促进软件开发领域知识与创新的传播

# 架构师

8月 ARCHITECT

## 特别专题

再谈REST

如何设计好的RESTful API？

REST之中的幂等指的是什么？

Web风格起过作用吗？

## 阿里技术嘉年华访谈

KISSY的自成体系是一种趋势

阿里巴巴M-DMP负责人许成访谈

小微金服前端技术专家伯约访谈

所有的团队都应该使用自动构建吗？

从OpenStack的角度看块存储的世界

对话微软开源技术子公司资深总监

InfoQ

每月8号出版

# 卷首语

## 人、技术、情怀

这个世界人很多，做技术的人也不少，但是有情怀的技术人不多。

人类发明了各种工具来方便自己、改善生活亦或是达到某种目的，虽然并不只是人类会制造工具，但是不同的是我们懂得 如何去思考、沉淀、积累。现在我们有了计算机、有了编程语言、有了网络，也许我们已经不再需要具体的工具形态就能改变世界， 一切就在弹指一挥间。

但是我们所造就的工具却正奴役着我们，多少程序员苦逼的吐槽自己是码农？多少人在2米见方的格子里敲着CURD？ 多少人每天熬到深夜只是因为Deadline逼近？这就像历史中描述的工业革命的那段日子，人们发明了蒸汽机、纺织机，但是却成了这些机器 的守护者。我们存在的意义也许只是按照机器的指示做出各种动作，什么梦想、什么情怀，已经烟飘云散。

我们不反对技术的进步，然而这种进步不应该以牺牲自己的生活为代价，二者是可以兼容的。如果你了解运维就应该知道这是一个很苦逼的差事，“没出事不被人想起，一旦有事就要随时待命”。在今年的QCon北京上，来自Github的运维工程师分享了他不同的“生活”，通过工具、脚本、辅助机器人，他可以逍遥在自己的小船上晒着太阳维护着机房。的确我们 需要对自己所做的工作负责，但是我们更要对自己负责。如果你还能想起十几二十年的梦想，如果你能在码字的间隙去环顾现在的生活，请问一个问题，这是否是我想要的！

技术人，不要把自己陷在技术的圈子中，世界太大，人生太短。我们习惯了敲错代码就按下ctrl+z，出问题了就debug，但是人生是一段不能回头的单行道， 不要迷惘选择是否正确，当你心之所向，那就前往。最后送大家一首叶芝的诗，献给也许已经逝去的“情怀”！ ：

> 当你老了，白发苍苍，睡意朦胧，
> 在炉前打盹，请取下这本诗篇，
> 慢慢吟咏，梦见你当年的双眼
> 那柔美的光芒与青幽的晕影；
>
> 多少人真情假意，爱过你的美丽，
> 爱过你欢乐而迷人的青春，

唯独一人爱过你朝圣者的心，
爱你日益凋谢的脸上的哀戚；

当你佝偻着，在灼热的炉栅边，
你将轻轻诉说，带着一丝伤感，
逝去的爱，如今以步上高山，
在密密星群里埋藏着它的赧颜。

**本期主编：水羽哲**

# InfoQ

促进软件开发领域知识与创新的传播

中文 | 英文 | 日文 | 葡文 | ……

内容投稿 editors@cn.infoq.com | 商务合作 sales@cn.infoq.com | 加入我们 hr@cn.infoq.com

# 目录


## 人物 | People

OSChina的技术选型：就是为了短平快

## 观点 | View

事物不断变化（流程也需调整）
函数式编程为何越来越受到重视

## 本期专题：再谈REST | Topic

虚拟研讨会：如何设计好的RESTful API？
REST之中的幂等指的是什么？
Web风格起过作用吗？

## 推荐文章 | Article

所有的团队都应该使用自动构建吗？
从OpenStack的角度看块存储的世界
对话微软开源技术子公司资深总监Gianugo：关键在于互操作性

## 特别专栏 | Column

拔赤谈前端：KISSY的自成体系是一种趋势
阿里巴巴M-DMP负责人许成访谈：从顾客管理到消费者管理的升级
小微金服前端技术专家伯约访谈：相信数据的力量

## 避开那些坑 | Void

那些年我们犯过的错（一）——豆瓣Xupeng：补齐那丢失的三分钟数据
Async/Await - 性能开销和其他陷阱

## 新品推荐 | Product

Java EE 7推出Expression Language 3
Visual Studio 2012发布新更新
.NET Reflector 8.2 支持Visual Studio 2013、 局部变量高亮显示和代码地图视图
Essential Studio for JavaScript已发布，用于开发基于业务线的Windows Store应用
Google Dart新进展：Polymer代替Web UI
GAE 1.8.2引入专属Memcache和模块化
Ruby on Rails 4发布：通过Turbolinks提速页面

Powered by InfoQ

# QClub

## 我们影响有影响力的人

北京 上海 广州 大连 西安 太原 成都 杭州 武汉 南京 深圳...

中文 | 英文 | 日文 | 葡文 | .....

内容投稿 editors@cn.infoq.com | 商务合作 sales@cn.infoq.com | 加入我们 hr@cn.infoq.com

# 人物 | People

# OSChina的技术选型：就是为了短平快

作者 张龙

OSChina.net，目前每天的访问IP在50万左右，每天的动态请求在2000万左右，高峰期并发请求达到8000。

这是一个纯Java技术开发的网站，没有使用任何框架。整个网站的底层只需要一台低端服务器：

- DELL SC 1435 (1台)
- CPU：双 AMD 2G
- 内存：4 * 2G + 4 * 1G
- 硬盘：250G * 2

这样一个技术选型是如何决策的？InfoQ编辑与OSChina最初的代码撰写者刘柄成（红薯）进行了沟通。

**InfoQ：OSChina网站仅采用了Servlet+Velocity这一在今天看起来颇为“简陋”的技术架构，与当今SSH打天下的做法“背道而驰”，请问是如何做出这个决定的，在网站的发展过程中，是否有过转型到框架上的想法？**

> 红薯：当初开发网站的时候是为了短平快，因为网站一开始功能非常的简单，只是一些开源软件的分类展示，因此整个系统最麻烦的地方其实是页面的展示。而且对于 Java 开发网站一个最麻烦的地方便是一旦修改了 Java 的代码就需要重启应用服务器。对一个互联网网站来说，不能因为这些技术的问题导致用户的访问频繁中断。因此刚开始架构选型的思路便是后端代码只提供一些数据的基本封装，由前端的模版自由组合数据。
>
> 在网站这五年的发展过程中，我们深刻体会这种设计带来的便利性。由于产品不断的在完善和更新中，经常要对页面做各种调整，包括版面、显示的数据等。由于是模版根据自身需要调用 Velocity Toolbox 的方法去获取数据，相当于每次的调整只需要修改模版页面即可，随时更新到服务器随时生效，无需

重启应用服务器，非常的便利。而常规的 SSH 做法，当我需要在页面上增加某些额外的数据显示时，往往需要在 Controller (Java类) 中读取数据，然后需要重新编译，重新发布……

目前尚未有转型到框架上的想法。

**InfoQ：你提到目前SSH框架结构流行，而且很多应用都采用了过于模式化的设计。能举几个例子说明一下这种过于模式化的设计吗？**

红薯：我接触过很多用 Spring 框架开发的项目，过份的接口化设计是普遍存在的问题。

好比说访问用户资料的方法，首先会做一个 UserDAO 的接口，然后再做一个 UserDAO_Impl 的接口实现类。理由是可以方便扩展其他读取资料的实现，或者部署为支持远程调用的接口方法等等

但你会发现，至始至终不曾要求过其他第二种用户资料的读取方式，换成是我比较极端，一个静态的 UserTool 类搞定即可。

**InfoQ：底层与数据库交互之处仅仅使用了JDBC么？有没有做过哪些具体的优化措施？**

红薯：OSChina 访问数据库使用了 dbutils 这个工具包以及阿里的 druid 连接池实现。

我曾是 Hibernate 的深度用户，深感 Hibernate 在实现跨数据库支持的便利性，我以前的一个 DLOG4J 开源博客产品，因为需要支持各种数据库，因此用 Hibernate 来开发非常之方便，你不需要关心数据库之间的差异性。

但对 OSChina 来说，从第一天上线就以前确定了它是运行在 MySQL 数据库上，不会改变。既然这样，我们就不认为 Hibernate 能给这个项目带来多少好处，而且又是这么庞大的一个 jar 包。相比之下 dbutils 简单得多得多。

优化方面主要是大量的缓存处理以及索引优化。

**InfoQ：OSChina网站的读写并发量如何，读与写的比例大概是多少？针对这个比例采取了怎样的缓存手段？**

红薯：我们还真没统计过读写的比例，目前会根据具体的功能需求来处理缓

存。

例如：

a）发布新闻、新帖，无需即时更新列表，通过缓存中配置的失效时间自动更新列表

b）回帖、评论、动弹等，用户发布完回帖必须马上看到新的内容显示，因此会在代码上手工去清除对应的缓存

整个 OSChina 的代码中有大量的代码是用来处理缓存的更新问题。

**InfoQ：之前是否做过评估，如果这个请求量按照流行的SSH框架结构来实现，预计会需要怎样的硬件配置进行支撑？**

> 红薯：这个真没做过评估，我觉得两种开发架构对硬件的要求不会差别太多，关键是不断的开发更新上会更加便利，此外使用 SSH 开发的应用在启动上也需花费更多的时间。

**InfoQ：服务端仅使用了Servlet，请问是如何实现MVC的，是通过制度强制开发人员遵循，还是采取了哪些具体的分层手段来实现MVC？**

> 红薯：OSChina 自己实现了一个简单的 MVC ，代码请看这里。
>
> 无需任何配置文件，依照“惯例重于配置”的思路。

**原文链接：http://www.infoq.com/cn/news/2013/07/oschina-on-tech-choice**

**相关内容**

- 58同城开源其轻量级Web框架Argo
- PrimeFaces 3.0发布：面向JSF2的Ajax、移动与IE 9组件
- Grails最佳实践
- ZK Web框架6.0发布：新型数据绑定系统
- Flash与Flex路在何方？

# 观点 | View

## 事物不断变化（流程也需调整）

**作者 Shane Hastie ，译者 李彬**

Jonathan Kohl是一位咨询顾问、作家和知名博主，他提出了这样的观点：时下团队在工作中身处的或正在构建的技术环境，与过去那些曾诞生了许多通用开发流程的环境已经有了很大不同。他鼓励团队在检查流程时保持清醒和活跃，并让自己做好准备，以能够适应当今他们工作中身处的现实。

他最近在Better Software杂志上发表了一篇题为“事物不断变化（流程也需调整）”的文章，并给出了一些在移动设备上使用自动化测试的例子：

> 广泛使用自动化单元测试，是一项得到普遍认可的敏捷开发实践。在过去十五年中，我们见证了这一领域中单元测试工具、框架、实践和知识体系方面的大爆炸——亲身经历这一过程也是件令人兴奋的事情。然而，在移动开发框架方面，自动化单元测试得到的支持程度很低。当我们习惯于在Web或其他较老的技术中，享受这些工具所带来的好处时，移动开发框架中的匮乏会成为让人们感到困扰的因素。对移动设备来说，状态至关重要——无线网络条件、设备的运动和传感器优化、人机交互、情感和知觉，甚至其他一些事情，例如天气和光线的变化等等。但实际上，自动化单元测试无法真正解决其中任何问题……由此，我熟识的移动开发者们倾向于很少依赖自动化单元测试，转而大量使用其他质量实践作为补充。

他提倡根据当前的需求，谨慎地运用多种不同技术；并建议团队做好准备，在选定的过程方法的“规则”之外进行探索。他说道：

> 我的一些敏捷教练朋友几乎丧失了理智，关切地观察某个移动开发团队是否使用（如果确实有的话）单元测试——只是为了发现团队使用了不同的工具，而且这些工具能够更好适应其最终用户要求的质量准则。移动团队一开始大多使用标准开发流程和工具集，随后会发现不足或是缺乏特定领域的支持，并且会调整流程。

Laurant Bossavit在其著作“The Leprechauns of Software

Engineering”中对软件工程方面的某些“根本事实”提出了质疑，他对于遵循近期Google+的一条thread里的流程或方法的空谈也表达了类似的观点。对于当今在团队中发现的许多行为，以及在许多组织机构中采用OO之后所发生的事情，他进行了对比并表示：

> “在任何敏捷流程中，我都可以让自己戴上耳机整天躲在屏幕后面”是“我可以使用任何面向对象语言编写过程代码”的新的体现。

他继续探讨了以往敏捷有多么处于实践的前沿：

> 那时候，敏捷是全新并且激进的；如果遇到某位对敏捷有兴趣的人，我们可以非常确定他们具有好奇、有积极性、聪明的特性，一言以蔽之，他们是与众不同的。他们并不一定完美或是能够自动成为任何环境中的绝佳人选，但至少他们远超同侪。

然而现如今的现实情况是：

> 敏捷不再是“激进的”，如果有人因为敏捷能让自己遇到不同寻常的人而对其产生兴趣，那么我可以完全理解他在寻找某些别的扮演这一角色的东西。

他并不赞成把事情丢给敏捷不管，而是建议投入时间和精力来确保从敏捷方法获得最大收益。这需要努力以及谨慎的思考：

> 类似地，只是因为有太多的误报（假阳性数据）——人们宣称预“敏捷”结合，却鲜有真正的经验或倾向——并不意味着这个标签是没有价值的。它意味着我们应该关注表面现象下面的内容，并找出那些难以评估的事情，去关注有价值的信号而不是那些廉价的。
>
> 同理，在那些最大声宣称自己拒绝敏捷的人之中，许多人将不可避免地进行模仿——练习一种在恰当时机改变自己花纹的古老的进化艺术。而只有那些没能拥有由艳丽色彩所象征的昂贵却又是实质性的属性的物种，才会练习这项艺术。

在文章中，Kohl总结道：

> 如果我们已经实现了一个流程，而且在最初的成功后，我们发现了质量问题、团队成员陷入困境、或是持续地延迟和超出预算，那么或许应该检查问题的根源是否在自己身上。不要谴责团队成员，而是去检查流程：它或许不再适用；它是否帮助团队中的所有人创造价值，或是已成为大家的阻碍？同时，牢记我们需要创造价值，不仅是为了客户、项目干系人和公司老板，还为了我们的团队成员和我们自己。任何不能积极创造价值的方面都会成为问题。

**查看英文原文：**Things Change (and so should processes)

**原文链接：http://www.infoq.com/cn/news/2013/07/things-change-processes**

# 观点 | View

## 函数式编程为何越来越受到重视

作者 崔康

函数式编程受到越来越多的关注，软件架构师Neal Ford分析了其中的原因。

Neal Ford首先强调，函数式编程的特点之一是存在强大的抽象，它隐藏了许多日常操作的细节（比如迭代）。用抽象来处理迭代等任务，使得需要维护的代码变得更少，因此可能出现错误的地方也就更少。

> Java 开发人员习惯于框架 级别的重用；在面向对象的语言中进行重用所需的必要构件需要非常大的工作量，他们通常会将精力留给更大的问题。函数式语言在更细化的级别提供重用，在列表和映射等基本数据结构之上通过高阶函数提供定制，从而实现重用。
>
> 在面向对象的命令式编程语言中，重用的单元是类以及与这些类进行通信的消息，这些信息是在类图中捕获的。在 OOP 的世界中，鼓励开发人员创建独特的数据结构，以方法的形式附加特定的操作。
>
> 函数式编程语言尝试采用不同的方式来实现重用。它们更喜欢一些关键的数据结构（如列表、集和映射），并且在这些数据结构上采用高度优化的操作。传递数据结构和高阶函数，以便 “插入” 这种机制，针对某一特定用途对其进行定制。
>
> 函数级的封装支持在比构建自定义类结构更细的基础级别上进行重用。此方法的优势之一已经体现在 Clojure 中。最近，库中的一些巧妙创新重写了 map 函数，使它可以自动并行化，这意味着所有映射操作都可以受益于没有开发人员干预的性能提升。

Neal Ford举了一个解析 XML 的例子。大量的框架可用于在 Java 中完成这个任务，每个框架都有自定义的数据结构和方法语义（例如，SAX 与 DOM）。

> Clojure 将 XML 解析为一个标准的 Map 结构，而不是强迫您使用自定义的数据结构。因为 Clojure 中包含大量与映射配合使用的工具，如果使用内置的列

> 表理解函数 for，那么执行 XPath 样式的查询就会很简单。
>
> 最初，Clojure 中令人生畏的细节之一是：与映射和其他核心数据结构进行交互的方法似乎有无限多种。然而，它反映了这样一个事实：在 Clojure 中，大多数内容都尝试解决这些核心的、优化的数据结构。它没有将解析的 XML 困在一个独特的框架中，相反，它试图将其转换为一个已存在相关工具的现有结构。

其次，函数式编程提供了新的工具类型，以优雅的方式解决棘手的问题。例如，Java 开发人员不习惯尽能延迟生成其值的惰性 数据结构。而未来的函数式语言将对这种高级特性提供支持，一些框架将此功能加装到 Java 中。

第三点是让语言迁就问题，大多数开发人员都将他们的工作误解为接受一个复杂的业务问题，将它转换成 Java 等语言。

> 他们的这种误解是因为 Java 并不是一种特别灵活的语言，它迫使您让自己的想法适应于已经存在的刚性结构。但是，当开发人员使用可塑语言时，他们看到了让语言迁就问题，而不是让问题迁就语言的机会。像 Ruby（它为领域特定语言 (DSL) 提供了比主流更友好的支持）等语言证明了这种潜在可能。现代函数式语言甚至走得更远。Scala 旨在协调内部 DSL 的托管，并且所有 Lisp（包括 Clojure）都可以提供无与伦比的灵活性，使开发人员能够让语言适应问题。

