促进软件开发领域知识与创新的传播

架构师

7月 ARCHITECT

特别专题

指尖上的世界

移动用户界面一瞥

使用ASP.NET MVC开发混合移动应用

让1.5亿移动端用户第一时间获取消息

Esri

2013 Esri空间信息技术开发者大会

Hadoop在GIS数据处理中的运用

基于ArchGIS的Javascript API

如何从组件开始构建一座城市?

ThoughtWorks技术雷达

管理技术债

InfoQueue
Enterprise Software Development Community

每月8号出版

卷首语

取舍

架构师应该具备什么样的技能？答案很多，不过有一条是公认的技能，即取舍的能力。做系统架构要平衡硬件和系统的资源，做软件架构要决定是优先改进性能、安全性还是添加新的特性。在重要的事情和紧急的事情之间要形成平衡是很难的事情，因此能够形成好的优先级排序并执行、反复调整优先级排序并持续执行，是非常难得的能力。

本期《架构师》的人物专栏介绍了ThoughtWorks的全球CEO郭晓。跟很多技术公司的管理层一样，郭晓也是从工程师开始做，在职业发展的过程中转变为管理的岗位。这样的转变一方面意味着逐渐远离技术，另一方面意味着学习另一个领域的技能，究竟是得到的多还是失去的多，对于每个人都是不同的。有意思的是，很多人走过了这条路之后都会感慨，做之前总觉得自己做不成，但真正用心去做了，会发现人的潜力是无穷的。当然，即使努力了也没做成，至少你也有收获，因为你知道自己以后可以避开这个坑。按郭晓的话说，alternatives exclude，所以一旦选择了一条路，就要认真走。

做人和做技术的道理其实是相通的。本期的观点专栏涉及到两个争论，一个有关REST，一个有关noBackend。好的REST设计是怎样的？noBackend有哪些好处？其实从应用的角度来说，没有好或不好，只有合适不合适，是否与团队、开发方法、遗留系统能够进行较好的匹配。比如，在最近的一个虚拟座谈会上，InfoQ编辑李锟在针对RESTful API设计的讨论中引用了这样一个事实：

“按照HTTP协议的规定，GET方法是安全的且幂等的，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方法都是不安全但是幂等的。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。”

如果对各个方面都了解的通透，在做取舍时才能够做到恰如其分。架构师就是要有这样的追求。

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

ThoughtWorks全球CEO郭晓谈软件人才的招聘与培养

观点 | View

REST的缺点是什么？
不用后端：noBackend是什么？有什么好处？

本期专题：指尖上的世界 | Topic

设计指尖上的世界：移动用户界面一瞥
使用ASP.NET MVC开发混合移动应用
让1.5亿移动端用户第一时间获取消息

推荐文章 | Article

如何从组件开始构建一座城市？
ThoughtWorks技术雷达（2013年5月）
管理技术债

特别专栏 | Column

2013 Esri空间信息技术开发者大会：打造可持续的开发者平台
Esri卢萌谈Hadoop在GIS数据处理中的运用
Esri何超谈基于ArchGIS的 Javascript API

避开那些坑 | Void

绕开 Android 应用开发的那些“坑”
影响可扩展性的十宗罪

新品推荐 | Product

Google 发布新一代Web UI库Polymer
Facebook发布用户界面库React，业界褒贬不一
Yahoo!开源运行在Hadoop上的Storm——Storm-YARN
PhoneGap 2.7发布，官方弃用1.9版本
Visual Studio 2013浮出水面
Entity Framework 6的重大变革
Oracle官方发布Java EE 7和Glassfish 4

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

ThoughtWorks全球CEO郭晓谈软件人才的招聘与培养

作者 杨赛

2013年4月18日，Trevor Mather宣布将在6月卸任ThoughtWorks全球CEO一职，同时正式任命由四位全球领袖联席组成的全新集体领导层。这四位联席主席分别为：郭晓——首席执行官；Chris Murphy——首席战略官；Sudhir Tiwari，Craig Gorsline——首席运营官。其中，郭晓在就任全球CEO之前，曾任ThoughtWorks中国区总经理，从零开始建立了ThoughtWorks的中国区业务，并负责ThoughtWorks在中国市场的业务开展。同时，ThoughtWorks中国区总经理一职由原中国区咨询总监、《精益软件度量》作者张松及原西安分公司负责人胡凯联席担任。

在ThoughtWorks成立20周年之际的这次人事交接之后，ThoughtWorks在业务发展和人才培养这方面会有哪些新的尝试？InfoQ于近日采访了郭晓，将相关信息整理分享如下。

InfoQ：ThoughtWorks下一步对人才引入和培养的计划是怎样的？比如在亚太区，现在成都、上海、新加坡、西安都有团队，不同地区的招聘定位有怎样的不同？

> 郭晓：从招聘的标准来讲，是没有任何区别的。我们招聘的标准有三个词：态度、能力和诚信。这个人不但能力要强，我们对他做事的态度，与同事合作的方式、方法，还有他个人的做事的准则都比较关注。
>
> 地区之间的差异肯定是有的，不同的城市的人才体系和为客户提供的服务会略有不同。成本上也有差异，比如中国的二线城市、印度、除了南非以外的非洲，成本优势比较大。
>
> 此外，我们更关注的一点，在于包容性，或者说多样性。关注包容性的原因有两个方面：第一，一个组织要想维持不断创新的能力和文化，其实最关键的一点是思路的多样性。如果一个组织的人同质化严重，全是一种思路、一种技能，那么它很快会失去创新的能力。对我们来讲，如何招到跟自己理念相近

——对卓越软件的追求和对社会公正的追求，同时在技能和思考方式上跟我们不同的人才，是一个挑战。

第二，当我们倡导社会公正的时候，我们发现自己进入了一个误区：如果我们招的人都是IT行业里面最优秀的人才，我们实际上不是在为行业做贡献，因为这是一个抽血的过程。尤其在不发达地区，我们所面临的挑战不是去找到已经非常成熟、技能非常优秀的人才，而是怎样去建立更好的人才培养体制。这个挑战在非洲就非常明显。很多国际上比较知名的大公司在非洲拓展业务的时候，从来没有招过一个本地的程序员；而我们想做的事情就是在非洲多招本地的程序员。这个挑战跟我们的业务模式是匹配的，因为我们的业务是高端业务模式的定制，我们需要有值得培养的人才来配合我们弹性的体系。每个新人加入公司，都是N带一的模式，每个人都能从周围人身上学到不同的技能和知识。

InfoQ：刚才说的是地域划分，那么对于领域方向又是怎样的规划？比如，大家在国内一直觉得ThoughtWorks是敏捷领域的代表，而现在很多企业的需求都在新的领域，比如移动互联网。ThoughtWorks在类似移动互联网的新领域的创新是怎样的一个情况？

郭晓：敏捷跟我们过去做的很多工作是有关系的，但长远来讲，这并不是我们唯一的领域。我们有一个叫做《技术雷达》的半年度报告，在很多中国以外的地区，人们想到ThoughtWorks的第一反应不再是敏捷，而是这个《技术雷达》。《技术雷达》是我们以自己的实践提炼出来的总结，包括在什么时候使用什么技术平台、方法论、编程语言等等。

在移动互联网这个领域，我们做的很多事情集中在几个方面：

第一，用户体验。移动平台跟PC最大的不同在于，它不是以功能为导向，而是以体验为导向。我们在用户体验能力上跟以前相比有很大的变化，以前我们很少有设计专家，而现在我们全球的设计团队已经有七八十人。

这其中包含一个问题，就是怎么和云计算接轨。更多的客户对移动平台的要求，在于在任何时间、任何地点都提供这种场景体验：在办公室里能做的事情，在回家的路上也可以做，到家后还能做。这里面的数据，业务逻辑，如何通过云平台来实现这种接轨，也是体验方面的一个方向。

第二，跨平台兼容。前一段时间，HTML5显然输了一仗，因为Facebook决定放弃HTML5，转而采用iOS的Native开发方式。但是长远来看，Native将会成为赢家么？也很难说，因为现在跨平台遇到了成本问题。大部分情况下，在Android、iOS和Windows Phone这三个方向做足够的投入去实现用户功能，在三个平台之间不断独立发展所产生的重叠，会在未来持续的维护和升级上遇到

很多挑战。

HTML5是解决这个问题的最好办法，但是它在体验上遇到了挑战。我们有一个开源软件叫Calatrava，它通过这样一种方式去解决这个问题：这个平台提供了很薄的一个显示层，或者叫界面层，能够利用本地的功能来实现展示；下面有一个很厚的业务逻辑层，通过JavaScript实现三个平台上的共享。这样就解决了Native体验和重用之间的一个平衡。这不是唯一的解决方案，但这是一个方向。

InfoQ：前一阵子在圈内有一个讨论，就是一个应届毕业生同时拿到了百度和ThoughtWorks的offer，在微博上问去哪里好，@左耳朵耗子 就建议他去一个能够实实在在学到东西的地方，而认为ThoughtWorks比较虚一些。您对于这个观点是怎样看的？

郭晓：在百度和ThoughtWorks这样的公司里面能够学到什么东西？从一个互联网公司的角度来讲，所做的技术和业务一定有自己重视的方向。我想，一个人如果机会好的话，无论在百度，在ThoughtWorks，还是在其他公司，都有可能学到很多东西；但机会不好的话，也可能挣扎很长时间都学不到东西。我们公司有很多人是从Google跳槽过来的，我想有一定的可比性。

同样，即使在一个很好的大环境里面，内部也有微环境，在不同的微环境里能学到什么东西是很难讲的。很多类似我们这样的公司，尤其是比较大的，往往会细分领域，比如你做策划，你做Oracle，你做SAP，你做Java，你做零售业务线，你做金融业务线等等。在ThoughtWorks，我们崇尚全才优于偏才。比如扁平化管理，我们不以业务线区分人才梯队，而是让每个人掌握不同技术领域的技术，做不同业务领域的项目，这样才能够有更广的视野。我们相信这样才能够培养出更好的人才，更好的程序员，更好的架构师。

在软件领域，我觉得最主要的一个技能就是模式识别。一个好的程序员、架构师，能够迅速的识别出一个问题背后有怎样的模式，从而解决它。这个模式识别的能力，一定来源于做过不同的项目，研究过不同的架构和语言体系。这就是为什么我们不去做人才的区分。

作为咨询公司，我们既和互联网公司合作，也和保险公司合作，也和物流公司合作。只要你有足够的耐心，就有机会接触到不同的技术平台和不同的业务领域。敏捷咨询只是我们很小的一块业务，我们大部分业务是研发，即定制软件开发。这个东西是很实在的。应届毕业生进来，我们很少派他们出去做敏捷咨询，这样不现实。我们会放他们在不同的项目上锻炼。

由于我们ThoughtWorks对人才是这样的要求，如果一个毕业生的目标是迅速的成为某个领域的专家，比如互联网领域的专家，那我们这个环境就不适合他。

> 但是，如果他的态度是希望接触到不同的平台和不同的客户，希望获得更多的视野，那我们这里就是不错的平台。

InfoQ：您对架构师的看法跟我之前听到的看法都挺不一样的。一般大家会认为架构师是T型人才，既有广度，又要在某一方面有深度。

> 郭晓：深和浅是相对的，每个领域都浅尝辄止肯定是不行的。我们的技术雷达背后的团队叫做TAB（ThoughtWorks Technology Advisory Board，即技术委员会），这些人都不是T型的，而是一条横线下面很多条竖线。如果有两个人，他们的水平线都是相同的广度，但一个人在三个不同领域都有相对不错的深度，而另一个人则是在一个领域有很深的深度，那我们认为前者会是比后者更为优秀的架构师。

原文链接：**http://www.infoq.com/cn/news/2013/06/tw-guoxiao-on-talents**

相关内容

- ThoughtWorks技术雷达（2013年5月）
- ThoughtWorks CTO Rebecca Parsons专访
- 持续交付加速了创新的步伐
- 全功能团队 - 数据篇
- 物理墙和虚拟墙之争

观点 | View

REST的缺点是什么？

作者 **Mark Little**，译者 马国耀

几年前，Ganesh Prasad问道，Internet比REST更基本吗？这些年，他不断围绕REST SOA、以及更时新的云计算提出相关讨论，并且钟情于REST的指导原则。然而，最近有人在LinkedIn REST架构师讨论组中的一片帖子中问道，“REST的缺点是什么？”Ganesh对此做了回复，然后又在其个人博客中重申了自己的观点：

> 我不能说REST有“缺点”。它说到的都做到了，而且做得很好。但是，请记住REST架构的实现唯一使用的协议是HTTP。我们可以肯定地想象REST在另一种传输协议上的实现，而且其中还包含诸多方面的增强。

接着，他谈到了四个可能改进的方面。值得一提的是，Ganesh和许多人一样，将REST等同于REST/HTTP，即基于HTTP的REST：

1. "HTTP是一个同步的请求响应式协议。这意味着它不能天然地支持服务端发起的通知（点对点），但这又是经常需要的。这就解释了为什么REST风格的应用程序中的回调需要使用应用层的设计模式，如Webhooks。现在，我们有了WebSockets这样的双向传输协议，也许工业界应该考虑在其之上根据REST原则设计一款新的应用协议。" 这一点非常有意思，因为就在差不多一年之前，我们看到有些人甚至在探讨WebSockets和REST是否能够兼容的问题。
2. Ganesh认为，REST社区从Web Service协议栈中至少能够学到一些东西："这些协议都是基于核心的SOAP和WS-Addressing消息机制能力之上的端到端的协议"。过去，也有人提到过类似的建议。Ganesh接着做了一个类比，他说Web Service之于WS-Addressing、WS-ReliableMessaging、WS-SecureConversation和WS-Policy好比Internet之于TCP、IP和IPSec等。Ganesh认为，REST的应用幂等性可以更好地实现可靠性（尽管并没有提到如何更好），而且可以找到REST事务的替代方案，但是他仍然留下了一个问题：

> 剩下的问题是安全。WS-SecureConversation和WS-Security是可路由

> 的，它们不同于SSL/TLS，而后者却是REST唯一的安全机制。有了WS-Sec*，就可以对消息进行部分加密，留下一些明文用于基于内容的路由和分支切换。这是REST缺乏对应的优秀方案的地方。SSL是点对点的，不受代理检测，并且违反了REST原则。REST对此只能忍受。

这一说法也非常有意思，我们也看到了有的人提出了一些想法，如Resteasy的Bill Burke，他提议，REST需要更好地实现安全的方法。

Ganesh继续谈到了REST对一般性QoS的问题:

> REST不能支持这些一般性QoS背后的原因是，它们都需要维护“会话状态”。有状态（statefulness）存在明显的缺陷（如，影响可伸缩性和错误恢复），但是随着Redis之类的宣称常数时间（即O(1)）和高性能的NoSQL存储的出现，就有可能将会话状态从内存中转移到这样的存储中，因而可支持在多个节点间共享会话，进而可以满足QoS的需要。

回到Ganesh谈到的另外两点：

3. 他认为，HTTP的动词太少，尤其对于需要做点对点交互的场景。他建议："INCLUDE（将资源加入到某个资源集合中并返回服务端设定的URI）、PLACE（使用客户端指定的URI向资源集合中添加资源）、REPLACE（全部替换）、FORCE（PLACE或REPLACE）、AMEND（部分更新，它是一个容器动词，为一个资源子集指定一个或多个动词）、MERGE（通过提供的表述合并部分资源）、RETIRE（比DELETE更好）和SOLICIT（替换GET，它也是一个容器动词，用于告知响应端对发起方的资源做些什么，因为现在是点对点的场景了）"
4. "HTTP合并了应用层和传输层的状态码（如，304 Not Modified[未修改]和400 Bad Request[无效请求]；407 Proxy Authentication Required[需要代理认证]和502 Bad Gateway[无效网关]）。 REST在另一个协议上的实现应该更清晰地隔离应用层协议和传输层协议。HTTP承担了双份工作，因此其结果也往往不尽完美。"

参照一下HTTP 2.0的初始草案，就能发现它不可能完全采纳Ganesh的建议。然而，Ganesh本人在过去5年里好像一直致力于提出一个新规约：

> 我正在编写一个称作Internet Draft的新应用协议，它可以绑定任何传输（发布/订阅、异步点对点或同步请求/响应）。该协议是被我称为ROMA（面向资源/表述的消息传输架构）的新型分布式计算架构的一部

> 分，它不仅包含数据模型和消息传输API，还包含更高级的功能（QoS、描述和流程协作）

该规约发布之后，如何解决这些已知的问题？是否能够获得REST社区的积极响应？将是很有意思的看点。

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

观点 | View

不用后端：noBackend是什么？有什么好处？

作者 李永伦

在近期举办的前端趋势2013大会上，Gregor Martynus做了一场题为“各位快看，不用后端！”的报告，站在前端的视角谈应用开发，服务端组件只退化到实现浏览器尚不支持的特性，这种方式与传统开发方式截然相反，后者首先要关注应用服务端，然后再利用前端技巧来提升应用。他们还建了一个名为noBackend的网站来深入传播这种理念。

InfoQ在会上对Gregor进行了采访，对noBackend进行了更深入的了解。

同时，阿里UED团队的王涛也针对这个报告进行了介绍，并认为：

> noBackend的开发模式其实对前端同学提出了更高的开发能力要求，当后端的模板层完全撤去，只剩下纯净的数据接口时，意味着js将负责更多的业务逻辑处理，代码的组织架构需要有更好的设计。责任大了，压力也自然不会小，据闻腾讯盛行这种开发模式，而其配有500+的前端开发队伍，也就完全可以理解了。

王涛认为，noBackend将成为前端的重要趋势。同时，InfoQ编辑黄玲艳（@hellena）也表示，noBackend已经是在前端领域被广泛讨论的趋势，是当前行业前沿关注的一个重要方向。

针对这个话题，InfoQ对王涛进行了更进一步的接触，了解noBackend究竟是什么，以及该趋势目前的扩散状态。

InfoQ：你觉得noBackend是什么？noBackend不是什么？

> 王涛：我觉得noBackend并非是一种新技术或是一种新架构，而是一种新的web开发模式。我们经历过前后端分离的开发阶段，就是后端同学提供数据接口，让前端同学调用来实现页面逻辑，其实在这个阶段，就已经奠定了

noBackend的架构基础了。

但如何增加后台接口的通用性，甚至可以做到和具体业务逻辑无关，能够让前端具有可按照业务逻辑的需求来调用和组装接口的能力，则是noBackend开发模式中对后台服务平台提出的要求。也是从前端的角度看到“noBackend”的关键所在。

业务开发的重担会落在前端层面，而多条产品线，甚至整个公司的产品都可以共用一套后台接口服务平台。

InfoQ：你有没有项目用到noBackend？或者你是否知道正在使用noBackend的项目？

王涛：我在个人的一个移动小项目中曾简单尝试过noBackend的开发模式，使用了Titanium Cloud Services，当时对这种新的开发模式的触动还是蛮大的，noBackend虽然因移动app开发而出现，但我觉得他完全适用于很多桌面应用。也相信国内的开放平台也会逐步的提供类似的服务平台。

