步处理与调用**：如果你的应用需要确保可靠性与安全性，那么请采用 SOAP。SOAP 1.2 为确保这种操作补充定义了 WSRM（WS-Reliable Messaging）等标准。
- **形式化契约**：若提供者/消费者双方必须就交换格式取得一致，那么采用 SOAP 更合适。SOAP 1.2 为这种交互提供了严格的规范。
- **有状态的操作**：如果应用需要上下文信息与对话状态管理，那么应采用 SOAP。SOAP 1.2 为此补充定义了 WS-Security、WS-Transactions 和 WS-Coordination 等标准。相比之下，REST 方法要求开发者自己来实现这些框架性工作。

正如你所看到的，REST 和 SOAP 各自有其用武之地。它们在安全性和传输层等方面有着自己的潜在问题，不过它们都能完成任务，而且在许多情况下，它们都丰富了 Web 的技术手段。因此，就这一论题，其实最好的原则就是灵活性原则；因为至少在现今的 Web 开发中，无论面对何种问题，Web 开发者们总有办法运用好这两种技术中的一种。

关于作者

Mike Rozlog 是 Embarcadero 科技公司的高级产品主管。他的主要工作是确保 Embarcadero 公司推出开发工具满足全世界开发者们的期望。他的大部分时间被致力于从技术和业务两个方面来介绍讲解 Embarcadero 的产品与服务，其听众是遍布全球分析师及其他听众。Mike 曾工作于 CodeGear，一个于 2008 年被 Embarcadero 收购的开发工具团队。之前，他为 Borland 公司工作了八年，担任过包括首席技术架构师在内的许多职位。作为一名知名作者，Mike 出版了很多书。他最近的作品《Mastering JBuilder》已由 John Wiley & Sons 出版。

原文链接：http://www.infoq.com/cn/articles/rest-soap-when-to-use-each

相关内容：

- SOA 反原则?
- RESTful HTTP 的实践
- 处理 REST 服务安全
- RESTful 服务的版本控制策略
- SOA 执行者应该首先定义标准

推荐文章 | Articles

又拍网架构中的分库设计

作者 周兆兆

又拍网是一个照片分享社区，从 2005 年 6 月至今积累了 260 万用户，1.1 亿张照片，目前的日访问量为 200 多万。5 年的发展历程里经历过许多起伏，也积累了一些经验，在这篇文章里，我要介绍一些我们在技术上的积累。

又拍网和大多数 Web2.0 站点一样，构建于大量开源软件之上，包括 MySQL、PHP、nginx、Python、memcached、redis、Solr、Hadoop 和 RabbitMQ 等等。又拍网的服务器端开发语言主要是 PHP 和 Python，其中 PHP 用于编写 Web 逻辑（通过 HTTP 和用户直接打交道），而 Python 则主要用于开发内部服务和后台任务。在客户端则使用了大量的 Javascript，这里要感谢一下 MooTools 这个 JS 框架，它使得我们很享受前端开发过程。另外，我们把图片处理过程从 PHP 进程里独立出来变成一个服务。这个服务基于 nginx，但是是作为 nginx 的一个模块而开放 REST API。

图 1：开发语言

由于 PHP 的单线程模型，我们把耗时较久的运算和 I/O 操作从 HTTP 请求周期中分离出来，交给由 Python 实现的任务进程来完成，以保证请求响应速度。这些任务主要包括：邮件发送、数据索引、数据聚合和好友动态推送（稍候会有介绍）等等。通常这些任务由用户触发，并且，用户的一个行为可能会触发多种任务的执行。比如，用户上传了一张新的照片，我们需要更新索引，也需要向他的朋友推送一条新的动态。PHP 通过消息队列（我们用的是 RabbitMQ）来触发任务执行。

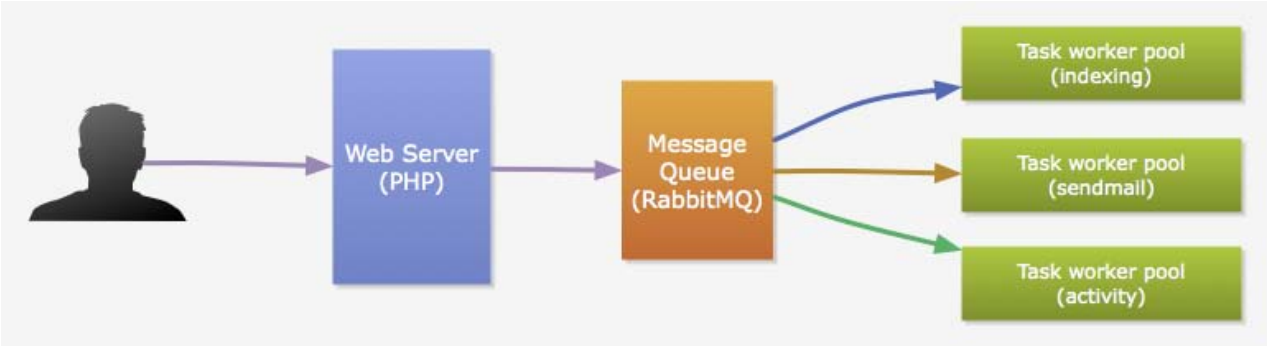


图 2：PHP 和 Python 的协作

数据库一向是网站架构中最具挑战性的，瓶颈通常出现在这里。又拍网的照片数据量很大，数据库也几度出现严重的压力问题。因此，这里我主要介绍一下又拍网在分库设计这方面的一些尝试。

分库设计

和很多使用 MySQL 的 2.0 站点一样，又拍网的 MySQL 集群经历了从最初的一个主库一个从库、到一个主库多个从库、然后到多个主库多个从库的一个发展过程。

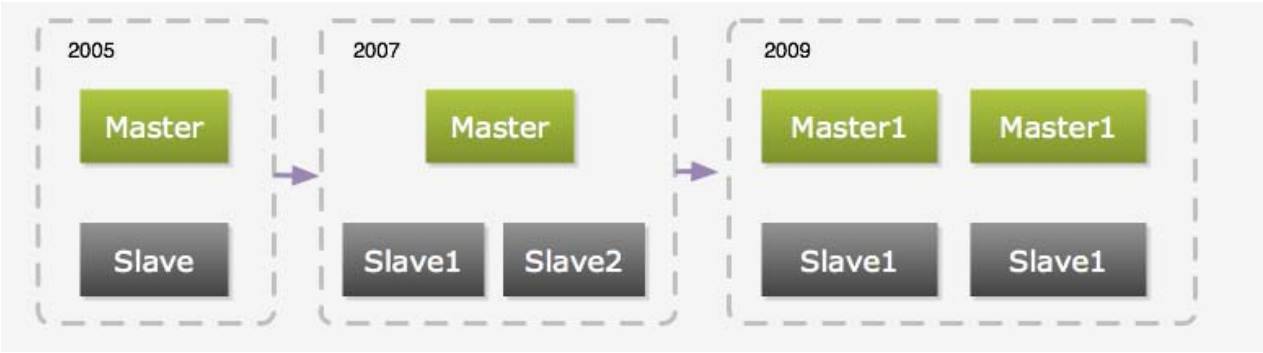


图 3：数据库的进化过程

最初是由一台主库和一台从库组成，当时从库只用作备份和容灾，当主库出现故障时，从库就手动变成主库，一般情况下，从库不作读写操作（同步除外）。随着压力的增加，我们加上了 memcached，当时只用其缓存单行数据。但是，单行数据的缓存并不能很好地解决压力问题，因为单行数据的查询通常很快。所以我们把一些实时性要求不高的 Query 放到从库去执行。后面又通过添加多个从库来分流查询压力，不过随着数据量的增加，主库的写压力也越来越大。

在参考了一些相关产品和其它网站的做法后，我们决定进行数据库拆分。也就是将数据存放到不同的数据库服务器中，一般可以按两个纬度来拆分数据:

- **垂直拆分**：是指按功能模块拆分，比如可以将群组相关表和照片相关表存放在不同的数据库中，这种方式多个数据库之间的表结构不同。
- **水平拆分**：而水平拆分是将同一个表的数据进行分块保存到不同的数据库中，这些数据库中的表结构完全相同。

拆分方式

一般都会先进行垂直拆分，因为这种方式拆分方式实现起来比较简单，根据表名访问不同的数据库就可以了。但是垂直拆分方式并不能彻底解决所有压力问题，另外，也要看应用类型是否合适这种拆分方式。如果合适的话，也能很好的起到分散数据库压力的作用。比如对于豆瓣我觉得比较适合采用垂直拆分，因为豆瓣的各核心业务/模块（书籍、电影、音乐）相对独立，数据的增加速度也比较平稳。不同的是，又拍网的核心业务对象是用户上传的照片，而照片数据的增加速度随着用户量的增加越来越快。压力基本上都在照片表上，显然垂直拆分并不能从根本上解决我们的问题，所以，我们采用水平拆分的方式。

拆分规则

水平拆分实现起来相对复杂，我们要先确定一个拆分规则，也就是按什么条件将数据进行切分。一般 2.0 网站都以用户为中心，数据基本都跟随用户，比如用户的照片、朋友和评论等等。因此一个比较自然的选择是根据用户来切分。每个用户都对应一个数据库，访问某个用户的数据时，我们要先确定他/她所对应的数据库，然后连接到该数据库进行实际的数据读写。