最后一点是与语言的趋势相一致，函数式编程的目标之一是最大程度地减少可变状态。像垃圾收集一样，现在缓存也可以降级用于语言。

> 由于运行时获得了更多的能力并且有多余的开销，开发人员可以将繁忙的工作割让给语言，将我们解放出来，去思考更重要的问题。Groovy 中的备忘功能就是众多示例中的一个；因为基础运行时允许这样做，所有现代语言都添加了函数式构造，包括 Totally Lazy 等框架。

微软的函数式编程语言F#最近随Visual Studio 2013预览版发布了3.1版本。3.1版的主要新增语言特性：

- 命名的联合类型字段——现在可以为每个联合类型中的联合字段命名
- 数组切片扩展——之前的数组切片只能用于同等维度的数组，现在这不是必须的了
- 为LINQ风格的方法增强了类型推断——改进了F#中使用函数作为参

数的重载方法的类型推断

- 支持第一个参数为类型参数的C#风格的扩展方法——3.1支持使用扩展方法（this关键字），比如：static member M<T>(this T input, T other)
- 在特性和字面量表达式中对常量进行组合——比如，可以使用“+”来连接字符串，或对枚举标记进行按位OR操作（|||）

**原文链接：http://www.infoq.com/cn/news/2013/07/functional-programming**

**相关内容**

- Fn.py：享受Python中的函数式编程
- Nashorn获投票通过，成为OpenJDK项目中Rhino的后继
- Dojo 1.8：向完美架构继续前行
- 带有基于Smalltalk的Ruby VM的NoSQL OODB：MagLev 1.0发布了
- Java EE 7推出Expression Language 3

QCon

International
Software Development
Conference

# 全球软件开发大会

## [上海站]
## 2013

2013年11月1-3日
上海光大会展中心
国际大酒店

优惠报名

**8**折限时报名中……
5人团购更享优惠

TICKET 抢购热线：010-64738142
票务咨询：qcon@cn.infoq.com

精彩内容策划中，讲师自荐/推荐，线索提供：
speakers@cn.infoq.com

更多经典专题 精彩内容 敬请登录：
qconshanghai.com

## 部分特色专题

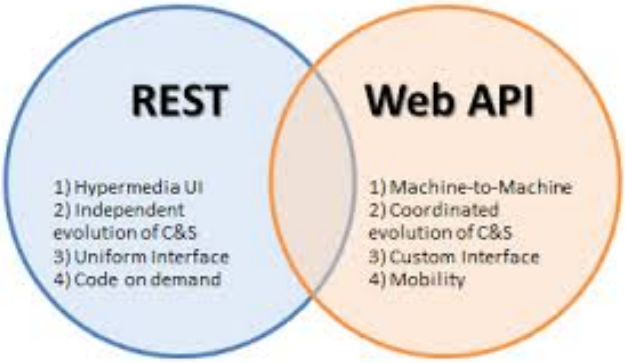
金融系统设计与架构
系统安全设计面面观
扩展性、可用性与高性能
推荐系统
移动应用案例分析
构建高效能团队
JavaScript生态圈
来自一线的敏捷实战
知名网站案例分析

# 专题推荐语

## 再谈REST

在现在的互联网世界中，大家都在忙着构建平台，外有Amazon、Google、Facebook、Twitter，内有百度、腾讯、阿里巴巴，在这些平台的背后不能忽视的是REST。是的，通过REST大家可以简单的实现异构的应用到应用通信，这种通讯的方式很轻，你可以抛弃SOAP了！

在7月份，我们的编辑马国耀也发起了一次虚拟的座谈会来谈谈如何构建优秀的REST API。在本次的专栏中，我们还推荐了关于REST中幂等的误用以及Web风格相关的内容，希望于您有用。



# 本期专题：再谈REST | Topic

# 虚拟研讨会：如何设计好的RESTful API？

**作者 马国耀**

REST架构风格最初由Roy T. Fielding（HTTP/1.1协议专家组负责人）在其2000年的博士学位论文中提出。HTTP就是该架构风格的一个典型应用。从其诞生之日开始，它就因其可扩展性和简单性受到越来越多的架构师和开发者们的青睐。它必将得到更大的发展。一方面，随着云计算和移动计算的兴起，许多企业愿意在互联网上共享自己的数据、功能；另一方面，在企业中，RESTful API（也称RESTful Web服务）也逐渐超越SOAP成为实现SOA的重要手段之一。为此，InfoQ将通过一系列深度文章和虚拟研讨的形式，深入剖析REST，以飨读者。

设计好RESTful API对于软件架构的可扩展性、可伸缩性和消费者的体验都具有至关重要的作用。本次虚拟研讨会的主题是，如何设计好的RESTful API。我们邀请了几位国内对REST风格架构及RESTful API设计颇有心得的专家，请他们就自己的实战经验谈谈对这一问题的看法。他们分别是李锟、李建业、丁雪丰、马钧（注：文末有他们的个人简介）。

## 问题

我们为本次虚拟研讨会设计的问题有：

1. 什么是好的RESTful API？相信每个人都有自己的评判标准。那么，您认为一个好的RESTful API应该具有哪些特征呢？
2. 安全是恒久的话题，对于基于WSDL和SOAP的Web Service，我们有WS-Security这样的安全规范来指导实现认证、授权、身份管理等安全需求。那么，RESTful API有无成熟可用规范或实现框架呢？如何保证RESTful API的安全性呢？
3. 如何对RESTful API进行版本控制，请分享您认为实用的做法？
4. HTTP1.1规范中给出的动词对于设计RESTful API够用吗？您在实际项目中会扩展自己的动词吗？在什么情况下需要扩展？

5. 今年5月份发布的JAX-RS 2.0规范对于RSTfulAPI的设计最有价值的特性是哪个（些）？它（们）用于解决什么问题？
6. 能否为InfoQ的读者们推荐一款实用的RESTful API开发框架，并说明您的推介理由。
7. HTTP2.0规范正在制定当中，您对它的期待是什么？

**InfoQ：什么是好的RESTful API？相信每个人都有自己的评判标准。那么，您认为一个好的RESTful API应该具有哪些特征呢？**

**李锟：**一个好的RESTful API，应该具备以下特征：

1. 这个API应该是对浏览器友好的，能够很好地融入Web，而不是与Web格格不入。

   浏览器是最常见和最通用的REST客户端。好的RESTful API应该能够使用浏览器+HTML完成所有的测试（不需要使用编程语言）。这样的API还可以很方便地使用各种自动化的Web功能测试、性能测试工具来做测试。Web前端应用（基于浏览器的RIA应用、移动App等等）也可以很方便地将多个RESTful API的功能组合起来，建造Mashup类的应用。

2. 这个API中所包含的资源和对于资源的操作，应该是直观和容易理解的，并且符合HTTP协议的要求。

   REST开发又被称作“面向资源的开发”，这说明对于资源的抽象，是设计RESTful API的核心内容。RESTful API建模的过程与面向对象建模类似，是以名词为核心的。这些名词就是资源，任何可命名的抽象概念都可以定义为一个资源。而HTTP协议并不是一种传输协议，它实际提供了一个操作资源的统一接口。对于资源的任何操作，都应该映射到HTTP的几个有限的方法（常用的有GET/POST/PUT/DELETE四个方法，还有不常用的PATCH/HEAD/OPTIONS方法）上面。所以RESTful API建模的过程，可以看作是具有统一接口约束的面向对象建模过程。

   按照HTTP协议的规定，GET方法是安全且幂等的，POST方法是既不安全也不幂等的（可以用来作为所有写操作的通配方法），PUT、DELETE方法都是不安全但幂等的。将对资源的操作合理映射到这四个方法上面，既不过度使用某个方法（例如过度使用GET方法或POST方法），也不添加过多的操作以至于HTTP的四个方法不够用。

   如果发现资源上的操作过多，以至于HTTP的方法不够用，应该考虑设计出

更多的资源。设计出更多资源（以及相应的URI）对于RESTful API来说并没有什么害处。

3. 这个API应该是松耦合的。

   RESTful API的设计包括了三个循序渐进、由低到高的层次：资源抽象、统一接口、超文本驱动。正是这三个层次确保了RESTful API的松耦合性。

   当设计面向互联网的API时，松耦合变成了一种“必须有”的强需求。紧耦合的API非常脆弱，一旦公布出去，服务器端和客户端都无法持续进化。尤其是服务器端，公布出去的接口根本不敢改，改了之后，几乎所有客户端应用立即无法正常工作。REST这种架构风格就是紧耦合API的解毒剂，这个话题可以谈的很深，这里就不展开了。感兴趣的读者可以参考《REST实战》。

4. 这个API中所使用的表述格式应该是常见的通用格式

   在RESTful API中，对于资源的操作，是通过在服务器端-客户端之间传递资源的表述来间接完成的。资源的表述可以有很多种格式，并且在响应和请求中的资源表述格式也会有所不同。GET/POST响应中的资源表述格式，常见的有HTML、XML、JSON；POST/PUT请求中的资源表述格式，常见的有标准的HTML表单参数、XML、JSON。

   这些常见表述格式，处理起来非常容易，有大量的框架和库提供支持。所以除非有很合理的要求，通常不需要使用自定义的私有格式。

5. 使用HTTP响应状态代码来表达各种出错情况

   HTTP响应状态代码，是HTTP协议这个统一接口中用来表达出错情况的标准机制。响应状态代码分成两部分：status code和reason phase。两部分都是可定制的，也可以使用标准的status code，只定制reason phase。

   如果一个所谓的“RESTful API”对于任何请求都返回200 OK响应，在响应的消息体中返回出错情况信息，这种做法显然不符合“确保操作语义的可见性”这个REST架构风格的基本要求。

6. 这个API应该对于HTTP缓存是友好的

   充分利用好HTTP缓存是RESTful API可伸缩性的根本。HTTP协议是一个分层的架构，从两端的user agent到origin server之间，可以插入很多中间组件。而在整个HTTP通信链条的很多位置，都可以设置缓存。HTTP协议内建有很好的缓存机制，可以分成过期模型和验证模型两套缓存机制。如果API设计者完全没有考虑过如何利用HTTP缓存，那么这个API的可伸缩性会

有很多问题。

**李建业：**首先说明一下，对REST这个概念，我一般把它理解为REST风格的架构，但是现在实践中最为广泛认知的是HTTP，而它是REST的一个实现，所以RESTful API也可以不太严格的指基于HTTP的API——当然，即使是不严格的时候，API本身也应该力求遵循REST架构风格。

我认为，一个RESTful API最重要的一点应该是——**“尽可能少的先验信息”**，这一条也同时是我判断一个好的RESTful API的标准。

比如HTTP动词，在实践中，大家可能会常常纠结于有效利用 HTTP 动词，但这却并不是特别重要的事情——除非你理解这么做的价值。HTTP 动词最重要的地方在于它是标准阐明了的行为，也就是说，如果我们的“客户端”遵循约定，那么就不必要发明新的动词，也就不必增加“先验信息”；但是，所谓“先验信息”，针对的是客户端——对API来说就是调用者，对于一些企业内部系统，或者一些传统系统，由于“资源”很稳定，对资源的操作也很稳定，这些系统的“调用客户端”不是浏览器而是另一个系统，此时如果强制对应到HTTP动词，反而会变成额外的“先验信息”，这时我就不会太拘泥HTTP动词，自己制定一套动词放在参数中也可以接受——只要动词不变化，这个系统依然是REST风格的。

再比如Response里面的Content-Type，这个有时会被新手忽略，但这其实很重要，因为一般涉及到系统间协同的API，往往不会使用普通的文本，比较常见的是使用json表达复杂结构，而这与通常的缺省理解不同（缺省一般会认为是text/plain和text/html），所以如果在API中忘记用Content-Type进行区分的话，后续对多种类型的客户端接入的支持就会变成陷阱（我们多次遇到过这个问题）。而如果一开始就检查是否增加先验知识（缺省Content-Type为plain或者允许指定Content-Type），那这一困难就可以避免了。

**丁雪丰：**首先，应该正确地使用HTTP的统一接口，比如HTTP的动词，如果不分青红皂白清一色POST那显然还有改进的余地；

其次，资源有合适的粒度，可以从三个方面来评判资源的粒度是否合理——网络的效率、表述的大小以及客户端使用时的易用程度；

最后，是表述的设计，除了表述的正文内容，还有其中的URI和链接，这些都是评判一个RESTful API好坏的标准。

**马钧：**在我看来，一个好的API标准，就是能尽量利用到HTTP协议的特性，将HTTP当成一种转移协议，而不是传输协议。包括但不限于：利用HTTP的各种动词来明确操作；包含有内容协商，可以根据请求头提供的参数选择一个资源最合适的媒体类型、语言、字符集和编码的表现；使用不同的返回代码来描述各种状态。但实际上见到过的很多声称RESTful API，包括国内的和国外的，能符合这些条件的并不多。parse.com提供的API是我见到过的较为不错的RESTful API，可以作为范例参考。

**InfoQ：安全是恒久的话题，对于基于WSDL和SOAP的Web Service，我们有WS-Security这样的安全规范来指导实现认证、授权、身份管理等安全需求。那么，RESTful API有无成熟可用规范或实现框架呢？如何保证RESTful API的安全性呢？**

**李锟：**保证RESTful API的安全性，主要包括三大方面：

a) 对客户端做身份认证

b) 对敏感的数据做加密，并且防止篡改

c) 身份认证之后的授权

对客户端做身份认证，有几种常见的做法：

1. 在请求中加签名参数

   为每个接入方分配一个密钥，并且规定一种签名的计算方法。要求接入方的请求中必须加上签名参数。这个做法是最简单的，但是需要确保接入方密钥的安全保存，另外还要注意防范replay攻击。其优点是容易理解与实现，缺点是需要承担安全保存密钥和定期更新密钥的负担，而且不够灵活，更新密钥和升级签名算法很困难。

2. 使用标准的HTTP身份认证机制

   HTTP Basic身份认证安全性较低，必须与HTTPS配合使用。HTTP Digest身份认证可以单独使用，具备中等程度的安全性。

   HTTP Digest身份认证机制还支持插入用户自定义的加密算法，这样可以进一步提高API的安全性。不过插入自定义加密算法在面向互联网的API中用的不是很多。

   这个做法需要确保接入方“安全域-用户名-密码”三元组信息的安全保存，另外还要注意防范replay攻击。

优点：基于标准，得到了广泛的支持（大量HTTP服务器端、客户端库）。在服务器端做HTTP身份认证的职责可以由Web Server（例如Nginx）、App Server（例如Tomcat）、安全框架（例如Spring Security）来承担，对应用开发者来说是透明的。HTTP身份认证机制（RFC 2617）非常好地体现了“分离关注点”的设计原则，而且保持了操作语义的可见性。

缺点：这类基于简单用户名+密码机制的安全性不可能高于基于非对称密钥的机制（例如数字证书）。

3. 使用OAuth协议做身份认证

   OAuth协议适用于为外部应用授权访问本站资源的情况。其中的加密机制与HTTP Digest身份认证相比，安全性更高。需要注意，OAuth身份认证与HTTP Digest身份认证之间并不是相互取代的关系，它们的适用场景是不同的。OAuth协议更适合于为面向最终用户维度的API提供授权，例如获取隶属于用户的微博信息等等。如果API并不是面向最终用户维度的，例如像七牛云存储这样的存储服务，这并非是OAuth协议的典型适用场景。

对敏感的数据做加密，并且防止篡改，常见的做法有：

1. 部署SSL基础设施（即HTTPS），敏感数据的传输全部基于SSL。
2. 仅对部分敏感数据做加密（例如预付费卡的卡号+密码），并加入某种随机数作为加密盐，以防范数据被篡改。

身份认证之后的授权，主要是由应用来控制。通常应该实现某种基于角色+用户组的授权机制，这方面的框架有不少（例如Spring Security），不过大多数开发团队还是喜欢自己来实现相关功能。

**李建业：**我不认为安全是RESTful API需要考虑的问题，事实上我觉得这是两个正交的问题。当然，如果使用RESTful API来提供认证、授权和身份管理，那也算是双方有关系，但是这和其它风格的API设计所要考虑的问题似乎没什么区别，不值得特别注意。

但是在具体设计层面，这两者的“正交点”上似乎确实有些问题，因为REST是一个推崇状态无关原则的架构风格，而认证和授权通常基于第三方解决方案，所以往往会出现违背有状态约束的问题，这个地方我也没有特别的想法，当然这个困难和原问题关系不大。

至于WS-族的协议，我不太了解，不太能参与讨论。

**丁雪丰：**对于RESTful API，常见的安全措施都是可以继续使用的。例如，为了防篡改，可以对全部参数进行签名；为了防范重放攻击可以在请求中增加一次性的Token，或者短时间内有效的Token；对内容加密可以实现数据防泄露……；对于DDoS攻击，各种HTTP流量清洗策略，都可以继续发挥作用，因为这就是基本的HTTP请求。

在授权和认证方面，OAuth 2.0已经基本成熟了，并且得到了广泛地应用。如果可以，接入第三方账户体系是个不错的选择，比如Google和Facebook的，国内的当然也有几个候选。

**马钧：**个人认为RESTful的安全性分为几个层次，在安全要求较高的场合，可以通过HTTPs这样的加密协议来保证网络层的安全，应用层的安全可以通过OAuth实现认证，而对于资源的访问授权，则只能依靠应用程序来实现了。

**InfoQ：如何对RESTful API进行版本控制，请分享您认为实用的做法？**

**李锟：**一个比较简单实用的做法是直接在URI中插入版本号，这样做允许多个版本的API并行运行。

另一个做法是在HTTP请求中加入自定义头信息，标明使用的版本号。不过这个做法其实对浏览器不够友好，简单地使用浏览器+HTML无法测试。

**李建业：**目前比较好的方式还是在uri设计中添加版本信息，其它方法都不如这个实用。

**丁雪丰：**个人认为最好的版本化，就是没有明显的版本。在对已发布的服务进行变更时，要尽量做到兼容，其中包括URI、链接和各种不同的表述的兼容，最关键的就是在扩展时不能破坏现有的客户端。例如，要变更一个参数，可以选择同时兼容新旧两种输入，或者保持老参数不动，提供一个新的参数，在文档中必须做出说明，不推荐新用户再继续使用之前的参数。