InfoQ：这些项目从noBackend中得到什么实在的好处？

王涛：从前端的角度看好处会很明显，你不需要关心任何服务器配置和搭建、不需要使用sql创建任何的数据库、也不需要用后端语言开发任何的服务API。你可以把关注度聚焦在业务层面的实现上，以最小的成本完成业务的构思和需求，并且建立在一套可扩展，高性能的后台底层架构之上。

InfoQ：什么样的项目不适合使用noBackend？

王涛：由于noBackend模式下搭建的平台接口通用性是第一优先级的，所以有着非常特殊需求的应用在使用中会遇到问题。比如登录和注册的通用接口，如果某应用对此有着特殊需求，而通用接口又不希望支持该特性，那么就需求开发个性化的接口。 但是接口一旦开发完成，又会进入到“noBackend”的良性循环中。

InfoQ：noBackend目前在国内的状况如何？

王涛：我觉得国内的各大公司不乏前后端分离开发模式的实践，但是能够进一步升华，成为“noBackend”开发模式的公司还比较少见。

> 这也是受国内人才的现状影响，因为在“noBackend”开发模式下，对前端开发人员的要求和数量会提升，而目前并不具备这样的人才结构，推动这种开发模式的转变会有阻力。但这种快速开发App的需求是存在的，所以相信国内也会出现类似Titanium Cloud Services这样的开放平台。

InfoQ：你是否认为**noBackend**会是今后的趋势？为什么？

> 王涛：能否成为趋势关键还是取决于喜欢这种开发模式的人有多少，我相信这些人更多的属于web前端开发人员。而目前来看，这些人员的整体数量和能力都需要一个成长的过程，所以短时间内还是比较难成为趋势的。但长期来看，这种开发模式对整体效率的提升是有利的，相信会有更多的人会发现并推动这种模式的发展。

原文链接：**http://www.infoq.com/cn/news/2013/06/wangtao-on-nobackend**

相关内容

- Web风格起过作用吗？
- WSO2 API Manager 1.4.0 引入多租户
- Esri何超谈基于ArchGIS的 Javascript API
- 京东云存储服务和应用探索
- Mobile离WebApp还有多远

JavaOne™

2013 年 7 月 22 - 25 日 | 上海

立即注册
7月21日前注册
节省500元

注册网址: oracle.com/cn/javaone2013

ORACLE

版权所有 © 2013, 甲骨文和/或其关联公司。保留所有权利。Oracle 和 Java 是甲骨文和/或其关联公司的注册商标。

推荐语

设计指尖上的世界

PC时代的软件以功能取胜，移动时代的软件以体验取胜。这是目前在做移动应用的很多开发者都感受到的一个现实。

6月百度技术沙龙的话题是前端开发，同时也是InfoQ编辑的聚会。我们在聚会上邀请了前人人网前端负责人闫强一起过来玩，聊到前端职业发展这个话题的时候，他感慨道：前端这个领域很少有科班出身的。他自己的经历很“狂野”：攒过机器，修过笔记本，开过黑车，做过面馆，后来到做设计、Flash开发，到写HTML、JS，完全是一路闯过来的。

在人人网，前端最早是属于产品部门的。在他看来，这是件羞于启齿的事情。后来，前端独立了出来。现在，前端能够给后端提意见，因为他们既懂设计，又懂产品，也懂技术。另一方面，百度的前端则提出了F.I.S.项目，项目负责人张云龙称其为“前端领域工业化”，提倡前端的自动化。不管从哪个角度来看，前端正在变成一个产业，这个产业涉及到设计、产品、技术，并越来越往工程化的方向前进。

归根结底，跟电脑相关的设计都可以追溯到一个学科：人机交互。没有人可以知道在五年之后，指尖上的世界会变成什么样。但可以肯定的是，好的体验不再可有可无，因为用户对它具有非常高的期望！

本期专题：指尖上的世界 | Topic

设计指尖上的世界：移动用户界面一瞥

作者 Forrest Shull ，译者 李彬

本文首先发表在IEEE软件杂志，现由InfoQ及IEEE计算机协会为您献上。

在计算机与软件领域，鲜有进步能够像智能终端的发展一样显著地影响着我们的世界。对于不断运行的各种任务和服务，消费者们似乎拥有无穷无尽的热情。同时，新应用涌现的速度也令人印象深刻。如果最近经常旅行的话，任何一个人在任何一个候机室观察自己周围的同路人，都肯定能够发现这样的场景：他们在使用移动服务，不仅在处理邮件，还在进行购物、支付、看电影，甚至是在进行抵押贷款。（有传言说，其中一些设备甚至在拨打电话）。

人机交互

我们能够取得现在的研究成果，与制造商和应用开发者在易用性方面所取得的成就是紧密相关的，包括快速的响应时间、直观的界面、精心设计的功能等方面。为了探寻这是如何产生的，我最近与Ben Sheiderman和Ben Bederson进行了交流，他们都曾是马里兰大学人机交互实验室（HCIL）的主管。该实验室是美国最早致力于人机交互领域的研究机构。在建设这个世界的过程中，人机交互领域扮演了意义重大的角色，即便它成功地将界面变得如此直观和透明，以至于我们往往不会明确地注意到它。Sheinderman这样说道：“HCI的研究员们对于帮助设计移动应用界面感到自豪，它使得50亿人能够受益于通讯服务，从而改善他们的生活、促进贸易、提升教育，并在需要的时候提供医疗服务。”

HCIL以能够与各种各样的干系人共事为荣，这里所谓干系人不仅包括开发者和移动用户，还包括政策制定者和其他自认不是技术人员的人群。Ben Bederson同意回答我的一些关于当代移动应用开发方面的关键问题——提问从一些背景知识开始：为何设计良好的移动应用界面，与设计良好的桌面应用截然不同。

在他看来，尽管目前在许多重要方法中，为移动设备进行开发实际上还没有从针对“传统”设备的开发中脱离出来，但二者还是有许多不同。移动应用必须面对其天生的资源限制——很小的屏幕使用空间、小容量内存、较弱的计算能力、有限的带宽等等。但这里有两个重要的因素来缓解有限资源与开发需求间的矛盾。首先，尽管资源是有限的，但一般而言用户利用设备执行的任务（注：原文所说的任务，是指用户使用移动设备完成的一项用户事件）也是相当轻量级的——例如查看天气或股价。这些应用倾向于避免那些复杂的或多步骤的活动，因为它们难以在资源有限的设备上运行得很好。

第二个缓解因素是，移动设备内建了大量的传感器并能够轻松地通过它们获取数据，因此开发者可利用很多信息来提升用户体验：例如，根据设备的能力，移动应用或许能够判断设备的朝向、剩余的电量，或是用户位置（通过GPS定位）。此外，也不难想象，通过使用设备上的摄像头获取图像的方式，应用能够对用户环境进行有依据的推测（例如，基于设备所面对的墙的材质）。

要求

由于在移动设备上的用户任务具有相对短小且分散的特点，用户一般不愿投入大量的时间来掌握如何完成它们。Bederson指出，由于PC应用的用户已经习惯面对一条学习曲线，他们期望完成的任务集一般比移动用户所面对的复杂许多，而且这些任务集天然需要时间上的投入。例如，如果我刚刚购买了一个软件产品，使用它帮助我处理税务问题，我一般不指望自己在刚刚使用时就能对它完全掌握。但如果一个用户任务仅需要一两分钟即可完成，那么克服学习曲线就不值得了——用户交互必须要非常直观，才能让用户认为任务是值得去做的。

这些观测结果引导Bederson提出了四条用户界面（UI）原则，它们对创造良好的用户界面至关重要。移动应用应该是：

- 普遍可用的
- 有价值的
- 高效的
- 有吸引力的

显然许多开发者已将这些放在心里。回想五年前我使用的手机——一个笨重的设备，配以笨拙、难用的基于文本的界面——我向Bederson询问，在这么短的时间内，UI何以能够有如此大的改善。他将其部分归结于在针对更好的UI技术的研究，部分归结于产品测试。但他也感到，这是一个很好的例子，能够说明有多少用户的反响实际上是来自于事物而非应用程序接口。例如，与PalmPilot（一个曾受大量用户喜爱的早期移动设备）与苹果的Newton（并未取得商业成功）：两个设备捆绑了相似的应用，并且实际上采用了相似的界面，

但Newton采用了大量的细节设计决断，实际上这些细节设计决断组合在一起打造了完全不同的用户体验。

未来

当我追问Bederson在接下来五年中，我们可以期待哪些UI领域革命性的改进，他非常谨慎的表示："要是我知道这个答案，我现在就已经在着手进行这些改进了。"他指出，虽然许多移动设备和应用已经明显在直观可用性方面达到了相当高的水平，但是，显然在未来仍旧可能诞生革命性的改进。例如，当用户使用iPhone编写短信或邮件的过程中，如果需要修正错误，用户会发现界面非常难用。这虽属细节，但却是用户每天都需要反复面对的问题（并非巧合的是，这个问题正在研究之中，并且Bederson预计很快就能解决）。其他当前正在研究的领域包括多设备、隐私与安全，以及声音。

多设备

我们大部分人拥有多个移动设备，它们往往来自不同制造商。我们不仅每天使用它们，还希望能够在其间无缝切换。我们应该能够从当前手边任何设备访问自己的日程表，此外通常当我们离开家或办公室的时候，也希望能够使用其他朋友的设备并继续工作。现在，用户可以在设备间同步他们的工作，但这一操作并不非常便捷。让多设备间的切换成为无缝、易操作的任务，将成为一个可能的重要改进领域。

隐私和安全

尽管实际上大量用户越来越注重隐私和安全问题，但该领域中已经做到的工作还不多。它还没有成为制造商们用来在竞争中脱颖而出的手段。但这是一个我们有可能在未来看到许多工作和进展的领域；新方法或许会采用某种形式保护在不同应用间使用的个人用户数据，又或许会开发一些特性——例如移动设备上的访客模式——可以让用户与朋友们分享硬件又不至于将他们的数据置入危险境地。

语音

Google和苹果的最新设备都已经很好地集成了语音输入并终于开始得到广泛使用。但即使近年来语音技术的质量得到了惊人的提升，Bederson相信这一技术还需要一到两处改进，才能够进一步流行起来。当语音识别在处理背景噪音和用户口音方面的可靠性能够达到键盘录入的水平时，它将具有更广泛的吸引力。此外，当诸如苹果Siri这样的问答系统与更多的数据源对接并能够回答更宽

泛的问询集时，我们就可以预见到更广阔的应用场景。尽管如此，即使这些成为现实，Bederson认为键盘录入也并不会消失，因为还有许多场景需要使用键盘录入，至少在某些时候你可能会希望输入一些隐私数据。

其他问题

当然，我们应该期望有更多关于理解和挖掘“触碰设计”以及更好的手势的研究——本质上，这意味着用更直观的方式与我们的设备进行互动。现在正在涌现许多非常创新的界面，而Bederson期望这一势头能够保持下去。

挑战

市场上有着为数众多的设备，我很好奇当开发者需要让他们的软件在数十款（姑且假定不是数百款）不同的设备上运行的时候，会感到多么头痛。在这样的多元空间里，开发者怎样进行测试？Bederson将其更多地视作工程难题而不是UI难题。开发者和测试者往往会针对市场上三到五款主要平台处理问题（每一个平台依赖不同的编程语言），而不是操心数十款设备。最后，解决这个问题离不开好的工程实践：与许多软件工程难题的处理方法类似，处理这个问题需要易于表达和理解的解决方案（例如代码模块化、使用整洁的设计将平台相关代码与其他代码分离），但这样的解决方案需要开发者投入大量的时间和精力。

最后，尽管好的UI正在变得无处不在，我还是询问Bederson：从他的专业眼光来看，是否发现了一些开发者们不断出错的常见问题。他说，一个常见的出问题的领域，是应用具有太多功能或过于复杂的界面。贯穿本次讨论的一个主题是保持简单——不要过度使用小尺寸屏幕，让用户快速完成一个具体任务。在这个环境中，复杂度对用户而言实在是一件糟糕的事情。

Bederson指出的另一个问题是应用之间或内部的不一致。不论开发者如何努力使应用保持简单，在应用程序中难免会出现一定程度的复杂性，而用户必须在一个非常小的屏幕上对其进行处理。设计和布局上的一致性能够帮助用户快速掌控复杂性，并使其直观的操作移动设备。尽管在理论上这一点很明显，但实际趋势恰好相反：大量开发者和供应商一直努力寻找新奇的、出乎意料的界面设计，以使自己在市场竞争中脱颖而出。

Bederson以一个由来已久的问题作为讨论的结尾：对设计原则缺乏重视。但好的设计不再是备选方案，因为用户对它具有非常高的期望！

尽管有这些困难，移动设备上无处不在的易用和直观的应用，是隐藏在视线之外的成功故事——因为应用自身的良好设计，我们经常将一项惊人的技术成就视作理所当然的。我认为Ben Sneiderman的另一段陈述适合于作为结尾，来描

述这项工作："移动应用强制要求设计师深入理解用户需求，从而通过易于学习和使用的界面提供恰到好处的功能。这些界面工作横跨语言和文化，适用于年轻人和老人，适用于专家和新手，也适用于各种不同类型的残疾用户。这是一个突出的成就——它是改变世界的伟大成功。"

Forest Shull是马里兰Fraunhofer中心软件工程实验方面的高级科学家，在这个非营利性研究和技术转让组织中，他领导测量和知识管理部门。他也是马里兰大学帕克分校的兼职副教授，同时还是IEEE软件的主编。可以通过fshull@computer.org联系他。

本文最早发表在IEEE软件杂志。IEEE软件的任务是构建领先的社区和未来的软件从业者。杂志提供可靠的、有用的、前沿的软件开发信息，使工程师和管理者能够跟上日新月异的科技变革。

原文链接：http://www.infoq.com/cn/articles/designing-a-world-at-your-fingertips-mobile-user-interfaces

相关内容

- FaceUI创始人访谈：移动应用的用户体验
- MobileCloud for TFS支持测试Windows Phone，Android，iOS及BlackBerry应用
- 让1.5亿移动端用户第一时间获取消息
- 移动应用开发者违反隐私法律
- Facebook发布用户界面库React，业界褒贬不一

本期专题：指尖上的世界 | Topic

使用ASP.NET MVC开发混合移动应用

作者 Daniel Jebaraj ，译者 邵思华

移动设备无处不在

十年前，微软专业开发者大会（PDC）上，主办方播放的一段视频，勾画了未来的移动领域蓝图。视频演示了一个颇具未来感的Windows手机设备，如何完成诸如定位最近诊所的任务。在那个Palm VII算是最接近智能手机的年代，这段视频所演示的未来确实令人向往。

时间飞驰到2013年：我们毫无偏差地来到了这个未来时代，智能手机和其它移动设备无处不在。各种价位任君选择，而且价格也越来越亲民。事实上，对于大多数发展中国家的人来说，他们唯一的电脑就是所拥有的智能手机。

移动应用开发：Gartner所预测的增长

Gartner预测[i]，到2016年，至少50%以上的企业email用户将会主要依赖于浏览器、平板电脑、或移动客户端，而不是桌面客户端。据移动设备的使用增长状况可预计，在今后的几年中，针对这些设备的软件应用开发也会飞速增长。Garnter继续预测，到2015年，针对智能手机及平板电脑的移动应用开发项目与原生PC项目之比将达到4：1。Garnter还预测，在今后的4年中，智能飞机与平板电脑在新设备中所占比重将会超过90%。

Apple的应用商店现已有超过50万的应用，Android也接近这个数字，而新贵Windows Phone近期也超过了5万，并且还在快速发展中。

移动应用行业：市场分化带来的挑战

在这个令人振奋的背景下，我们能够确信，不久的将来移动平台上将出现大量行业应用。与机遇相随的是，移动应用开发也将面临着它自身的一系列挑战。

主要的挑战之一是市场分化的问题。2012第三季度的调查显示，移动操作系统市场的分化问题非常严重。在这个季度所售出的设备中，Android的各种型号占了大约72%，iOS占了大约14%，黑莓（RIM/Blackberry）占了5%，而Windows Phone平台占了2%。（基于Gartner统计）

要开发一个能运行于所有设备上的商业应用，需要应用到以下这些的不同技术：

Platform	Primary development platform	Primary development language	Primary IDE	Development platforms
Android	Java based	Java	Eclipse	Windows, Mac OSX, Linux
iOS	Cocoa Touch framework	Objective C	Xcode	Mac OSX
RIM	Java ME	Java	Eclipse	Windows, Mac OSX
Windows Phone 8	.NET/native	C#/C++	Visual Studio	Windows

这些平台、语言及开发工具之间有着本质上的区别，一个能够运行在所有平台上的解决方案，其开发所需的投入也是很高的。

值得一提的是，即使在同一平台之下，分化的程度也很严重，当前处于领先局面的Android平台尤为明显。由于Android是开放的，并且厂家可以任意改动，导致如今市面上基于Android的设备已有几百种。其中的大部分只能使用特定版本的Android API。而即使应用在支持的API版本之内，仍然有一部分设备在使用这些应用时会遇到问题。因而结论是，处理移动市场上的市场分化问题没有捷径可言。这使得实现一个运行于多个平台的原生解决方案异常困难。

移动web应用：市场分化的解决方案？

Web应用是原生应用的一种替代方案。所有主流的移动平台都提供了可用的浏览器。而且，除Windows Phone浏览器之外，其它多数平台浏览器都是基于开源的WebKit浏览器平台，正是它实现了桌面版本的Apple Safari与Google Chrome。这些浏览器平台对JavaScript都有很好的支持，jQuery也在多数主流移动设备上得到完善的支持。另外，各浏览器也在提高对HTML 5与相关的web标准的遵循，这使得浏览器成为一个更吸引人的开发平台。使用当前的技术创建功能丰富的网站，并同样在移动设备上良好运行是完全可能的。

移动web应用：其它观点

移动web网站无法与原生应用的用户体验媲美。特定硬件平台上的用户已经习惯于原生应用所提供的良好体验，这些应用已经原生地安装于设备中，只需通过启动界面就能立即使用。而且原生应用遵守该设备上的用户界面约定，比如在Android设备上，按侧边栏分类菜单键通常会显示一个上下文菜单，这是用户所期望的表现。Web应用可以在多数设备的启动界面上安装一个快捷方式，但它们在所安装的设备上无法遵守特定的用户体验期望。Web应用的另一个劣势是，除了HTML与相关的web标准所提供的输出显示之外，它们对硬件没有任何原生访问的能力。比方如对于设备上的联系人、图片或摄像头不能直接的访问。对于许多应用来说，直接访问设备硬件上的关键元素是很重要的。

混合应用：web与原生应用的最佳结合

混合应用是指在完全的原生应用中嵌入一个特定平台的web浏览器控件。所有主流的移动平台，包括Android、iOS、Windows Phone 8与Blackberry/RIM都支持将这些平台上的web浏览器控件嵌入应用中。由于包装器是完全原生的，用户甚至不太会查觉到他们正在与一个web应用在交互，原生的应用就完全可能提供一个无缝的浏览体验。