那么，怎么样对应用户和数据库呢？我们有这些选择：

按算法对应

最简单的算法是按用户 ID 的奇偶性来对应，将奇数 ID 的用户对应到数据库 A，而偶数 ID 的用户则对应到数据库 B。这个方法的最大问题是，只能分成两个库。另一个算法是按用户 ID 所在区间对应，比如 ID 在 0-10000 之间的用户对应到数据库 A，ID 在 10000-20000 这个范围的对应到数据库 B，以此类推。按算法分实现起来比较方便，也比较高效，但是不能满足后续的伸缩性要求，如果需要增加数据库节点，必需调整算法或移动很大的数据集，比较难做到在不停止服务的前提下进行扩充数据库节点。

按索引/映射表对应

这种方法是指建立一个索引表，保存每个用户的 ID 和数据库 ID 的对应关系，每次读写用户数据时先从这个表获取对应数据库。新用户注册后，在所有可用的数据库中随机挑选一个为其建立索引。这种方法比较灵活，有很好的伸缩性。一个缺点是增加了一次数据库访问，所以性能上没有按算法对应好。

比较之后，我们采用的是索引表的方式，我们愿意为其灵活性损失一些性能，更何况我们还有 memcached，因为索引数据基本不会改变的缘故，缓存命中率非常高。所以能很大程度上减少了性能损失。

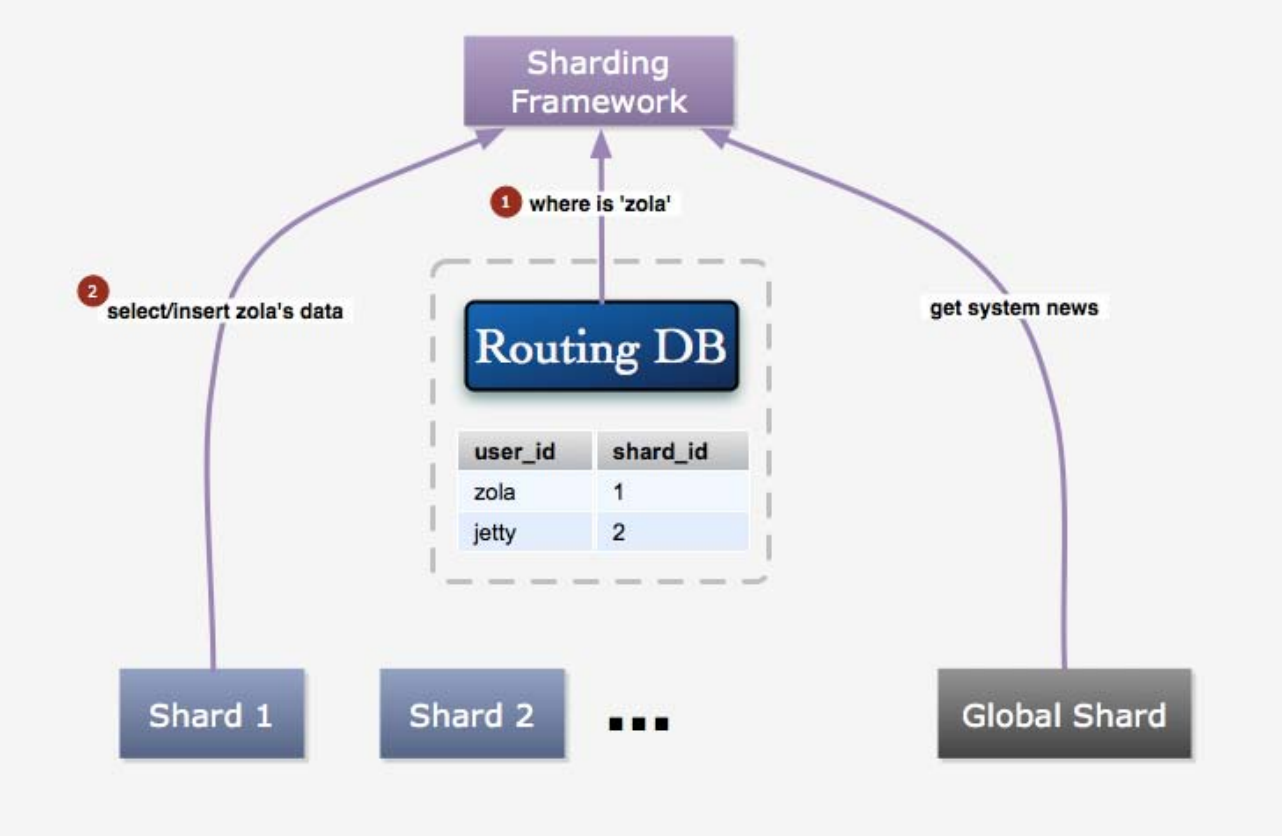


图 4：数据访问过程

索引表的方式能够比较方便地添加数据库节点，在增加节点时，只要将其添加到可用数据库列表里即可。当然如果需要平衡各个节点的压力的话，还是需要进行数据的迁移，但是这个时候的迁移是少量的，可以逐步进行。要迁移用户 A 的数据，首先要将其状态置为迁移数据中，这个状态的用户不能进行写操作，并在页面上进行提示。然后将用户 A 的数据全部复制到新增加的节点上后，更新映射表，然后将用户 A 的状态置为正常，最后将原来对应的数据库上的数据删除。这个过程通常会在临晨进行，所以，所以很少会有用户碰到迁移数

据中的情况。

当然，有些数据是不属于某个用户的，比如系统消息、配置等等，我们把这些数据保存在一个全局库中。

问题

分库会给你在应用的开发和部署上都带来很多麻烦。

不能执行跨库的关联查询

如果我们需要查询的数据分布于不同的数据库，我们没办法通过 JOIN 的方式查询获得。比如要获得好友的最新照片，你不能保证所有好友的数据都在同一个数据库里。一个解决办法是通过多次查询，再进行聚合的方式。我们需要尽量避免类似的需求。有些需求可以通过保存多份数据来解决，比如 User-A 和 User-B 的数据库分别是 DB-1 和 DB-2，当 User-A 评论了 User-B 的照片时，我们会同时在 DB-1 和 DB-2 中保存这条评论信息，我们首先在 DB-2 中的 photo_comments 表中插入一条新的记录，然后在 DB-1 中的 user_comments 表中插入一条新的记录。这两个表的结构如下图所示。这样我们可以通过查询 photo_comments 表得到 User-B 的某张照片的所有评论，也可以通过查询 user_comments 表获得 User-A 的所有评论。另外可以考虑使用全文检索工具来解决某些需求，我们使用 Solr 来提供全站标签检索和照片搜索服务。

photo_comments

column	type
photo_id	int
comment_id	int
author_id	int
posted_at	datetime
content	text

user_comments

column	type
user_id	int
comment_id	int
photo_owner_id	int
photo_id	int
posted_at	datetime

图 5：评论表结构

不能保证数据的一致/完整性

跨库的数据没有外键约束，也没有事务保证。比如上面的评论照片的例子，很可能出现成功

插入 photo_comments 表，但是插入 user_comments 表时却出错了。一个办法是在两个库上都开启事务，然后先插入 photo_comments，再插入 user_comments，然后提交两个事务。这个办法也不能完全保证这个操作的原子性。

所有查询必须提供数据库线索

比如要查看一张照片，仅凭一个照片 ID 是不够的，还必须提供上传这张照片的用户的 ID（也就是数据库线索），才能找到它实际的存放位置。因此，我们必须重新设计很多 URL 地址，而有些老的地址我们又必须保证其仍然有效。我们把照片地址改成 /photos/{username}/{photo_id}/的形式，然后对于系统升级前上传的照片 ID，我们又增加一张映射表，保存 photo_id 和 user_id 的对应关系。当访问老的照片地址时，我们通过查询这张表获得用户信息,然后再重定向到新的地址。

自增 ID

如果要在节点数据库上使用自增字段，那么我们就不能保证全局唯一。这倒不是很严重的问题，但是当节点之间的数据发生关系时，就会使得问题变得比较麻烦。我们可以再来看看上面提到的评论的例子。如果 photo_comments 表中的 comment_id 的自增字段，当我们在 DB-2.photo_comments 表插入新的评论时，得到一个新的 comment_id，假如值为 101，而 User-A 的 ID 为 1，那么我们还需要在 DB-1.user_comments 表中插入(1,101...)。User-A 是个很活跃的用户，他又评论了 User-C 的照片，而 User-C 的数据库是 DB-3。很巧的是这条新评论的 ID 也是 101，这种情况很用可能发生。那么我们又在 DB-1.user_comments 表中插入一行像这样(1,101...)的数据。那么我们要怎么设置 user_comments 表的主键呢（标识一行数据）？可以不设啊，不幸的是有的时候（框架、缓存等原因）必需设置。那么可以以 user_id、comment_id 和 photo_id 为组合主键，但是 photo_id 也有可能一样（的确很巧）。看来只能再加上 photo_owner_id 了，但是这个结果又让我们实在有点无法接受，太复杂的组合键在写入时会带来一定的性能影响，这样的自然键看起来也很不自然。所以，我们放弃了在节点上使用自增字段，想办法让这些 ID 变成全局唯一。为此增加了一个专门用来生成 ID 的数据库，这个库中的表结构都很简单，只有一个自增字段 id。当我们要插入新的评论时，我们先在 ID 库的 photo_comments 表里插入一条空的记录，以获得一个唯一的评论 ID。当然这些逻辑都已经封装在我们的框架里了，对于开发人员是透明的。为什么不用其它方案呢，比如一些支持 incr 操作的 Key-Value 数据库。我们还是比较放心把数据放在 MySQL 里。另外，我们会定期清理 ID 库的数据，以保证获取新 ID 的效率。