如果必须要进行不兼容的变更，那么可以选择标记不同的版本号，这时可以选择在路径或参数中增加版本信息。也有做法是增加HTTP标头，只是在调用时会稍有不便，推荐前两种方法。

**马钧：**RESTfulAPI的版本升级，尽量兼容之前的版本，保证原有的API都能正常工作，可以通过HTTP 301转跳到新的资源。另外一种实用的做法就是在url中保留版本号，同时提供多个版本供客户端使用，如 v1.rest.com 或者 rest.com/v1/ 这样。

**InfoQ：HTTP1.1规范中给出的动词对于设计RESTful API够用吗？您在实际项目中会扩展自己的动词吗？在什么情况下需要扩展？**

**李锟：**这个问题取决于设计者如何看待和设计资源。如果资源抽象做的很好，对于某个资源的任何操作，通常都能够映射到CRUD四个类别中。CRUD四个类别对于操作资源来说，绝大多数情况下是完备的。HTTP的GET/POST/PUT/DELETE四个方法，对于CRUD四个类别的操作来说是足够的，映射关系是Create-POST/Retrieve-GET/Update-PUT/Delete-DELETE。

我们通常不会选择创建自己的动词，这样做对于客户端开发者来说，需要更多的学习成本。如果在资源上定义的操作过多，我们会选择拆分出更多的资源。

**李建业：**一般是够用的，有时一些“不够用”的场景是由于我们没有设计出合理的资源，比如批量操作。但是，正如之前所说的那样，对于某些内部的、传统的（因此模型稳定且已知）系统，API提供者和调用者会有自已的固定动词表，此时没必要拘泥。另外，我不建议扩展动词，一旦扩展了动词，其实已经破坏了我之前说的*“**尽可能少的先验信息**”*，那么，扩展动词和重新设计动词的成本差别不大。基于这个考虑，我建议尽可能保持动词不变，除非你想重新设计动词表。

**丁雪丰：**一般情况下，常用的HTTP动词是够用的，并没有出现一定要自己扩展动词的情况。其实，最常用的也就是GET、POST、DELETE和PUT，而HEAD、OPTIONS、TRACE则基本用不太到。如果出现一时找不到合适的动词，安全幂等的操作用GET，其他都可以用POST，在设计资源时稍加考虑即可。

**马钧：**在我的实际项目中，只用到了POST，PUT，DELETE，GET这四个动词。

**InfoQ：今年5月份发布的JAX-RS 2.0规范对于RSTfulAPI的设计最有价值的**

**特性是哪个（些）？它（们）用于解决什么问题？**

> **李锟：**REST开发框架RESTEasy项目负责人Bill Burke，去年写了一篇文章介绍JAX-RS 2.0。
>
> 我同意Bill在文章中的观点，在JAX-RS 2.0增加的内容中，最重要的三部分为：
>
> a) Client API——用来规范化JAX-RS客户端的开发方式。
>
> b) Server-side Asynchronous HTTP——用来实现服务器端推送功能，而不需要依靠低效的轮询方式。
>
> c) Filters and Interceptors——用来分离关注点，将鉴权、日志等逻辑与业务逻辑分离开，更好地实现代码重用。
>
> 这三部分的内容对于开发者来说都很有用。遵循JAX-RS规范做开发，可以确保服务器端以及客户端代码的可移植性。

> **李建业：**我个人关注异步API这部分，主要是因为流式服务将会越来越多，那将大量需要这类支持。

**InfoQ：能否为InfoQ的读者推荐一款实用的RESTful API开发框架，并说明您的推介理由。**

> **李锟：**这个问题我就不详细回答了。不同的编程语言有不同的REST开发框架，对于REST的支持程度也不同。开发RESTful API的需求范围很广，可选择的开发框架的范围也很广。保持多样性是繁荣生态环境的基础。像Java就有支持JAX-RS规范的Jersey、RESTEasy、Restlet、Apache CXF，和不支持JAX-RS规范的Spring MVC等等很多框架。这些框架目前都做的不错。我对框架的选择没有倾向性。RESTful API设计的最佳实践应该是通用的，而不是必须依赖某种特定的开发框架。

> **李建业：**不好意思，这个我不太重视，没法推荐，不过我可以解释一下为什么对RESTful API框架不感冒的原因。
>
> REST作为一个架构风格，对我们的系统开发有很大影响，但是这些影响一般是针对架构（例如状态无关）或者设计（例如资源识别）上的，所以一旦涉及到

具体实现，主要工作就基本结束了，此时开发框架能做的事也就只有简化编程了（相较而言，有的框架还能起到引导设计的作用），而由于RESTful会抽象动词，所以实现层面中和API规范相关的工作本来就不多，那么框架的价值就更小了。

当然，我们也不可能直接基于servlet/rakc/wsgi来开发，不过一般的编程语言都会提供一些简单的url route/match策略，我们使用这些就足够了。另外，有些框架能帮我们生成全部的动词支持，但这也未必是好事，我一般倾向于按需实现——用到了再支持，这就更不需要太关注开发框架对RESTful的支持了。

**丁雪丰：**由于本人是Spring的拥护者，工作中也一直在使用Spring，所以在选择框架时会更多地倾向Spring MVC（并不是说别的框架不好，这里有些个人主观的成份）。如果一定要选择其他框架，也要选择能够方便与Spring集成的框架。如果在项目中已经使用了Spring，那么没有什么理由不选择Spring MVC，鉴于目前Spring在各种项目中的高出镜率，相信一般情况下都会选择Spring MVC。

REST的成熟度模型中，第三层就是HATEOAS，Spring目前还提供了Spring Hateoas子项目，对链接、资源等方面的支持都做了一定的增强。

**马钧：**我目前在实际项目中使用的是Spray，这是一个开源的 REST/HTTP 工具包和底层网络 IO 包，基于 Scala 和 Akka 构建。轻量级、异步、非堵塞、基于 actor 模式、模块化和可测试是Spray的特点。

**InfoQ：HTTP2.0规范正在制定当中，您对它的期待是什么？**

**李锟：**我的期待包括两个方面：应该做的和不应该做的。

HTTP/2.0规范应该做的：

1. 与HTTP/1.1协议保持兼容。兼容的含义是说两者可以并存，客户端应用可以根据服务器端的能力，自由地选择使用HTTP/2.0还是HTTP/1.1，而且选择过程对应用来说是透明的。
2. 改进HTTP协议（作为资源的统一接口）之中操作语义表达方式的语法，提高网络传输效率。
3. 更好地模块化，这样HTTP/2.0协议的实现能够更好地模块化。应用程序可根据需要选择适当的模块，而不是要么全有、要么全无。

4. 废弃掉HTTP/1.1协议中一些很少有人用到的部分，例如采用管道（pipelining）方式发送请求。
5. 增加更多的动词，以适应除CRUD之外的其他场景。

HTTP/2.0规范不应该做的：

1. HTTP/2.0协议不应该把底层的数据加密机制（即SSL）作为必选项。
2. HTTP/2.0协议不应该背离REST架构风格的约束，尤其是要确保操作语义对于中间组件的可见性。

在上面这两个方面，Roy Fileidng曾经与SPDY协议设计者Mike Belshe发生过激烈争论，详情请看：Roy Fielding谈Google SPDY协议

**李建业：**对此规范关注不多，不知道会不会有对于**流**的支持，目前我所知道的只有chunk方式进行简单的支持，但是真正的流需要区分数据通道和控制通道——哪怕是逻辑上的区分，这样就直接对REST风格产生了很大冲击，考虑到流式服务在未来的发展潜力，我特别期待业界在这方面有所进展。

**丁雪丰：**HTTP 2.0很大程度上是借鉴了Google的SPDY，就我而言，首先，希望这个规范能做到与HTTP 1.1的兼容，使用者如果只认识1.1，那么2.0能优雅“降级”；其次，希望2.0能带来更好的性能，SPDY在这方面还是有所改进的，希望HTTP 2.0能再接再厉；最后，希望这个规范能在最终定稿时附带一个最佳实践，正确引导人们合理地使用HTTP 2.0。

**马钧：**没研究过，估计即使出来，1.1还有很长的生命周期，不会很快被取代。

# 关于专家们

**李锟：**从2003年开始尝试Ajax开发，意识到这种开发方式的优点后，次年（2004年）在知名技术论坛JavaEye上与技术爱好者深入讨论过相关问题。他也是轻量级Java开发框架的拥护者，曾参与JavaEye论坛组织的《J2EE Development without EJB》一书的翻译。在2007年Ruby on Rails 1.2版支持REST开发之后，他对搞清楚REST的来龙去脉产生了浓厚兴趣，同年组织翻译了Roy T. Fielding的博士论文《Architectural Styles and the Design of Network-based Software Architectures》（中文版名为《架构风格与基于网络的软件架构设计》）。在2011年又翻译了《REST实战》一书，目前仍然

致力于在国内普及对于REST架构风格的深入理解，以及相关实战经验的交流。目前主要感兴趣的技术领域包括DDD、DSL、SOA、REST。

**李建业：**2002年毕业，之后一直从事软件开发，涉及办公自动化、电信网管/增值业务系统以及互联网；2009年12月加入淘宝的广告应用开发团队；从2011年底开始，关注软件研发本身，涉及运维和测试相关工作，目前关注持续集成。李建业还经常受邀出席一些技术大会并发表主题演讲，比如InfoQ，Ruby大会等。

**丁雪丰：**一线攻城师一枚，InfoQ中文站小编，常年混迹于各种社区，业余时间写作翻译、汉化软件，《RESTfulWebServices Cookbook中文版》、《Spring攻略》等多部图书的译者。

**马钧：**全端程序员，15年以上多种程序编程经验，爱好开源软件，积极参加各类社区活动，《REST实战》等技术图书的译者。

**原文链接：http://www.infoq.com/cn/articles/how-to-design-a-good-restful-api**

**相关内容**

- 设计并实现超媒体API
- TeamToy2的开发故事：面向移动的API设计，代码重用，快速迭代
- IQueryable对于API是个错误的选择吗？
- HTTP API可演进性最佳实践
- 计划FM为人人网提供首个开源Ruby SDK

# 本期专题：再谈REST | Topic

## REST之中的幂等指的是什么？

作者 Mark Little ，译者 张卫滨

REST（或者至少是HTTP）的一个重要方面就是一些操作（动作）是幂等的。正如 Gregor Roth在多年前所说的那样：

> PUT方法是幂等的。幂等的方法意味着请求成功执行所得到的的结果不依赖于该方法被执行的次数。

幂等也是在SOA设计模式中所讨论的一个问题。多年以来，围绕着REST以及相对于其他方式它所能带来的收益已经有了很多的讨论，在这个过程中，幂等只是一个方面，通常我们认为已经对其有所理解了，实际上并没有进行充分地研究。但是，最近在面向服务架构（Service Oriented Architecture）邮件列表中有一个帖子，针对这个看起来很直观的理念，激发了很多的讨论。初始的帖子是这样的：

> 目前我正在审视我们服务的幂等性并提出一个应用于企业的规定性指导，这个指导将会用于创建幂等服务的场景之中。问题在于，所有的服务都需要是幂等的吗？这现实吗？

它引用了另外的一篇blog，在这篇文章中，作者这样说：

> 当然，对于服务的提供者来说，实现幂等会比较复杂。而复杂性就会花费金钱。这可能就是幂等的服务如此稀少的原因。[...] 但是，服务只要成为幂等的，那么它就会非常简单，并且能够保证跨集成边界的数据一致性。服务*应该*是幂等的。这应该是架构和服务设计的原则。尤其是对于那些被很多消费者使用的服务更应如此。你不能将保持数据一致的复杂性推给你的消费者，对吧？消费者就是你的客户，你应该这样善待他们。如果你能为他们带来便利的话，你会有更多的消费者。

初始的这个帖子引起了很多的响应，包括来自Cap Gemini的Steve Jones，他这样说到：

> 幂等是IT行业中巨大的垃圾词汇之一，它的意思是“内存中的幂等”（idempotent in memory）。如果你有一个评论的服务，它具备更新的功能，从服务调用者的视角来看，服务必须要维护和管理状态。那你的含义就是它能够在内存中横向扩展。如果你更新状态的话，包括在数据库中，那么就不能保证是幂等的。幂等指的是当你使用相同的值调用相同的功能时，结果完全一致，这是它的数学定义。如果你改变过状态，那么就不是幂等的，而“数据库更新”并不会改变任何事情。

Mark Baker曾经在REST这个话题上曾经有过很多回复，回应了Steve：

> 就像你的评论样例所述，确实存在非幂等的状态更新，将一些数据更新为特定的值也会是幂等的更新...因为很显然如果用相同的输入执行多次的话，结果会是相同的。

Steve在这一点上与Mark的观点是一致的：如果具有相同输入的一个操作没有改变任何的事情，那么它是幂等的。但是，如果这个操作改变了任意的状态，比如说它记录了这个请求的最后时间，那么它就不是幂等的。

> 在IT行业中，这个术语长久以来被人们滥用了，这一点我“感触颇深”，我曾经多次见过这样的场景，人们宣称一些事情是“幂等的”只是因为它在内存中是无状态的，尽管其消费者要达到的效果是记录性的事务。

Mark同意Steve的严格定义，但是他相信这不能应用于初始请求的上下文之中，也就是：分布式系统中的幂等性：

> [...] 这个词汇只对于接口有意义而不是实现。所以，如果我定义一个操作，比如说“set”，并将其定义为幂等的，那么它会只关系到客户端，即便在这个接口的实现中会产生日志。

另外一个参与者Ashaf Galal，断言所有“读的服务”（reading services）都是幂等的，因为这样的服务只会返回数据。但是，Steve指出，这是一个过于简单的视角，有时候一个操作呈现出来的是幂等的，它的实现可以做任何事情但它需要是幂等的就可以（这再次提出了这个问题，那就是这种区分是否重要）：

> [...]读的功能不一定是幂等的。“getNextIterator()”是一个读的功能，它不是幂等的，因为它会渐进迭代器。银行账号请求余额可能也不是幂等的，因为会创建审计日志。两个连续调用返回的结果可能是相同的（如果没有变化发生的话），但是日志条目是不同的。

Ashaf回应了Steve，他说幂等是服务的属性而不是接口的属性。

> 客户端需要从服务那里得到适当的响应，他并不关心服务是不是幂等的，是不是要记录日志。

那么，其他人怎么想的呢？对于幂等的含义，是否还有误解？如果一个被视为幂等的操作实际上会改变一些状态，如Steve所提到的更新日志，那么这有关系吗？

**原文链接：http://www.infoq.com/cn/news/2013/05/idempotent**

**相关内容**

- REST之中的幂等指的是什么？
- REST会是SOA的未来吗？
- RESTful Web Services Cookbook中文版
- Remote Object、SOA和REST
- RESTful Web服务框架Jersey 2.0发布，实现了JAX-RS 2.0规范

# 本期专题：再谈REST | Topic

## Web风格起过作用吗？

作者 Mark Little ，译者 方盛

在最近的一篇博客文章中，Jean-Jacques Dubray（JJ）提醒我们，自从Tim Bray预测SOA的死期，已经过去了差不多7年：

> 我曾经做过一个采访和一次播客[...]但在这两方面都浮现出了同一个问题，连走廊谈话也不例外：“你觉得我们该拿SOA怎么办？”，奇怪的是，以前从没有人问起过我这个问题，我能找到的唯一答案就是：“什么也别做。‘SOA’在以前可能还意味着什么，但是现在，它只是供应商们的胡扯罢了。”

Tim最后声称（预测）SOA不是未来的趋势，Web风格才是。如JJ所提到的，该预测开启了其他人在接下来的几年里不断追随效仿的大门，这其中就包括了Anne-Thomas Manes等人。它导致的其中一个结果就是许多SOA的项目都被搁置甚至取消了，JJ也有自己的一些前车之鉴：

> 我的经理在那时告诉我，Tim Bray的观点当时流传于整个IT部门，而且他也不知道怎么为他的管理工作找到一条出路。他的团队在当时已经构建了属于他们自己的ESB，而在那个时候还只有很少人听过XML，就因为Tim Bray写的一段话，就让我们多年的辛勤工作和不断增加的事务量（在2007年初的时候就有每天10M多的请求）毁于一旦。

JJ这几年在InfoQ和其他地方都花费了相当多的精力来探讨Web风格所忽视的问题。在他最近的文章里，他依然在关注那些打算实现Web风格的服务：

> 据我对[可编程的Web目录里的]9000多个API最好的估计，只有不到1%的API遵循了Tim Bray的Web风格。他们大部分只是遵循了“API”的风格，缺乏对RPC的支持。

JJ针对他上面所提到的内容列举了一些他认为最有代表性的示例：

> - **AskZiggy** 提供了定义“actions”的能力（sic）（例如Play，NextSong，Previous Song，Shuffle...）
> - **WhatLanguage**说明了你可以任意使用GET（如果你的请求少于7500个字符）或者POST请求来向同一个URI发送请求从而来判断一个给定的字符串使用的是哪种语言。
> - **DO.com** 可能看起来是在提供一个Web风格的API，但是它并没有真的做多少工作，它只是简单地对五种资源（tasks，project，users，...）进行了CRUD的操作。
> - **SkyBuffer**也同样遵循了Web风格，但是就像DO.com一样，也只是实现了对一些实体对象进行CRUD的操作。
> - 被称作“云API中心”的**MaShape**非常有意思，因为他们为开发者提供了一种更好的方式来使用API。那么他们是怎么做到的呢？他们邀请开发者“学习如何在MaShape上来描述你的API从而自动生成客户端类库和文档”。是的，你没有听错，不过经过多年的抨击以后，开发者们已经开始在讨论客户端类库代码自动生成的话题了。