浏览器中显示的web页面也可以通过某种JavaScript桥接方式与原生的硬件进行交互，在每种主流的平台上都有该方式的某种实现。通过这种对原生平台的回调，使得访问联系人，抓获与选择图片，以及播放媒体文件成为可能。实际上，任何通过原生代码可以完成的操作都可以通过这种桥接方式实现。当然，对每一种目标平台，桥接代码都必须重新编写，不过这部分代码通常只占用你的整个应用代码的一小部分。

另外，现今已经出现了几种JavaScript桥接框架了，最流行的一种是开源的PhoneGap平台，它提供了这种桥接的一个重要部分。不过我们在这篇文章不打算使用任何框架，而是通过一个简单的Android包装器来演示这个概念。

ASP.NET MVC：一个优雅的后端框架

混合应用可以使用任何web后端技术，不过我们相信由ASP.NET MVC实现混合应用是个理想的选择[ii]。下文阐述了ASP.NET MVC作为这些良好选择的原因。

清晰的职责分离

MVC环境提供了清晰的职责分离，这使得对于HTML输入的精确控制成为可能，也使得生成适合于移动平台的HTML变得非常简单。而那些内建的、独立的驱动能对所生成的标记控制。

与桌面或平板web客户端共享大部分代码

如果你有一个现成的针对桌面浏览器的ASP.NET MVC应用，那么许多代码都可以在你的移动应用中重用。Controller和Model的代码基本可以完全重用，只需要修改视觉的部分。在当前版本的ASP.NET MVC中为移动客户端指定一个定制的视觉就不是什么难事了，而下一个版本的ASP.NET MVC会使得这一切更简单。关于即将到来的ASP.NET MVC版本中，移动开发新特性的其它细节，请参考ASP.NET MVC文档。

通过以下web开发模型将市场分化问题降至最低

ASP.NET MVC对于无状态的web应用并没有创建很多抽象层，而是提供了一个非常简单的模型，它能够与底层的平台相集成，使得在客户端发起AJAX调用或使用jQuery变得非常简单。不必担心处理像ASP.NET Web Forms中的ViewState那样复杂的抽象了。

除此之外，值得一提的是，现有的.NET应用中的业务层和数据库层可以重用在ASP.NET MVC应用中。ASP.NET MVC完全不受业务层和数据库层的影响，它能够高效地应用在任何现有系统中。

混合应用示例

我们现在来看一个非常简单的示例，以演示使用ASP.NET MVC平台开发一个端到端的混合应用开发过程。该示例将显示某个虚拟的大学 - Contoso大学的学生信息，它包含一些常用信息的链接，以及能够根据姓名查找学生信息的目录。为了使这个示例更清晰，它没有包含安全或错误处理的部分。代码本身并不复杂，因为示例本身并不是为了展现ASP.NET MVC平台的强大功能，而是想表现它作为一个后端平台，非常适于开发混合的原生移动应用。

示例的完整代码可以在bit.ly/mvc-native-mobile-apps找到。

运行示例代码的条件：

- ASP.NET MVC 3及Visual Studio 2010（包括Expression在内的任何版本）
- Android SDK的功能安装，及Eclipse的Android开发工具插件。
- 详细的指示及需求可以在这里找到。
- jQuery及jQuery Mobile，这里不需要创建本地的拷贝，因为示例代码会引用jQuery CDN。

ASP.NET MVC后端部分

在示例代码中，_Layout.cshtml包含了对jQuery与jQuery Mobile库的脚本引用。它们对创建ASP.NET MVC移动应用并不是必需的，但它们解决了很大一部分工作。我们在示例中使用jQuery Mobile以简化对移动设备上内容的格式化操作。

```
<link rel="stylesheet" href="http://code.jquery.com
/mobile/1.0/jquery.mobile
;-1.0.min.css" />
<link rel="stylesheet" type="text/css" href="@Url.C
ontent(" site.css?)? Content ~ />

<script type="text/javascript" src="http://code.jqu
ery.com/jquery-1.6.4.min.js">
</script>
<script type="text/javascript" src="http://code.jqu
ery.com/mobile/1.0/jquery.mobile
-1.0.min.js"></script>
```

多数的移动web客户端都假设web页面会缩放至900像素左右的大小，并且能够自适应大小以便在设备上显示整个页面。为了将移动网站的显示优化在某个更小的设备上，我们可以提示设备不要自动伸缩，而是使用该设备的宽度。这可以通过使用viewpoint这个meta标签来实现，如下所示：

```
<meta name="viewport" content="width=device-width,
initial-scale=1.0 ">
```

在Home controller中默认的Index行为方法将映射到以下的view标签。

```
<nav>
     <ul id="menu" data-role="listview">
         <li>@Html.ActionLink("About Us", "AboutUs"
, "Home")</li>
         <li>@Html.ActionLink("Contact Us", "Contac
tUs", "Home")</li>
         <li>@Html.ActionLink("Student Directory",
"StudentDirectory", "Home")</li>
```

```
    </ul>
</nav>
```

我们有一个简单的无序列表，包含三个行为链接。通过在jQuery Mobile运行时中使用“data-role=listview”这个属性设置，我们指定该列表将自动格式化为一个列表视图。这样就能够在移动设备上显示以下的初始UI。

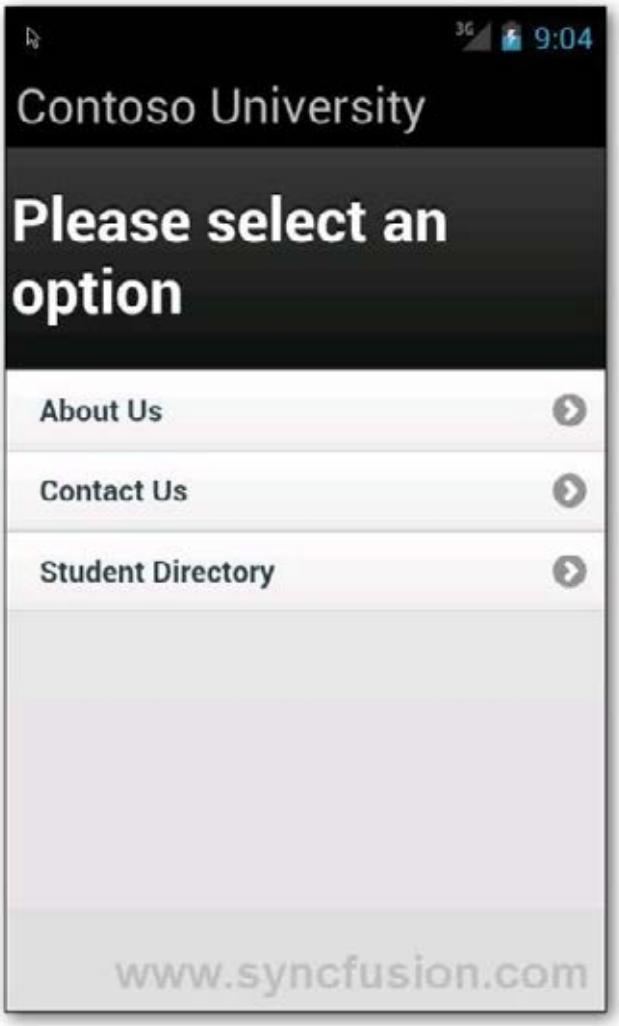


图1，初始界面

jQuery Mobile运行时会负责将它格式化为一个列表视图。如之前所说，jQuery Mobile并非必需，你可以选择最适合你的格式化及脚本的应用方式。

该示例也包含了选择“关于我们”及“联系我们”等选项时所显示的视图，这几个界面非常直白，无需进一步的说明。

打开学生目录链接将显示一个按首字母分组的学生名称页面，并显示每个字母下的学生数目。

图2，学生目录初始界面

点击任一选项将显示学生列表，如下所示。

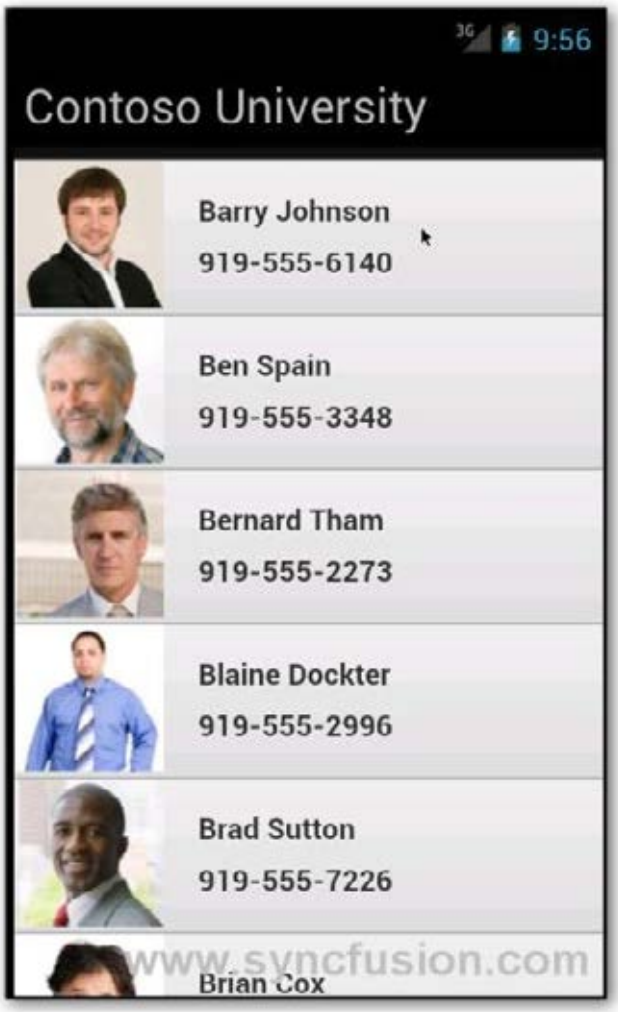


图3，学生目录

学生目录的view也非常简单，仅仅是将列表中的数据迭代显示。显示学生详细信息的view如下所示。

```
@{
    ViewBag.Title = "Student Directory";
     Layout = "~/Views/Shared/_Layout.cshtml";
     var random = new Random();
}

<ul data-role="listview">
@foreach (string student in ViewBag.Students)
{
     <li>

         @{var number = random.Next(1000, 9999); }
```

```
        <img src="@Url.Content("~/Content/images/U
serImages/80-80/" + student +
".jpg")" alt="@student"/>
        <h3>@student</h3>
        <h4>919-555-@number</h4>
    </li>
}
</ul>
```

在使用Android包装器进行查看之前，可以先在桌面游览器中运行ASP.NET MVC后端并进行完整的测试，这也不失为一个好主意。稍后我们再进行Android包装器的开发。

如果你的测试设备能够访问待测试的网站，那么你也可以直接在它的移动游览器上进行测试。如果你的开发PC与设备测试处于同一个网络中，也可以通过修改ASP.NET开发浏览器或IIS Express的配置，以允许你的测试设备访问该web应用。而在默认情况下这种访问是被禁止的。

一种较简单的替代方案是使用代理，它能够将对外部端口的访问重定向到内部的服务器，我们就经常使用这种方式。我们所使用的代理可以从GitHub[iii]下载。

Android包装器

Android包装器将在一个原生的Android应用中托管这个web应用，其代码所下所示。

```
package com.syncfusion.contoso;

import android.app.Activity;
import android.os.Bundle;
import android.util.Log;
import android.view.KeyEvent;
import android.view.View;
import android.webkit.WebView;
import android.webkit.WebViewClient;

public class ContosoActivity extends Activity {
```

```
    WebView mWebView;

    private class ContosoWebViewClient extends WebV
iewClient {
        @Override
        public boolean shouldOverrideUrlLoading(Web
View view, String url) {
            view.loadUrl(url);
            return true;
        }
    }


    /** Called when the activity is first created.
*/
    @Override
    public void onCreate(Bundle savedInstanceState)
 {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);

        mWebView = (WebView) this.findViewById(R.id
.webview);

        // Disable scrollbars
        mWebView.setVerticalScrollBarEnabled(false)
;
        mWebView.setHorizontalScrollBarEnabled(fals
e);

        // Scrollbar Overlay Content
        mWebView.setScrollBarStyle(View.SCROLLBARS_
INSIDE_OVERLAY);

        mWebView.getSettings().setJavaScriptEnabled
(true);
        mWebView.getSettings().setAppCacheEnabled(f
```

```
alse);
        mWebView.loadUrl("http://your-web-link");
        mWebView.setWebViewClient(new ContosoWebVie
wClient() );
    }

    @Override
    public boolean onKeyDown(int keyCode, KeyEvent
event) {
        if ((keyCode == KeyEvent.KEYCODE_BACK) && m
WebView.canGoBack()) {
            mWebView.goBack();
            return true;
        }
        return super.onKeyDown(keyCode, event);
    }
}
```

这段代码也很容易理解。

- WebView在Android上等同于WebBrowser控件，它是默认基于WebKit的Android浏览器的包装类。
- 我们获取了对某个Android WebView控件的实例（它定义在某个XML layout文件中，并由Android运行时进行创建）的访问。
- 我们为这个WebView实例开启了JavaScript功能，因为在WebView控件中，JavaScript默认是关闭的。接下来我们对拖动框的显示做了一些调整，其实就是把它关闭，以模拟原生应用的外观。
- 再接下来，我们调用了WebView实例的loadUrl API，以加载实际的web应用。你应该修改web链接，以指向你的web应用。
- 最后一部分代码处理了对硬件的回退按钮的调用，以使得嵌入的WebView回退到之前的页面。

如你所见，这段代码对web应用并没有直接的依赖关系，也不需要在不同的应用之间进行大量的修改。当你需要访问设备上的特定硬件功能时，你只需要添加一些额外的代码。在这里我们对这个主题不会做深入研究，如果你有兴趣进行更深入的钻研，请查看WebView中的addJavascriptInterface方法的相关内容。

出于简便，我们只介绍了Android包装器的内容，类似的包装器与扩展机制也存

在于所有其它主流移动平台上。

图4，在原生应用shell中使用Android 4.0模拟器显示的“联系我们”页面

结论

对于任何移动行业应用来说，混合应用都是一种值得考虑的有前景的解决方案。对于需要对原生硬件进行大量访问（例如在游戏中）的场景来说它并不适合，但对于其它多数场景它都有良好的表现。由web后端实现的解决方案很有可能不会过时。这些年来HTML的标准在缓慢地演化中，它也不太可能像专利解决方案那样倾向于做出重大的改变。这就为应用的创建提供了一个稳定的基础，也确保这些应用在可预见的将来能够继续工作。Mobile平台的供应商为HTML 5与相关标准的实现投入了巨大的精力，它们将使web应用更加强大，并能够完成很大一部分在原生应用中可以实现的工作。

你可以充分利用你现有的.NET web开发技术，并在更多的设备上创建强大的解决方案。在Syncfusion，我们为混合应用所提供的无限潜力感到激动不已。

关于作者

Daniel Jebaraj作为副总裁，主导Syncfusion产品的开发。总体掌控产品开发并制订发布计划。通过积极地与客户沟通，Daniel确保每个新产品都能基于客户反馈进行改进。早前，作为开发团队的副总裁，她专注于驱动Syncfusion的产品开发。在2001年加入Syncfusion之前，Daniel管理Rogue Wave Software的开发团队。获得克莱蒙森大学（Clemson University）的工学硕士学位。

[i]Gartner Reveals Top Predictions for IT Organizations and Users for 2012 and Beyond

[ii] Ignore ASP.NET MVC at Your Own Peril: Lessons Learned from the Trenches.

[iii] 更多细节请参考James的代码

原文链接：http://www.infoq.com/cn/articles/Hybrid-Mobile-Apps-MVC

相关内容

- 让你的ASP.NET Web应用工作在移动设备上
- ASP.NET MVC 4中的单页面应用程序
- ASP.NET MVC 4 路线图
- ASP.NET and Web Tools 2012.2发布，增强了MVC，Azure及SignalR的功能
- 使用标识和访问工具保护ASP.NET应用程序

QCon 全球软件开发大会 [上海站] 2013

International Software Development Conference

2013年11月1-3日
上海光大会展中心

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

部分特色专题

知名网站案例分析
来自一线的敏捷实战
游戏服务器实践
可穿戴硬件
移动应用案例分析
金融系统设计与架构
扩展性、可用性与高性能
JavaScript生态圈
系统安全设计面面观

更多经典专题 精彩内容 敬请登录：qconshanghai.com

本期专题：指尖上的世界 | Topic

让1.5亿移动端用户第一时间获取消息

作者 霍泰稳

在内容同质化严重的时候，同样的内容，如何让用户第一时间打开自己的移动应用阅读，而不是其他家的？如何将以前需要数小时甚至数天才能发送完毕的消息，在半小时内搞定？特别是类似爱奇艺这样体量巨大的公司，面对自己的1.5亿移动端用户，想维持他们的活跃度，也绝对是一个挑战。目前，云推送技术有望帮助解决这一困境，而且可应用多个移动平台，包括iOS和Android等。日前，InfoQ编辑采访了爱奇艺移动业务事业部的产品总监任衡，以及产品经理周政、吕苏等，就云推送技术在爱奇艺的应用做了探讨，相信他们的经验对其他有类似需求的移动研发人员有一定的借鉴价值。

维持用户活跃性——爱奇艺的大难题

爱奇艺2010年成立，自2012年开始发力移动端，据说不论是在iPad还是iPhone端，都基本上是国内最早做移动端的，到目前为止，也几乎覆盖了所有的平台，包括iPhone和iPad、Android手机和平板、WinPhone和WinPad、Symbian和Bada等。另外，爱奇艺还和三星、华为、步步高、Oppo等手机厂商合作，在它们的手机中预装其应用。根据艾瑞的数据， 爱奇艺与PPS合并后，移动端的用户覆盖规模达到第二名的两倍。

但话说回来，用户装了你的应用不一定就是你的忠实用户，根据一个非官方的说法，很多用户下载了某些应用之后就再也没有打开过。也就是说，如果及时唤醒用户并维持他们的活跃度，比如上了新片或者最新的娱乐八卦消息等如何推送到用户眼前，这才是众多厂商首要解决的问题，这也是爱奇艺所面临的挑战。从技术层面上分析，不论是成本、实时性、可扩展性和个性化，每个都不是容易的事情。

成本维度考量：在一个相对较短的时间内，把数亿条消息全都推送出去，对于服务器、带宽等都是很大的挑战。另外，苹果也有自己的消息中心，它的容量也是有限的，如果一股脑将全部的信息推送给它，它也承受不住，而且达到率会大打折扣。这种情况下，就需要通过一种缓释的方式，一部分一部分地推送