实现

我们称前面提到的一个数据库节点为 Shard，一个 Shard 由两个台物理服务器组成，我们称它们为 Node-A 和 Node-B，Node-A 和 Node-B 之间是配置成 Master-Master 相互复制的。虽然是 Master-Master 的部署方式，但是同一时间我们还是只使用其中一个，原因是复制的延迟问题，当然在 Web 应用里，我们可以在用户会话里放置一个 A 或 B 来保证同一用户一次会话里只访问一个数据库，这样可以避免一些延迟问题。但是我们的 Python 任务是没有任何状态的，不能保证和 PHP 应用读写相同的数据库。那么为什么不配置成 Master-Slave 呢？我们觉得只用一台太浪费了，所以我们在每台服务器上都创建多个逻辑数据库。如下图所示，在 Node-A 和 Node-B 上我们都建立了 shard_001 和 shard_002 两个逻辑数据库，Node-A 上的 shard_001 和 Node-B 上的 shard_001 组成一个 Shard，而同一时间只有一个逻辑数据库处于 Active 状态。这个时候如果需要访问 Shard-001 的数据时，我们连接的是 Node-A 上的 shard_001，而访问 Shard-002 的数据则是连接 Node-B 上的 shard_002。以这种交叉的方式将压力分散到每台物理服务器上。以 Master-Master 方式部署的另一个好处是，我们可以不停止服务的情况下进行表结构升级，升级前先停止复制，升级 Inactive 的库，然后升级应用，再将已经升级好的数据库切换成 Active 状态，原来的 Active 数据库切换成 Inactive 状态，然后升级它的表结构，最后恢复复制。当然这个步骤不一定适合所有升级过程，如果表结构的更改会导致数据复制失败，那么还是需要停止服务再升级的。

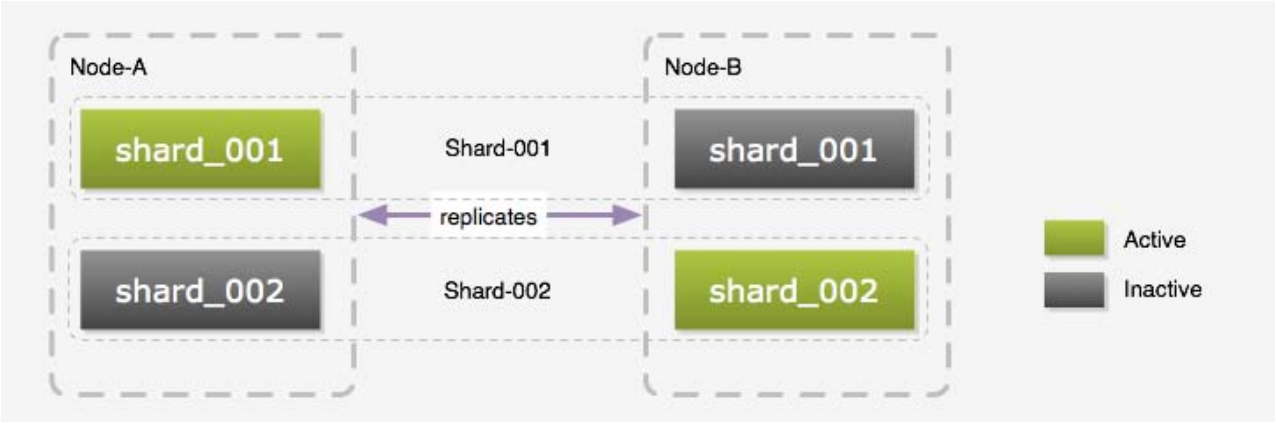


图 6：数据库布局

前面提到过添加服务器时，为了保证负载的平衡，我们需要迁移一部分数据到新的服务器上。为了避免短期内迁移的必要，我们在实际部署的时候，每台机器上部署了 8 个逻辑数据库，添加服务器后，我们只要将这些逻辑数据库迁移到新服务器就可以了。最好是每次添加一倍的服务器，然后将每台的 1/2 逻辑数据迁移到一台新服务器上，这样能很好的平衡负载。当然，最后到了每台上只有一个逻辑库时，迁移就无法避免了，不过那应该是比较久远的事情

了。

我们把分库逻辑都封装在我们的 PHP 框架里了，开发人员基本上不需要被这些繁琐的事情困扰。下面是使用我们的框架进行照片数据的读写的一些例子:

```
<?php
   $Photos = new ShardedDBTable('Photos', 'yp_photos', 'user_id', array(
              'photo_id'     => array('type' => 'long', 'primary' => true,
'global_auto_increment' => true),
              'user_id'     => array('type' => 'long'),
              'title'      => array('type' => 'string'),
              'posted_date' => array('type' => 'date'),
          ));

   $photo = $Photos->new_object(array('user_id' => 1, 'title' => 'Workforme'));
   $photo->insert();

   // 加载ID为10001的照片，注意第一个参数为用户ID
   $photo = $Photos->load(1, 10001);

   // 更改照片属性
   $photo->title = 'Database Sharding';
   $photo->update();

   // 删除照片
   $photo->delete();

   // 获取ID为1的用户在2010-06-01之后上传的照片
   $photos = $Photos->fetch(array('user_id' => 1, 'posted_date__gt' =>
'2010-06-01'));
?>
```

首先要定义一个 ShardedDBTable 对象，所有的 API 都是通过这个对象开放。第一个参数是对象类型名称，如果这个名称已经存在，那么将返回之前定义的对象。你也可以通过 get_table('Photos')这个函数来获取之前定义的 Table 对象。第二个参数是对应的数据库表名，而第三个参数是数据库线索字段，你会发现在后面的所有 API 中全部需要指定这个字段的值。第四个参数是字段定义，其中 photo_id 字段的 global_auto_increment 属性被置为 true，这就是前面所说的全局自增 ID，只要指定了这个属性，框架会处理好 ID 的事情。

如果我们要访问全局库中的数据，我们需要定义一个 DBTable 对象。

```
<?php
   $Users = new DBTable('Users', 'yp_users', array(
              'user_id'    => array('type' => 'long', 'primary' => true,
'auto_increment' => true),
              'username' => array('type' => 'string'),
          ));
?>
```

DBTable 是 ShardedDBTable 的父类，除了定义时参数有些不同（DBTable 不需要指定数据库线索字段），它们提供一样的 API。

缓存

我们的框架提供了缓存功能，对开发人员是透明的。

```
<?php
   $photo = $Photos->load(1, 10001);
?>
```

比如上面的方法调用，框架先尝试以 Photos-1-10001 为 Key 在缓存中查找，未找到的话再执行数据库查询并放入缓存。当更改照片属性或删除照片时，框架负责从缓存中删除该照片。这种单个对象的缓存实现起来比较简单。稍微麻烦的是像下面这样的列表查询结果的缓存。

```
<?php
   $photos = $Photos->fetch(array('user_id' => 1, 'posted_date__gt' =>
'2010-06-01'));
?>
```

我们把这个查询分成两步，第一步先查出符合条件的照片 ID，然后再根据照片 ID 分别查找具体的照片信息。这么做可以更好的利用缓存。第一个查询的缓存 Key 为 Photos-list-{shard_key}-{md5(查询条件 SQL 语句)}，Value 是照片 ID 列表（逗号间隔）。其中 shard_key 为 user_id 的值 1。目前来看，列表缓存也不麻烦。但是如果用户修改了某张照片的上传时间呢，这个时候缓存中的数据就不一定符合条件了。所以，我们需要一个机制来保证我们不会从缓存中得到过期的列表数据。我们为每张表设置了一个 revision，当该表的数据发生变化时（调用 insert/update/delete 方法），我们就更新它的 revision，所以我们把列表的缓存 Key 改为 Photos-list-{shard_key}-{md5(查询条件 SQL 语句)}-{revision}，这样我们就不会再得到过期列表了。

revision 信息也是存放在缓存里的，Key 为 Photos-revision。这样做看起来不错，但是好像列表缓存的利用率不会太高。因为我们是以整个数据类型的 revision 为缓存 Key 的后缀，显然这个 revision 更新的非常频繁，任何一个用户修改或上传了照片都会导致它的更新，哪怕那个用户根本不在我们要查询的 Shard 里。要隔离用户的动作对其他用户的影响，我们可以通过缩小 revision 的作用范围来达到这个目的。所以 revision 的缓存 Key 变成 Photos-{shard_key}-revision，这样的话当 ID 为 1 的用户修改了他的照片信息时，只会更新

Photos-1-revision 这个 Key 所对应的 revision。

因为全局库没有 shard_key，所以修改了全局库中的表的一行数据，还是会导致整个表的缓存失效。但是大部分情况下，数据都是有区域范围的，比如我们的帮助论坛的主题帖子，帖子属于主题。修改了其中一个主题的一个帖子，没必要使所有主题的帖子缓存都失效。所以我们在 DBTable 上增加了一个叫 isolate_key 的属性。

```
<?php
$GLOBALS['Posts'] = new DBTable('Posts', 'yp_posts', array(
       'topic_id'    => array('type' => 'long', 'primary' => true),
       'post_id'         =>  array('type'  =>  'long',  'primary'  =>  true,
'auto_increment' => true),
       'author_id'   => array('type' => 'long'),
       'content'    => array('type' => 'string'),
       'posted_at'   => array('type' => 'datetime'),
       'modified_at' => array('type' => 'datetime'),
       'modified_by' => array('type' => 'long'),
   ), 'topic_id');
?>
```

注意构造函数的最后一个参数 topic_id 就是指以字段 topic_id 作为 isolate_key，它的作用和 shard_key 一样用于隔离 revision 的作用范围。