JJ认为这种API的方式跟Tim等人所推崇的纯粹的Web风格是相违背的：

> Web风格不就是讨论的“统一接口”、书签和自动配置的HATEAOS吗？不要忘却标准的IANA类型？是的，这些日子你是没有听过这方面的相关争论了。API规则。人们已经不再耻于在他们的URL和POST（复杂的）请求里使用动词了。最重要的是，MongoDB告诉我们跟这四个简单的动词和贫乏的URL语法比起来，还有更多的地方需要使用CRUD。开发者和架构师更渴望使用“Web风格”以至于他们甚至都尝试在JSON里添加命名空间了。

当看着这些似是而非的Web风格服务时，JJ得出了一个结论，它实际上已经无法兑现当时的炒作，实际上它已经“死”了。不管怎么样，JJ更加深入地进行了思考，并声明Web本身已经死了：

> [...]（对那些不知如何使用HTML5来创造任何有价值的东西来与Native应用竞争的）开发者，和（那些对Web业务模型核心“产品”的绝妙想法逐渐冷淡的）最终用户来说，整个Web几乎已经死了。TinBerner-Lee每六个月就现身一次来宣传“Web万岁”的理论，但是当安全滥用引爆众怒以后，貌似KBE（译者注：一种骑士爵位）也无法拯救Web了。

幸运的是这篇文章并不只是留给我们一个对于过去的悲观概述和高昂“技术债务”。JJ纵观今日以及诸如移动通讯等新浪潮的影响力，他坚信这可能代表了有史以来最大的范式转换（paradigm shift ）计算：

可能很少有人会记得，软件工程就是构建在一个非常非常古老的范式——“文件处理”之上的，而它在UFS上达到了极致。桌面隐喻和个人电脑的主要使用模式依然停留在“文件处理”上。移动设备却不再是单纯的文件了，移动终端确实给我们生活中的各个领域都带来了便利，如果不出意外的话，那么未来的操作系统将是活动中心。

然而，他相信为了成功我们也必将超越Web技术：

> 最好的用户体验将会胜出，任何曾经、正在或者试图与之反抗的人终将会失败。如果Web崛起了，也是因为曾经它提供了更好的用户体验。如果它没有崛起，那是因为它仍然还是“Web”。

最后，JJ声明我们应该在解决问题的方式上更加务实，并且真正从过去的经验中吸取教训：

> 就算是使用像MongoDB那样设计良好的API，更多的增删改查也并不能把工作做得更好。我们还是需要形成这样一种理解，OO并不是描述分布式组件之间相互交互的正确范式。因此我们必须停止将我们所做的任何事情都实例化成无状态的单例方法调用。类上的注解不够强大以至于无法驱动由SOA开启的语义革命，而我们现在需要终止它。

但是即使认可JJ所提到的所有变化，那什么才是终极的目标呢？JJ认为它是一个健壮的复合编程模型（Composite Programming Model）。在该编程模型中，模型和视图是相互独立的，但依然是适当地相互关联的，而且还遵循activity/action/lifecycle的范式。不幸的是，JJ并没有在这篇文章中针对该模型做过多的细节探讨，但是他的意图很可能在一些相关的出版物中也会有所体现。

**原文链接：http://www.infoq.com/cn/news/2013/06/web-style**

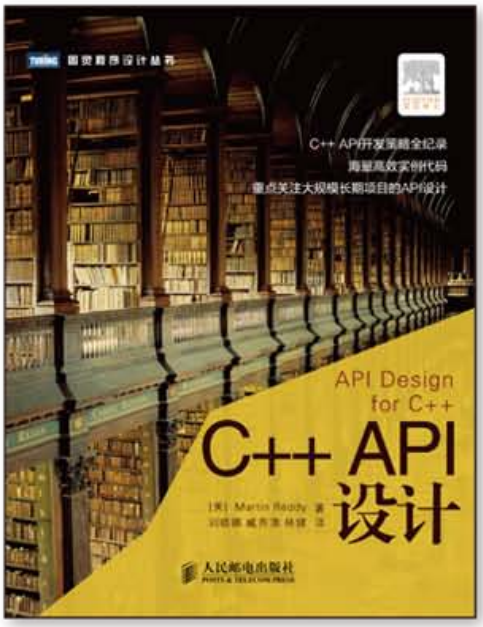
**相关内容**

- StackMob的乐趣
- 设计并实现超媒体API
- WSO2 API Manager 1.4.0 引入多租户

TURING

微信账号：
图灵教育

# 图灵教育推荐




- 软件 API 设计经典著作
- Pixar 动画系统首席工程师悉心讲解
- 软件工程效率之源

本书通过大量 C++ 代码阐述每一个概念，介绍了世界级 API 的开发策略。作者有 15 年从业经验，探讨了接口设计、文档、测试，以及脚本处理和插件可扩展性，并通书聚焦生成优雅、持久代码库的各种 API 样式和模式，讲述了 C++ API 的开发策略。

书名：C++ API 设计
作者：Martin Reddy
译者：刘晓娜 臧秀涛 林健
书号：978-7-115-32299-9
定价：89.00 元

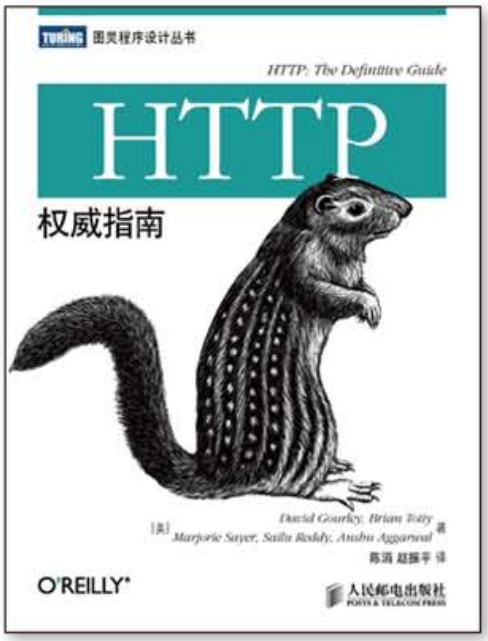
详情请点击：http://www.ituring.com.cn/book/923

本书由具有多年实践经验的专家编写，通过简洁、精确的语言和大量翔实的细节图解帮助读者形象地理解 Web 幕后所发生的事情，详细说明了 Web 上每条请求的实际运行情况。要想高效地进行 Web 开发，所有 Web 程序员、管理员和应用程序开发者都应该熟悉 HTTP。很多书籍只介绍了 Web 的使用方式，而本书则深入说明了 Web 的工作原理。

本书是 HTTP 及其相关核心 Web 技术方面的权威著作，主要介绍了 Web 应用程序是如何工作的，核心的因特网协议如何与架构构建块交互，如何正确实现因特网客户和服务器等。

本书适合所有想了解 HTTP 和 Web 底层结构的人阅读。

详情请点击：http://www.ituring.com.cn/book/844




书名：HTTP 权威指南
作者：David Gourley Brian Totty
Marjorie Sayer Sailu Reddy
Anshu Aggarwal
译者：陈涓 赵振平
书号：978-7-115-28148-7
定价：109.00 元

TURING 北京图灵文化发展有限公司

联系我们：
官方微博：@图灵教育
联 系 人：@图灵郭志敏
邮　　箱：guozm@turingbook.com
Q　　Q：52398720

# 推荐文章 | Article

## 所有的团队都应该使用自动构建吗？

作者 Jonathan Allen ，译者 邵思华

我们与Team Foundation Server的高级项目经理Chris Patterson讨论了在现代化应用程序开发中，自动构建与持续交付所扮演的角色。Team Foundation Service是由Microsoft托管的Team Foundation Server。

**InfoQ：你是否觉得所有的团队都应该使用自动构建？如果答案为是，为什么？如果为否，那么在何种环境下能体现出它的重要性呢？**

> **Chris：**简短的回答是：是的，所有团队都应该使用。首先，我想区分一下代码编译与构建。我认为一个构建应该起码包含运行你的单元测试，否则的话就仅仅是代码编译而已，而这仅针对那些无法通过编译的项目才能起到作用。随着HTML5和JavaScript越来越多地被项目所采用，甚至使用在原生的Windows应用中，单纯的编译步骤已无法给你初级的反馈了。因此，单元测试现在成为了你和你所引入的bug之间的第一道防线。我甚至建议只有一个人的团队也要搭建自动构建系统。

> 记着这一点，你会发现使用自动构建的诸多好处。
>
> 首先它为流程的规范化带来好处。自动构建促使开发者或开发团队在创建软件的同时建立并维护一个可重复的、文档化的流程。多数开发者都在不同场合被“在我的机器上能够运行”这种bug所困扰。典型的原因是因为开发者工作机上的某些依赖项没有安装在测试或者客户环境上。持续的、可重复的自动构建流程能促使团队保证他们的软件在IDE和开发者工作机之外也能够编译并运行基本的单元测试集，这有效地避免了上述bug的发生。
>
> 除此之外，自动构建也有助于发现及突出一些其它常见问题：
>
> - 某个开发者的签入破坏了构建
> - 某个开发者忘了签入某个重要文件或者配置变更
> - 你的构建脚本不能运行，或者你的软件不能通过单元测试了

关于软件工程流程中自动构建的重要性的好文章有很多，Martin Fowler就有一个极好的例子。

**InfoQ：你是否认为要想成功应用自动构建，必需了解如何编写MSBuild脚本？**

**Chris：**MSBuild是一门创建自动构建脚本的技术，也在Visual Studio中使用。从这个角度来看，如果你使用Visual Studio进行工作，你应该熟悉MSBuild并了解如何编写脚本。

.NET社区中还流行着其它一些构建脚本工具，例如NAnt和PSake等等。我个人对它们虽然并不十分精通，不过我的确知道大家对它们热情十足。

我们在Team Build中组合使用了MSBuild和Windows Workflow。MSBuild用来处理Visual Studio解决方案和项目文件，我们用它进行代码自动编译。从TFS中获取你的代码，执行MSBuild，运行单元测试并将结果写回TFS这一整套流程则是由Windows Workflow Foundation完成的。这种组合建立了一个非常强大而灵活的系统，开发者可以利用它实现完全的自动化，包括构建、测试环境管理甚至是生产环境的部署。

不过，如果你使用其它技术，那么你就需要了解如何运行它的构建系统。

**InfoQ：你觉得TFS在怎样的自动化部署（例如持续交付）场景中表现得特别优秀或者糟糕？**

**Chris：**Team Build的一大优势是你可以做任何想做的事，但这也可能是它最大的弱点。Windows Workflow运行时功能非常强大并且适用。不过这对开发者而言意味着一定的学习曲线。对这个平台有所钻研并真正了解它的人能完成连我们都想象不到能为客户实现的功能，而对于不了解WF的开发者来说，即使是简单的构建定制也会是一个挑战。这意味着，Team Build对于任一种自动化部署机制都是一个好的选择，但对开发者来说未必能马上了解如何使用它实现目标。

为了改善开发者在持续交付方面的体验，我们和Azure团队合作，提供了从Team Foundation Service到Windows Azure之间一个无缝的持续交付工作流。只需数次点击，开发者就能够将他们的Team Foundation Service帐号与他们的Azure订阅连接起来。这个流程同时还创建了一个预配置的构建定义，能够将web网站或云服务部署到该订阅中。开发者所要做的只是在他们的解决方案中指明该构建定义，其余的部分都将由我们完成。

今后，我们期待将更多的此类方案提供给Team Foundation Service的客户，以及在内部的防火墙中搭建TFS的客户。

**InfoQ：有人主张持续交付不仅仅针对QA，也可以应用到生产环境本身。这意味着，一旦自动化测试通过，它就会在没有任何人工干预的情况下自动部署到生产环境。你对这个观点的看法是什么？**

**Chris：**我非常提倡为QA实现持续交付，我认为能够将可运行的产品及变更交付给你的测试人员，是提高你的团队速率及软件质量最好的方法之一。再有，如果你都不能为QA实现持续交付，那更不可能应用到生产环境上。

为生成环境实现持续交付在某些地方正在流行起来，我相信对某些类型的业务和产品来说这是个长远之计，但对其它业务和产品就未必适用。它对开发者的自律性有着极高的要求，并且对你的自动构建要有足够的自信，确保它能够捕获任何类型的错误，而不让问题发布到客户那里。

**InfoQ：对比那些独立的构建服务器，例如CruiseControl，你觉得使用TFS构建agent的优势是什么？**

**Chris：**主要的优点在于其集成度。Team Build是TFS的一部分，在安装包和service中都有提供，这意味着你拥有了整套设计的方案。如果你遇到任何问题，我们也将为你提供世界级的支持。Team Build包含了一套预配置的模板，支持Visual Studio解决方案和Visual Studio test runner，这使开发者能在几分钟内轻松上手并运行构建。对于Team Foundation Service来说，开发者甚至不需要安装构建agent。我们为service提供了构建agent，能够构建绝大多数你能在Visual Studio中创建的项目。

当然，独立产品的一个优势就是来自社区方面的众多支持。CruiseControl或Jenkins等工具有着丰富的插件，以提供不同的功能。另外，它们的agent都能够运行在Linux和MacOS平台上。我们有许多客户同时使用Team Build和独立的构建服务器。所有Windows平台上的产品使用Team Build，而运行在其它平台上的产品使用独立的构建服务器。

**InfoQ：是否有计划推出Java、iOS及Android平台上的构建agent？**

**Chris：**在这个新平台不断出现的软件开发世界，我们将持续改善在其它平台上的功能。

我们提供了一系列的工作流activity以及MSBuild任务，以允许你使用在Java社区中非常流行的Ant和Maven实现自动构建。实际上，我们在内部构建Team Explorer Everywhere产品时也使用了这些用Java编写的activity。由于Android开发环境同样基于Eclipse和Java，我们也可以使用相同的技术。

目前，iOS应用还只能在Mac上开发，而我们还没有能够运行在Mac上的agent。

# 关于作者

Chris Patterson是参与Team Foundation Server开发的高级项目经理，他在软件行业已工作了15年，有着多种技术的工作经验，包括经典的ASP、VB、Java及.NET。在2005年加入Microsoft之前，他就职于TogetherSoft及Borland，通过指导和培训帮助客户改进他们的软件开发流程。在Microsoft，他负责Visual Studio 2008及2010中负载、web及手动测试的特性，目前他负责Team Foundation Server和Team Foundation Service中的构建自动化组件的工作。他乐于帮助客户解决问题，阅读和学习技术和陪伴家人。你可以在Twitter中通过@chrisrpatterson，或者在产品论坛中找到他。

**原文链接：http://www.infoq.com/cn/articles/Chris-Patterson**

**相关内容**

- 微软发布Team Explorer Everywhere 2012 Update 3，支持Eclipse及FreeBSD
- MobileCloud for TFS支持测试Windows Phone，Android，iOS及BlackBerry应用
- Visual Studio 2013浮出水面
- 物理墙和虚拟墙之争
- Visual Studio 2012 Update 2最终预览版发布

# 推荐文章 | Article

# 从OpenStack的角度看块存储的世界

作者 王豪迈

块存储，简单来说就是提供了块设备存储的接口。用户需要把块存储卷附加到虚拟机或者裸机上以与其交互。这些卷都是持久的：它们可以从运行实例上被解除或者重新附加而数据保持完整不变。

本文重点介绍块存储服务。我们对目前主流的块存储服务提供商和开源的块存储软件做了一个简要分析，希望能给从事块存储开发的工程师对于块存储一个全局的认识。

下面会先介绍常见的单机块设备工具，以建立对块存储的初步印象。

## 单机块存储

首先，一个硬盘是一个块设备。内核检测到硬盘后，在/dev/下会看到/dev/sda/。为了用一个硬盘来得到不同的分区来做不同的事，我们使用fdisk工具得到/dev/sda1、/dev/sda2等。这种方式通过直接写入分区表来规定和切分硬盘，是最死板的分区方式。

## 1. LVM & Device-mapper

LVM是一种逻辑卷管理器。通过LVM来对硬盘创建逻辑卷组和得到逻辑卷，要比fdisk方式更加弹性。如果你目前对LVM用途还不熟悉或者不大清楚，可参考以下链接：

- 通用线程：学习 Linux LVM，第 1 部分
- 通用线程：学习 Linux LVM，第 2 部分

LVM基于Device-mapper用户程序实现。Device-mapper是一种支持逻辑卷管理的通用设备映射机制，为存储资源管理的块设备驱动提供了一个高度模块化的内核架构。以下链接对Device-mapper架构进行了极好的说明：

- Linux 内核中的 Device Mapper 机制

# 2. SAN & iSCSI

在接触了单机下的逻辑卷管理后，你需要了解SAN，目前主流的企业级存储方式。

大部分SAN使用SCSI协议在服务器和存储设备之间传输和沟通，通过在SCSI之上建立不同镜像层，可以实现存储网络的连接。常见的有iSCSI，FCP，Fibre Channel over Ethernet等。

SAN通常需要在专用存储设备中建立，而iSCSI是基于TCP/IP的SCSI映射，通过iSCSI协议和Linux iSCSI项目，我们可以在常见的PC机上建立SAN存储。

对于如何建立在PC机上的SAN，可以参考：

- iSCSI建立

不过，这篇文章提到的iSCSI target管理方式不太方便，通常我们会用targetcli管理target。targetcli可以直接建立和管理不同backstone类型的逻辑卷和不同的export方式，如建立ramdisk并且通过iSCSI export，非常方便。操作方式见targetcli screencast Part 2 of 3: ISCSI – YouTube。

以上都是我们经常接触的单机块存储。接下来是本文主要分享的内容：公共云技术服务提供的块存储服务，开源的块存储框架，以及OpenStack目前对块存储的定义和支持情况。

# 分布式块存储

在面对极具弹性的存储需求和性能要求下，单机或者独立的SAN越来越不能满足企业的需要。如同数据库系统一样，块存储在scale up的瓶颈下也面临着scale out的需要。我们可以用以下几个特点来描述分布式块存储系统的概念：