给苹果消息中心，这都需要N多服务器和带宽的支持。对于公司来说，当然也希望通过规模化的操作来降低成本。

实时性维度考量：对于熟悉娱乐界的同学来说，这就比较好理解了，比如最近爆出八卦，某大众偶像已经结婚，这是一个很劲爆的消息，那么爱奇艺获得了这个新闻之后，如何第一时间发出去？要知道，现在每个用户的终端设备上几乎都装着多个视频客户端，而谁的客户端先发出消息，那么谁也就抢得了先机，用户也就可能选择谁。另外，用户也并不是每天24小时都拿着手机看的，一般中午12点和晚上6点是两个明显的小高峰，饭前和下班路上大家都有看手机的习惯，如果在这个时间段让用户收到所推送的消息，开启率和激活率无疑会有比较大的提升。

扩展性维度考量：如果说自己架设服务器，倒也是能做，只是扩展性上不是很好把握，比如要用多少服务器，每个服务器的容量大小等都是需要考虑的。即使爱奇艺能做，相比于专门提供这样服务的公司，比如百度，可能就不如他们将这样的服务做得精细而全面。所以说，通过百度云推送，爱奇艺能够以一种比较低成本、高效的方式解决自己的问题，而且不用考虑任何负载、带宽需求。

个性化维度考量：个性化的优势不言而喻，不同的手机不同的特点，不同的用户也有不同的需求，比如功能手机只能接收文字，而Android手机还可以接收图片，以后也许会直接包含视频等。那么多的手机型号，而且各个厂商可能都有自己的推送服务，对于爱奇艺来说，能够维护好这个复杂的系统也是很大的挑战。如果有一个专门的技术，让开发者不用考虑手机型号，各类推送服务，只需要将精力放在业务上面，比如不同用户的特性，然后依此推送个性化的服务，给喜欢电影的人推送新片，给女性推送娱乐八卦等。大家各司其职，各取所长，工作效率就会提高很多。

基于云的推送服务的引入

在苹果没有在国内架设自己的数据中心之前，类似爱奇艺这样的公司，在推送消息时，经常遇到的问题就是丢包。其原因也比较简单，一是因为苹果的带宽较小，本来总量可能比较大，但是分配给国内的公司就少很多。

在引入百度云推送服务之前，爱奇艺一直没有找到特别好的办法解决丢包的问题。在测试了多次之后，他们才决定选用百度云推送的服务，其原因主要是，虽然爱奇艺的技术团队对自己的服务端做了很多改进，但是相比于百度云的规模效应，还是小巫见大巫。举个简单的例子，原来自己需要花费四、五个小时才能发完所有的消息，现在则是几十分钟的事情，这就是云技术的能量。而且对于Android平台，因为不像iOS那样还要走自己的消息中心，百度云推送可以让爱奇艺的信息做到实时发布。

对于百度云推送服务的特点，爱奇艺根据自己的理解，也做了一些总结。以自己以前自建服务器完成信息推送服务，和现在的云推送服务相比，任衡举了一个快递配送的例子，虽然现在的快递公司很多，但是大家很多时候还是愿意选择顺丰，其原因就在于其服务可以及时、安全，服务质量也不错。可以说，百度云推送就是一个功能完善、经验丰富的配送站，比如从前自己通过苹果消息中心推送信息时，不知道那个道路只能跑两辆车，于是一下子塞进去了一万辆，结果自然是堵塞。而百度知道那个是双车道的，每次只能塞两辆，基于这些经验，于是就将客户发送到他那儿的一万辆车分次发，每次两辆，问题就轻易解决了。

另外，很多中小型公司，其实包括爱奇艺在内，面对苹果的消息中心，有时候也不知道如何调整自己的策略才更好，而百度知道，鉴于自己在互联网领域的多年经验，它知道何时中国的带宽波动最大，而何时苹果的消息发送中心效率最高，因为走过很多遍，也就知道了路上有哪些坑。要知道百度地图、百度音乐都是体量很大的移动客户端，在其发展的过程中，相信也都遇到了各种各样的问题。而解决这些问题之后的经验，通过百度云推送服务都可以无缝传递给企业。

谈到在使用百度云推送时的经验，以及如何让后来者少走一些坑。任衡和周政提到，因为百度云推送的接口是标准的，所以只要严格按照文档中的指南，很快就可以上手。爱奇艺是差不多用了一个月的时间完成了从测试到调试到上线的三部曲。其实，在选用百度云推送之前，爱奇艺也测试了其他一些国内的云推送服务公司，之所以最终选择百度云，一方面是公司间的关系，另外就是百度在运维、云计算方面的技术积累，以及从公司品牌和个性的服务等方面，都比较占优势。

期待——更多的个性化数据分析

虽然现在百度云推送已经实现了比较好的消息推送功能，能够满足企业的基本需求，但是用户的需求是无止境的。在交谈中，任衡也列举了一些例子，类似淘宝的数据魔方，通过收集整理买家的数据，能够了解他的喜好，然后推送不同的广告信息。而百度云推送也许将来可以通过和爱奇艺合作，基于所发送的信息，深入了解用户对信息的敏感度，然后按照电影、电视剧、娱乐八卦对人群细分，从而不仅可以将消息推送到用户面前，而且推送的都是他们感兴趣的。如果能做到这一点，那么就善莫大焉。

在InfoQ编辑和爱奇艺的研发经理沟通的过程中，即使是体量大的移动端，即使有着强大的技术研发力量，也是希望能够将自己的精力放在业务上，让用户有片看，让视频更加流畅，让用户体验更好，然后将非核心的事情都外包出去，比如将消息推送交给百度云推送去做等。对于体量不如爱奇艺这么大的中小型公司，也许需求更是如此吧。在接下来，InfoQ的编辑会走进更多的有代表性的

移动产品，与其技术负责人、产品经理沟通，获取更多在移动产品开发过程中的难题，以及可能的解决方案。各位读者如果在研发过程中遇到什么难题，也欢迎以评论的方式与编辑讨论，InfoQ的编辑会带着这些问题和有经验的专家沟通，为大家尽可能答疑解惑。

原文链接：http://www.infoq.com/cn/articles/iqiyi-cloud-push-practices

推荐文章 | Article

如何从组件开始构建一座城市？

作者 Aliaksei Papou ，译者 李彬

为什么越来越多的企业应用开发正在转向组件框架和解决方案？组件架构是否有前途？我相信答案是肯定的，而且很快所有开发框架都将会是基于组件的——这是近在眼前的事情。下面让我来向你揭示这一切的原因。

你怎么来建设你的房子？一般你会从砌块开始。我们可以将构建Web应用与构建你的乡间小屋进行对比。你能够快速构建一个非常好看的应用，而且它具有所有必需的功能。同样，在你的房子里面，每一间房间都是针对具体的需求来创建的，例如厨房、起居室、卧室或浴室。房子的布局使你能够通过走廊和楼梯很方便地在房间之间移动。

现在你能够做得更好，而且能够承担建设一座更大更好的房子的投入——你也许希望拥有桑拿房、游泳池、影院以及一座满是爬行动物的巨大的水族馆☺。但要想改变房子的设计却是件非常困难的事情。若要添加额外的设施，房子最终看起来也许就不那么漂亮了。此外，由于你添加的这些设施必须放在不太方便的位置，它们也会影响房子使用的便利性，例如你必须穿过主卧室才能进入台球室。

最后，你那漂亮又整洁的房子将拥有一堆不同的功能，但它会变得笨拙又不舒适。同样的道理也适用于应用开发。

问题是，有没有可能设计一款应用，能够根据你的需求成长和改变？

组件是应用的积木式构件

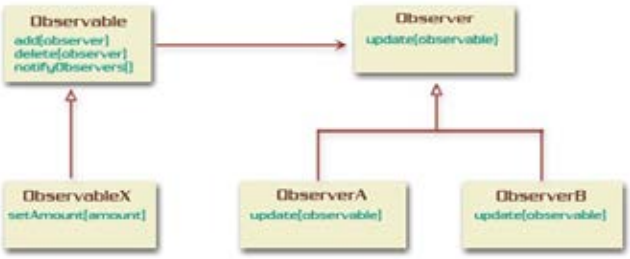
组件是扩展应用功能的首要方法。创建组件的过程，与基于组件创建应用的过程有一些差异。组件不止应该提供有用的功能，还应该从一开始就设计成可复用的。

组件复用

组件应该采用松耦合方式设计以便于复用。为实现这一目标，不同的框架往往基于观察者模式实现其事件模型。该模式允许多个接收者订阅同一事件。

观察者模式的实现最早出现在Smalltalk中。Smalltalk是一个基于MVC的用户界面框架，现在它已经成为MVC框架的关键部分。我希望你能注意到，自Java 1.0版本起，观察者模式就已经在Java中存在。下面让我们深入了解它。

下面的UML图展现了观察者模式：



以下则是一段基本的Java实现：

```
public class ObservableX extends Observable {
  ...
  public void setAmount(double amount) {
    this.amount = amount;
    super.setChanged();
    super.notifyObservers();
}

}
  public class ObserverA implements Observer {
  public void public void update(Observable o) {
  // gets updated amount
}

}
  public class ObserverB implements Observer {
  public void public void update(Observable o) {
```

```
  // gets updated amount
}

}
//instantiate concrete observableX
ObservableX observableX = new ObservableX();
//somewhere in code
observableX.addObserver(new ObserverA());
observableX.addObserver(new ObserverB());
//much later

observableX.setAmount(amount);
```

它是这样工作的：

首先我们创建一个ObservableX类的实例，将ObserverA和ObserverB实例添加到observableX对象中，然后在代码中的某个位置，我们使用setAmount方法设定“一定数量”的值。被观察者（observable）类的功能将接收到的数量通知所有注册的观察者。

观察者担当着中介的角色，维持接收者的列表。当组件里有事件发生时，该事件将被发送到列表上的所有接收者。

由于这个中介角色的存在，组件并不知道其接收者。而接收者可以订阅某个特定类型的不同组件中的事件。

当一个类使用事件将自身的变化通知观察者时，该类就可以成为一个组件。而这可以通过观察者模式来实现。

使用组件比创建组件容易

通过使用组件，你能够快速创建各种窗体、面板、窗口以及界面中的其他合成元素。不过，为了能够复用新的由组件创建的合成部分，应该将它们转化为组件。

为了实现这一目标，你需要决定组件所要生成的外部事件，以及消息传递机制。例如，你至少需要创建新的事件类并且定义接口或回调方法以接收这些事

件。

这个方式让实现可复用的应用组件变得更复杂。当系统只是由少量合成元素组成时没什么问题——这时合成元素最多不超过10个。然而当系统包含数以百计的此类元素时，又当如何？

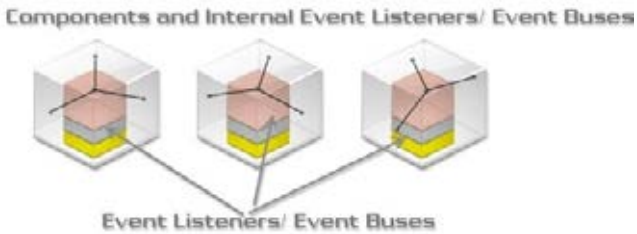
与之相反，不遵从这一方式将导致元素间的紧耦合，并且会把复用的机会降低到0。这反过来会导致代码复制，从而让未来的代码维护变得更复杂，并将导致系统中的bug数量上升。

由于组件使用者往往不了解如何定义和传递他们自己的新事件，问题将变得更为严重。但他们可以轻松地使用组件框架提供的现成的事件。他们知道如何接收但不知道如何发送事件。

为了解决这个问题，让我们考虑如何简化应用中使用的事件模型。

太多的事件监听者

在Java Swing、GWT、JSF和Vaadin中，观察者模式被用于实现多用户能够订阅同一事件的模型，并将用于添加事件监听者的列表作为其实现方式。相关事件一旦发生，将被发送到列表上的全部接收者。



每个组件为一个或多个事件创建自己的事件监听者集合。这将导致应用中类的数量不断增多。反过来，这也会使系统的支持和开发变得更复杂。

借助注解机制（annotation），Java找到了一条让单个方法订阅特定事件的道路。例如，考虑Java EE 6里的CDI（Contexts and Dependency Injection，上下文和依赖注入）中对事件模型的实现。

```
 public class PaymentHandler {
      public void creditPayment(@Observes @Credit P
aymentEvent event) {
        ...
      }
}

public class PaymentBean {
```

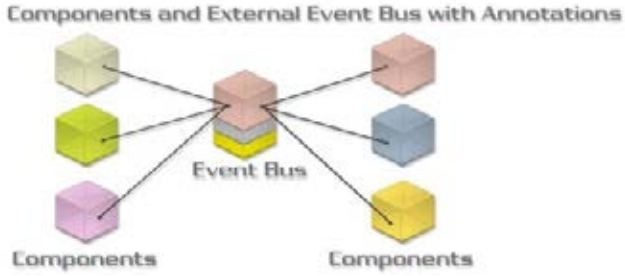
```
    @Inject
    @Credit
    Event<<paymentevent> creditEvent;

   public String pay() {
     PaymentEvent creditPayload = new PaymentEvent(
);
            // populate payload ...
            creditEvent.fire(creditPayload);
      }
}
```

你可以看到，当PaymentBean对象的pay()方法被调用时，PaymentEvent被触发。接下来PaymentHandler对象的creditPayment()方法接收了它。

另一个例子是Guava类库中事件总线的实现：

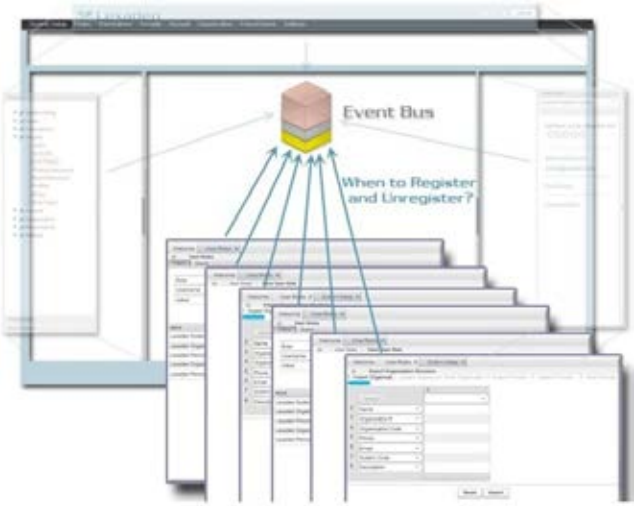
```
// Class is typically registered by the container.
class EventBusChangeRecorder {
  @Subscribe public void recordCustomerChange(Chang
eEvent e) {
    recordChange(e.getChange());
  }
}
// somewhere during initialization
eventBus.register(new EventBusChangeRecorder());
// much later
public void changeCustomer() {
  ChangeEvent event = getChangeEvent();
  eventBus.post(event);
}
```

EventBus注册了EventBusChangeRecorder类的对象。接下来对changeCustomer()方法的调用会使EventBus接收ChangeEvent对象并调用EventBusChangeRecorde对象的recordCustomerChange ()方法。



现在你不需要为你的组件实现若干事件监听者，在应用中使用事件也变得更简单了。

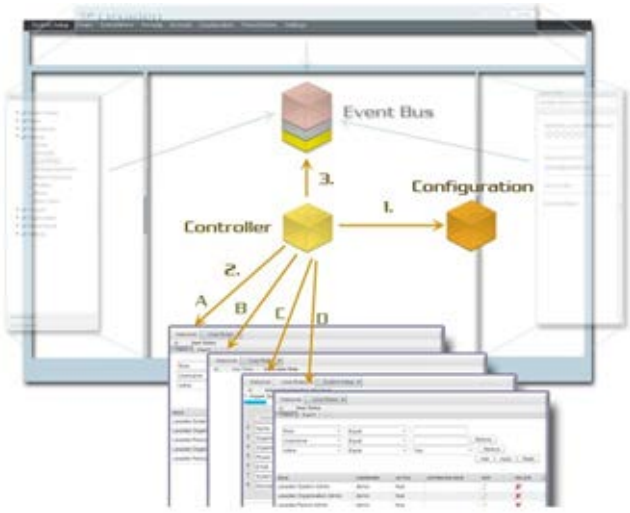
当所有组件都同时在屏幕上展现时，使用事件总线是很方便的。如下图所示，它们使用事件总线进行消息交换。



这里，所有元素——标题、左侧的菜单、内容、右侧的面板——都是组件。

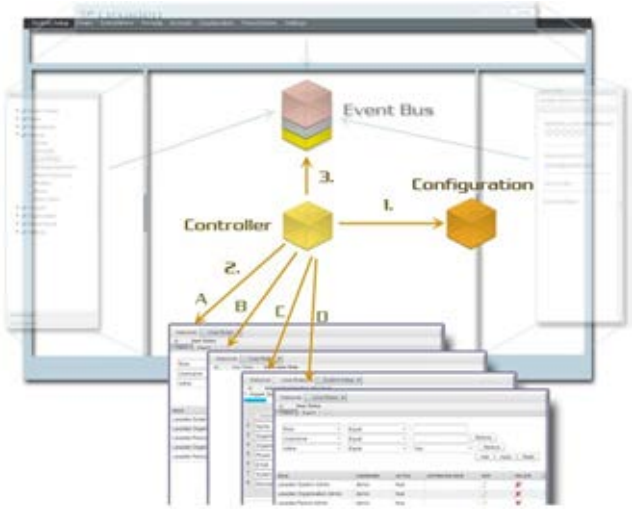
订阅事件——别忘记取消订阅

通过将事件监听者替换为注解，我们在简化事件模型使用的道路上前进了一大步。但即使如此，系统中的每个组件依旧需要连接到事件总线，然后，必须订阅上面的事件并在正确的时间取消订阅。



当相同的接收者多次订阅同一个事件时，将会出现许多重复提醒，这种情况很容易出现。而相似的情况还会在多个系统组件订阅同一事件时发生，这将会触发一系列级联事件。

为了能更好地控制事件模型，将工作与事件一起迁移到配置中，并让应用容器负责事件管理是很有意义的。由于特定的事件仅在特定条件下有效，将这些事件的状态管理迁移到配置中也是合理的。



下面是一段配置的例子：

```
<?xml version="1.0"?>
<application initial="A">

    <view id="A">
        <on event="next" to="B"/>
    </view>

    <view id="B">
        <on event="previous" to="A"/>
        <on event="next" to="C"/>
    </view>

    <view id="C">
        <on event="previous" to="B"/>
        <on event="next" to="D"/>
    </view>

    <view id="D">
```

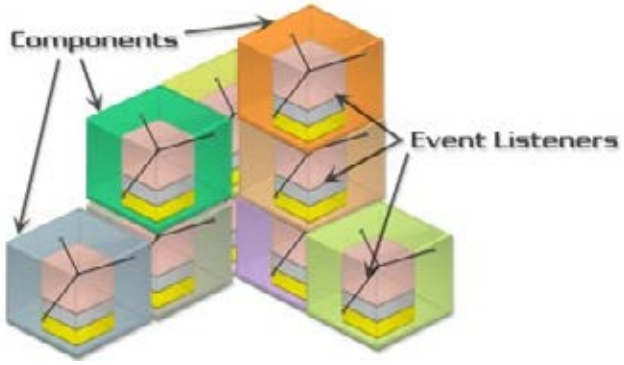
```
        <on event="previous" to="C"/>
        <on event="finish" to="finish"/>
    </view>

    <final id="finish" />

</application>
```