ShardedDBTable 继承自 DBTable，所以也可以指定 isolate_key。ShardedDBTable 指定了 isolate_key 的话，能够更大幅度缩小 revision 的作用范围。比如相册和照片的关联表 yp_album_photos，当用户往他的其中一个相册里添加了新的照片时，会导致其它相册的照片列表缓存也失效。如果我指定这张表的 isolate_key 为 album_id 的话，我们就把这种影响限制在了本相册内。

我们的缓存分为两级，第一级只是一个 PHP 数组，有效范围是 Request。而第二级是 memcached。这么做的原因是，很多数据在一个 Request 周期内需要加载多次，这样可以减少 memcached 的网络请求。另外我们的框架也会尽可能的发送 memcached 的 gets 命令来获取数据，从而减少网络请求。

总结

这个架构使得我们在很长一段时间内都不必再为数据库压力所困扰。我们的设计很多地方参考了 netlog 和 flickr 的实现，因此非常感谢他们将一些实现细节发布出来。

关于作者

周兆兆（Zola,不是你熟知的那个），又拍网架构师。6 年 IT 从业经验，不太专注于某项技术，对很多技术都感兴趣。

原文链接：http://www.infoq.com/cn/articles/yupoo-partition-database

相关内容：

- 洪强宁谈豆瓣网技术架构
- Raven——一个.NET 上的文档数据库
- 百度大规模数据处理
- 实施云计算之前需要问的 6 个问题
- 豆瓣数据存储实践

推荐文章 | Articles

增强的恶意软件检测

作者 Carlos Rozas 译者 吕骏

摘要：最近几年，出现在恶意软件领域中值得注意的新事态是黑客利用被攻破的平台获得金钱利益，主要有如下 3 种方式：（1）出售盗取的信息，（2）使用平台的资源进行非法或者未经授权的活动，或者（3）绑架平台内的信息以勒索赎金。攻击者在平台中可以捞取的好处越多，平台中未能检测出的恶意软件也就越多，那些能骗过传统安全代理的恶意软件已经再度泛滥。这种类型的恶意软件被称为匿踪恶意软件（stealth malware）。研究人员和工业界发现了云计算的新用途：恶意软件检测。在这篇文章里，我们简要描述了这些新用途，并指出其不足之处。我们提出了一个基于云计算的架构，它能扩展现有解决方案的可用范围，并且我们还提供了一个基于现有 Intel 平台的原型。我们在此仔细分析了这个增强现有架构可靠性的新固件。这种基于平台的新工具有助于安全提供商对抗恶意软件的威胁。

简介

在过去三年间，恶意软件编写与开发者为达到其利用系统漏洞获取经济利益的新目标，对其产品进行了升级换代。在经济利益驱动下，能隐藏自身于平台中的恶意软件得到了很大发展。某些恶意软件甚至会从受侵害的计算机中清除容易露出马脚的同类，以免受其牵累而被发现。

恶意软件在全球范围内造成的损失每年超过百亿美元：仅在 2006 年就有 143 亿美元[1]。

在与恶意软件的斗争中，IT 安全面临多方面的挑战。其中最重要的是恶意软件样本的急剧增长。Panda Security 报告说，在 2008 年每天平均检测到 35,000 个恶意软件样本，全年合计超过 1500 万个样本。McAfee 公司报告说他们收集的恶意软件样本数量从 2008 年 3 月的 1000 万个增加到了 2009 年 3 月的 2000 万个[3]。样本数量的急剧增长意味着没有客户能及时更新已知恶意软件列表。此外，这也意味着安全代理需要花费更多资源来检查文件，以对抗铺天盖地而来的恶意软件特征码。在某些情况下，安全代理会消耗掉 50-60%的 CPU 资源。

恶意软件现在已经泛滥成灾，学术界与工业界一致认为应用云计算来检测恶意软件的时机已

经到来[5，6]。现在已经有很多种可以使用的云计算解决方案模型。图 1 显示一种基于云计算、用于防病毒服务的通用系统架构。第一种模型是服务模型，在主机上运行轻量级的进程以收集相关样本（例如文件），然后将其传送到网络服务上。而网络服务则会进行分析，以鉴定样本是否包含有恶意软件，如果确认此样本已经中毒，就直接指示主机上的轻量级进程将其隔离。在另外一种方式中，主机代理仅仅维护已知恶意软件特征码的一个子集以及一个常用软件应用列表。

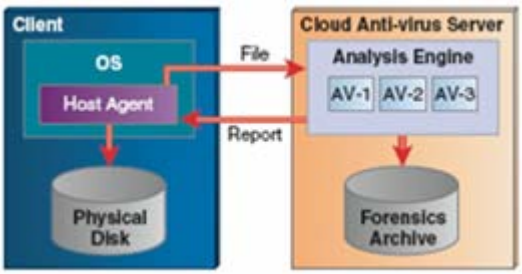


图 1：基于云的防病毒服务（来源：Intel 公司，2009）

恶意软件检测从云计算中受益良多。以云计算为中心会减少客户端占用的存储空间与计算资源，并能简化特征码文件的管理。此外，无论何时，只要以前未能成功鉴别出的恶意软件样本出现在云中，安全提供商就能使用非常复杂并且计算密集的启发式搜索方法将这一恶意软件的危害剖析得清清楚楚。

然而，云计算却不能使主机代理免遭恶意软件的伤害。为防止或检测到遭到禁用、破坏的代理，主机代理需要采用特别的方法与途径。在中提出了很多方法为主机代理提供更安全的保护。其中一些方法集中在使用虚拟技术为安全代理提供隔离的运行环境。在这篇文章里，我们仔细分析了一些平台功能特性，它们能用来隔离主机代理以应对来自各方面不同威胁。

文章组织方式

开篇讨论对主机代理的威胁。然后勾画出一个基于增强型云计算、用于恶意软件检测的通用架构。接下来描述了用来增强计算解决方案的 Intel 平台技术，最后用前面讨论的方法对对威胁进行了分析。

主机代理面临的威胁

正如只能在主机运行的恶意软件检测系统必须得到真实可靠的信息才能有效工作，平台上的主机代理也必须为云服务提供真实可靠的信息才能确保发挥其防病毒作用。如果恶意软件能

利用系统中的漏洞（例如，浏览器插件中的缓冲溢出），然后破坏主机代理，它就能在未被发现的情况下安然运行。

可以使用多种方式破坏主机代理：

- **窜改主机代理**。修改主机代理的可执行属性，使之不能再对恶意软件样本构成威胁。此类窜改可以简单到不再发送文件到云服务，也可以复杂到允许恶意软件的代理过滤发送到云服务的文件。
- **禁用主机代理**。恶意软件修改系统配置，或者使主机代理下次系统启动时不再运行，或者立即中止主机代理的运行。
- **输入过滤**。恶意软件在系统 API 中设置钩子（hook），并插入恶意代码，从而过滤提供给主机代理的信息。众所周知的钩入点（hook points）包括引入表（import table）和系统调用参考表（system call table）。然而，还存在有更多钩入点（hook points）；Wang 及其他人员在 Red Hat Fedora core 中找到了 41 个可能用来进行文件隐藏的内核钩入点（hook points）。

在最近几年，恶意软件发展的重点在于使用更具破坏性的方式攻破系统安全。其中一种由暗影行者（Shadow Walker）使用的方法是在中断描述符表（interrupt descriptor table ,IDT）、页面出错处理程序（ page-fault handler ）中设置钩子。当处理器读取内存中的内容作为数据时会返回恶意软件所指定的值，而当处理器读取内存中内容作为代码时会返回其他值。另外一种安全研究人员提出的方式是安装恶意虚拟机监视程序（VMM）来超劫持（hyperjack）操作系统（OS）。VMM 使得研究者能对系统一览无余，却不需要对 OS 进行任何修改或者挂钩（hooking）。

增强基于云的恶意软件检测

在图 2 中描述了一个系统架构，用以增强基于云的防病毒服务。

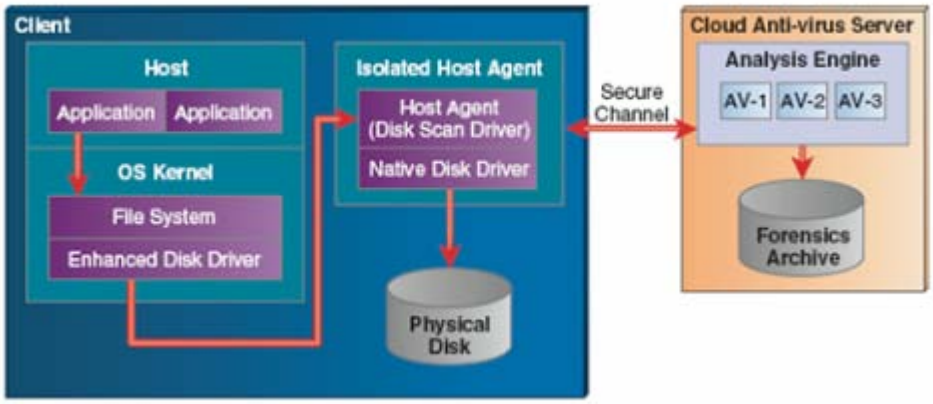


图 2：基于云的防病毒增强型解决方案（来源：Intel 公司，2009）

通过在主机环境中隔离出主机代理，并提供对诸如外存和内存等平台资源的直接访问，主机上的恶意软件不能再直接攻击或者操纵主机代理。恶意软件不得不转而攻击主机代理分区。由于主机代理分区不需要支持多用途计算，它能以更加健壮的解决方案配置得更加安全。下面描述了相关架构组件：