1. 分布式块存储可以为任何物理机或者虚拟机提供持久化的块存储设备
2. 分布式块存储系统管理块设备的创建、删除和attach/detach
3. 分布式块存储支持强大的快照功能，快照可以用来恢复或者创建新的块设备
4. 分布式存储系统能够提供不同IO性能要求的块设备

# 3.1 Amazon EBS

Amazon作为领先的IaaS服务商，其API目前是IaaS的事实标准。Amazon EC2目前在大多数方面远超其他IaaS服务商。Amazon EC2的产品介绍是快速了解Amazon EC2的捷径。

EBS是Amazon提供的块存储服务。通过EBS，用户可以随时增删迁移volume和快照操作。

Amazon EC2实例可以将根设备数据存储在Amazon EBS或者本地实例存储上。使用Amazon EBS时，根设备中的数据将独立于实例的生命周期保留下来，使得在停止实例后仍可以重新启动使用，与笔记本电脑关机并在再次需要时重新启动相似。另一方面，本地实例存储仅在实例的生命周期内保留。这是启动实例的一种经济方式，因为数据没有存储到根设备中。

Amazon EBS提供两种类型的卷，即标准卷和预配置IOPS卷。它们的性能特点和价格不同，可以根据应用程序的要求和预算定制所需的存储性能。

标准卷可为要求有适度或突发式I/O的应用程序提供存储。这些卷平均可以提供大约100 IOPS，最多可突增至数百IOPS。标准卷也非常适合用作引导卷，其突发能力可提供快速的实例启动时间（通常十几秒）。

预配置IOPS卷旨在为数据库等I/O密集型随机读写工作负载提供可预计的高性能。创建一个卷时，利用预置IOPS为卷确定IOPS速率，随之Amazon EBS在该卷的生命周期内提供该速率。Amazon EBS目前支持每预配置IOPS卷最多4000个IOPS。您可以将多个条带式卷组合在一起，为应用程序提供每个Amazon EC2数千IOPS的能力。

EBS可以在卷连接和使用期间实时拍摄快照。不过，快照只能捕获已写入Amazon EBS 卷的数据，不包含应用程序或操作系统已在本地缓存的数据。如果需要确保能为实例连接的卷获得一致的快照，需要先彻底地断开卷连接，再发出快照命令，然后重新连接卷。

EBS快照目前可以跨regions增量备份，意味着EBS快照时间会大大缩短，从另一面增加了EBS使用的安全性。

总的来说，Amazon EBS是目前IaaS服务商最引入注目的服务之一，目前的OpenStack、CloudStack等其他开源框架都无法提供Amazon EBS的弹性和强大的服务。了解和使用Amazon EBS是学习IaaS块存储的最好手段。

## 3.2 阿里云存储

阿里云是国内的公共云计算服务商。阿里云磁盘目前仅支持在创建云主机的时候绑定云磁盘或者在升级云主机的进行云磁盘扩容，这从根本上就是传统的虚

拟主机的特点而不是所谓的“云磁盘”。

从目前的阿里云磁盘的限制：

1. 无法快速创建或删除volume，在进行扩容时需要升级云主机才能达到，而升级云主机只有在下月云主机套餐到期时才能生效（中国移动套餐的模式）
2. 一个云主机最多只能绑定3个云磁盘

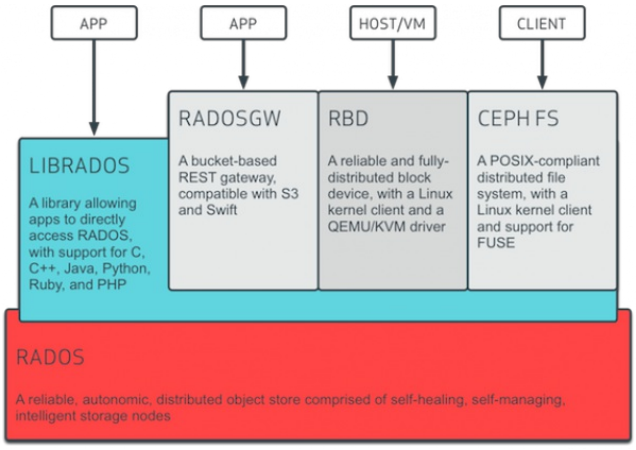
从阿里云磁盘目前的使用分析，以下是我对阿里云磁盘实现的推测：

1. 阿里云主机是跟磁盘绑定的，这意味着阿里云的云磁盘是local volume（因此性能还是挺可观的）。用户需要扩容、减少都需要下个月更说明了这点，整个主机在扩容时去调度合适的有足够存储空间的host，然后进行扩容。
2. 阿里云磁盘是分布式块存储系统，但是由于其QoS无法保证和其他资源调度原因，无法提供足够的块存储支持。

从演讲回顾：阿里云存储技术的演进，以及云服务用例最佳实践中了解到阿里云是基于自家的“盘古”系统，那么从实际使用来说，远没达到一般的分布式块存储系统的要求。

## 4.1 Ceph

Ceph是开源实现的PB级分布式文件系统，其分布式对象存储机制为上层提供了文件接口、块存储接口和对象存储接口。Inktank是Ceph的主要支持商，也是目前Ceph开源社区的主要力量。




Ceph目前是OpenStack的开源块存储实现系统（即Cinder项目的backend driver之一），其实现分为三个部分: OSD，Monitor，MDS。OSD是底层对象存储系统，Monitor是集群管理系统，MDS是用来支持POSIX文件接口的

Metadata Server。从Ceph的原始论文《Ceph: Reliable, Scalable, and High-Performance Distributed Storage》来看，Ceph专注于扩展性，高可用性和容错性。Ceph放弃了传统的Metadata查表方式（HDFS）而改用算法（CRUSH）去定位具体的block。

利用Ceph提供的RULES可以弹性地制订存储策略和Pool选择，Monitor作为集群管理系统掌握了全部的Cluster Map，Client在没有Map的情况下需要先向Monitor请求得到，然后通过Object id计算相应的OSD Server。

Ceph的文档可以参考以下链接：

- Ceph：一个 Linux PB 级分布式文件系统
- 分布式文件系统Ceph调研1 – RADOS
- Ceph Architecture
- Ceph的现状
- ceph的CRUSH数据分布算法介绍
- Ceph INTERNAL DEVELOPER DOCUMENTATION

Ceph支持传统的POSIX文件接口，因此需要额外的MDS（Meatadata Server）支持文件元信息（Ceph的块存储和对象存储支持不需要MDS服务）。Ceph将Data和Metadata分离到两个服务上，跟传统的分布式系统如Lustre相比可以大大增强扩展性。在小文件读写上，Ceph读写文件会有[RTT*2]，在每次open时，会先去Metadata Server查询一次，然后再去Object Server。除了Open操作外，Ceph在Delete上也有问题，它需要到Metadata Server擦除对应的Metadata，是n(2)复杂度。Ceph在Metadata上并非只有坏处，通过Metadata Server，像目录列表等目录操作为非常快速，远超GlusterFS等其他分布式文件系统的目录或文件元操作。

如果要用Ceph作为块存储项目，有几个问题需要考虑：

1. Ceph在读写上不太稳定（有btrfs的原因）。目前Ceph官方推荐XFS作为底层文件系统
2. Ceph的扩展性较难，如果需要介入Ceph，需要较长时间
3. Ceph的部署不够简易并且集群不够稳定

## 4.2 Sheepdog

Sheepdog是另一个分布式块存储系统实现。与Ceph相比，它的最大优势就是代码短小好维护，hack的成本很小。Sheepdog也有很多Ceph不支持的特性，比如说Multi-Disk, Cluster-wide Snapshot等。

Sheepdog主要有两部分，一个是集群管理，另一个是存储服务。集群管理目前使用Corosync或者Zookper来完成，其存储服务的特点是在client和存储host有Cache的实现可以大大减小数据流量。

目前Sheepdog只在QEMU端提供Drive，而缺少library支持，这是Sheepdog目前最主要的问题。但是社区已经有相关的Blueprint在讨论这个问题。

了解Sheepdog的一些相关资料：

- Sheepdog Overview
- Sheepdog 淘宝核心系统团队
- Sheepdog wiki：Sheepdog的一系列Wiki如同它的代码一样简短出色

目前Taobao是Sheepdog主要用户和社区贡献者。

# 5. Cinder

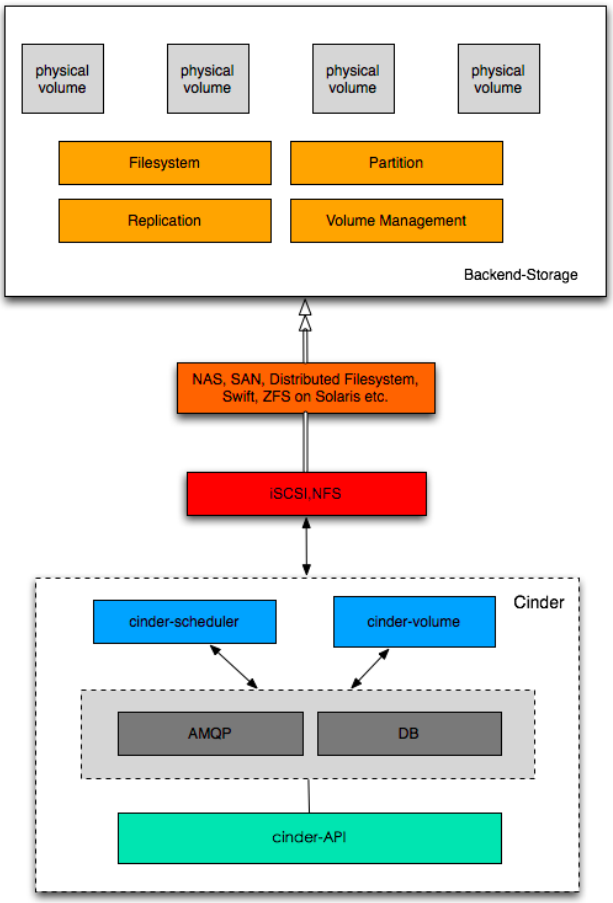
OpenStack是目前流行的IaaS框架，提供了与AWS类似的服务并且兼容其API。OpenStack Nova是计算服务，Swift是对象存储服务，Quantum是网络服务，Glance是镜像服务，Cinder是块存储服务，Keystone是身份认证服务，Horizon是Dashboard，另外还有Heat、Oslo、Ceilometer、Ironic等等项目。

OpenStack的存储主要分为三大类：

- 对象存储服务，Swift
- 块设备存储服务，主要是提供给虚拟机作为“硬盘”的存储。这里又分为两块：
  - 本地块存储
  - 分布式块存储，Cinder
- 数据库服务，目前是一个正在孵化的项目Trove，前身是Rackspace开源出来的RedDwarf，对应AWS里面的RDC。

Cinder是OpenStack中提供类似于EBS块存储服务的API框架。它并没有实现对块设备的管理和实际服务，而是为后端不同的存储结构提供了统一的接口，不同的块设备服务厂商在Cinder中实现其驱动支持以与OpenStack进行整合。

后端的存储可以是DAS，NAS，SAN，对象存储或者分布式文件系统。也就是说，Cinder的块存储数据完整性、可用性保障是由后端存储提供的。在CinderSupportMatrix中可以看到众多存储厂商如NetAPP、IBM、SolidFire、EMC和众多开源块存储系统对Cinder的支持。




从上图我们也可以看到，Cinder只是提供了一层抽象，然后通过其后段支持的driver实现来发出命令来得到回应。关于块存储的分配信息以及选项配置等会被保存到OpenStack统一的DB中。

# 总结

目前分布式块存储的实现仍然是由Amazon EBS领衔，其卓越稳定的读写性能、强大的增量快照和跨区域块设备迁移，以及令人惊叹的QoS控制都是目前开源或者其他商业实现无法比拟的。

不过Amazon EBS始终不是公司私有存储的一部分，作为企业IT成本的重要部分，块存储正在发生改变。EMC在一个月前发布了其ViPR平台，并开放了其接口，试图接纳其他厂商和开源实现。Nexenta在颠覆传统的的存储专有硬件，在其上软件实现原来只有专有SDN的能力，让企业客户完全摆脱存储与厂商的绑定。Inktank极力融合OpenStack并推动Ceph在OpenStack社区的影响力。这些都说明了，无论是目前的存储厂商还是开源社区都在极力推动整个分布式

块存储的发展，存储专有设备的局限性正在进一步弱化了原有企业的存储架构。

在分布式块存储和OpenStack之间，我们可以打造更巩固的纽带，将块存储在企业私有云平台上做更好的集成和运维。

## 作者简介

王豪迈，UnitedStack工程师。本文根据作者的博客文章《块存储的世界（入门级）》修改而成。

**原文链接：http://www.infoq.com/cn/articles/block-storage-overview**

### 相关内容

- 云计算时代的运维与安全
- 演讲回顾：阿里云存储技术的演进，以及云服务用例最佳实践
- 腾讯云Party：分享稳定云和质量云
- 腾讯陈磊：大数据是解决开放平台效率的重要手段
- 林仕鼎谈数据中心计算（三）：学术界与工业界，以及未来展望

# 推荐文章 | Article

# 对话微软开源技术子公司资深总监Gianugo：关键在于互操作性

作者 杨赛

Gianugo Rabellino差点成为一个律师。在大学实习的时候，Gianugo对法律工作的理解被颠覆，转而投向了程序员的行列，并成为了开源社区的活跃份子。在开源界活跃的20年间，他做过技术支持，做过CTO，成立过自己的公司，并在Apache社区中的数个项目中被推选为committer和副主席。2010年，他加入了微软。

现在，Gianugo是微软开源技术公司（Microsoft Open Technologies, Inc, 微软全资子公司）的资深总监。MS Open Tech成立于2012年4月，其前身是微软的互操作性部门。根据公司官网的说明，MS Open Tech的目的主要是“在微软与非微软技术之间搭起桥梁，通过开放标准和开放源码提升互操作性”。MS Open Tech在HTML5、Hadoop、Eclipse等开源技术与Windows、Azure、Visual Studio等微软产品之间构建项目，并提供了VM Depot服务，为公共领域提供VM镜像共享的平台。

一个开源界的资深人士为何会加入微软？MS Open Tech又是一个什么样的组织，它的目的和职责都有哪些？InfoQ中文站近日跟Gianugo进行了一次交流，了解到一些以上问题的答案。

**InfoQ：Gianugo，先跟大家做一下自我介绍吧。**

> **Gianugo：**我是来自MS Open Tech的Gianugo。MS Open Tech是一个独立的子公司。我在2010年10月加入了微软，到现在已经快三年了。一开始，我加入的团队是互操作性团队，这个团队后来分离出来，成为了现在的MS Open Tech子公司。Open Tech的成立是为了进一步推动开放——在我们看来，这包括开放源码，开放标准与互操作性。
>
> 互操作性团队到现在已经有7年的历史。虽然名字变了，不过做的事情是差不多的。成立独立的公司，意味着我们更加敏捷，更加轻便，能够更好的适应快速响应的挑战——在开源领域，快速响应就是一切。

加入微软之前，我创办了Sourcesense Europe BV公司——欧洲最大的开源服务供应商之一。我同时为Apache工作——这是一个基于志愿者模式的组织。顺便说一句，Apache基金会的主席最近也加入了MS Open Tech。我呢，是个副主席。所以这也挺有意思的：这相当于他在MS Open Tech里给我报告，而我在Apache里向他报告。

当然了，Apache是个志愿者模式的社区，也就是说，你在里面想干啥都行，并没有公司一样的等级划分，也没有老板规定你要做这做那的。我们有各自的职责——Apache基金会的主席承担发言人的职责，同时要做一些行政执行方面的工作，比如签发支票之类的。每一个项目都报告给委员会，这样只是为了确保委员会了解项目的进程。

委员会的任务在于确保Apache社区的健康。社区是他们唯一关心的对象。好的代码来自好的社区。Apache的座右铭就是：社区大于代码。这很有意思，因为它是个事实。

**InfoQ：这挺有意思。MS Open Tech在开源这方面主要是跟随Apache的文化么？**

**Gianugo：**MS Open Tech毕竟是个公司，它具有公司所具备的所有属性。不过，参与社区是我们的主要工作之一。我在开源领域已经工作了20年，在这个过程中，我深刻的意识到，开放源代码只是整体中的一部分，还有很多其他的东西。在MS Open Tech，我们必须了解这一点。对我来说，重要的不是我们做什么，而是我们怎么做。

**InfoQ：其他很多公司也有开源技术中心等对开源方面的投入。他们的做法是，雇一些工程师到公司内部，让这些工程师参与社区项目，将自己的代码贡献给上游。上游的committer或maintainer会审查、提交他们的代码，而他们往往来自不同的公司。MS Open Tech也是这样的运作方式吗？**

**Gianugo：**还有比这些工作更重要的事情。将代码贡献给上游，相当于请求其他人为你维护代码。这就好像是你送给人一份礼物，但这份礼物会加重别人未来的工作量。你必须以正确的方式去做这个请求，而不是直接将代码扔到墙的另一边，仅仅因为它是开源的，就认为这个项目应该接收这段代码。实际上，很多代码提交是被项目拒绝掉的。正确的做法应该是跟社区进行沟通，让他们知道你来了，你会留下，而不是扔下代码就走人。这是开源过程中很重要的一部分。

我们一直在复盘我们的贡献。如果我们贡献一段代码，那是因为我们认为它很重要，所以我们会继续贡献，而不是打了就跑。

**InfoQ：比如说，对于Azure的Eclipse插件这个项目，整个流程是怎样的呢？**

**Gianugo：**这个项目跟上面说的情况不同，因为这是个我们自己运作的项目。对这个项目来说，我们进行开发和分发，同时我们欢迎社区人士为这个项目贡献补丁。我们自己就是这个项目的维护者。所以，这个项目主要是看其他人是不是愿意参与进来。如果有人带了一个很大的特性过来，这个特性会加重我们的工作量，那我们需要做一个评估：这希望加入的20000行新代码是否有这个价值，让我们愿意处理这里面可能包含的bug——或者应该说，肯定会包含的bug。