视图A中的“下一个（next）”事件触发了向视图B的转变。在视图B中，用户可以通过“前一个（previous）”事件回到A，或是通过“下一个（next）”事件进入C。D视图中的结束事件将转入“最终（final）”状态，将通知应用结束其中的工作流。

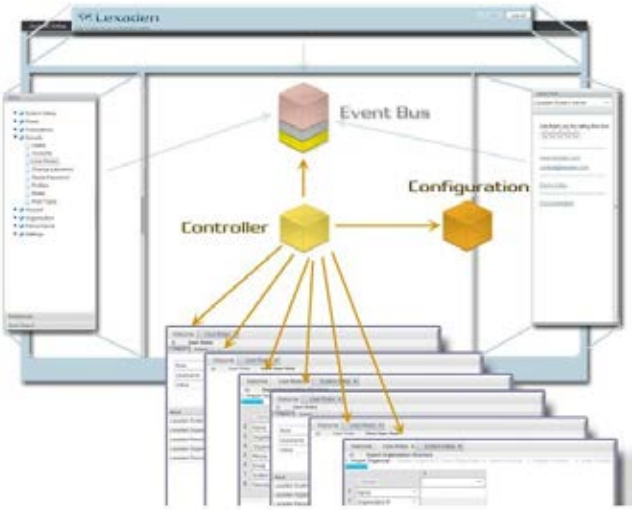
有限状态机是专为这样的需求设计的。状态机是一种数学计算模型。它被设想为一种抽象的机器，可以处于有限数量的状态中的一个，并且在同一时间里只会处于一个状态——这被称为当前状态。事件或条件将触发向另一个状态的转变。使用这一方式，你能够轻松地定义活动画面，并让某事件来触发向另一个画面的转变。

使用有限状态机来配置应用的好处

大部分情况下，应用配置是静态定义的。使用依赖注入配置应用，我们在启动时定义应用结构。但我们忘记了在探索应用的过程中它的状态可能会改变。在应用的代码中，状态改变往往属于硬编码，它让未来的调整和维护变得复杂。



将状态间的转变迁移到配置中可以提高灵活性。而且这正是为什么我们在创建诸如窗体、窗口或面板等复杂应用元素时，无需为了应用应该进入哪个状态而操心。你可以稍后来处理它们，在配置中设定其行为。



所有组件都可以使用标准的事件发送机制进行交流——即通过事件总线。同时，状态机能够控制组件事件到事件总线的订阅。这一方式将应用的全部组件（窗体、窗口、面板）变为可复用组件，可以通过外部配置轻松地管理它们。

如果有兴趣，你可以看一下Enterprise Sampler中一些配置的例子。

你也可以将状态配置看作城市的公路图，把事件看作递送商品的汽车，而将城市里的人看作目的地。

我确信采用这样的方式，不仅能够轻松地设计和构建一间规模虽小却做好了成长准备的房子，还能够建设拥有摩天大楼、公路和高速公路的城市。

关于作者

Aliaksei Papou是Lexaden.com的CTO、软件架构师和联合创始人，他拥有超过10年的企业级应用开发经验，他对于技术创新有着强烈爱好。他与Lexaden.com的CEODenis Skarbichev（另一位联合创始人）一同开发了可以创建大规模敏捷企业级应用的 Lexaden Web Flow语言。

原文链接：http://www.infoq.com/cn/articles/component-city

相关内容

- Dagger: 一种Android平台的依赖注入框架
- 58同城开源其轻量级Web框架Argo
- 服务器端代理拥有事务的设计模式

推荐文章 | Article

ThoughtWorks技术雷达（2013年5月）

作者 ThoughtWorks技术战略委员会

技术

多年来，团队和组织都已经看到了围绕技术学科进行专业细分的危险性。当我们就高级应用听取专家的建议时，开发人员至少应当了解用户界面，数据库和数据科学等业界新宠的基础知识。当高级应用需要深入的专业知识时，我们要在所有开发人员使用基础统计分析理论和工具的领域，推动协作分析与数据科学，以此做出更好的决策，并在事情愈趋复杂时，开展与专家的密切合作。

技术趋势已经冲破了原本保护着企业IT网络的围墙，成为没有边界的企业。员工们频繁使用自己的消费设备通过云服务和Web API访问企业数据，而企业往往对此并不知情。随着设备不断扩散以及更多的应用程序迁移到云端，迫使企业重新考虑数据访问和网络安全的基本假设。

在云端搭建的开发环境意味着开发的基础设施能够外包，开发人员只需要一台笔记本电脑和网络连接。通过组合使用最佳的服务，如GitHub的私有资料库加上云端的Snap CI持续集成环境，你的团队也许永远都不再需要为自建IT基础设施而劳神了。

廉价或免费的视频会议工具，质量正在不断改善，选择也越来越多，让分布式团队有了一种全新的工作方式。即便团队分散在各地，全天候（Always-on）视频连通也可以帮助建立一种虚拟的临场感。事实上，这已经成为我们部分离岸交付中心的标配。我们还看到像ScreenHero这样的屏幕共享工具已经开始被广泛用于远程结对编程。但我们也必须警惕寻找银弹消除物理上同处一地的需求，毕竟没有什么可以替代面对面沟通时所产生的相互理解和情感交流。

一旦开始使用一件全新的工具，管理不同环境的部署，或者尝试着理解应用程序的行为为何在不同环境表现各异，对应用的配置就会成为痛苦的根源。作为最小化应用配置的拥趸，我们正尝试去保证应用能在最小配置下能够正常工作。

多数虚拟化技术都提供了通过加载镜像启动虚拟机的方法。在构建管道中要尽早创建虚拟机镜像作为构建产物, 随着它通过后续的测试套件对其进行提升，如此就能实现从测试环境到产品环境机器上的可靠部署 。此技术剔除了导致snowflake server*（注释）反模式的多数原因。

蓝绿部署（Blue-green deployment）是一种执行软件升级的模式。首先，将最新版本的应用部署到一个和当前产品环境完全一样的应用栈副本中。这样，当新版本的应用通过了相应的测试并得到了业务上的许可后，访问就可以被瞬间切换这个新版本的应用上。尽管这不是一项新技术，但基础设施自动化和云端资源使它值得被重新考虑。

以前，像Chef和Puppet这样的工具都缺乏对Windows的支持，导致构建简单的基础设施自动化任务都需要大量的Powershell脚本。所以，想要使Windows到达和Unix一样的自动化水平可谓困难重重。不过在过去的一年里，Chef和Puppet对Windows的支持得到了很大改善。这种支持与强大的PowerShell相结合，使得W**indows基础设施自动化**极为可行。

HTML5存储，也被称为本地存储或Web存储，是一种在现代浏览器里存储客户端数据的机制，其中也包括iOS和Android移动浏览器。 在几乎所有情况下，我们都建议使用**HTML5存储，而不是cookie** 。 HTML5存储最多可容纳5MB的数据，而cookie则被限制在4KB。 Cookie的数据在每一次请求中都会被发送，这会拖慢你的应用程序，并且有可能将数据暴露于不安全的HTTP连接中。 与此相反，HTML5存储只需要由浏览器来保证数据安全。 Cookie则只应该用来保存像session id这样的简单数据。

使用**Promise进行异步编程**是一项古老的技术，也被称为futures。由于JavaScript在客户端和服务器端的的广泛使用，它又再次得到了关注。这项技术消除了深层嵌套回调、标志位和轮询，同时得到了像jQuery库的一等支持。如果项目涉及到非常复杂的JavaScript代码库，团队就应该利用这项技术。

捕获客户端JavaScript错误，已经被我们的交付团队用来识别与浏览器或插件配置相关的影响用户体验的问题。在过去一年中，一些服务提供商业已开始对这一需求提供支持。除了将这些错误储存于应用数据存储中，一些Web应用还会将这些错误记录到网络分析工具，或像New Relic这样监测工具中，以减轻存储的需求。

随着HTML5模糊了传统本地应用和Web应用程序之间的界线，我们正开始试验**移动设备上的持续交付**。 如TestFlight服务可让你在一天内多次部署本地应用程序到实际设备上。对于全部或部分基于HTML5的应用程序，可以直接部署修改，而无需向应用商店提交一个新的应用。如果你的组织有企业内应用程序商店，就可以轻松的发布应用。我们注意到，当移动设备上的持续交付技术实现不断改进时，测试实践则相对落后。要取得成功，你需要更加专注于自动化测试，以确保应用被部署到设备上时能够正常工作。

我们越来越多的看到移动应用在开发和测试过程中工作良好，但在发布出去后却问题多多。移动网络的移动测试（Mobile testing on mobile networks）揭示了应用是如何在不同条件下运行的。你可能需要测试3G，LTE，或者故意使用一个访问超载的糟糕的WiFi网络。先获取目标环境下的网络性能，然后用延迟和丢包诱导工具去模拟相同的环境。 此外，使用像Wireshark这样的工具去检测你的设备和软件到底如何使用网络，有时也是必要的。

NoSQL数据存储正继续成为主流，但团队需要充分认识到对迁移数据库到**NoSQL**的需求。特别是使用隐式或动态schema，你可能会随着时间的推移需要重新配置数据。这里提供了几种方法：比如在部署新版本应用时进行显式地迁移，或者通过代码在加载和处理文档时进行动态迁移。

失败的测试揭示了产品代码中的bug。然而，针对其他属性进行分析性测试则可以揭示一些有趣的信息。 一个简单的例子是监测频繁失败的测试，然后在构建管道中尽早运行它们，从而快速得到反馈。同理，跟踪其他属性，如测试执行时间和耗时测试的比例，也可以为达成快速测试提供可操作的指标。

在以前的雷达中，我们曾经建议延长自动化验收测试的周期，在我们称之为语义监控（semantic monitoring）的阶段，不断在产品环境中运行这些测试。 现在我们仍然相信，这是一项使团队可以提前预测情况的重要技术。但现在, 尤其是在初创公司中, 我们还看到这种方法的一个变体, 就是在减少测试次数的同时，增加监控与自动报警。这使关注点从避免可预见的问题转移到对所有问题减少**平均的恢复时间**。

虽然单元测试和验收测试被广泛接受并作为标准的开发实践，但这一趋势还没有延伸到性能测试领域。目前，常用的测试工具促使测试人员养成编写一次性代码, 对点击进行脚本化的心态。 给予**性能测试一等公民待遇**，可以创造更好的测试，涵盖了更多的功能，引导出更好的工具来创建和运行性能测试，从而

使得测试套件可以被维护，并能自我测试。

平台

PostgreSQL 正在逐渐成为SQL数据库中的NoSQL选择。版本9.2支持存储JSON格式的数据，并在其之上提供完备的查询功能。另外，PostgreSQL还提供了允许用户以键值对（key/value pair）的形式存储和查询数据的扩展。该扩展让你可以直接利用稳定的数据库底层存储和事务的能力， 而无需绑定在关系数据模型上.对于那些希望同时开发SQL和NoSQL应用，却又倾向于使用一套他们熟知的可靠基础设施的开发者而言，提供这一特性就再理想不过了。

网站的访问量即使再低，其产生的数据量也可能是巨大的。一旦你把各种分析结果，业务图表， 统计数据，用户资料以及对多设备的支持囊括进来，数据量可能会大到难以想象。许多组织使用数据仓库来存储从各部门收集上来的数据。这样做的挑战是数据仓库会变成一座“数据森林”。即使是获取及时的业务图表也已变成一种挑战，更别提在整个数据集上运行探索性查询了。而基于**BigQuery**之类的云技术对此会有帮助。它们即时付费的模式和可以运行任意查询的能力使你无需购买特殊的软硬件也能获得对数据的全面掌控。数据驱动的商业模式应该将数据送到决策者手中, 而不是隐藏在技术障碍和官僚主义中。

对于适用文档数据库模型的问题来说，**MongoDB**是当下最流行的选择. 除了易于使用和提供高可靠性的技术实现之外，开发社区及整个生态系统对其成功也贡献良多。我们也听说了一些团队抵受不住MongoDB流行的诱惑而产生了问题，比如并不适用文档数据库的应用, 又或者团队没有充分理解其内在的复杂性。然而若能用对， MongoDB已经在很多项目中证明了自己。

Redis 已经在多个ThoughtWorks项目中被证明是有用的工具。它被用作结构化的缓存或者跨越多个国家的分布式数据存储。

Hadoop最初的架构基于水平扩展数据，垂直扩展元数据的模式。 尽管slave节点可以从容的进行数据的存储和处理，管理元数据的master却是瓶颈，同时也限制了规模化网络的使用。 **Hadoop 2.0** 重新设计了HDFS和Map Reduce框架

的架构，以此来解决这些问题。现在，HDFS的命名空间可以通过使用同一个集群的多个name nodes来联合在一起，并且部署为高可用性(HA)模式。MapReduce也被YARN取代了。YARN解耦了集群的资源管理和任务的状态管理，并且通过JobTracker消除了伸缩性和性能问题. 最重要的是，这些变化鼓励在Hadoop集群上开发除了MapReduce之外的分布式计算模型。

在过去的一年中，我们发现采用Elastic Search（弹性搜索）作为开源的搜索平台呈逐步上升趋势。Elastic Search基于Apache Lucene, 是一个可扩展的，多租户的， 支持水平伸缩的搜索解决方案。它允许索引复杂数据, 并可通过JSON获取。它支持从集群, 故障转移及备份中自动恢复各节点。Elastic Search可以通过基于插件系统的REST API来扩展，它提供了一个优雅的操作模型, 允许添加新功能, 或者改变现有行为。围绕着Elastic Search的社区是非常活跃的。一个证据就是已经开发出了包括Java、 C#、 Ruby 和 JavaScript 在内的多个客户端类库。

Node.js 是一个轻量级web容器。它在开发微服务, 移动服务端以及单页面web应用方面都是极好的选择。因为其天然的异步特性, 开发人员可以通过使用promise类库来简化应用代码。 随着在node.js社区promise应用的成熟, 我们期望看到更多为node.js开发的应用。 对那些不愿在产品环境中尝试node.js的团队, 用node.js来实现一些开发任务仍然值得考虑, 比如在浏览器之外运行JavaScript测试, 或者从类似CoffeeScript, SASS和LESS之类的工具中生成静态web内容。

Zepto.js 是一个轻量级的JavaScript类库。它的API与JQuery相同, 但并非100%兼容.因其压缩显著的文件大小, Zepto在开发响应式web应用方面颇有竞争力。

PhoneGap, 现在叫 **Apache Cordova**, 允许你使用HTML, CSS和JavaScript 来开发跨平台的移动应用。它通过一组JavaScript API对平台特定的本地代码做了抽象, 同时还保持了不同移动平台之间的一致性。Cordova支持非常多的平台, 包括iOS、Android、 Blackberry、 Windows Phone 以及 WebOS。

在云存储和云计算领域, 尽管AWS还在持续的增加功能, Rackspace Cloud 却悄悄发展为一名不可忽视的竞争者。部分用户会非常看重Rackspace更加周到的客户支持, 以及混合多种传统托管模型的能力。 我们并不为之而感到激动人心，因为Rackspace是我们的客户, 我们在参与该平台的开发过程中已经收获了许多乐趣。我们在另外多个客户的项目中已经成功地应用了Rackspace Cloud, 目前正期待着它能辐射到更多的地区。

开源**OpenStack**项目的发展可谓蒸蒸日上, 在最近几个月，它已经逐渐成为部署私有云的不二选择。 许多原来导致**OpenStack**难以使用以及运行不稳定的问题都得到了解决. 新功能也一直都在增加。显然，OpenStack联盟包括它的成员如Rackspace，Redhat，HP等都在不断为其添砖加瓦, 并以此作为自有的基于

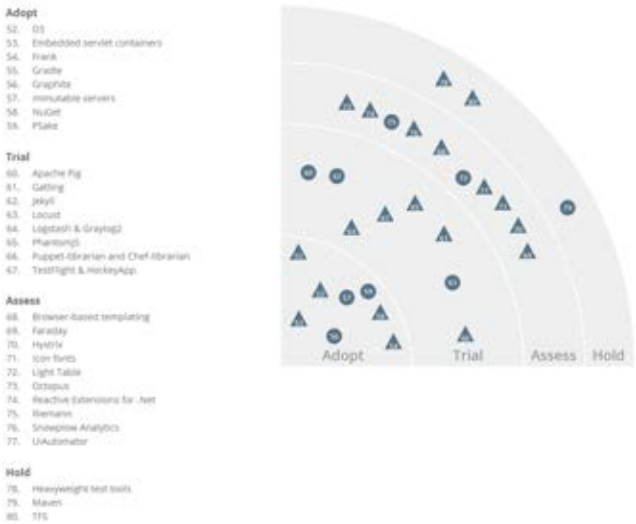
OpenStack的云服务基础。

在去年全球售出的手机中, 58%其实是老式的功能手机, 而非智能手机。 在很多发展中国家, 这一比例甚至更高。如果你的客户需要你为这些地区开发应用, 你需要考虑到这些限制。这些手机使用**SMS**和**USSD**作为用户界面。SMS是存在已久的发送消息的技术, 而USSD允许你通过安全会话来发送SMS之类的消息。你应该把SMS和USSD看做另外一个UI和UX的平台, 给予它们一等公民的待遇。

Vumi是一个可伸缩的开源的消息引擎。 它在移动设备上通过一些廉价的方式来完成通信。 Vumi帮助公司和它们的客户, 医疗服务机构和它们的病人, 政府和市民, 以及更多的团队之间通过SMS , IM 和 USSD进行交互. Vumi与电信公司进行了集成, 使你能够很容易的基于它构建应用, 而你只需要向运营商支付通信费。

“企业级”商业软件包提供的功能, 和实际需要的功能之间的差距正在拉大。这对面向互联网的应用来说更是如此。那些真正能够扩展自如的创新方案, 易于支持的现代技术如持续交付等, 都是实践者写给实践者的。它们肇始于许多互联网规模的公司, 而后作为开源软件得到了提炼。庞大的企业解决方案通常会阻碍高效的交付，因为这些方案经过长久累积，具有臃肿而繁琐的许可限制；它驱动出来的功能集来自于检查列表，依靠假想出来的脱离多数团队开发现实的需求。

工具



在上期的技术雷达中，我们谈到的嵌入式servlet容器。如今，这已经在我们的项目中广泛采用。像 SimpleWeb 和 Webbit 这样的工具在追求简洁和嵌入式的方向上更进一步，提供了未实现 Java Servlet 规范的原生HTTP 服务器功能。同时，最流行的 Java 应用服务器 Tomcat 在嵌入式安装中应用的逐渐增多，以及微软为 .NET 框架提供的自托管服务器，都加强了这一趋势。

D3 作为一个用来在浏览器上创建丰富可视化效果应用的 JavaScript 库，也在持续获得更多的关注。此前，使用 D3多少 显得有点底层， 比起那些更简单更有针对性的库，它需要更多的工作量完成常用的可视化效果。自从发布了上一次技术雷达，随着诸如用于图表的 Rickshaw 以及用于在浏览器中进行数据探索的 Crossfilter 等库的出现，使得 D3 比以前更容易使用了。