- **隔离的主机代理环境**。主机代理被包含在隔离的运行环境中。主机能利用此环境提供的接口发送请求。它提供直接访问主机外存、主机访问；磁盘 I/O 请求能直接交付给此环境。
- **隔离的主机代理**。主机代理包含安全、经过授权的通道，用来连接到防病毒云。主机代理监控主机磁盘 I/O，如有需要，经由安全通道发送文件到防病毒云网络服务进行分析。主机代理包含文件系统操作功能，负责访问主机文件系统，并且代理能定期扫描物理磁盘以发现哪些文件被更改；然后发送被更改的文件到防病毒云网络服务。
- **增强型磁盘驱动**。增强型磁盘驱动用来通过文件系统将磁盘 I/O 请求从主分区转发到运行在安全容器内的主机代理，以进行下一步处理。
- **原生磁盘驱动**。原生磁盘驱动用来从被隔离的分区中访问主机磁盘硬件。

图 3 描述了怎样扩展防病毒云服务，使其有能力提供内核匿踪攻击（Rookit）检测，还有在磁盘/文件上扫描恶意软件。架构组件描述如下：

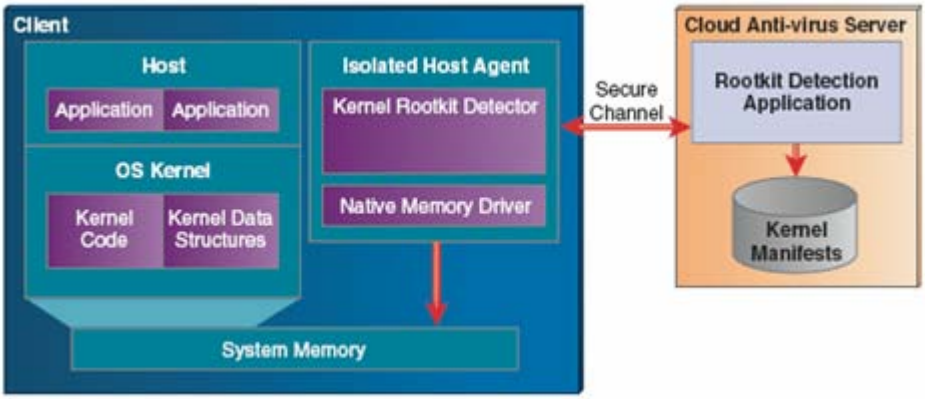


图 3：基于云的内核匿踪攻击（Rookit）检测（来源：Intel 公司，2009）

- **内核匿踪攻击（rootkit）检测程序。**本地内核匿踪攻击（rootkit）检测程序，运行在客户端隔离分区，向运行在防病毒云中的内核匿踪攻击（rootkit）检测应用暴露出安全的远程接口。用这种方法，匿踪攻击（rootkit）应用程序能访问核心内存页，并在内核内存区域内执行基本哈希（hash）比较操作，以进行完整性检查。完整性确认操作在远程服务器端执行。内核哈希值也会保存在防病毒云服务器上，在需要时提供给客户端 PC 的内核匿踪攻击（rootkit）检测程序。
- **原生内存驱动。**原生内存驱动运行在隔离的分区上，用以确保能够安全访问系统内存区，主机 OS 的内核内存位于系统内存区中。

远程内存完整检查操作会产生两个问题：安全与网络延时。我们使用下面三种方式来解决网络安全问题：提供客户端 PC 与防病毒云服务间的安全通道、提供内存哈希比较接口并限制远程内存访问。由于绝大多数进行内存完整检查的核心内存区驻留在客户端平台的不可分页（nonpageable）内存，内存验证时的网络延时问题就显得不是太严重。

通过检测诸如引入表挂钩、内存代码与静态数据修改、IDT、系统调用参考表以及直接内核对象窜改等常用攻击方法，内核匿踪攻击（rootkit）检测程序有助于缓解未知威胁或者内存内（in-memory）威胁。

对抗匿踪恶意软件的原型架构

我们开发出两个系统原型，用来检验系统设计。原型 1 基于 Intel®管理引擎（Intel®ME），原型 2 基于虚拟机监视程序（virtual machine monitor,VMM），两者都与主机 OS 隔离。因为这两种方式都与主机 OS 互相隔离，攻击者更难破坏其运行环境。

原型 1：基于 Intel®管理引擎

自从 Intel®主动管理技术（Intel® AMT） [16, 17]及运行 Intel® vPro™技术的平台面世以来，平台中就包含了一个嵌入式微控制器，称为 Intel ME。Intel ME 是一个集成设备，独立安装在 PCI 总线中。Intel ME 集成了多种硬件引擎，比如总线控制器、加密加速器、DMA 引擎等等。Intel ME 运行的固件（firmware）由实时操作系统（real-time operating system ,RTOS）、操纵硬件的驱动程序以及管理应用程序组成。在我们的原型中，我们利用 DMA 引擎访问内存区。

我们首先实现一个代理来扫描固件中的内存。这个代理是传统的基于黑名单的扫描代理。因为 Intel ME 在计算能力与存储空间两方面的共同限制，我们只能在 Intel ME 固件中实现一个功能受限的扫描代理。能安全存放的黑名单的长度限制在 192KB，扫描操作的频率也不能太高。由于这些限制，我们只好在主机 OS 中实现了一个主机代理，其功能是扫描基于黑名单的内存。在此原型中，我们向 Intel ME 代理中加入完整性检查固件，以检验其主机代理的完整性。此外，Intel ME 带外（ out-of-band ,OOB）接口能与远程防病毒云服务进行通讯，如果主机代理在运行时被修改，就会通知防病毒云上的软件。Intel ME 在其存储区域内保存主机代理的哈希值，并且定期验证主机代理运行时映像的完整性。在我们的论文《实时内核匿踪攻击（Rootkit）检测》（Runtime Kernel Rootkit Detection）中，我们描述了用来验证主机代理运行时完整性的清单（manifest）产生进程以及 3 阶段（3-phase）算法。对于此原型，我们使用 Intel ME 度量主机代理的完整性：进程执行所需时间在毫秒级。我们打算进一步研究事件驱动、主机代理扫描等方法，以解决定时攻击问题。图 4 显示了我们的原型架构：

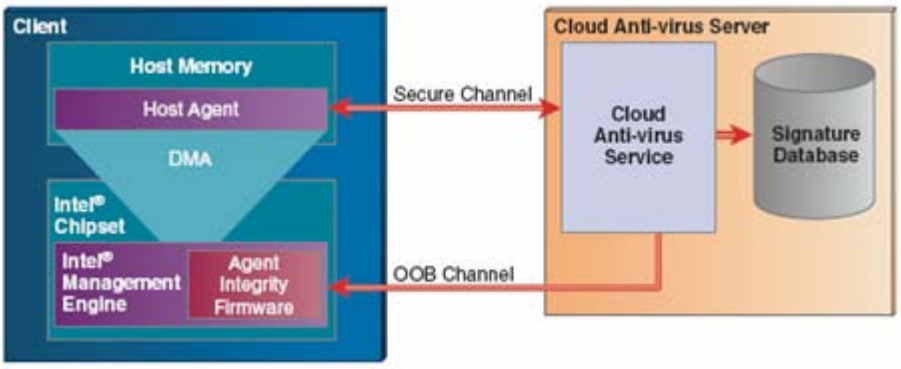


图 4：Intel®管理引擎架构（来源：Intel 公司,2009）

原型 2：基于虚拟机监视程序（Virtual Machine Monitor）

在这里，我们的第二种有原型中，我们用虚拟机（virtual machine,VM）作为隔离环境，并且扩展我们的第一个原型来度量虚拟机监视程序（VMM）的完整性。

为理解我们为何选择 VMM 作为隔离环境，让我们首先来了解一下使用硬件虚拟的基于虚拟的系统。我们在平台上使用 Intel® Virtualization Technology （Intel® VT）。虚拟是指一种将处理器或者芯片组的物理资源到分割到不同 VM 上，并在 OS 下插入比 VM 更高权限的执行程序的技术。这个执行程序就是 VMM。在 IA-32 的 Intel® Virtualization Technology （Intel® VT）、Intel® 64 和 Intel® Architecture （Intel® VT-x）中，称这一权限级别为 VMX-root 模式。将控制转换到 VMM 称为 VMExit，将控制转换到 VM 称为 VMEntry。通过使用 VMCALL 指令，VM 能显式地、强制地进行 VMExit。在 VMX non-root 模式下运行的客户 OS 进行关键 OS 操作时必定会产生一个 VMExit。这使得 VMM 能强制执行隔离策略。通过增加一个轻量级 VMM 到[9]中所描述的原型，我们增强了其功能。为能拦截换页事件以及对数据结构的改动，我们利用 VMM 监控所需的系统事件，并可以在需要时建立影子页（shadow page）表。系统组件、清单生成以及完整性验证算法的细节在中详细讨论。我们在研究原型中加入了 Intel ME 固件，用其来验证 VMM 自身的完整性。在图 5 显示了此架构。