一个更贴切的例子是我们在WebKit、Blink和Firefox上的贡献。我们根据他们的规则贡献代码。对于Hadoop和Linux内核也是同样。

在Linux内核这个领域，不乏自己开发驱动、分发驱动的公司，有些驱动甚至还是二进制的。我们也可以这样做，但我们没有。我们去找到了Linux内核社区的维护者，向他们了解我们应该做的。一段代码进入内核的流程非常复杂，需要进入测试分支，需要让一位维护者审核你的代码，需要回应他们的各种反馈，比如“我们不是这样处理系统调用的”，或者“我们不是这样处理空格的”。不管怎么说，这是别人家。各家有各家的家规，我们需要遵守。如果要成为Linux世界的好公民，WebKit世界的好公民，Node.js世界的好公民，我们就必须遵守他们的规则。这是参与社区活动的原则。

**InfoQ：从MS Open Tech的官网来看，过去一年大概有60个项目左右，还有不少被归档的项目。**

**Gianugo：**是的。正如我所说，我们在这块已经做了7年，所以项目是做过很多。

**InfoQ：公司现在有多少人了？**

**Gianugo：**70个左右。这是比较合理的团队尺寸。另一方面，我们有一个叫MS Open Tech Hub的项目，通过这个项目，微软母公司的工程师可以临时加入Open Tech，参与开源项目。这些人都加起来，人数就很多了。

对于我们做的事情，可以参考这个例子：目前，ASP.NET的大部分代码都开源

了，这是MS Open Tech的工作成果之一。开源的部分包括MVC、ASP.NET网页，Entity Framework，Razor模板引擎，Actor FX，RX，TX等等，差不多是全部了。

同时，这个项目接受来自外部的贡献，这意味着微软之外的用户可以给.NET提交补丁，如果补丁被接受，下一个版本的微软.NET产品中将会包含这些补丁。如果你去ASP.NET的codeplex页面查看，就能够了解整个流程是怎样的。签署一个贡献者协议，就可以提交补丁。我们现在已经有为数不少的外部贡献者了。ASP.NET已经是一个社区项目。

**InfoQ：可以说，MS Open Tech主要的工作就是两个：给现有的开源项目贡献代码，以及将微软的现有项目开源。**

**Gianugo：**正是如此。这是一个趋势。这样做的原因有很多：将我们的技术更多的开源，会吸引社区的兴趣；参与更多的社区项目，可以使我们的产品也能支持这些技术，提供给我们的客户。Azure和Hadoop就是这个关系。

**InfoQ：你当时为什么决定加入微软？**

**Gianugo：**因为我看到了这个趋势。你听过我的故事就明白了。

长话短说，我的故事是这样的：我在很多开源项目里做过贡献：Firefox，很多Apache的项目，PHP，等等。我是Apache软件基金会的成员，还被推选为副主席。总的来说，可以说我是个不错的开源开发者。

最初，我是一个律师。我在学校念的是法律，不是工程。在学校念完法律之后我才发现，我并不想做一名律师。所以，我不得不“重新创造”自己。但是呢，我又没有足够的资源再去读一个学位。这时候，我在开源界学到了我需要的一切。社区是个非常神奇的存在。想想看，一个意大利小村庄里的失意少年，这一天还在想着，“完了完了，我这辈子就要做一个律师了”，结果过了一天，我就在跟HP、IBM和Sun的一群工程师们谈天说地了，谈论的话题是一个叫做Linux的新鲜玩意儿。这种协作感深深地打动了我。

为什么加入微软？因为我看到微软在开源领域的投入越来越多。更重要的是，他们做事的方式是完全对路的。

当时，我还在运作着自己的公司，是个局外人。有一天我们要帮助微软做一个项目：创建一个开源库，可以用Java读写，打开HTML文件。我是在那个时候认识微软，并认识了我最终加入的团队里面的成员。当时我就发现，他们有很强的意愿去做正确的事，并且用正确的方式去做。所以，当我决定离开自己的

公司，让它自己去成长的时候，微软给了我一个offer，我感到很兴奋，就过去了。

微软有一点我很喜欢，就是他在开源这块有很清晰的界线。这年头，自称开源的公司多如牛毛，但实际上，很多公司的开源态度都很暧昧，他们的产品总是有那么一撮是不开源的，但他们又总是顶着开源的帽子行事。我们对于开源是很清晰的：这些是开源的，那些是商业的，就这么简单。清晰的界限非常重要。

**InfoQ：这跟Apache许可协议有关么？微软对GPL协议总是有问题，你们是否总是倾向于Apache协议？**

**Gianugo：**我们在Linux内核上贡献的代码自然是GPL的。对于我们发起的项目，Apache协议是我们的最爱。不过，这并不妨碍我们在各类协议的项目中活动：BSD，GPL，乃至于AGPL。MongoDB就是AGPL协议的。

当然了，协议只是工具，有各自的限制，优点和缺点。我们总是要想办法选择比较合适的。

**InfoQ：那么，当你们进入一个新项目——比如Hadoop或者MongoDB——的时候，设立的目标是什么呢？**

**Gianugo：**首先，我们做这一切的最初原因，都是因为我们听从我们客户的声音，我们听从市场的声音。客户想要用某种技术。开发者们正在使用某些技术。我们的目标就是要让我们的客户和我们的开发者社区能够用上这些技术。

举个例子。我在这边参与的第一个项目差不多在两年半前，当时我们发现Node.js开始火了。所以我们就看看这个Node.js到底是什么东西，看了之后觉得这东西很有趣，有潜力。当时我们已经在琢磨JavaScript和HTML 5这档子事儿了，所以觉得这个东西真的不错。然后呢，我们试着在Windows上运行它，结果我们被惊呆了：这玩意儿压根在Windows上它就跑不起来！Node.js仅仅是为Linux设计的。

于是我们跑到社区里，表示我们想要解决这个问题。社区方面的回应是标准的开源社区最佳回复：提交一个补丁吧。不过，他们说的更加具体：补丁你们随便发，不过，由于解决这个问题涉及到性能、可扩展性等，肯定要对原本的核心代码进行大手术，需要使用很多的UNIX语义，创建结构库等等。所以，无论你们做什么，请保证你们的补丁不会影响到Linux上的性能。这当然是很合理的要求。

> 几个月之后，我们和Node社区发布了一个叫做libuv的新平台，说白了就是可以让你在Linux和Windows上都能进行event I/O。libuv现在已经被不少其他的开源项目使用。有了这个之后，Node.js在Windows上的性能提升了一个数量级，总算可以用了。有意思的是，与此同时，Node.js在Linux上的性能也提升了5%！当时社区发了一个声明说，这个新版本是首个可以在Windows上跑起来的版本，同时呢，它在Linux上也跑得更快了。这就是跟社区合作产生的力量。为什么说理解社区的运作机制那么重要，这就是原因。
>
> 所以现在呢，我们在Azure上可以提供Node.js的服务给我们的开发者。Azure上现在可以用PHP，可以用Java，可以用Python，可以用.NET，可以用Node.js——我们为它们都提供了SDK。而且，这些SDK都是开源的。而且，你可以给这些SDK提交补丁。而且，我们为这些SDK提供了开放的RESTful API，全都有完备的文档。
>
> 在这家公司工作仍然令我很兴奋，因为我们在各方面做的路子都是正确的。

**InfoQ：所以简单来说，你们的目标是两个：让微软的客户可以使用他们想要的各种技术；让其他技术领域的开发者可以在微软的平台上做开发。**

> **Gianugo：**是的。
>
> 看看我们现在所在的这间屋子：我们有这样多种类的设备，但它们打开网页的方式都是一致的。在今天，互操作性非常非常重要。市场上那些不重视互操作性的玩家最终都会被抛弃。客户有选择权，他们有权选择自己想用的技术，同时可用的技术又如此之多。你不可能将他们囚禁在一个技术领地里面。
>
> 曾经，我们有.NET阵营，Java阵营等等。你们做QCon，QCon大会上讲的编程语言何其多？分享的技术何其多？所以，我们必须要让不同的技术之间可以通话。
>
> MS Open Tech的使命是成为桥梁。互操作性是微软平台和非微软平台之间的桥梁。我们希望提供选择。有选择，客户才会来；没选择，客户就走了。

**原文链接：http://www.infoq.com/cn/articles/interoperability-is-the-key**

**相关内容**

- 微软将进行大规模重组
- 微软部门重组引发各方评论
- 微软将停用TechNet订阅服务

# 百度开发者中心

Baidu Developer Center

**应用万象　云创未来**

百度秉持“平等地成就每一个开发者”的理念，围绕百度云平台能力整合开放给开发者四大服务体系：开发支持、 运营支撑、渠道推广、商业变现。

我们希望藉由这些服务的开放，让开发者在开发、运营、推广、变现各环节有所受益，最大程度地保证开发者的利益,，共建开放共赢的局面！

开发支持

运营支撑

渠道推广

商业变现

Baidu 百度

http://developer.baidu.com

# 专栏推荐语

## 阿里技术嘉年华访谈录

7月13日，第三届阿里技术嘉年华在杭州召开，大会为期两天，涵盖“前端技术”、“业务架构&后端技术”、“搜索”、“大数据应用”、“无线技术”以及“测试”等专场，有将近3000人参与。

InfoQ在两天的活动中采访了诸位技术专家整理成合集奉献给各位，希望于您有用！

# 特别专栏 | Column

# 拔赤谈前端：KISSY的自成体系是一种趋势

**作者 杨赛**

Kissy是阿里前端自己开发的前端框架，目前最新的版本是Kissy 1.3.0，下一个版本1.4.0计划将在今年十月发布。在最近的阿里技术嘉年华上，InfoQ编辑对淘宝前端工程师李晶（拔赤）进行了采访，了解有关Kissy框架和他分享的Kissy Mobile相关的一些信息。

嘉宾简介：李晶，花名拔赤，淘宝前端工程师，具有多年前端开发经验，在团队协作、组件开发、移动Web App等方面有深入研究，曾经参与淘宝首页、KISSY等项目开发。他翻译过《JavaScript Web富应用开发》、《JavaScript权威指南（第六版）》。

**InfoQ：先简单的介绍一下Kissy吧。在你看来，最初开发Kissy的时候已经有了jQuery，淘宝自己也在用YUI2，为什么还会开发这样一套框架呢？**

> 拔赤：KISSY是阿里前端自己开发的开源前端框架，根据阿里自己的业务有很多定制和本地化，力争大而全。KISSY最初是由玉伯发起，最开始是一款设计精密的编辑器控件，后来提供一个轻的核心代码集合，为了弥补YUI2在语法、编程模式和代码复用模式上的缺失。YUI2和jQuery在当时都有一些棘手的缺陷是无法满足淘宝团队协作的场景。
>
> 1）YUI2体积过大，在完成同样功能的情况下页面JS体积要比KISSY多一倍，5年前网速没有像现在这么快，这样的体积严重影响页面载入速度。所以当时我们需要体积更小的库。
>
> 2）jQuery虽然提供基础的Ajax和Dom封装，但缺少Loader，这样就无法管理团队协作和多人开发，而淘宝从08年开始团队协作就已经是主流了。
>
> 因此，体积小巧、支持团队开发在当时看来是最迫切的需求，而当时的开源类库都无法满足（YUI3之后才内嵌了Loader、像第三方JS加载器requirejs和labjs在几年以后才陆续出现），所以KISSY从开始就走了一条自主探索的道

路，一直到今天。

**InfoQ：你主要参与的Kissy组件有哪些？能说说你是怎么参与到里面的吗？**

拔赤：从最开始的Loader、Ajax，到后来的Calendar、Slide，我从开始就参与到KISSY的核心代码贡献中。因为KISSY源码托管在Github上，所以只需fork一份代码，就可以为它贡献代码。这也是大多数开源类库的组织方式。

**InfoQ：Kissy在描述中表示要服务使用中文的工程师，所以项目说明、文档当中都有大量中文。这一方面对中文用户友好，另一方面也制约了Kissy在国际化方面的发展。你会不会觉得这样太迁就那些不愿意学英文的工程师了？**

拔赤：KISSY的首要目标是服务阿里集团，其次是国内用户，再次是国际用户，所以我们目前会优先考虑中文。所以不管是注释还是文档，我们都会优先使用中文，对国内用户友好。再一个，我们自己的英文水平也高不到哪里去，没必要强求开始就写英文文档和英文注释，还是把主要精力放在代码结构和质量上。关于KISSY的国际化，我们会在KISSY 1.4.0时打一些补丁，支持一些组件的国际化。至于是不是迁就那些不愿学英语的工程师，他们学英语不会选择用KISSY来学习，这也不符合KISSY给自己的定位。

**InfoQ：根据Kissy Mobile的项目介绍来看，它重点关注pageview的切换。这方面目前有没有比较业界推荐的标准处理方法？Kissy Mobile试图解决什么问题？**

拔赤：KISSY Mobile是一套面向移动端的功能特性集合，pageview切换是其中一个，名叫Mobile App Toolkit，也是我这次分享的主题。关于处理多页切换的框架也有一些，比较流行的是jQuery mobile，但正像第一个问题那样，jQuery Mobile也无法解决目前我们的业务上的关键问题，比如页面切换是基于post请求？或者面向低端移动机型提供降级方案？

KISSY Mobile App Toolkit主要解决那些想将Web做App化的产品，并且Web会和Native整合，并力争做到和Native一致的用户体验。如果你的项目有这种需求，你需要尝试下KISSY mobile app toolkit。

**InfoQ：Kissy即将发展到1.4版本。新版本有哪些值得关注的特点？**

> 拔赤：1，种子文件会变的更小；2，CSS选择器会面向移动端做性能优化，性能全面超越zepto。这两个特性应该是接下来这个版本最重要的功能。面向移动端的优化是一种趋势，而KISSY应该在这个方面做得更好，更懂得企业应用场景和需求。

**InfoQ：现在整个国内前端领域，各家大公司自己各搞一套框架，各自维护；小公司只能跟随，如果跟错了队就会很惨。你觉得未来的趋势会怎样？会有统一的上游，大家一起维护，还是大家渐行渐远？**

> 拔赤：从KISSY创立起到现在，可以看到KISSY的自成体系是一种趋势，但凡大一点的网络公司都不约而同的选择自建技术体系，也就是之前很多人提到的我们在教堂和集市之间选择前者。原因很简单，要想用好jq/zepto/bb之类的“小”库，必须配合模块加载器（不管是静态打包还是动态合并）建设起一套组件机制并生长出一个组件库，因此不如从开始就自建。所以大公司的技术体系都趋向封闭或半封闭，真正的开放只存在于没有公司背景的Github社区里。因此，很可能我们最终只能一定程度的打破KISSY的“封闭”。而其他公司包括全球顶尖互联网公司也是一样的路数，所以要想做大，早晚要自创体系。拥抱开源只是创立之初处于成本考虑的一种权宜之计，真正做精只有从0开始。

**原文链接：http://www.infoq.com/cn/news/2013/07/bachi-on-kissy**

**相关内容**

- 百度技术沙龙第39期回顾：前端快速开发实践（含资料下载）
- 小脚本，大世界--专业化前端的自我修养
- 博文共赏：国内前端行业十日谈
- FIS2.0全新的百度前端解决方案
- 小微金服前端技术专家伯约访谈：相信数据的力量

# 特别专栏 | Column

# 阿里巴巴M-DMP负责人许成访谈：从顾客管理到消费者管理的升级

作者 水羽哲

7月13日，第三届阿里技术嘉年华在杭州召开，大会为期两天，涵盖“前端技术”、“业务架构&后端技术”、“搜索”、“大数据应用”、“无线技术”以及“测试”等专场，有将近3000人参与。在14号的“广告”分论坛上，来自阿里巴巴的高级产品专家许成以“淘宝DMP平台”为主题分享了阿里巴巴DMP平台（Data management Platform，基于对消费者数据的管理进行更高效营销和广告投放的整合平台）的概况，InfoQ在现场对他进行了采访。

**InfoQ：许成老师首先请您介绍一下自己？**

> **许成：**我是许成，2010年来到阿里巴巴，负责阿里巴巴的直通车定向推广业务，2011年的时候，我们做了第一个淘宝DSP，据我了解，应该是中国第一家真正意义上的DSP，2012年开始做DMP平台，在来阿里之前，我在硅谷做了两家创业性的小公司，一家是广告相关，另一家是与数据相关。

**InfoQ：这次您在主导的产品是M-DMP，能不能给我们介绍一下它的产生背景和现状？**

> **许成：**DMP这种产品在国外是很多的，在中国也不少，他们有不同的关注点：有些是关注数据交易、有些关注后台数据的管理，我们的关注点是营销，因为我们认为数据，尤其是大数据是要在应用场景中才有价值的东西，我们关注在广告渠道里面的营销，就把它叫做M-DMP，它是DMP的一个变种，能够通过数据的整合管理，让整个营销的效率更高。同时为了达到这个目的还需要更丰富的数据，因此需要数据的交易、整合、和无线数据打交道等，这些环节都相当于M-DMP的一个功能或者是我们的一种方式，但目标都是能够让消费者管理做起来，使得商家能够从顾客管理升级到消费者管理。