我们看到一些 JavaScript 框架正在拥抱基于浏览器的模板化的趋势，它们把更多的布局工作转移到客户端来做。虽然这种方法在很多情况下很有效，但它也会引入一些涉及缓冲、性能和搜索方面的运营问题。我们认为在使用这些工具前，需要仔细对这些工具进行评估，以确保满足目标部署环境的要求。

通过把 IObservables 和 IObservers 放到与 IEnumerables 和 IEnumerators 同等的地位上，.NET 上的响应式编程框架 Rx 允许开发者通过observable事件流的通用底层抽象，使用已有的 LINQ（Language INtegrated Query） 操作符，来查询和编写异步操作和基于事件的代码。

微软也发布了 RxJS 库把响应式编程的好处带到 JavaScript 中。整个 Rx 框架的开源，也使其在 Windows 富客户端应用和单页面 JavaScript 应用中发挥更大的作用。

一些 ThoughtWorks 的团队特别提到 Ruby 的 HTTP 客户端类库 Faraday 的实用性。它为各种适配器提供了通用的接口，并且和Rack 中间件有良好的集成。

针对第三方库的包管理系统继续在所有平台上获得认可，也都在不断添加新的功能。我们最近把 NuGet 和 Chocolatey NuGet 列入到技术雷达，也证实了这个重要的敏捷工程实践的先进性。

基于Windows的基础自动化应该采用，然而仍然要比在 Unix 上进行基础设施自动化要困难得多。Chef 和 Puppet 正在增加对 Windows 基础设施自动化的支持，同时也有正在研发中的像Octopus这种专门针对 Windows 的解决方案。Octopus 可以在自动部署 ASP.NET 应用程序和 Windows 服务时减少对 PowerShell 的依赖。它也可以与 NuGet 和 TeamCity 一起使用，以此创建一个完整的构建、打包、部署管道。

Puppet 和 Chef 都需要处理社区针对通用服务和任务而共享的module和 manifest。Puppet Forge 和 Chef的Cookbook 资料库能在一定程度上解决这个问题，但结果是人们总是复制和粘贴这些脚本到自己的代码中，这样他们就不能享受到最近的错误修正和改进的好处。Puppet-librarian 和 Chef-librarian 通过让你更容易地声明自己的模块依赖关系，试图解决这个问题。其做法包括从这些社区网站取得已知版本的代码。

在分布式系统上的依赖关系管理非常复杂，这也是越来越多人在迁移到细粒度的微服务（micro-services）时所需要面对的问题。来自 Netflix 的 Hystrix 是一

个实现了处理各种下游故障模式的 Java 类库，它提供连接的实时监控、缓存和批处理机制，以更有效地管理服务间的依赖关系。

不需要应用程序商店，TestFlight 和 HockeyApp让你能够管理移动应用程序的部署，让用户测试更容易。它们提供了崩溃报告和实际数据分析的能力。HockeyApp 支持 iOS、Andriod 和 Windows Phone 平台，而 TestFlight 只支持 iOS 和 Android 平台。我们已经成功地使用这两种工具来帮助移动应用程序的发布。这显然是一个快速发展的领域。

Frank是一个开源库，可以使用Cucumber 编写iOS 的功能测试，并在远程设备上执行。

以前在 iOS上，验收测试驱动开发非常麻烦，该库填补了一项重要的空白。

UIAutomator允许测试时对用户界面组件的细粒度控制，以及为在多个设备上的测试提供帮助，因此被认为是Android 用户界面测试方面最有前途的工具。

随着具有多种外形因素和像素密度的设备的兴起，呈现全尺寸高品质图标的问题已经变得非常重要。图标字体利用浏览器对WebFonts和SVG的支持，而不是使用缩放图像或维护不同的图标集，解决了这个问题。但当大量使用 SVG 时，需要注意移动设备的功耗问题和旧设备上的性能问题。

随着我们构建的系统比以往涉及越来越多细粒度服务，分布在越来越多的机器上，如何获得集中的信息以便更方便地识别和解决问题，比以往任何时候都更具挑战。Logstash已经成为一种解析和过滤日志再将它们转发到单一聚合节点的简单方法。虽然 Logstash 也提供了一些日志的搜索和过滤功能，但配合Graylog2使用，能提供更全面的日志查询和报告功能。

我们认为 Snowplow Analytics 在网络数据分析方面具有远大前景。它是一个开源网络分析平台，基于公开数据原则和云存储，并能从常规的网络分析中得出智能信息。

我们在 ThoughtWorks 的项目中看到对 PhantomJS 的兴趣，它是一个针对现实目标进行功能测试的 headless 网络测试工具。

Gatling 是另一个在自动化性能测试领域的新成员。它和 Locust 类似，都比传统的 JMeter 和 Grinder 更轻量级。Gatling 的DSL构建在Scala基础上，提供了易配置的数据源和响应断言等广泛的开箱即用功能。在需要个性化配置时，也可以很容易直接使用Scala 来进行扩展。通过 Highcharts 默认生成数据的各种动态视图的功能，更增加了它的吸引力。

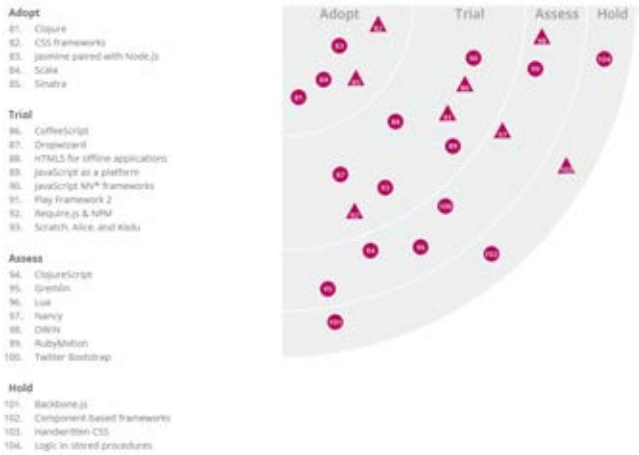
许多已经迁移到敏捷工作方式的组织还在继续使用重量级的测试工具。这些工具并不适合快速发展的软件交付。重量级测试工具较陡的学习曲线及对专业技

能和培训的要求，使团队很难使用这些工具来进行测试。通常这会导致随着其他团队的加入，对于每一个版本的发布都造成不必要的开销。这些工具昂贵的软件授权费用更加剧了这个问题。一些重量级的测试工具使用“模型驱动”的方法，试图精确地对应用程序的使用模式进行建模，但软件缺陷的误报却导致测试脚本昂贵的的开发和维护代价。其实我们经常看到一些简单的开源解决方案能以少得多的时间、精力和预算，给予对所要求的软件质量的信心。

基于语言的构建工具 Gradle 和 Rake 继续提供比Ant 和 Maven 等基于XML和插件的构建工具更细粒度的抽象和更灵活的长期解决方案。这使得它们在项目变得更加复杂时，还能保持优雅的增长。

我们继续看到尝试使用 TFS 作为版本控制系统的那些团队遭遇到生产力下降的问题。那些团队在实践作为持续集成核心部分的频繁代码提交时，发现TFS重量级的方式显著降低了生产效率。这往往导致团队更少地进行代码提交，导致产生诸多问题的代码合并。我们建议使用 Git、Perforce和 Subversion。

语言和框架



随着支持数据绑定、客户端模板和校验等功能的框架的蓬勃发展，单页面Web应用开发继续高歌猛进。在ThoughtWorks， 诸如AngularJS、Knockout和Ember.js 这样的 JavaScript MV* 框架有着广泛的应用，且各有其倡导者和反对者。我们期望在此充满活力的领域有更多的创新。

随着基于Node.js的服务器端应用程序的持续增长，单页面应用和基于移动浏览器的应用渐渐融入主流， CoffeeScript 也因其能简化 JavaScript 代码而越来越多地得到采用。作为一种将源代码编译成 JavaScript 代码以供运行时执行的语言，CoffeeScript应用程序的调试难度备受关注。CoffeeScript 1.6.1 引入了 Source Maps 以帮助开发工具生产厂商解决这个问题，同时，像Dropbox 这样的著名技术公司也开始使用 CoffeeScript，我们预计它将获得更广泛的应用。

我们在新产品中对node.js的使用越多，对可靠的 JavaScript 包管理工具的需求

就越迫切。Node Package Manager（npm）作为 node.js 应用的打包工具，是其生态系统的重要组成部分。对于要面对大量 JavaScript 或 CoffeeScript 代码的浏览器应用开发人员来说，应考虑使用 Require.js 组织代码结构以及在运行时装载依赖。

在客户端和服务器端，微框架正成为解决应用程序复杂度日益增加问题的新兴方式。Sinatra正是这一领域的先驱，它能够使用轻量级DSL在服务器端构建快速的、易于组合的服务。其它语言也有类似的框架，例如Java平台下的Spark、Python语言的Flask、面向Scala的Sclatra、基于Clojure的Compojure和 .NET平台下的Nancy。

在.NET平台上，构建一个多样的、开源的Web开发生态系统始终步履维艰，原因之一就是对IIS和ASP.NET 框架的过度依赖。OWIN之于.NET平台，正如Rack之于Ruby 社区，它定义了一个开放的HTTP操作接口，从而将Web服务器从应用程序中解耦出来。我们为此激动不已，因为OWIN打开了一扇大门，使得在.NET平台下诞生简单的、可独立开发模块的Web开发工具成为可能。Nancy 正是一个利用这种可能性的完美例子。我们也希望它能促使人们在.NET平台下将web应用程序部署为单独的、自托管服务方面进行更多的实践。

最近发布的Play Framework 2.1.1 支持控制器依赖注入、异步、非阻塞I/O、Code - Reload工作流、数据库迁移、资源管道以及灵活的部署选项，这些特性对开发者产生了更大的吸引力。基于这个原因，Play重新出现在我们的技术雷达上，我们希望团队在 JVM 上构建 Web 应用程序和服务时，认真考虑选用Play。然而需要提醒的是，Play 使用了函数式编程风格，在结合使用 Java时， Play 会产生过多的静态方法，因此很难在运行中的服务器外围进行单元测试。

和 JavaScript、HTML 一样，CSS 是构建网站的核心技术。很不幸，CSS 本身缺乏一些关键特性，这导致代码具有高度重复性并且缺乏有意义的抽象。虽然CSS3 旨在纠正其中的一些问题，但要等到大多数浏览器都正确支持 CSS3，恐怕还需很多年。幸运的是，通过 SASS，SCSS，LESS 这样的 CSS 框架可以帮助解决这个问题。基于这些框架的质量和相应支持，我们相信，除非需要细微的样式调整，手写 CSS 的时代已经一去不复返了。

CSS 框架简化了大规模 CSS 代码的开发，而无需每次都从头开始。面对琳琅满目的各种框架，选择一个支持持续增强与维护，而不仅仅是能够快速上手的框架就显得尤为重要。就这一点而言，基于 mix-ins 的框架如Compass，或者Susy这种有针对性的框架，将是更合适的选择。

Twitter Bootstrap 是一个流行的 CSS 框架， 它可以帮助你快速构建流畅的、具有响应式布局（responsive layouts）的精美网站。基于我们长期的使用经验，在此版技术雷达中，我们把 Bootstrap 从试用阶段移到了评估阶段。如果想要

替换或定制应用程序外观，其与 HTML 标记的深度集成将成为需要面对的挑战。但这并不妨碍你选择Twitter Bootstrap，只是做决定时需要考虑到这些限制。

原文链接：http://www.infoq.com/cn/articles/thoughtWorks-technology-radar

推荐文章 | Article

管理技术债

作者 Sven Johann & Eberhard Wolff，译者 张卫滨

技术债被广泛视为一件坏事，它应该避免或者要尽快进行偿付。

你应该这样做吗？我们并不这么认为。首先，我们对比了技术债与财务债，阐述了它与战略设计（Strategic Design）的相似性以及它的利益相关者。然后，我们列出了识别代码中技术债的各种可行的方式，这可能是你所关心的。

最后，我们描述了项目中可以偿还技术债的不同方式，并且阐述了当你在决定要偿还债务、转移债务或者只是支付利息哪个方案更好时，必须要考虑的因素。

什么是技术债

开发人员在实现新特性的时候，有两种不同的方式：一种是快速且混乱地完成，这会使得将来的变化很困难。另一种是整洁（clean）和明智的方案，它需要更长的时间来实现但是将来的变化会更为容易（亦可参见Martin Fowler）。不过，如果同一个特性以较为凌乱的方式实现时，能够具有相同的功能和较低的成本，那么项目的赞助者为什么会接受一个较高成本的整洁实现呢？他们为什么会花费金钱在自动化测试覆盖率上呢？测试不是什么特性，因此不会交付业务价值！

如果凌乱的代码和没有测试的代码能够交付期望的业务价值，对于客户来说也会运行得很好，但是这将会导致难以控制的代码基、极其专门化的开发人员并最终形成缺乏灵活性的软件产品。大量的混乱代码有可能会使得整个工程部门停滞（stand-still）。

“技术债”的比喻说法——与“财务债”的相似和不同

1992年，在与非技术的利益相关者沟通这个问题时，Ward Cunningham第一次使用了“技术债”这个比喻说法。低质量的或没有自动化测试覆盖的代码可以与

财务债进行类比。这样的代码就像财务负担，它会将所有的利益相关者牵连进来——并不仅仅是开发人员——债务会导致将来要支付的利息。本金就是将代码基重构为整洁的设计所付出的代价，整洁的代码会简化将来发生的变化。如果团队将来工作在混乱的代码基之上，相对于良好的代码基所付出的的额外成本就是利息。

与财务债的不同之处在于，你不一定必须要偿还技术债。有时候将其偿清并没有太大的意义：有些代码是很少甚至从来不会阅读或修改的。因此，技术债也需要考虑发生的概率——混乱的代码将来要接触到的可能性和频率如何？另外一个与财务债的不同之处在于，偿还这个债务的人并不一定是技术债的最初创造者，而是随后维护代码基的开发人员。

类似于财务债，技术债未必是一件坏事。如果你知道如何偿付的话，举债买房也是一种负责任的行为。但是，如果明明知道无力支付账单，依然通过信用卡购买大量的奢侈品，那么通常会导致灾难性的后果。考虑到软件的技术债可能会在提前释放版本方面带来优势，并且组织从中的获益要超过偿还技术债的成本。在财务上，如果本金加利息低于投资的收益，那么举债是一件好事。对于软件来说，这依然是成立的。如果你牺牲了内部质量来获取市场第一的地位，那么相对于市场中稍微靠后的位置但是具备较好的内部质量，你需要获取更高的收益，只有这样才是值得的。但是，这里会有风险，因为很难预先估计这些收益，这里会有一定程度的不确定性。

技术债与战略设计

Eric Evans所提出的战略设计理念为我们处理技术债提供了一个思路。战略设计（Strategic Design）表明在一个系统中，不可能在系统的任意地方保持高质量。因此，团队就有两种选择，要么对系统中哪一部分的质量高低听天由命，要么主动地对质量进行控制。很少会被读到或接触到且没有实现重要需求的混乱代码并不需要绝对得完美，因此我们没有必要花费大把的工夫将其重构为好的代码。所以，问题在于哪一部分的代码要具备高的质量？系统中的某个实现如果设计很糟糕，却想有好的质量是不可能的——如果这部分实现并不需要好的设计的话。这种观点当然是有问题的：如果代码的每一部分都不需要好的质量，那么最终你所创建的将是一个无法维护的系统。

理解战略设计和技术债能够使得项目内部进行更好地沟通，因此软件项目的利益相关者就能做出更好的决策。开发人员也会意识到如果要付出巨大努力的话，其实并不是所有的需求都需要非常优雅地实现。客户会理解快速且脏乱的解决方案将会导致债务的出现，而这最终是要偿还的。技术债包含了隐藏的技术问题，这就像冰山一样，一旦出现就会导致项目的失败，如出现大量的缺陷，就成本收益来说，这些质量问题解决起来可能太迟了。

技术债的利益相关者

技术债涉及到软件项目中的众多利益相关者：

- 客户（**Customer**）将会因为低生产率导致的缺陷或功能缺失而恼怒。
- 这会导致客服（**helpdesk**）的额外成本，因此也会导致那里的人产生不满。
- 对于市场部门（**marketing**）来说，增加的开发时间和质量问题也是一个麻烦。
- 缺陷会导致频繁地打补丁，这会引起运维团队（**operations team**）的厌烦。
- 众多带有恼怒情绪的当事人显然不会让管理人员（**management**）觉得很舒服——尤其是因为缺陷、延迟或安全问题导致很坏的影响时。
- 最后一项但同等重要的是开发人员（**developer**） 也会因此遭罪。没有人愿意交付糟糕的成果。另外，没有人愿意接手别人的糟糕成果。

技术债是不可避免的

但是，众多的项目为什么不在开始的时候就把事情做好从而避免技术债呢？假设开发人员并不懒惰同时对相关的技术和模式经验丰富，那么技术债基本上都是因为时间压力所导致的。开发人员可以借助快速且脏乱的实现获取较高的短期生产率，同时降低当前释放版本的成本——这实际上会导致将来的版本具备更低的效率和更高的成本，他们对此是心知肚明的。但是，团队永远不能确定当前的释放版本真的能够获取较高的生产率和较低的成本。可能的情况是，采取捷径产生的回力镖（此处的原文是boomerang，指的应该是采取短期方案所产生的问题提前爆发了——译者注）可能回来地比预期要早，因此团队要早于预期就支付利息。

但是，我们作为开发人员其实并不想在质量上有所损失或妥协。有时候，外部的条件使得必须要快速地完成版本释放，因为如果不这样做的话，我们就永远不会有下一个版本了。或者团队假设这些代码不会经常被读到或修改。前面所讨论到的“战略设计”明确允许在质量上的妥协。在那种情况下，好的代码可能就显得有些“画蛇添足（gold-plating）”了，这样做有些过犹不及。

因此，我们可以看到技术债的比喻说法能够帮助所有的利益相关者对糟糕质量的问题进行交流，否则的话，这些问题可能一直得不到注意，而意识到的时候就已经太晚了。糟糕的代码质量以及质量的持续退化所带来的危害通过利息这个比喻也能很好地得以展现。但是，这个比喻也有些欠缺：并不是所有的技术