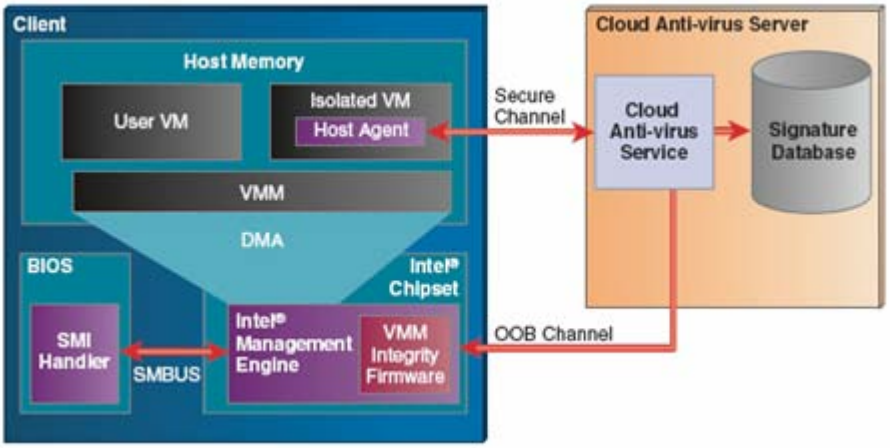


图 5：虚拟化环境架构（来源：Intel 公司，2009）

第二种原型的的关键架构组件基于深视技术（Deep Watch），它是由位于 Intel ME 固件中的完整性验证模块和防病毒云服务中的 VMM 完整性应用服务组成。在 BIOS 系统管理中断（system management interrupt,SMI）处理程序提供的简单支持下，能得到处理器状态和寄存器信息，然后 Intel ME 能从这些信息中为 VMM 重建虚拟内存页表。系统管理模式(System management mode,SMM）是一种能达成特定目标的操作模式，此模式能管理全系统范围内的功能，例如电源管理、系统硬件控制，甚至私有 OEM 设计代码。SMM 带来的主要好处是提供了明确的并且使用简便的独立处理器环境，此环境对 OS 或者执行程序及软件应用都是透明的。当通过 SMI 调用 SMM（SMI 处理器），处理器保存当前处理器的状态（处理器上下文 context），然后转换到隔离的、存放在系统管理 RAM（system management RAM ,SMRAM）中的操作环境。这个处理器状态被收集到 SMI 处理程序，并通过硬件接口传送到 Intel ME。

Intel ME 中存放的处理器状态可以用来重新构建内存页表，以验证 VMM 的实现完整性。此外，Intel ME 可以传送所有信息（处理器状态和内存页）到防病毒云服务。远程防病毒服务就能验证 VMM 的实时完整性，也就克服了 Intel ME 的计算能力限制。我们还建立了一个类似研究原型通过 PCI DMA 设备来度量主机 OS 驱动器的完整性。

威胁分析

我们假设攻击者已经完全能访问及控制 OS，包括内核，并能插入、修改或者删除内核驱动程序；然而，我们假设攻击者不能修改 Intel ME 固件或者 SMRAM。我们的假设意味着攻击范围十分广泛，从简单用户攻击到各种修改关键内核数据结构的攻击，其目的是破坏用户 OS 或者 VMM 本身。这种类型的攻击例子包括有在导入表、IDT 或者系统调用参考表中挂钩、内核代码及静态数据修改，以及直接内核对象窜改。

下面是我们在原型 1 中解决的不同攻击方式的威胁：

- **主机代理的威胁**。Intel ME 中的内核匿踪攻击（rootkit）检测程序有效防止了内核代码修改、导入表、IDT 以及系统调用参考表挂钩。Intel ME 使用 OOB 接口通过其 DMA 接口只读访问内存，确保匿踪攻击（rootkit）检测程序能得到一个全面而且未经篡改的内存视图。
- **未知内核攻击**。如果 Intel ME 中的内核匿踪攻击（rootkit）检测程序检测到任何可疑行为或者模式，它就会通知防病毒云服务器进行更加细致地扫描。

在原型 2 中，除了原型 1 所能解决的威胁以外，我们还解决了 VMM 攻击。

- **对主机代理的威胁**。VMM 保护主机代理不受恶意软件的攻击。
- **未知内核攻击**。主机代理，经我们的内核匿踪攻击（rootkit）检测程序加强后，可以检测任何可疑行为或者模式。
- **VMM 攻击**。被破坏的 VMM 可以通过 Intel ME 中的直接内存访问以及实时内存匿踪攻击（rootkit）检测（Runtime Kernel Rootkit Detection ,RKRD）甄别出来。

总结

在这篇文章中，我们解释了学术界与工业界共同提议采用云计算对抗病毒的缘起。我们仔细研究了基于云的防病毒服务所面临的威胁，首要风险直指运行在客户端上的主机代理，它用

于向防病毒云引擎输送数据。我们然后提供一些基于平台的功能特性，基于 Intel 架构，能用来减少恶意软件对主机代理的危害。我们研究了结合了 Intel 虚拟技术及 Intel 芯片组技术的原型，这些技术包括有 Intel ME，能有效减少防病毒云环境中对主机代理的大部分威胁。因此，这些技术的新应用能为我们的终端用户提供更加有效的防病毒云服务。

原文链接：http://www.infoq.com/cn/articles/malware-detection-intel

相关内容：

- 攻击 Google 的 Jarlsberg，学习安全漏洞知识
- 网络安全：采访 Intel 安全与加密经理
- 加密因特网
- 关注网银系统：安全模型和架构设计
- 评论：微软云计算平台的安全性

推荐文章 | Articles

解读 NoSQL 技术代表之作 Dynamo

作者 陈臻

NoSQL 在过去的一年里，逐渐已经成为了家喻户晓的东西，我（54chen）自从去年开始人人网的 NoSQL 系统 Nuclear 的研发以来，一直看 NoSQL 越来越热，越来越引来大家的围观。受 InfoQ 中文站编辑之托，特作此文，一来作为过去一年的总结，二来希望对 NoSQL 系统在国内的发展和推广尽绵薄之力。

NoSQL 背后的两种模式

NoSQL 其实并不是什么妖魔鬼怪，相反，NoSQL 的真谛其实应该是 Not Only SQL，其产生背景是在数据量和访问量逐渐增大的情况下下，人为地去添加机器或者切分数据到不同的机器，变得越来越困难，人力成本越来越高，于是便开始有了这样的项目，它们的本意是提高数据存储的自动化程度，减少人为干预的时间，让负载更加均匀等。在国际上，真正的代表之作有来自 Google 的 BigTable 和 Amazon 的 Dynamo，他们分别使用了不同的基本原理。

MapReduce

这是历史最久的一种模型，典型的代表是 BigTable。Map 表示映射，Reduce 表示化简。MapReduce 通过把对数据集的大规模操作分发给网络上的每个节点实现可靠性（Map）；每个节点会周期性地把完成的工作和状态的更新报告回来（Reduce）。大多数分布式运算可以抽象为 MapReduce 操作。Map 是把输入 Input 分解成中间的 Key/Value 对，Reduce 把 Key/Value 合成最终输出 Output。这两个函数由程序员提供给系统，下层设施把 Map 和 Reduce 操作分布在集群上运行。

Dynamo

这里我把 Dynamo 专门归纳成为了一种，其原因是它与 MapReduce 有很大的不同，自成一派。先说一下历史，Amazon 于 2006 年推出了自己的云存储服务 S3，2007 年其 CTO 公布了 S3 的设计方案，从此江湖中就不再太平了，开源项目一个个如雨后春笋般地出现了。比较

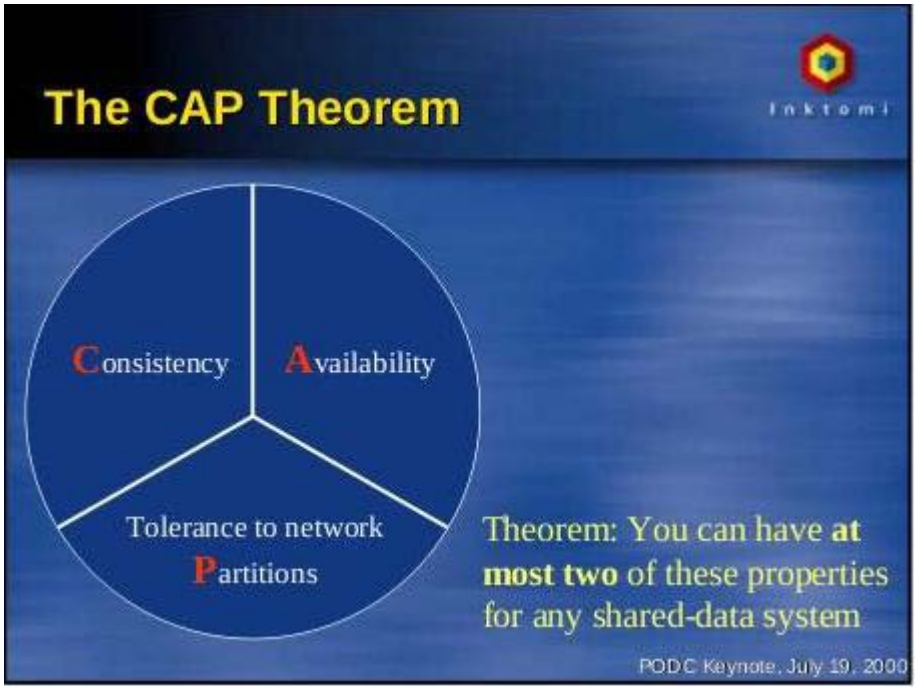
常见的有 Facebook 开发的 Cassandra（如果没有记错，在去年浏览他们项目网页的时候，上面还写着他们之中的一个开发人员是 Dynamo 的设计人员，现在风头紧，去掉了），还有 Linkedin 的 voldemort，而在国内话，有豆瓣网的 beansDB，人人网的 nuclear 等等。这里我主要讨论的也是 Dynamo 的方案细节。

入门基础

Dynamo 的意思是发电机，顾名思义，这一整套的方案都像发电机一样，源源不断地提供服务，永不间断。以下内容看上去有点教条，但基本上如果你要理解原理，这每一项都是必须知道的。