**InfoQ：您提到数据M-DMP的核心，现在阿里有一整套完善的数据体系，但**

**是你们还会引入第三方的一些数据，比如说高德地图或者新浪微博，那么你们会提供怎样的一种机制让他们更方便的融入进来？**

> **许成：**最主要的就是两个方面：
>
> 1. 数据的关联；这是一个偏技术性的机制，因为不同的数据体系有自己的key，不同的Key怎么样才能够关联、打通起来其实有很多的技术细节。现在我们工作中特别重要的是移动数据和非移动数据、线上数据和线下数据的关联、打通，这个是一个持续进行的过程。从我分享中举得那个例子也可以看到，移动和非移动数据的关联，有的时候成功率能达到30%，这是一个起点，最终是希望越高越好；
>
> 2. 数据的授权和交易的机制；因为数据要流动起来，不可能说你就把数据通过光盘这种方式给我就行了，这种方式之前有人用过，但是后果很严重，为什么？因为数据流失了，而且数据的产权保护也成了一个大问题，所以这种方式是不能够持续下去的！我们的机制是把数据的授权和交易做到一个平台之内，相当于一个市场的概念。这样每一个数据提供方，比如新浪微博，他们可能就有动力去把他的数据变得更好，而且也愿意拿出来分享。

目前这两个点我们现在都在进行当中。

**InfoQ：也就是把数据做成一个闭环，在可控的范围内。那么通过DMP，我们希望能够进行精准的营销，开始的用户是一个很大的范围，现在能够压缩到很小，然后对这部分人群进行精准投放，比如说有些品牌他们做品牌知名度的推广，越多人知道越好，这时希望另一种方案，怎么办？**

> **许成：**把DMP等同为精准投放其实不完全对！我们的目标更多的是对人群有一个全方位的了解，这个人群可能是一个很大的人群，所以我也一直在强调，我们的产品定位是消费者管理。只要你对消费者有一个比较好的了解，那么有可能发现能够找到很大的人群量。比如我刚才举得戴尔电脑的例子，比如曾经购买过戴尔品牌的回头客量很少，但是我们对消费者了解的非常精准，我们知道18到25岁的学生是我们的潜在消费者，那么就可以通过满足各个维度需求的数据透视分析，实现大规模用户的精准、定向投放。也能满足对于追求比较高效果的高精准投放需求。
>
> 这些其实都不需要DMP帮他们来做，完全是客户自己来判断，我们只是设计了一个平台，品牌也好、淘宝店也好，都可以去定义他们期望的消费者人群。

**InfoQ：相应的也能够做关联？**

**许成：**是的，比如客户有自己的CRM系统，当时他们对客户的了解是片面的，而这个客户在全网的消费情况是怎样的？他们没有办法做到，比如有人在联合利华买了一块肥皂，他们只是知道他是联合利华的客户，但是作为一个消费者，他有更加丰富的信息，这些信息对于联合利华来说远远超过了他在店里买了一块肥皂的价值，这个时候客户就可以通过系统做关联对他进行更精准的推送，价值就凸显了！

**InfoQ：比如说现在新浪微博的广告现在有一个问题，如果我是用户，在购买了一口锅之后我的新浪微博就全是锅了，但是这个锅我已经买过了，所以从需求上讲我现在就不需要这些信息了，就感觉广告是没有上下文的？**

**许成：**这不是数据平台的问题，是DSP投放策略的问题，就如第二场品友互动所讲的，对他们来讲数据是他的上游，那么广告投放、消费者就是他的下游。从某种意义上讲，比如我们在新浪微博上面非常了解这个消费者，知道消费者最近买了一口锅，但是针对这个消费者特定的标签该怎么去管理、怎么去定义？这还是没有做得特别好的一件事情。可能买了锅，只能说明他是一个对日用品或者厨房用品有兴趣的消费者，那么更好的一个方法是卖厨房用品的品牌，针对这个消费者去做投放。我觉得DMP接入之后，这个情况会有所改变。

**InfoQ：刚才您提到DMP赋予了广告主很多的能力，实现对人群更精准的细分，但是DMP后台目前看起来有些复杂，有时候广告主也不知道这些设置对他们的选型有怎样的影响，所以他们必须有一个试错的过程？**

**许成：**这其实是技术挑战，主要有这几个方面：

1. 用户界面；如何把这么多维度以一种直观的方式展示出来其实是很复杂的，并且还要有很好的互动，这是一个技术挑战！

2. 建模的挑战；比如说广告主经常是说我要把这么多的钱花掉，而且我希望广告效果还可以，这个时候就是一个质和一个量的博弈，需要有较强的算法建模能力，目前我们的平台里还没有，后期可能会采用开发平台的方式，让算法人员能够接入进来；

3. 实时性；因为客户需要在线去试，这才是真金白银的东西，客户试出来的结果也是特别有价值的，这时就需要有一个非常快的机制能够给出结果。比如说我的前台，设定了一个人群，然后能够在很快能得到反馈，就可以去做优化的动作。

> 那么这三个方面其实都有很多的技术挑战，现在M-DMP平台还刚开始，很多技术能力我们现在都没有，包括阿里在内现在都做不到，我不知道国际上能不能完全做到，对于这些维度很大、实时性很高、建模要求很高的技术其实难度很大的。我理解这种业务需求是在推动技术向边缘在发展，如果真的是这样，就会把诸如大数据技术等推上一个新的台阶。

**InfoQ：你们也用到了比如说阿里数据平台等其他技术部门的支持？**

> **许成：**我们跟他们有非常紧密的合作，也经常在推动他们，希望他们去能够在实时性、高纬数据透视等技术方面提供更强大的支持。

**InfoQ：刚才演讲中您提到会把第三方接入进来，让他们提供一些定制的服务，这个能介绍一下吗？**

> **许成：**我们在最开始设计这个产品就考虑到这个问题，希望能够把数据开放出来，让开发者能够通过提交算法来把数据转变成标签，然后开放给广告主。由于底层的数据是各种具体的行为，不会提供给广告主的，那么在我们和广告主之间就有一个空缺，需要开发者来填补。

**InfoQ：比如说我要把第三方接入进来，肯定会谈到用户隐私问题，因为这个DMP他这个平台就是在对用户的很精准的一个跟踪，这块用户隐私，刚才你也谈到他不会破坏用户隐私的现状。**

**许成：**对于用户隐私，我们的理解是这样：

> 1. 诸如用户身份证号、电话号码等隐私性最强的信息，如果被泄露出去，甚至会有法律的一些后果，所以我们绝对不会把这部分数据暴露出去；
> 2. 对于一些比较弱的信息，例如用户兴趣、购买行为等，用户可以自己设置是否希望公开给广告商作为精准投放的分析数据源，当然，用户既希望能够获得精准的推送，又不希望隐私泄露，其实也是一个博弈的过程，我们只能不停的去尝试，让大家获得一个相对满意的结果。

**InfoQ：刚您说到在接入第三方过程中有一点，就是关于移动这块的接入，这块能不能具体提一下？**

**许成：**我们认为移动是很关键的未来，因为从消费者的路径上来讲，移动变得越来越重要，而且移动有个特点，它更像是消费者的第三只手。消费者会随身携带这个设备，PC做不到这一点！那样，移动设备能搜集消费者的信息，也能够让消费者受到自身的影响，但目前移动数据的搜集和打通还是一个国际性的挑战，在我的分享中也提到了一些方法。

**InfoQ：比如说嵌入SDK这样的？**

**许成：**现在是这样的，但是我知道美国有一家公司，他们好像想出了另外一种方法，大致是通过统计建模的方法来预测。比如说用户拥有手机、iPad，但是平台不知道这些设备属于同一个人，他们能够利用搜集到的IP地址信息、应用信息等通过概率的手段判断相关性，这个过程不涉及隐私的问题，只要概率上能够判断大致的人群就能够做广告，这是一个好的方法，我相信也有其他的方法。

**InfoQ：从刚才您谈到DMP现状，我觉得它的一个构想是很好的，也希望能够推动现在广告的发展，那么对于它今后一个发展路线，是怎么来设想的？**

**许成：**数据有点像我们手中的钱，本身是没有任何价值的，数据底层就是01字符串，但是它一旦进入流通就出现了价值，这个流通过程包括广告、金融领域等，数据能够帮助这些业务发展。广告是DMP的一个核心应用，我个人认为这只是第一步，也因为他比较直接，能够立刻见效。但数据驱动业务，对于健康、医疗，保险等都是非常关键的。所以，我们要在底层技术上要不停的往前走，上层的应用方面找更多的出口，这两个方面结合我觉得才有可能做出更好的类似与DMP的应用出来。

**InfoQ：所以您的那个角度还是从数据为出发的原点，然后让它推动其他的产品，现在比如说广告只是推它的第一个？**

**许成：**只是其中一个而已，所以说大数据这个词本身有点虚，只是把数据都放在一起就好了吗？不是这样的！数据放在一起是就是一些结构化的数据信息，其实没有什么太大意义，但是如果把它放在一个业务里面，比如说帮助广告主能够提升他的ROI、做更好的品牌推广或者说帮助银行去做更好的贷款投放，亦或是帮助病人更好和其他相关的人来沟通等等，我认为这些目标才是真正的驱动力。

# 特别专栏 | Column

# 小微金服前端技术专家伯约访谈：相信数据的力量

作者 水羽哲

7月13日，第三届阿里技术嘉年华在杭州召开，大会为期两天，涵盖“前端技术”、“业务架构&后端技术”、“搜索”、“大数据应用”、“无线技术”以及“测试”等专场，有将近3000人参与。在“前端技术”分论坛上，来自小微金服的王磊（花名：伯约）以“用户行为监控”为主题分享了小微金服的用户体验验证体系、Session&Tracker用户行为监控解决方案和用户行为动画还原方案等，InfoQ在现场对他进行了采访。

**InfoQ：首先请您做一下我介绍？**

> **伯约：**我是伯约（王磊），来自支付宝，主要是负责用户行为监控体系的建设，也会兼顾去做核心产品的一些用户体验工作。

**InfoQ：您主要是做通过数据监控来指导产品的改进工作，那么现在如果支付宝有新的产品设计方案时，需要在监控方面提前做哪些工作？**

> **伯约：**现在支付宝的产品设计环节，我们会考虑后期的数据需求，设计师会告诉前端在设计中在哪里进行埋点、布控。前端根据设计师所提供的信息，将对应的代码部署上线，这是一个人肉的方式。另外，我们还有一套自动化布点的方案，它会通过一个脚本，根据页面元素的一些特点，例如元素的父对象、URL的域名前缀等规则，自动拼接出当前元素的名称，实现自动埋点。

**InfoQ：对于数据采集，如果早期可能采集了三个点，那么后期我才发现这三个点不够，如何来补数据？**

> **伯约：**就像刚才说的，我们有自动化埋点，可以通过这套自动化埋点的方案去补全数据，但是现在有一个问题是自动化埋点的可读性并不是那么好，因为要

考虑到设计师，包括产品经理可能是非技术出身的，如果说那个点不是他自己预设的，那么他们在后期自己去看的时候会非常困难。

**InfoQ：所以现在你们想到的、想不到的元素都会有埋点，会不会采集的数据量太大了？**

**伯约：**这个会考虑，但是目前来看，支付宝的数据存储能力应该是非常强大的，我们现在对数据也非常关注，比如说刚才分享中提到的案例数据，我们会对数据做一些分级，保留对以后数据挖掘有用的数据，剩下的数据则可能存一、两个月就定期的去做一些清理。

**InfoQ：也就说现在数据存储是没问题的，在数据分析方面，你们是和数据仓库团队来进行合作？**

**伯约：**是的，和数据仓库团队合作。

**InfoQ：产品的设计可以通过后期的数据来做论证，那是否可以在产品上线之前就建立数据模型提供预警机制呢？**

**伯约：**这个是可以的，我们在产品上对应的地方去做埋点，通过一个类似于产品魔方的平台，去做这个事情，进而去监控用户在操作过程中所遇到的问题。我们可以通过通用的全局数据，比如说在一段时间的走势、访问量波动等来判断产品是否达到预期。也可以通过细节的数据，比如预设的产品用户操作路径分析具体的产品体验问题。

**InfoQ：之前您提到的那个案例最后得出结论是：对于一个具体的问题，并不是说在一个页面，或者两个页面就能够看出问题所在的，因为支付宝毕竟是流程中的一环，那么可能在他之前别的网站时遇到了一些问题了，这时该怎么分析？**

**伯约：**这个可能就需要其他网站授权了，说实话是挺难的，现在我们看到公司也有一些动作，把外围相关的数据打包进来，这样我们就可以看到用户在整个购物链路里面的体验问题了。我觉得这可能跟我们在数据上面的一些思考是有些关系的，有了这些外围的数据之后，整个链路真的就能打通了！

**InfoQ：刚才分享你提到了一个Session&Tracker的解决方案能够实现用户状**

**态的还原动画，同时您还提到这个方案还存在一些问题？**

> **伯约：**主要是数据权限控制的问题，因为如果通过这种方式会涉及到用户的敏感信息，因此需要确定哪些人可以看、哪些人不可以看，这真的是一个非常大的系统工程，如果没有办法去界定哪些人可以看，万一真的用户的隐私受到侵犯，这是坚决不允许的。

**InfoQ：国外一些公司，比如Google或者Facebook他们应该有一套方案，你们是否了解他们的一些做法？**

> **伯约：**市面上现在比较现成的一些行为监控的方法，我们都有去看过，但其实很多是开放给所有的个人甚至企业的，这一套监控体系可能跟我们想要的还有一点不一样，他们可能更关注全局面数据的走势，但我们其实很多时候除了这部分的数据之外，还希望知道用户的操作细节，通过结合这些具体的信息才能更好的去做一些体验的优化。

**InfoQ：我看到你的介绍里提到你十分相信数据的力量，能否谈一下您对数据的看法？**

> **伯约：**我觉得有些时候数据只是一种习惯跟信仰，可能长期以来，我无论做什么都会去做一些数据的跟踪和采集工作，比如说我前段时间在看一些人做的同步并行下载，国外其实已经有很多大牛在做这个事情了，但我自己觉得任何东西都是需要亲自去验证，所以我会做对应的一些数据测试，因为大牛们的文章里是没有具体操作的，所以还需要有一些具体的数据去验证，那么验证之后我们再看到这个信息时，可以通过数据更直观的把结果展现出来，跟大家单纯看文章的收获是不一样的。
>
> 我觉得无论做什么，特别是用户体验，其实还是偏感性的工作，如果说没有数据，也是不科学的。有数据指标，我们就可以去做相对的比较，而不是单纯去看它的绝对值，对于体验也更可衡量。

**InfoQ：是的，感性的设计需要理性的支撑，那最后一个问题，作为一个前端的大牛，希望能给这些前端的同学提一些建议如何提升自己的技能？**

> **伯约：**我觉得其实还是习惯，平时遇到问题是学习跟成长的一个最好的机会。比如，在我们团队中就会发现有很多同学很忙、压力很大，在项目过程里面会

> 遇到非常多的问题，但是可能解决完问题就接着忙了。很多时候去解决一个问题是很容易的，但是知道所解决问题的背后的原理是什么，才是成长中最重要的！如果你一直忙，一直在重复做你以前在做的事情，而没有去问为什么，是不会有成长的！但如果说有这种积累的习惯，不用多久，自然而然就会觉得自己变成大牛了，当然我不觉得自己是大牛。

**原文链接：http://www.infoq.com/cn/news/2013/07/believe-data-power**

### 相关内容

- 拔赤谈前端：KISSY的自成体系是一种趋势
- 对话百度前端工程师张云龙：F.I.S与前端工业化
- FIS2.0全新的百度前端解决方案
- 百度技术沙龙第39期回顾：前端快速开发实践（含资料下载）
- 小脚本，大世界--专业化前端的自我修养

# 避开那些坑 | Void

## 那些年我们犯过的错（一）——豆瓣Xupeng：补齐那丢失的三分钟数据

作者 员旭鹏

2013年6月底，下厨房由于技术人员操作失误，导致近两个月数据丢失。下厨房的技术博客对整个事件进行了总结，记录了整个事故的隐患、发生、数据修复、以及如何避免此类错误再次发生的一些措施。

大部分有经验的运维、DBA都少不了做过误删数据的事情，这次事件引起了很多工程师的共鸣与讨论。就在该事件发生后的第二天，豆瓣的Xupeng撰写了一篇博客，记录自己《这几年犯的错》。他表示：

> 这几年犯过很多次严重影响线上服务的错误，像重启了错误的节点这样的事情应该算作能够对线上造成影响的最微不足道的错误。

在文中，Xupeng简单罗列了这些年犯过的一些错误，包括停用线上memcache 集群、软件 bug 导致线上 memcache 集群被污染、恢复数据时删除了更多数据等。

犯错误的经验其实是跟源代码一样宝贵的信息。一个有经验的工程师不仅要了解项目是如何实现的，更重要的是要了解实现的过程中可能会遇到哪些坑，以及如何绕过它们。那么，为什么不把犯错误的经验也像源代码一样，将错误是什么、如何发现的、如何解决的、之后如何预防等步骤，详细的记录、共享下来呢？

从今天开始，InfoQ将开辟《那些年我们犯过的错》话题，隶属“故障排除”专题下，专门邀请经验丰富的工程师们分享他们处理故障的经验。

第一位受邀分享的嘉宾，正是上面介绍的Xupeng同学。

嘉宾简介：Xupeng，真名员旭鹏，Linux 爱好者，Python 程序员，Mac 用户，豆瓣首位全职 Linux SA，关注 Linux 性能优化、虚拟化，擅长 Linux 及其上软件系统的 troubleshooting，同时管理豆瓣若干 MySQL 集群。

本文分享的是Xupeng在博客中提到的最后一个事件，即“误操作并误删数据文件”事件。

# 发生了什么事？

有一个这样结构的 MySQL 集群：

```
   -----          -----

   | A |    <->   |B|

  -----       -----

  /   \

-----  -----

|C| |D|

-----  -----
```

A 是 master 节点，接受线上的读写请求，B 和 A 同构是 A 的热备，A 和 B 互相同步，当 A 有故障时可以把 B 提升为 master 接受读写请求，C 和 D 是 A 的 slave，用于离线查询和备份等目的，这是一个比较常见的 MySQL 架构。

有一次对一张比较大的 MySQL 表做 schema 变更，先在 slave (B) 上做完变更之后做了主从切换，使线上使用新的 schema，但切换之后我发现新的 slave (切换之前的 master: A ) 的数据不对，后来在仓促的修复过程中又误删了数据文件，现在想起来还一身冷汗。