债都需要进行偿付。团队也不能明确知道债务有多少以及何时需要偿还。欠下债务的人并不一定是要偿还债务的人，通常会是另外一个人。现实生活中的事情是难以预料的！

识别技术债

技术债的一个主要问题在于它并不是显而易见的。通过账户信息，任何人都能看出财务债的数额。但是，一个团队怎样有效地识别技术债呢？有哪些指标呢？

- 团队成员在自己的言语中所表现出的常见“坏味道”：“能够修改这块代码的只有Carl”、“让我们复制并粘贴这段代码就可以了”、“如果我动这块代码的话，那所有的地方就都不好用了”或者代码中有太多的TODO或FIXME（Nico Zazworka很好地描述了这一点。）
- Scrum团队所保持的速度。如果团队或外部环境都没有发生变化，但是速度降低了，这可能意味着软件系统有过多的技术债。
- 软件的寿命。技术债的一个指标就是系统使用了非常旧的库，这些库已经不再维护了或者新的版本更有生产率（如EJB2 VS EJB3）。在最坏的场景下，库太陈旧了以至于这些库的开发人员都不再支持它了。
- 自动化测量技术债在一定程度上是可行的。正确地配置像Sonar、SonarJ或者Structure101这样的工具能够发现严重违反最佳编码实践的地方。正如这篇博客所言，这并不是终极的解决方案，却是一个很好的方案。借助于Sonar，你能确保开发人员遵循了重要的代码指标，比如适当的类和方法规模以及较低的循环复杂度。像Structure101这样的工具会发现结构化的问题。比如循环依赖。如果两个元素互相依赖，任何一个元素的变化都会潜在地影响另一个元素。这两个元素必须要一起发生变化——尽管它们实际上应该进行分离并独立开发。此外，Structure101还会发现复杂的代码：它所关注的就是元素之间的依赖。如果一个系统的类或包有太多对其他类或包的依赖，那么这个系统会很难理解和维护。如果开发人员想要做一些变动，他必须要理解很多的类和包并且还要考虑很多依赖。这些问题其实就是架构中最主要的挑战。
- 代码覆盖度工具能够探测出自动化测试到底覆盖了多少代码。这个指标使用起来要当心，因为并没有通用的指导原则。一般来讲，超过90%的覆盖率就是就是一个好的信号，它标志着有足够的测试用例。相反，低于75%的覆盖率就标志着会有严重的问题。
- 但是，在很多场景下系统中有技术债，却并不能直接通过代码测量出

来，比如笨拙的解决方案、选择了错误的技术、解决方案的设计虽然执行得很好但设计本身很差、不容易被工具检测到的有害的hack方式。在这些情况下，技术债必须要以不同的方式来进行管理：每个释放版本的缺陷都在快速增长、工作速度在持续降低或者团队在接近版本释放的时候面临特别大的压力。
- 一个很坏的指标就是在生产环境经常出现问题。这意味着系统中的问题非常广泛以至于无法进行可靠的操作。

所有的这些指标都是可测量的，但是并不一定总是直接产生在代码之中。这些问题越早解决，带来的成本越低廉。如果开发人员发现了技术债并对此进行了交流，但是没有做什么针对性的事情，那么业务人员也将感知到这些问题。这个陈述清楚地表明：技术债并不是那些想编写漂亮代码的开发人员的噱头。它是真实的成本元素并且可能会成为项目的风险。因此，技术债必须可见并且可管理。在现实生活中，债务不一定是什么坏事，但是必须要慎重并恰当地使用。这意味着对于软件项目来说，偿付技术债完全是一个业务决策。它绝不是开发人员就能做出的简单决定。

如何管理技术债？

我们已经看到了，技术债不能视而不见。即便是客户那一边的非技术管理人员也必须要对技术债的管理保持关注，从而尽可能地保持短期、中期以及长期成功之间的平衡。一个团队如何既能避免不把时间浪费在不重要的美化上，又能做出有意义的业务决策从而提高其代码质量呢？

简单粗暴地说法，如“什么都不做”或者“如果代码不能再维护下去了，那就开发一个新的”，并不会提供什么帮助。

我们在本文中考虑了两种有前景的方式，在多个项目中已经发现它们很有用处了：

- 技术化的Backlog（Technical Backlog）
- 在需求评估的时候包含技术债的成本
- 在此之前，我们想要讨论在特定的上下文中很有价值的两种其他重要方法：
- 针对重构的缓冲任务（Buffer-task）
- 清理类型的释放版本（Cleanup-release）

当讨论技术债的时候，通常会有一个绕不过去的问题需要回答：技术债一定要进行偿还吗？Frank Buschmann 描述了3种策略，我们稍后会进行讨论：

- 偿付债务

- 债务转换（debt conversion）
- 只支付利息

缓冲任务（Buffer-task）

团队在每个释放版本中都创建一个缓冲任务，例如它占用可用时间的10%。团队成员可以将没有预定的重构时间记录在这个任务上。所以，它预留给未知的问题，而这些问题在将来会出现。这样的缓冲任务很容易安排和使用。但是，这也会带来一定的风险，那就是时间被浪费在了不重要的工作上。缓冲任务并不强制任何人考虑他们是不是将时间花在了有用的重构上。开发人员只是记录花在缓冲任务上的时间。最可能出现的情况是缓冲的时间并没有最好地加以利用——特别是要确定要做哪一个重构，尽管这应当是业务上的决策。令人遗憾的是，使用缓冲任务就意味着要做什么并没有被真正地定义。

清理类型的释放版本（Cleanup-release）

有些团队会不时地发布技术化的版本（technical release）来提高代码基的水平。如果已经列出了有必要重构的地方，那这种方式才会起作用。否则的话，团队会有风险将时间浪费在并不重要的重构上。这种方式也必须要得到业务方面的支持，因为它可能会延误新的特性。当然，这需要业务人员理解技术债。如果有些地方需要付出特殊努力的话，你应该考虑一个纯粹的清理版本来对代码基进行清理并对架构进行修订（rework）。例如，如果相同部分的代码在开发和运维期总是导致问题的出现，那么目前的架构可能已经不适合当前的需求了。这样的问题不能通过小的重构来解决。清理版本允许进行大量的变化。

在非常匆忙和时间要求比较重要的释放版本之后，发布清理版本是不合适的，这样的版本会有大量的技术债。对新的代码基只有很少的经验，所以没有人能够清楚哪部分代码需要提高。存在一种风险那就是修改了本来没有真正提高必要的代码。

技术化的Backlog（Technical Backlog）

技术化的backlog是一个既定的最佳实践，它用来定义纯粹的技术化工作。这种目的的任务会创建在任务跟踪或需求管理工具之中。每项任务会有一个简短的描述，包含了要做的技术变化、对于项目来说这些技术变化为何是重要的以及变化提高了哪一部分的代码。类似于其他的任务，我们还要评估开发足够好的解决方案需要多长的时间。除此之外，我们还要评估已有的代码会有什么样的利息。精确的评估是很难的，但是通常粗略的估计如“较小”、“中等”或者“较高”就足以指导做出决策。最后我们还需要一个概率：这些代码在不远的将来被

读到或修改的可能性？

这种方式有多项优势：

- 对每个人来说，技术都是可见和明确的。基于对难度的评估以及将来代码发生变化的影响，在是否以及何时完成重构任务方面，可以取得一致的决策。
- 每项任务的成本可以很容易地跟踪。
- 技术化的任务和特性任务间并没有混合。
- 但是，这种方式也有一些不足：
- 客户必须要确定backlog和任务的优先级。客户只有在一定程度上关注技术问题才能做到这一点，因为如果他不了解软件细节的话，就不能确定技术化任务的价值。
- 另外，客户可能也不能完全理解纯粹的技术化任务所带来的业务收益。在很多情况下，这里会存在一定程度的不信任感，那就是为什么一项与任何特性无关的技术化任务是真切需要的。

依我们来看，只有在一些特定的场景下，客户才能够进行决策，如软件更新。对于大多数客户来说，很显然过时的软件会导致问题。是否要进行一项更新具体取决于成本、风险以及更新的必要性。定期地更新像Java运行时环境（Java Runtime Environment）或O/R映射这样的组件通常不会太费事也不会有太大的风险。非常广泛的修改如替换Web框架或者将关系型数据库变更为NoSQL解决方案必须要有业务上的理由，比如应用的性能或用户体验必须要提高。为了达到这样的目的，技术化的backlog也会是很合适的。但是对于这样的任务你可能会初始化一个单独的项目，这取决于预期所付出的努力程度。

如果有一个新的需求要基于已有的代码基进行实现，但是这个代码基对新的需求来说设计得并不好，在这种场景下技术化的backlog就不是那么合适了。在这种场景下，在实现需求之前可能需要一个重构。为了做到这一点，在评估需求的时候，必须将重构所需要的额外努力考虑进去。

考虑技术债的需求评估

战略设计告诉我们并不是代码基的任何部分都需要很好，只有那些可变性会带来业务价值或者因为其他原因经常发生变化的那些部分需要如此。所以，很少或从不发生变化的技术债代码可以忽略。同样重要的是：如果没有好的原因，代码不要进行重构。所以，如果一个新的需求会带来切实的业务价值，那么我们就不要基于写得很糟糕的代码实现这个需求，而是要预先进行重构。按照这种方式，真正非常重要的需求能够实现为“足够好”的标准。当团队在将来的释放版本中要实现一个这样重要的特性时，应该将重构预算进去然后才是特性实

现。当然，这也有例外的情况，我们将会在“偿付还是不偿付”部分进行讨论。

这样的过程能够确保没有不必要的美化工作并且重构都是直接与需求相关的。这样客户就能与团队一起讨论并确定重构的优先级。客户也能决定一个需求是不是要基于债务和利息来实现。可能还会有其他的因素要考虑，如上市时间。客户可能也会明白基于这些需求，代码未来会发生的变化。为了节省将来的成本，他可能因此决定重构为整洁的代码。

在实现的过程中，如果团队遇到代码基的问题，那么将会产生新的问题。此时，团队必须要决定怎样做：可能按时交付承诺的功能是非常重要的。那么，就必须要接受因此导致的技术债并支付利息。或者，基于糟糕的代码来进行所有的开发会被视为一种浪费并且没有时间截止点的压力。那么，团队应该安排重构，这会导致一个稍晚的发布日期。这只能取决于环境尤其是客户的需求。

偿付还是不偿付

我们已经看到只有代码的重要部分必须要是整洁的。所以，代码应该只有在必要的时候进行重构。但是，对于特定的场景很难确定什么样的解决方案是最佳的。Frank Buschmann描述了3种策略：

- 偿还债务：将那些被视为技术债的代码、框架或平台进行重构或替换。
- 债务转换：将当前的解决方案替换为“好的，但并不完美的”方案。这个新的方案会降低利率。如果完美的方案需要特别高昂的代价才能进行构建的话，这可能是一个较好的选择。
- 只支付利息：与这些代码共存，因为相对于使用这些不太好的代码，重构的代价更为高昂。

开发人员和客户必须要基于成本、风险以及紧急程度来进行决策。总而言之，重要的重构应该是业务决策。关于支付利息，另外一个重要的方面就是：软件的生命周期结束时间（end-of-life）——如果一个系统马上要重建或下线的话，那重构还有什么价值吗？

结论

技术债可以被视为一个捷径，它可以在当下减少团队的时间、努力程度以及/或金钱，但是可能会导致将来成本的增高。在软件项目中，技术债并不能真正避免。如果处理得当的话，它并不一定是坏事。

做到这一点是很困难的：糟糕代码给未来需求所带来的影响很难甚至不可能进

行预估的。但是，完全忽视技术债会给软件带来灾难性的影响，它应该保持可管理的状态。我们已经展现了多种不同的方式来应对技术债。每种可选的方案只能用于特定的环境上下文中。接受以下几点是很重要：

- 技术债会始终存在
- 技术债并不一定总是坏事
- 并不是在任何的场景下都必须要对技术债进行（完全的）偿付。

任何的偿付行为都应该是业务决策。即便这个偿付很小，以至于没有必要与客户进行磋商，开发人员依然要反问自己，相对于花费的时间和努力，所带来的代码提升是不是值得。

本文的英文原文下有不少有价值的评论，感兴趣的读者可以移步查看。——译者注

关于作者

Sven Johann是Trifork Amsterdam的软件开发人员。Sven是XP实践的狂热采纳者，如结对编程、TDD以及小版本释放（small release）。目前，他正在基于Trifork的QTI引擎为瑞士以及荷兰的学校开发在线评估软件。

Eberhard Wolff是德国adesso AG的架构师和技术管理人员。他关注于Java、云计算以及软件架构。他是一些国际会议的固定撰稿人并且是多本图书以及文章的作者。

原文链接：http://www.infoq.com/cn/articles/managing-technical-debt

特别专栏 | Column

2013 Esri空间信息技术开发者大会：打造可持续的开发者平台

作者 水羽哲

位置应用正在逐渐获取更多的线下入口，Google对于Waze的收购也更加坚定了地理位置信息类的开发者的信念。在6月5~6日的2013 Esri空间信息技术开发者大会上，我们可以从地理位置技术平台的未来、GIS现状、ArchGIS开发等角度，一窥空间信息技术的现状与趋势。在会前InfoQ也参访了Esri的卢萌与何超，听他们谈Hadoop、Javascript与GIS的结合场景等。

目前，Esri正在做的是通过把GIS的服务云端化、智能化、个性化以及移动化等，让更多的开发者能够在这个平台之上用更少的成本获得更多的机会。对于如何构建一个有效、健康的开发者生态环境，Esri中国首席技术官王昊这样认为：

> 平台是规范与体系，开发者是持续的生命力。一个好的平台应该具有云、软件、数据、生态、社区、持续投入与可进化的特性，通过平台，开发者会与用户产生更多的互动，从幕后到台前，随时被聚焦！

围绕这样的平台，本次大会主要是为开发者推送最新的技术实现与最佳实践，主题包括：

- 桌面产品扩展以及开发
- 在ArcGIS中玩转影像
- 三维城市构建和应用开发
- ArchGIS for Server 面面观
- ArchGIS拥抱云时代
- 基于ENVI/IDL的遥感应用以及开发技术
- ArchGIS移动，变化于多端

通过浏览Esri的公开课来获取GIS开发的更多信息，目前Esri也在积极推动他们的开源项目的进展。

• 想成为技术**牛人**？ • 想获得技术**干货**？ • 想结识技术**圈内朋友**？

百度技术沙龙

与技术大咖一起，讨论时下技术热点

牛人 + 大咖 + 圈内朋友 + 热点 + 干货

百度技术沙龙第40期

7月27日，贝塔咖啡，一起来聊聊LBS

salon.baidu-tech.com

畅想 • 交流 • 争鸣 • 聚会

百度技术沙龙是由百度主办，InfoQ负责策划、组织、实施的线下技术交流活动，每月一期，每期一个主题，由2场演讲以及Open Space开放讨论环节组成。旨在为中高端技术人员提供一个自由的技术交流和分享的平台。每期沙龙会邀请1名百度讲师分享百度在特定技术领域的成果及实践经验，同时还会邀请1名优秀的互联网公司或企业技术负责人对同一话题进行分享。活动主要面向开发者、技术负责人、项目经理、架构师等IT技术人员。

新浪微博：infoqchina

微信帐号：bdtechsalon

特别专栏 | Column

Esri卢萌谈Hadoop在GIS数据处理中的运用

作者 张月

在现代人的日常生活中，缺少不了地图的身影，而地图已经超越了简单的工具属性，它的价值正在逐步被大家挖掘出来，例如：通过Nike+，你可以看到自己每天的运动路径和热量消耗、商家可以通过地图对消费者进行定向广告投放等。越来越多的互联网公司或个人开发者尝试通过结合地图去做更多不一样的事情，对于地图数据本身也有很多有意思的事情，InfoQ借此机会采访了Esri的卢萌，听他谈谈在GIS数据是如何运用Hadoop进行并行处理的，他也会在6月5~6号的Esri空间信息技术大会上对这个问题做分享。

InfoQ：请您向InfoQ的读者做下自我介绍吧！

> 卢萌：我叫卢萌，2005年大学毕业后就进入了GIS这个行业，碰巧的是2005年正是谷歌地图发布的那一年，各种WebGIS在中国开始大行其道，所以我应该算是国内做WebGIS比较早的那一拨人了。在这些年中，做了不少项目，后来进入了Esri中国，目前在Esri中国政府行业技术部担任高级产品咨询师的职务，主要研究的方向是传统GIS领域中的空间分析、空间统计等技术，同时也对云GIS、海量数据挖掘等新技术应用有过一定的涉猎。

InfoQ：请问卢老师GIS领域目前国内外发展情况如何？

> 卢萌：借用WEB时代的说法，我们也把网络时代的GIS发展，分为1.0、2.0和正在迈向的3.0时代。
>
> 1.0时代，GIS是专业人士玩的东西，主要是通过一小批专业人士做出各种地图或者应用，然后通过网络发布到出去，广大的用户通过浏览器来查看地图，也就是一个人发布，大家来看。 2.0时代是以交互为标志的时代，GIS也是一样，在业界，Esri最先推出的ArcGIS Online这类的平台，提出了空间信息共享和协作的概念，我们能够很轻易的获取到各种专业的空间数据，并且这些数据的来

源除了专业机构，还有广大的使用。在这个平台中，每一个用户都成了空间信息的发布者，也是信息的享用者。

而未来的3.0时代，将以一种更加智能的方式来获取、使用空间信息，并且能够直接获取到我们需要的结果，而不仅仅是一堆地图和数据。3.0时代是以服务为核心的时代。就像我们购买了一堆服务器，一堆软件，部署了各种环境，编写了很多代码，那么我们的目的是什么？是想要那些服务器？那些软件和那些代码吗？当然不是，我们是通过这些基础建设来解决我们的实际问题，获取需要的结果。3.0的时代，我们更多的将这些基础建设、分析模型和处理过程托管到以网络为基础的云环境中，我们只需要提出问题，然后获取结果就可以了。

InfoQ：Hadoop目前涉及的科学领域很多，不仅包含众所周知的互联网行业，也包括如能源开采、节能环保、生物医学等领域。请问卢老师，Hadoop在GIS（地理信息系统）这个领域有哪些典型的应用场景，又是如何实现呢？

卢萌：传统的GIS数据存储，大多依托于各种关系型数据库，但是数据库由于其在海量数据管理、高并发读写、难以扩展等方面，已经开始的制约了GIS的发展。而且关系型数据库一直在非结构化数据领域难有作为，这一点更是致命伤。世界上90%以上的数据都是非结构化的，GIS里面大量数据如影像数据等，都是非结构化的。 而Hadoop以其高可靠性、高扩展性、高效性和高容错性，特别是在海量的非结构化或者半结构化数据上的分析处理优势，给我们提供了另外一种思路。 Hadoop的核心算法就是“分而治之”，这个与GIS里面很多算法是相通的，GIS里面很多应用场景都是要去分析不同区域内的各种信息，把这样的计算放到Hadoop上，正好利用了Hadoop的分布式计算特性。特别是一直让GIS界最头痛的海量影像数据存储和分析，在Hadoop的分布式存储和分布式运算架构上，更是能够体现出Hadoop在GIS应用上的优势。

InfoQ：Esri公司推出开源环境“GIS Tools for Hadoop”，其中还包含了基于Hive的UDF，这将给地理信息这个行业带来哪些新的机会呢？

卢萌：GIS Tools for Hadoop是Esri推出的基于Hadoop上的一套完整的空间大数据量处理的环境，包含有一套工具、一套API和一系列的框架。此环境的推出，扩展在Hadoop上geomtry的类型和对空间数据的操作，让开发人员能够方便的在空间数据上构建MapReduce应用程序。