CAP 原则

先来看历史，Eric A. Brewer 教授，Inktomi 公司的创始人，也是 berkeley 大学的计算机教授，Inktomi 是雅虎搜索现在的台端技术核心支持。最主要的是，他们（Inktomi 公司）在最早的时间里，开始研究分布计算。CAP 原则的提出，可以追溯到 2000 年的时候（可以想象有多么早！），Brewer 教授在一次谈话中，基于他运作 Inktomi 以及在伯克利大学里的经验，总结出了 CAP 原则（文末参考资料中有其演讲资料链接）。图一是来自 Brewer 教授当年所画的图：



图一：CAP 原则当年的 PPT

- Consistency（一致性）：即数据一致性，简单的说，就是数据复制到了 N 台机器，如果有

更新，要 N 机器的数据是一起更新的。

- Availability（可用性）：好的响应性能，此项意思主要就是速度。
- Partition tolerance（分区容错性）：这里是说好的分区方法，体现具体一点，简单地可理解为是节点的可扩展性。

定理：任何分布式系统只可同时满足二点，没法三者兼顾。

忠告：架构师不要将精力浪费在如何设计能满足三者的完美分布式系统，而是应该进行取舍。

DHT——分布式哈希表

DHT（Distributed Hash Table，分布式哈希表），它是一种分布式存储寻址方法的统称。就像普通的哈希表，里面保存了 key 与 value 的对应关系，一般都能根据一个 key 去对应到相应的节点，从而得到相对应的 value。

这里随带一提，在 DHT 算法中，一致性哈希作为第一个实用的算法，在大多数系统中都使用了它。一致性哈希基本解决了在 P2P 环境中最为关键的问题——如何在动态的网络拓扑中分布存储和路由。每个节点仅需维护少量相邻节点的信息，并且在节点加入/退出系统时，仅有相关的少量节点参与到拓扑的维护中。至于一致性哈希的细节就不在这里详细说了，要指明的一点是，在 Dynamo 的数据分区方式之后，其实内部已然是一个对一致性哈希的改造了。

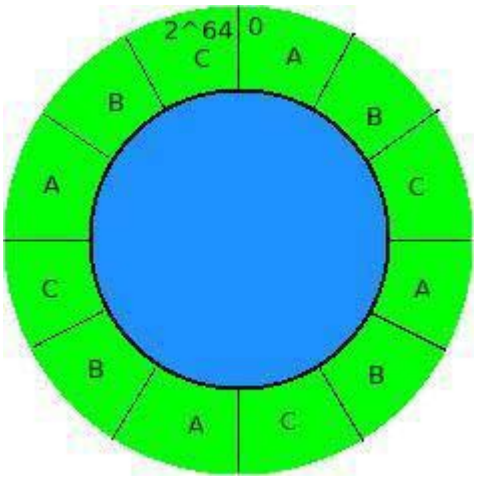
进入 Dynamo 的世界

有了上面一章里的两个基础介绍之后，我们开始进入 Dynamo 的世界。

Dynamo 的数据分区与作用

在 Dynamo 的实现中提到一个关键的东西，就是数据分区。假设我们的数据的 key 的范围是 0 到 2 的 64 次方（不用怀疑你的数据量会超过它，正常甚至变态情况下你都是超不过的，甚至像伏地魔等其他类 Dynamo 系统是使用的 2 的 32 次方），然后设置一个常数，比如说 1000，将我们的 key 的范围分成 1000 份。然后再将这 1000 份 key 的范围均匀分配到所有的节点（s 个节点），这样每个节点负责的分区数就是 1000/s 份分区。

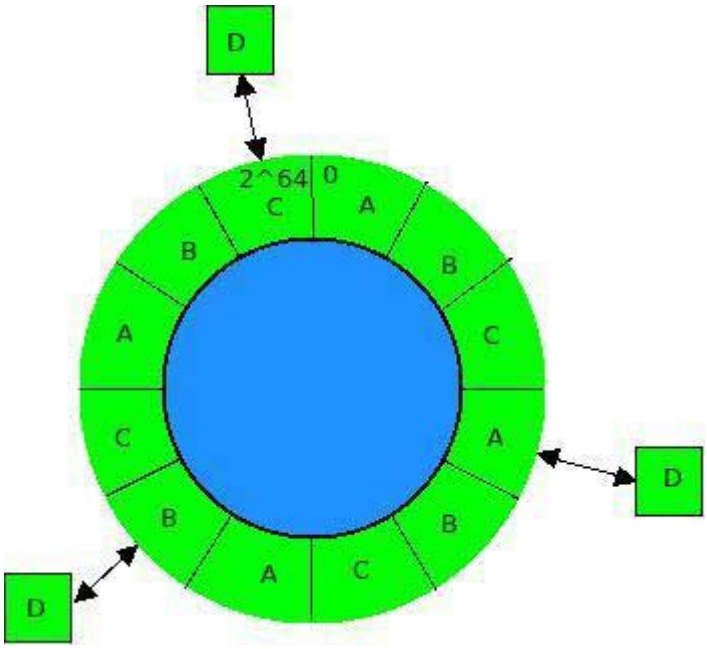
如图二，假设我们有 A、B、C 三台机器，然后将我们的分区定义了 12 个。



图二：三个节点分 12 个区的数据的情况

因为数据是均匀离散到这个环上的（有人开始会认为数据的 key 是从 1、2、3、4……这样子一直下去的，其实不是的，哈希计算出来的值，都是一个离散的结果），所以我们每个分区的数据量是大致相等的。从图上我们可以得出，每台机器都分到了三个分区里的数据，并且因为分区是均匀的，在分区数量是相当大的时候，数据的分布会更加的均匀，与此同时，负载也被均匀地分开了（当然了，如果硬要说你的负载还是只集中在一个分区里，那就不是在这里要讨论的问题了，有可能是你的哈希函数是不是有什么样的问题了）。

为什么要进行这样的分布呢，分布的好处在于，在有新机器加入的时候，只需要替换原有分区即可，如图三所示：



图三：加入一个新的节点 D 的情况

同样是图二里的情况，12 个分区分到 ABC 三个节点，图三中就是再进入了一个新的节点 D，从图上的重新分布情况可以得出，所有节点里只需要转移四分之一的数据到新来的节点即可，同时，新节点的负载也伴随分区的转移而转移了（这里的 12 个分区太少了，如果是 1200 个分区甚至是 12000 个分区的话，这个结论就是正确的了，12 个分区只为演示用）。

从 Dynamo 的 NRW 看 CAP 法则

在 Dynamo 系统中，第一次提出来了 NRW 的方法。

- N：复制的次数；
- R：读数据的最小节点数；
- W：写成功的最小分区数。

这三个数的具体作用是用来灵活地调整 Dynamo 系统的可用性与一致性。

举个例子来说，如果 R=1 的话，表示最少只需要去一个节点读数据即可，读到即返回，这时是可用性是很高的，但并不能保证数据的一致性，如果说 W 同时为 1 的话，那可用性更新是最高的一种情况，但这时完全不能保障数据的一致性，因为在可供复制的 N 个节点里，只需要写成功一次就返回了，也就意味着，有可能在读的这一次并没有真正读到需要的数据（一致性相当的不好）。如果 W=R=N=3 的话，也就是说，每次写的时候，都保证所有要复制的点都写成功，读的时候也是都读到，这样子读出来的数据一定是正确的，但是其性能大打折扣，也就是说，数据的一致性非常的高，但系统的可用性却非常低了。如果 R+W>N 能够保证我们“读我们所写”，Dynamo 推荐使用 322 的组合。

Dynamo 系统的数据分区让整个网络的可扩展性其实是一个固定值（你分了多少区，实际上网络里扩展节点的上限就是这个数），通过 NRW 来达到另外两个方向上的调整。

Dynamo 的一些增加可用性的补救

针对一些经常可能出现的问题，Dynamo 还提供了一些解决的方法。

- 第一个是 hinted handoff 数据的加入：在一个节点出现临时性故障时，数据会自动进入列表中的下一个节点进行写操作，并标记为 handoff 数据，在收到通知需要原节点恢复时重新把数据推回去。这能使系统的写入成功大大提升。
- 第二个是向量时钟来做版本控制：用一个向量（比如说[a,1]表示这个数据在 a 节点第一

次写入）来标记数据的版本，这样在有版本冲突的时候，可以追溯到出现问题的地方。这可以使数据的最终一致成为可能。(Cassandra 未用 vector clock ,而只用 client timestamps 也达到了同样效果。)

- 第三个是 Merkle tree 来提速数据变动时的查找：使用 Merkle tree 为数据建立索引，只要任意数据有变动，都将快速反馈出来。
- 第四个是 Gossip 协议：一种通讯协议，目标是让节点与节点之间通信，省略中心节点的存在，使网络达到去中心化。提高系统的可用性。

后记

Dynamo 的理论对 CAP 原则里的可扩展性做到了很方便的实现，通过创造性的 NRW 来平衡系统的可用性和一致性，增加了系统在实际情况下遇到问题的可选择方案。可以相像，在 NoSQL 的道路上，这只是个开端，在分布式计算的道路上，已经是 MapReduce 之后的再次革命。

关于作者

54chen（陈臻），人人网分布式存储研究人员，业余时间混迹于各技术组织且乐此不疲。目前关注实施 PHP 培训。对 flex 等前端技术有一点研究。个人技术站点：http://www.54chen.com/。可以通过电子邮件 czhttp@gmail.com 联系到他。

参考资料

- LinkedIn 公司的伏地魔项目主页
- FaceBook 公司的 cassandra 项目主页
- Dyanmo 最原始的一个 pdf 文档
- Brewer 教授的主页
- 伏地魔项目的中文文档
- 提出 CAP 原则的文档
- DHT 的介绍