# 如何发现错误？

在主从切换之后，我像以前一样在新的 slave (A) 上执行 `show slave status`，检查新的 slave 节点同步是否正常，Slave_IO_Running 和 Slave_SQL_Running 都是 Yes，Seconds_Behind_Master 为 0，这表明 slave 的 IO thread 和 SQL thread 在正常工作，但 Exec_Master_Log_Pos

却一直保持不变，`show master status` 的输出也表明 slave 上没有新的 binlog 写入，这一定是哪里出了问题，线上不可能没有新的数据写入。

立刻到 master (B) 上确认了的确是有写操作的，但是 B 上却没有 binlog 记录，随即发现了 B 上的 binlog 是被停止了的：

```
mysql> show global variables like 'sql_log_bin';



+--------------+-------+

| Variable_name | Value |

+--------------+-------+

| sql_log_bin   | OFF   |

+--------------+-------+
```

这是我之前在 B 上做 schema 变更时，为了避免执行时间很长的 DDL 被同步到 A 上阻塞线上应用，本意是停止当前 session 的 binlog 记录，应该执行的是 `set sql_log_bin=0`，但实际上却执行了 `set global sql_log_bin=0` 停止了整个 server 的 binlog 记录，于是主从切换之后，新的 master 就不再记录 binlog。

## 当时所采取的措施？

从完成切换到我意识到问题和可能的原因，已经过去了三分钟，这意味着 A, C 和 D 节点与 B 相比，缺少了最近三分钟的数据，并且只有 B 这一个节点才有完整的数据，于是第一时间恢复了 B 节点的 binlog 写入和 A 节点的同步。

紧接着就开始考虑该如何在 A，C 和 D 上补齐缺少了的那三分钟数据变更（有新写入的数据，被修改的数据和被删除的数据）。

我们在数据库节点上使用 tcpdump 抓取并记录 MySQL 的所有通信，这样做的目的主要有三个：

1. 当事后发现 MySQL 有异常比如突发的性能问题时，可以分析之前抓取的数据来定位问题
2. 安全审计
3. 在有计划地切换主从之前，使用从 master 上抓取的查询对 slave 进行热身

因此首先想到的就是使用 tcpdump 记录的 MySQL 通信数据来恢复，但是很快就发现了问题：由于 A 节点是在一个错误的数据集基础上重新开始同步的，接下来的几分钟内同步就出错了，原因是在那缺失的三分钟变更内有新记录写入但没有记录 binlog，恢复 binlog 写入之后，记录又被删除，删除操作被同步到 A 上，但由于 A 上之前并没有写入过这条数据，删除操作就失败了，我这才意识到第一时间恢复 A 的同步是错误的，正确的做法应该是第一时间恢复 B 的 binlog，但暂停 A 的同步。

如果没有率先恢复 A 的同步，那么就可以先尝试在 A 节点上回放 tcpdump 记录下的 query，但由于 A 已经在错误的基础上进行了几分钟数据同步，这条路基本上是走不通了，并且即便没有立刻恢复 A 的同步，也几乎没有希望仅仅依靠 tcpdump 抓取的数据进行准确的恢复，原因是 tcpdump 抓取到查询顺序和 master 上执行的顺序并不一致，在数据恢复这件事上，tcpdump 和 binlog 不等价。

紧接着考虑使用 Percona 的工具 pt-table-sync 来同步 A 和 B，但考虑到同步几百张表的过程会非常漫长，在这个过程中还会持续地给线上带来压力，也放弃了在紧急状况下使用这个方案。

能想到的最安全的方式只剩下了一个：使用 Percona Xtrabackup 对 B 进行热备份，重建新的 slave。

# 经验总结

这次事故的教训是，应该尽量使用软件而不是手工来执行确定性强的重复性操作。在执行重复性操作上，软件远比人靠谱，即便是人对着 checklist 一步一步执行，犯错误的几率也比软件大得多。应当使用经过验证的工具比如 Percona 的 pt-online-schema-change 进行 schema 变更，避免手工操作，数据库切换前后的检查也应该使用工具而不是手工，故障恢复之后第一时间改进工具以免这样的事情再次发生，对于 SA 和 DBA 来说，工具开发和改进是一件需要持续投入精力去做好的事。

---

《那些年我们犯过的错》话题正在征稿！投稿方式：

1. 回答以下问题：
    1. 介绍一下你印象深刻的、你犯过的一个错误。
    2. 你是如何发现/捕捉到这个错误的？
    3. 发生了错误之后，你尝试做了哪些事情？
    4. 你是如何从错误的症状跟踪到错误诞生的原因的？
    5. 之后，你做了哪些工作防止此类错误再次发生？
2. 撰写一段你的个人介绍。
3. 发信给 editors@cn.infoq.com ，邮件标题注明《那些年我们犯过的错》投稿，将上述内容粘贴入邮件正文当中。

期待您的来信！

**原文链接：http://www.infoq.com/cn/news/2013/07/xupeng-mysql-mistake**

**相关内容**

- 豆瓣石岩谈豆瓣阅读前端排版工作的感受
- 按需搭建灵活多变的 Web 应用 —— 以豆瓣阅读计划为例
- 如何恢复丢失的两个月数据——“下厨房”技术团队分析总结6.26数据库事故
- MySQL运维感悟
- 选择或者放弃MySQL的理由

# 避开那些坑 | Void

# Async/Await - 性能开销和其他陷阱

**作者 Roopesh Shenoy ，译者 姚琪琳**

Async/Await是C# 5引入的最强大的语言特性。但有哪些陷阱需要注意呢？与这两个关键字相关的开销又有哪些？

MSDN上的一篇文章“异步编程最佳实践”强调了以下几点：

- 选择异步Task方法而非异步void方法，事件处理程序除外。它们采用不同的错误处理语义。Marker Metro的Keith Patton详细解释了这一点。
- 避免混合使用阻塞和非阻塞代码，这可能会导致死锁、更加复杂的错误处理以及上下文线程的意外阻塞。
- 如果后续代码不需要使用原始上下文，就用ConfigureAwait(false)来获得更好的性能。这会使后续代码运行在线程池上下文中。如果你不得不混合编写同步和异步的代码，它还可以避免死锁。注意，在服务端使用时会有略微不同的考虑。

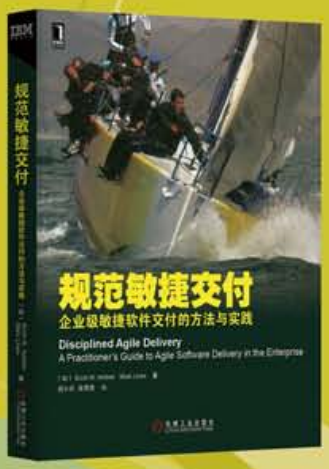
RedGate的软件工程师Chris Hurley解释了async-await的CPU开销，并通过一小段示例代码进行了演示：

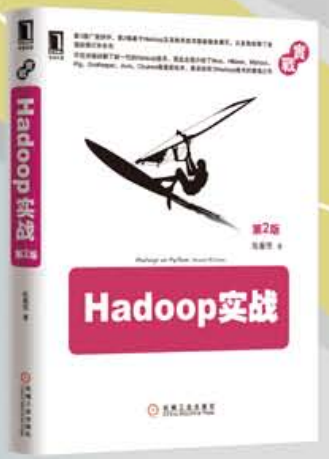
- 用async关键字调用一个方法，会创建一个状态机，并构建一个Task对象来包含要执行的工作，以及获取执行上下文和同步上下文。
- 在该展示示例中，第一次调用共运行了963个框架方法，来初始化相对简单的async方法。
- 上下文将被缓存，因此后续调用会减少很多开销。
- 对于在很短的时间内（1毫秒）就能同步运行完的方法，异步开销将阻塞调用线程更长的时间。在示例中，调用线程解除阻塞差不多要用45毫秒的时间。即使在循环中，后续调用会大幅减少开销的情况下，调用线程也看不到任何性能优势。
- 结论——避免对（耗时）短的方法使用async/await，避免在小循环内使用await语句（可以将整个循环放到一个async方法内）。

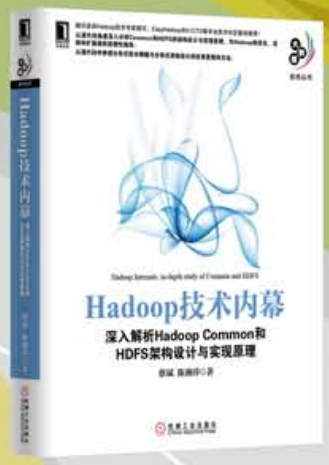
我们之前已经介绍过一些在使用这两个关键字时会遇到的其他常见陷阱。

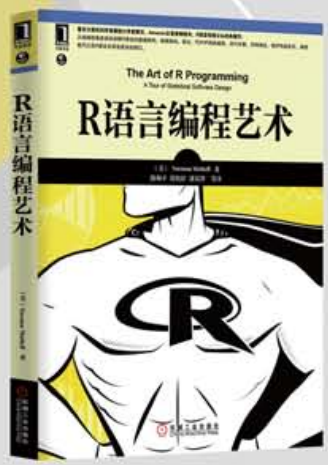
# 华章精品图书推荐

HZ BOOKS 华章科技









## 诚征合作，欢迎投稿

如果您有写作或翻译计算机图书方面的计划，欢迎通过以下方式与我们取得联系。

邮箱：linux1689@gmail.com 电话：010-88379525

微信：linux1689 联系人：杨福川（副总编辑）

机械工业出版社
China Machine Press

# 新品推荐 | Product

## Java EE 7推出Expression Language 3

**作者 Dan Woods ，译者 马德奎**

Java EE 7对Java表达式语言API进行了大幅度调整，并命名为Expression Language 3。根据JSR-341，EL API的增强包括对lambda表达式的支持、静态字段和方法访问、集合处理和独立处理器模式。

**原文链接：http://www.infoq.com/cn/news/2013/07/el3**

## Visual Studio 2012发布新更新

**作者 Jeff Martin ，译者 孙镜涛**

针对Visual Studio 2012的开发工作并没有结束，因为Microsoft宣布将发布它的第四个更新包。2012.4修复了一些bug并提升了与Visual Studio 2013的兼容性。

**原文链接：http://www.infoq.com/cn/news/2013/07/vs2012_4RC1**

## .NET Reflector 8.2 支持Visual Studio 2013、局部变量高亮显示和代码地图视图

**作者 Anand Narayanaswamy ，译者 孙镜涛**

Reg Gate Software公司发布了.NET Reflector 8.2，支持Visual Studio 2013、局部变量高亮显示和代码地图视图，另外还解决了仅启用我的代码选项的bug。

**原文链接：http://www.infoq.com/cn/news/2013/07/NET-Reflector-8-2**

## Essential Studio for JavaScript已发布，用于开发基于业务线的Windows Store应用

**作者 Anand Narayanaswamy ，译者 李彬**

Syncfusion于近期发布了Essential Studio for JavaScript，提供了超过30个控件来帮助开发者构建直观的Windows Store业务线应用。

**原文链接：http://www.infoq.com/cn/news/2013/07/essential-studio-for-javascript**

## Google Dart新进展：Polymer代替Web UI

**作者 Abel Avram ，译者 马德奎**

Google Dart将用Polymer代替Web UI，从表面上看，它们的主要差别是数据绑定和事件处理。

**原文链接：http://www.infoq.com/cn/news/2013/07/dart-polymer-web-ui**

## GAE 1.8.2引入专属Memcache和模块化

**作者 Abel Avram ，译者 马德奎**

Google应用引擎的开发人员现在可以使用最大20GB的专属内存缓存，并且可以将应用分成模块来提供有状态和安全的服务。

**原文链接：http://www.infoq.com/cn/news/2013/07/gae-memcache-module**

## Ruby on Rails 4发布：通过Turbolinks提速页面

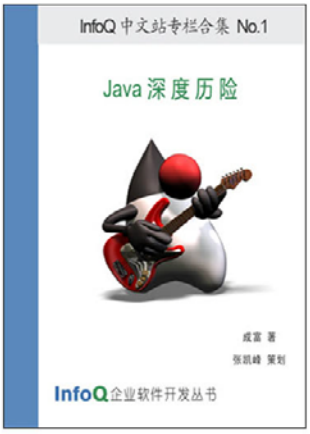
**作者 Mirko Stocker ，译者 臧秀涛**

Ruby on Rails 4已于近期发布。该版本利用Turbolinks改进了页面速度，缓存也更容易使用了。Rails 4不再支持Ruby 1.8，建议开发者使用Ruby 2.0。

**原文链接：http://www.infoq.com/cn/news/2013/07/rails4**

# InfoQ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

时刻关注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

商务合作：sales@cn.infoq.com

读者反馈/投稿：editors@cn.infoq.com

# 推荐编辑 | 马德奎

大家好，我是马德奎，很高兴成为本期《架构师》的推荐编辑，这是对我的莫大鼓励。与InfoQ的接触由来已久，好多次“百度一下”都把我带到这里；有段时间研究ESB，我还在InfoQ上读过ESB专题的多篇文章，深知它的专业性和严谨性。一次Q聊中，同窗老友臧秀涛问我是否愿意加入InfoQ，才知道原来只要你热衷于IT技术且有较好的专业英语及文字功底，就有机会加入这个严谨高效且宽松的团队。从业四年来，虽然我一直保持着对技术的热情，也阅读了大量英文技术文档，但敲开InfoQ这扇门并没有想象中那么容易。试译段落看上去不难，但第一次完成800字的翻译却用了一晚上，深叹“读懂”与“译成汉语”之间的巨大差距。译文提交一天后，收到反馈，试译评级为A+，于是正式成为了InfoQ的编辑。在此，特别感谢老友的建议和帮助。 经过仔细翻译及反复斟酌，我提交了第一篇译文的校核申请。InfoQ高级策划编辑杨赛对我的译文进行了审校，内容涉及标点符号、术语准确性和语句流畅性等方面，批注非常仔细，有原因有建议，甚至还提供了参考资料，评语平和有说服力，使我颇受鼓励。值得提出的是，InfoQ的每位编辑都十分认真负责、谦虚严谨，大家都本着对内容负责的态度坦诚交流讨论。若在翻译过程中遇到难题，还可以到“InfoQ好译者”群里寻求帮助，大家都非常热心。现在阅读和翻译InfoQ新闻成了我茶余饭后的必要活动。一来可以呼吸InfoQ带来的业界新空气，开阔视野，增长知识，二来可以为业界前沿资讯的传播助力。

即使是屌丝盛行的时代也不能让头脑也“屌丝”，学如逆水行舟，不进则退，尤其是在技术瞬息万变IT行业，要想立足，就要时刻关注前沿动态，把握时代脉搏。我一直秉持着这样的态度来不断地提高自我，而InfoQ提供了一个很好的平台。如果读者朋友和我有同样的看法，那么现在就开始关注InfoQ吧!我深信，包括我在内的每一位同仁都在用最饱满的热情浇灌着InfoQ这棵大树，同时我也不断从中汲取养料，希望可以贡献更大的力量。期待着InfoQ更加枝繁叶茂！

如果你也有意成为InfoQ大家庭的一员，请邮件至：editors@cn.infoq.com。也欢迎大家与我（新浪微博@ConfuciusSaid）交流。

# 封面植物

## 蔷薇

蔷薇（学名：Rosa multiflora），又称野蔷薇，蔷薇科植物，主要分布在北半球温带、亚热带及热带山区等地区。蔷薇花是一种蔓藤爬篱笆的小花，花盘环绕萼筒口部，有白色、黄色等多种颜色，叶互生，奇数羽状复叶，其耐寒，有野生的，可用压条或嫁接法繁殖等种植方法。蔷薇可以药用，由其制成的蔷薇花粥有良好的营养价值。其不同颜色有不同花语，如红蔷薇代表热恋； 粉蔷薇代表爱的誓言。全属约有200种，广泛分布亚、欧、北非、北美各洲寒温带至亚热带地区。我国产82种。生长习性蔷薇喜阳光，亦耐半阴，较耐寒，在中国北方大部分地区都能露地越冬。对土壤要求不严，耐干旱，耐瘠薄，但栽植在土层深厚、疏松、肥沃湿润而又排水通畅的土壤中则生长更好，也可在粘重土壤上正常生长。不耐水湿，忌积水。新株定植时要施入腐熟有机肥。霜植后头一二年可于每年深秋开沟施一次基肥，以利生长和开花。萌蘖性强，耐修剪，抗污染。花期一般为每年的4-9月，次序开放，可达半年之久，由于温室效应而导致全球变暖，某些地方的蔷薇提早在4月，甚至是3月份便开始开花。中国人在文学作品和日常生活中、口语中的蔷薇，一般指黄蔷薇和野蔷薇。严格的说来，蔷薇属除少数玫瑰及月季外都称作蔷薇，而每一种蔷薇前面都有形容词，如矮蔷薇、藏边蔷薇、多腺小叶蔷薇等。此属品种全世界约有200种，品种繁多，变异极大，有许多变种及园艺栽培种，多产北半球温带、亚热带及热带山区。

1kg.org 多背一公斤

爱自然 | 更爱孩子

# EMBARCADERO:
## MULTI-DEVICE APP DEVELOPMENT

Delphi, C++

FM³ Framework

Future

# 架构师 8 月刊

**每月 8 日出版**

本期主编：水羽哲

美术/流程编辑：水羽哲

顾问： 杨赛

发行人：霍泰稳

读者反馈：editors@cn.infoq.com

投稿：editors@cn.infoq.com

InfoQ 中文站新浪微博 :http://weibo.com/infoqchina

商务合作：sales@cn.infoq.com

本期主编：水羽哲，InfoQ 中文站内容运营编辑

爱动漫（火影忍者、海贼王）、爱代码（C#、Python）、爱读书（冰与火之歌、德川家康），信条：沉默亦诚恳；

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。