- 一致性哈希的介绍

原文链接：http://www.infoq.com/cn/articles/nosql-dynamo

相关内容：

- MemBase，NoSQL 家族的一个新的重量级的成员
- Neo4j：基于 Java 的 NoSQL 图形数据库
- 将 CouchDB 作为个人数据库
- LLBLGen Pro 3.0 现已支持多种持久层框架
- 使用微软 Live 实验室项目 Pivot 诠释海量数据

时刻关注企业软件开发领域的变化与创新

Java — .NET — Ruby — SOA — Agile — Architecture

Java社区：企业Java社区的变化与创新

.NET社区：.NET和微软的其它企业软件开发解决方案

Ruby社区：面向Web和企业开发的Ruby，主要关注Ruby on Rails

SOA社区：关于大中型企业内面向服务架构的一切

Agile社区：敏捷软件开发和项目经理

Architecture社区：设计、技术趋势及架构师所感兴趣的话题

讨 论 组：groups.google.com/group/infoqchina
编辑电邮：editors@cn.infoq.com
广告合作：sales@cn.infoq.com

新品推荐 | New Products

Internet Explorer 9 Preview 3 提供了更好的 HTML 5 支持

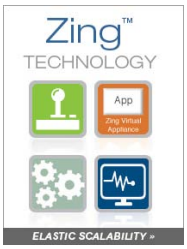
作者 Abel Avram **译者** 张龙

Internet Explorer Preview 3 对 HTML 5 的增强又更进了一步，主要有音频、视频和 canvas、更快的 JavaScript 引擎、支持更多的 DOM 与 CSS 特性、支持嵌入式字体、逐步缩小了与其他浏览器在 HTML 5 实现上的差距，通过硬件加速提供更棒的性能。

原文链接：http://www.infoq.com/cn/news/2010/06/IE9-Preview-3

Azul 发布 Zing：面向 x86、基于软件的 JVM 虚拟化/弹性运行时

作者 Charles Humble **译者** 张龙



近日，Java 应用专业级硬件厂商 Azul Systems 发布了 Zing：其硬件栈的虚拟化实现。Zing 使用 RedHat 的 KVM 和 VMWare 的 vSphere，面向流行的各种操作系统，特别 是针对 Intel 最新的 x86 进行了优化。

原文链接：http://www.infoq.com/cn/news/2010/06/zing

Eclipse Helios 发布了

作者 Alex Blewitt 译者 张龙

近日，Eclipse 基金会发布了 Eclipse Helios，随之而来的是 39 个不同的项目。Eclipse 项目采用敏捷方法管理，今年已经发布了 7 个里程碑版本，在本月前又发布了很多 RC 版。

原文链接：http://www.infoq.com/cn/news/2010/06/eclipse-helios

Java Lambda 语法的首个版本引发众多争论

作者 Dionysios G. Synodinos **译者** 张龙

近日，Oracle 的 Maurizio Cimadamore 在 OpenJDK Mercurial 仓库中发布了初始的 lambda 实现，这让我们有机会首次目睹该新语法的容貌，同时也引发了社区的广泛争论。

原文链接：http://www.infoq.com/cn/news/2010/06/lambda-syntax-debate

HyperSQL 2.0：期待 5 年，新版本终于发布

作者 Josh Long **译者** 张龙

在酝酿了 5 年之后，HyperSQL 数据库（HSQLDB）2.0 终于发布了。HyperSQL 2.0 是 HSQLDB 1.8 的后继，而 HSQLDB 1.8 已在大大小小的应用中都展示过身手，包括使用广泛的 OpenOffice Base 应用。相对于其他开源数据库来说，新版本的 HyperSQL 引入了更多的特性。

原文链接：http://www.infoq.com/cn/news/2010/06/hsqldb2

代码质量管理工具 Sonar 提供了设计与架构度量

作者 Srini Penchikala **译者** 张龙

近日，新版的开源代码质量管理工具 Sonar 提供了设计与架构度量。Sonar 2.0 引入了针对 Java 应用的设计分析、架构与面向对象的度量，而前不久发布的 Sonar 2.1 可以检测到未使用的方法以及对不建议使用方法的调用。

原文链接：http://www.infoq.com/cn/news/2010/06/sonar-2.1

Google 向开发者提供云存储

作者 Abel Avram **译者** 张龙

Google Storage for Developers（GSD）是个全新的 RESTful 服务，提供的数据存储可以在位于美国的几个数据中心间复制。GSD 叫做“for Developers”的原因在于传输与访问数据的 API 基于常规的 HTTP 命令，比如 GET、POST、PUT、HEAD 和 DELETE。

原文链接：http://www.infoq.com/cn/news/2010/06/Google-Storage-for-Developers

推荐编辑 | Agile 社区编辑 郑柯

“I was so much older then; I'm younger than that now"。

这是 Bob Dylan 在《My Back Pages》中反复吟唱的一句歌词，偶然的机会，我从豆瓣上看到了他在自己从艺 30 周年演唱会上这首歌的视频，其中尤其打动我的就是这句词。

“I was so much older then"。刚刚接触 InfoQ 是在 3 年以前，那时我在一家行业公司，所掌握的技术知识和项目管理方法说“so much older”一点也不过分。虽然在那家公司做了 4 年之久，但是真正的成长和积累也就是第一年，后来的 3 年基本上都是低层次的重复。

搜索敏捷相关资料的时候，发现了 InfoQ 的英文站，也就顺藤摸瓜找到了中文站，并在敏捷中国的邮件组里面响应主编霍泰稳的号召，自投罗网，做了中文站的编辑。

打那之后，我终于找到了上佳的理由告别无聊的电视。翻译新闻，不仅学习到最新的东西，而且还能带来一些额外的收入，虽不足以养家糊口，但一篇新闻翻译下来，一个人吃顿烤翅，喝瓶啤酒还是没有问题的。不久，我也告别了前一家公司，去一家技术媒体就职，而这也要感谢霍泰稳同学的介绍。在这两家技术媒体的工作，为我大开新技术之门，更重要的是：从此有机会跟业界的诸多专家面对面亲密接触，在短时间内聆听他们分享多年的积累和经验。

三年以来，InfoQ 成长了，我也为自己未来的发展确定了云计算的方向。十年来工作经历告诉我，在各种各样的职场人士中，技术人是最单纯、最善良的一群人。和他们在一起，不断学习新的技术，这会让你变得更年轻，要不怎么说“younger than that now"呢。证明？你来试试看不就知道了？

P.S.：强烈推荐大家有时间去看看 Bob Dylan 那段现场表演，与他一起的还有“吉它之神”Eric Clapton、Beatles 乐队吉他手 George Harrison、盲人歌手 Stevie Wonder、乡村老牛仔 Neil Young 等等一干老枪，绝对不容错过。

封面植物

四药门花

四药门花，稀有种。常绿灌木要或小乔木，高 12 米。分布于香港及广西海拔 590 米。有一定耐阴力。花期 3 至 4 月，和金缕梅亚科相似，仅具 1 粒种子，是比较原始的属，又和常绿的活塞木属及落叶的继木属与金缕梅属有十分密切的亲缘关系，因此在系统学研究上具有重要科学意义。

分布与习性：四药门花产于北纬 22℃上下两侧的狭窄地区，分布区年平均温为 20-22℃，1 月份平均温为 13-14℃，年降雨量 1350-2200 毫米，龙舟的更岗山一带属峰丛石山，旱季多雾。本种生于富含腐殖质的肥沃赤红壤土。

进化史：四药门花于 1857 年至 1858 年间在香港岛发现，于 1861 年收录于乔治·班逊姆（en:GeorgeBentham）出版的《香港植物志》（FloraHongkongensis）一书，一直被认为是香港独有的品种，100 年内也未于其他地方被发现。直至 1957 年 6 月才在广西龙州的常绿阔叶林里再次找到，植株数目稀少。现已被加载《中国保护植物红皮书》[1]及《广东省珍稀濒危植物图谱》。

开花动态：四药门花的花期为 9 月至次年 2 月，其中 9–10 月为开花高峰期，之后为零星开放；有时刚刚凋谢的花序所在的叶腋会长出新的花序继续开放。在一个头状花序中，下方的花会先开放，但整个花序的开花时间都相隔不远，并且在一天中没有特定的开花时间。花瓣完全伸展开需要 6–24h，此时柱头已有活性。开花后 12–24h 间花药开裂，同一朵花内花药开裂基本同步，在一天中也没有特定的开裂时间。

1kg.org 多背一公斤

爱自然 | 更爱孩子

架构师 7 月刊

每月 8 日出版

本期主编：崔康

总编：霍泰稳

总编助理：刘申

编辑：李明 胡键 宋玮 郑柯 朱永光 池建强

读者反馈：editors@cn.infoq.com

投稿：editors@cn.infoq.com

交流群组：
http://groups.google.com/group/infoqchina

商务合作：sales@cn.infoq.com 13911020445

本期主编：崔康，InfoQ 中文站 Java 社区编辑

崔康，InfoQ 中文站 Java 社区编辑，毕业于天津大学，计算机应用技术硕士，现就职于某外企研发部门。关注 Web 应用的开发与测试，感兴趣的领域包括 Web 性能诊断与调优等。希望能在工作之余，为中国软件技术的发展做一些力所能及的事情。联系方式为：cuikang[at]gmail.com。

《架构师》月刊由 InfoQ 中文站出品。

所有内容版权均属 C4Media Inc.所有，未经许可不得转载。

时刻关注企业软件开发领域的变化与创新

架构师

www.infoq.com/cn/architect

每月8号出版

InfoQ neue
Enterprise Software Development Community

商务合作：sales@cn.infoq.com
读者反馈/投稿：editors@cn.infoq.com