Hive是由Facebook首先研发的一个基于Hadoop的数据仓库工具。可以将结构化的数据映射成数据表并提供类SQL的语言进行数据查询，适合于数据参考的统计分析。GIS Tools for Hadoop扩展了Hive在Esri Geometry API上的应用，

使用户可以通过hive进行空间数据的处理。

数据仓库一直都是一个比较热门的领域，而在空间数据库上构建数据仓库，进行数据挖掘，更是一个比较新的话题。一个单位中需要在海量空间数据上做分析有各种不同的人员，有工程师也有专业数据分析人员，他们之间的计算机知识层次不一。所以构建一套满足各种人员和程序需求的基础设施就很有必要。Hive在维持Hadoop灵活性和扩展性的基础上提供类似关系型数据库的基本功能，在Hadoop文件系统上提供了方便的数据查询和管理功能。

GIS Tools for Hadoop中包含了基于Hive的UDF（User-Defined-Function）这一功能，可以让开发人员和数据分析人员能够构建面向业务的分析应用，在GIS中有很多特定的算法和模型，通过UDF都可以加入到Hadoop中去。这样看来，未来势必会使用它来定制开发和搭建各种面向业务的个性化工具和软件。

InfoQ： 既了解**GIS**又熟悉**Hadoop**开发，这样的人目前是不是比较稀缺？请您给大家一些学习和就业方向上的建议吧。

卢萌：虽然经过了这些年的发展，熟悉GIS的开发人员一直还是供不应求，而Hadoop在互联网上作为“大数据分析”的代名词，熟悉Hadoop的人员更是稀缺。据麦肯锡全球研究院的研究预测在未来6年，仅在美国就可能面临缺乏14万至19万具备深入分析数据能力人才的情况。

对于Hadoop来说，最大的优点就是开放性，因为它是开源的，所以每天有数以万记的程序员和爱好者在学习和研究Hadoop系统。作为一个新手，要学习Hadoop，最重要的就是动手去操作，从安装部署到优化到开发，一步一步的掌握MapReduce编程方式和各种应用场景，了解Hadoop的实现原理和算法。当然我们还可以去阅读他提供的源码，这也是开源软件最大的好处。

学然后知不足，教然后知困。我们可以在一些社区或者博客上记录在学习过程中的点点滴滴，当有朝一日，回过头来看我们曾经走过的足迹，你会发现，在不知不觉中，当年的小树苗已经成长成为了参天大树。

虽然目前IT界主流的开发还是依托在关系型数据库上面的，但是我们可以在工作中去逐步推广Hadoop的思想和应用，去潜移默化的改变周围的环境，未来Hadoop前景一定是光明的。

InfoQ： 除了**Hadoop**外，**GIS**还有其它并行计算的解决方案吗？

卢萌：从ArcGIS 10.1开始，ArcGIS for Server变成了纯64位的架构，即不再支持32位操作系统。 另外在ArcGIS 10.1的SP1包中，提供了ArcGIS 10.1 for

Desktop—Background Geoprocessing (64-bit)补丁包，实现64位后台地理处理，并且增加了新的“并行处理因子”环境，可以分跨多个进程来划分和执行处理操作。在10.1 SP1中新增支持使用最多4核处理器进行并行处理的功能，现已实现于一些空间分析工具当中。 而在即将发布的ArcGIS 10.2中，更是更新了600多个软件节点，其中主要集中于软件的质量和性能增强。

InfoQ：最后一个问题，请问卢老师，您眼中的大数据的未来是什么样的？

卢萌：大数据是现今非常时髦的IT词汇之一。什么叫大数据？如果单纯以量来看，在以MB为单位的硬盘存储量的时代，TB甚至是GB可能就算是大数据了，而现在TB硬盘普及的时代，恐怕要PB甚至是EB才算大数据了。 在IDC最新的《数字宇宙》（Digital Universe）中预计，到2020年，全球的数据存储总量将达到35ZB。而且这个数字还是受到了存储能力的限制。 对于大数据，IBM提出了“三V”的概念，即大量化（Volume）、多样化（Variety）和快速化（Velocity）。最近这两年，着眼于数据应用的专家们提出了“四V”的概念，就是在原有的“三V”的基础上加上了Value（价值）。 毋庸置疑，最后一个V（Value），才是我们最值得关注一个V，做数据挖掘和数据分析的主要目的也就是为了实现数据的价值。 未来是一个大数据的时代，在不久的将来，数据可能成为最大的一类交易商品。在互联网上，继“流量为王”、“应用为王”之后，下一个概念应当就是“数据为王”。

原文链接：**http://www.infoq.com/cn/news/2013/05/hadoop-usage-in-esri**

相关内容

- 虚拟化和云计算让Hadoop变得更弹性
- Windows Azure更新，支持Hadoop、HTML5/JS、CORS、PhoneGap、Mercurial和Dropbox
- 基于Hadoop的内部海量数据服务平台
- Greenplum Pivotal HD结合了SQL和Hadoop的优势
- Hadoop的现在和未来

特别专栏 | Column

Esri何超谈基于ArchGIS的Javascript API

作者 李雪韬

地理位置信息服务本身具有良好的线下和线上属性，通过暴露数据接口的API实现多终端、多语言的调用，可以将其应用场景扩大化。而Web端在各种需求场景中又是最常见的，在6月6日的Esri空间信息技术开发者大会上，来自Esri的何超以Dojo Widget和Angular.js的开发流程为出发点，分享了他们对于构建大型Web地图应用的一些实践和想法。InfoQ借此机会采访了何超，听他谈谈ArchGIS Javascript API：

InfoQ：我们知道ArcGIS是一款非常给力的国际化地理信息系统软件，在提供Javascript API的时候，与ArcGIS本身的数据结构相比，有哪些特殊处理？栅格、矢量地图数据的存储方式分别是怎样的？

> **何超：**ArcGIS本身提供了从ShapeFile到企业级数据库的各种存储方式以满足不同用户不同场景的需求。Javascript API在处理栅格和矢量地图的时候并不直接去读取Map Data，更多的是依附于ArcGIS Server所提供的RESTfull风格的各种不同类型的服务，如地图服务（Map Service）、影像服务（Image Service）等。因此，对于JS的开发者来说，大可不必去关心底层的数据是如何存储的，只需要了解ArcGIS Server所暴露的服务的相关信息即可。举个例子来说，对于大数据量的影像数据，我们在后台可能采取的是镶嵌数据集（Mosaic Dataset），这种方式进行存储，但是Arcgis Server并不会直接将地图数据暴露给用户，提供给用户的是一个类似于`http://sampleserver6.arcgisonline.com/arcgis/rest/services/CharlotteLAS/ImageServer`这样的地址。用户是通过ArcGIS Server 的RestAPI来实现客户端与ArcGIS Server之间的交互。 JS API所接受的是ArcGIS Server所返回的图片或者是json数据。而对与用户本身在前端所加载一些譬如gps轨迹等数据信息，都是以json的格式处理的。

InfoQ：地图未来很重要的一个发展趋势是实景、三维、立体。那么，基于ArcGIS的JS API，会提供哪些地理三维模型的API方法呢？比如实现街景的

图层拼接和建模的底层API接口、或实现全3D效果的地理物体，比如山川、立交桥、建筑物等。

> **何超：**关于Web 3D，这个一直是Esri中国研发中心在开发，会在今年的开发者大会上给大家带来精彩的Demo演示。从我目前了解的情况，3D API跟之前的2D API非常类似，开发者可以很容易上手，其代码也很容易移植。Web 3D的API在原有2Dapi的基础上扩展了Geometry、Layer、Symbol、Renderer等，例如，symbol针对3D场景新增了BoxMarkerSymbol、CylinderMarkerSymbol、SphereMarkerSymbol、MediaMarkerSymbol。3D Map将平滑加载原有的2D Layer，支持跟2D几乎一样的Geometry和Symbol，以及更多的3D Symbol,如Box、Cylinder、Sphere等。同时针对Layer，Web3D新增了支持3D Model（如城市模型）ArcGIS3DServiceLayer和ArcGISElevationServiceLayer，以及贴地模式的Graphic。此外，3D API无缝结合2D API，部分类甚至可以共用（如InfoTemplate和Query等)。Web 3D的API还会提供一个类似于Flex Viewer的3D Viewer，实现3D应用的快速搭建。3D的API紧密的跟Esri的CityEngine、ArcGIS Server结合在一起，实现从建模到服务再到应用的一体化。

InfoQ：我们知道，现在移动终端发展很快，iOS、android两大平台的JS运行时差别较大，很多标签和接口的执行效果差别明显，webkit内核的实现在各个平台上都有所不同，性能差异很大。 那么，基于ArcGIS的JS API，在移动终端有哪些特殊的性能优化、机型适配方面的考虑呢？

> **何超：**为了适配移动端的需求，ArcGIS API针对移动互联网做了很多的优化。首先就是对API library的大小进行了优化，提供Compact版本的API（~125kb）。至于浏览器和平台的差异，mobile版本的API也是基于dojo mobile的，所以这种浏览器的差异性就交给dojo来处理，可以说Dojo支持的移动设备我们API都是支持的。可以使开发者尽量少的去关心浏览器的差异，实现对开发者的友好。
>
> 举个例子来说，ArcGIS API在不同的平台上采用什么样的绘图引擎来绘制2维地图？我们知道不同webkit上在2D绘图上的选择是很多，可以用Canvas、可以用SVG，那对于不同的平台究竟用那一种？针对这种问题，ArcGIS JS API会基于dojo提供的dojo.gfx根据浏览器的不同来自己去判断、选择最适合该浏览器的绘图引擎。
>
> 再举个例子，ArcGIS API在PC浏览器上的信息弹出窗口POPUP和InfoWindow是很大很大的一个div，但是Mobile版本的API针对移动设备屏幕小的特点，提供了Infowindow lite和mobile popup这两个widget。

> 在对移动设备的兼容性上，目前3.5版本的API实现了对Android 2.2+、Blackberry 6+、iOS 3+等平台的全覆盖。此外，esri还体现了极大的兼容性，可以用在其他的移动端框架譬如（Sencha Touch、jQuery Mobile）的时候来调用我们的api。所以，可以说api在对移动设备的兼容性还是很好的。

InfoQ：ArcGIS功能强大，但是如果使用正版，需要很高的费用，不知在JS API方面的收费情况是怎样的？未来会以怎样的形式与互联网公司合作？会成为一个独立运营的平台吗？

> 何超：ArcGIS一直对开发者十分友好，所以JS API在用作开发、测试、评估等非商业用途的时候，是不收取开发者任何费用的。对于利用JS API开发的网络APP或者是移动APP需要用于商业用途的时候，则需要购买ArcGIS Server或者是需要购买ArcGIS Online的订阅（subscription）。 Esri一直以积极、开放的姿态，期待并正在与一些互联网公司展开合作。GIS中不仅仅有地图，在地图之外，还有地理，人文地理、自然地理、经济地理等。Esri希望依托ArcGIS Online这个云GIS平台，结合ArcGIS的强大的地理分析、预测功能，为互联网公司提供除地图之外的，更多专题信息、更多具有“地理”印记的信息。

原文链接：**http://www.infoq.com/cn/news/2013/06/archgis-js-api**

相关内容

- MDC Windows Phone专场：基于地图API的开发及经验分享
- WSO2 API Manager 1.4.0 引入多租户
- 采访Ross Mason：MuleSoft的新API平台
- Google发布用于Google Glass的Mirror API
- API社区关于治理与创新的争论

避开那些坑 | Void

绕开 Android 应用开发的那些“坑”

作者 “豌豆洗白白”团队 发布于 六月 03, 2013

自从推出“豌豆洗白白”之后，我们也在不断听取用户和开发者的意见来改进产品。不过，部分开发者提出的奇怪的场景，也会让我们感到疑惑：比如突然更换了签名文件，或者包名与别的应用撞车造成的误洗等。这些事故，很多情况下是可以避免的，所以我们也在本文中总结一下 Android 开发中容易被忽视的一些注意事项吧：

一、谨慎选择包名

包名 (Package Name) 就相当于一款应用在户口本上登记的名字，是系统用来区分不同应用的字段。重复的包名会被认为是同一款应用，不能同时安装在一个系统里。一般来说，包名建议使用域名反序。比如豌豆荚的网站域名是 www.wandoujia.com，那么豌豆荚的一系列 Android 应用的包名就是 com.wandoujia.***。这种约定俗成的办法可以最大程度地避免“撞名”的事故。

案例一

某个手机 ERP 客户端的开发者，在开发应用时使用了 Adobe 的工具，同时也参考了 Adobe 的官方教程，从而将应用命名为 air.Main 这样的通用包名，跟另一个游戏重名了。于是，在升级和洗白白的过程中就都产生了混乱……所以取名要慎重啊！

案例二

还有个开发者，不知为何给几个不同的应用都使用了同一个包名。于是“com.hexin.qs.app.android”这个包名，对应了“申国万银证券”、“中信建投手机证券”、“联讯证券手机炒股”等好几个完全不同的应用。带来的后果就是，用户不能在手机上同时安装这几个应用，而且在升级过程中也容易从一个应用升级到另一个应用上。所以，不能给好几个孩子取同一个名字啊！

二、一款应用只应该有一个签名

签名文件就相当于开发者的“身份证”，我们平时生活中应该只有一张身份证，那么在手机里也一样，签名也应该是唯一的。而且因为签名的加密机制，使复制和伪造签名变得几乎不可能，所以“豌豆洗白白”也是将签名作为判断是否官方版的最主要标准之一。

如果一个应用使用了不同的签名，那么对于用户来说，可能会造成应用无法升级，因为 Android 系统会在升级应用的过程中比对签名，不同的签名无法升级。如果用户选择强制升级的话，因为签名不同，所以需要先卸载旧版再安装新版，那么之前存在本地的应用数据、游戏记录等也会直接丢失。

案例一

就好像我们会不小心弄丢身份证，开发者也会不小心弄丢签名文件。我们曾遇到过四五例类似的例子，开发者找上门来说要求更换签名，原来是因为工程师离职把签名文件带走了，老板要不回来了，只好把原来的签名也全都改一遍。所以签名文件要保管好啊！

案例二

我们曾经统计过，网游“我叫 MT”在各大市场上一共有 48 种不同的签名。据“我叫 MT”的开发者说，他们是通过不同的签名来区分不同的渠道的，而一共放出过多少种签名难以统计。这么多个不同的签名文件，除了自己管理起来非常麻烦之外，也给用户的升级过程带来了困扰，一不小心升级到了不同签名的版本上，游戏记录就全部丢失了。

Android 签名机制的目的是为了检验应用是否被人更改过，而不是为了让你区分渠道的。那么正常来说应该用什么方法来区分渠道呢？请看下一条。

三、使用 xml 配置文件来区分渠道

一般来说，我们在 `Android manifest` 文件中使用 `meta-data` 来区分渠道。例如：

```
<meta-data android:name="CHANNEL" android:value="wandoujia"/>
```

每次正式打包完成后，修改 `android:value`，再重新打包即可生成一个新的渠道包，所以：

> 不要再用签名来区分渠道了!

四、正确填写版本号

在 Android 应用中，有两个参数与版本号相关。其中，version Name 表示版本名称，是字符串，version Code 表示版本号，是整型数字。

一般来说，用户直观看到的是 version Name，所以这里应该填写形似“4.15.1”这样的版本号。而真正用来判断新版本旧版本的参数是 version Code。在应用发布第一个版本的时候，version Code 应该填 1，然后每次发布的时候都递增，这样才是以规范的格式告诉各大市场你的应用的更新程度。

有的开发者在 version Code 上非常随意，这个版本发布的时候碰上结婚纪念日，于是用老婆的生日当 version Code；下个版本发布的时候运气不太好，于是用自己的幸运数字当作 version Code……这样在用户看来的结果就是，明明从官网安装了最新的 2.2.0 版本，可是各大市场却提醒“升级到 2.1.3 版本”，越升级版本号越小了。

以上四件小事，虽然都比较琐碎，而且没什么技术含量，但却是很多入门的 Android 开发者容易忽略的问题。希望本文能帮助广大开发者，在写出好应用的同时，避免这些“坑”给应用带来不必要的损失。

关于作者

“豌豆洗白白”基于豌豆荚对应用特征的深入了解，致力于帮用户找到干净无通知栏广告的官方版应用。“豌豆洗白白”团队由 8 名豌豆组成，平均年龄 27 岁。项目负责人丁吉昌是豌豆荚创始团队成员。

原文链接：http://www.infoq.com/cn/news/2013/06/bypass-android-dev-pits

相关内容

- DCCI与UC联合发布2012中国Android用户数据报告
- MDC Android专场：账户同步备份框架与Web&Native混合开发
- 移动应用平台首选iOS还是Android？
- 社区讨论：Android应用和iOS应用的开发成本比较
- 扩展Android框架功能和底层模块的技术要点

避开那些坑 | Void

影响可扩展性的十宗罪

作者 郑柯

Sean Hull是一名工作于纽约的技术咨询顾问，同时也是Oracle & Open Source一书的作者。最近，他在自己的博客上发布了一篇文章，指出5种“罪过”会对可扩展性产生致命影响。他在2011年发布过一篇文章，其中也提到5种罪。加起来，正好是影响可扩展性的“十宗罪”。接下来我们一一列举，看看到底是哪十宗罪。

第一宗：磁盘I/O慢，使用RAID5，使用多租户EBS

磁盘是所有服务器的基础，也是服务器性能性能的基础。虽然主内存变得越来越大，很多都可以作为缓存使用，但是服务器仍然需要不时从磁盘上读取数据，从内存清出数据。所以磁盘对于性能和可扩展性非常重要。

Raid 5有什么问题？

Raid 5是为了用更少磁盘提供更多的空间。常常用于磁盘插槽比较少的服务器，或者就是因为运维人员不知道它对性能的影响有大。在数据库服务器上用会特别不好。

所有的写操作都会影响性能。更大的问题在于：如果你失去了一块磁盘，虽然RAID原则上还在线，但是会**巨慢无比**，就跟掉线了一样。重建需要耗费N多小时。更糟糕的是：在重建过程中，可能会有另外一块磁盘也失效。要是你刚订购磁盘要好几天才能到，那该怎么办？

解决方案是采用RAID 10。每两个磁盘做镜像，然后对它们做Stripe操作。即使只有四个硬盘插槽，这么做也值得。读写性能都很好，并且有保护。

多租户的问题是尼玛怎么回事？

在云上，可以共享服务器、网络和磁盘，就像共享某间公寓一样。Amazon的EBS（Elastic Block Storage，弹性块存储）是对这个比喻的扩展，为你提供存储网络令人欣喜的灵活性。但是你的瓶颈就是：与其他租户争抢同一个存储网络。

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

Amazon提供的默认服务器存在这个问题，不过AWS解决了这个严重的问题，用一个不太为人熟知但是**超级有用**的特性——Provisioned IOPS。用它你可以锁定可靠的磁盘I/O。

第二宗：用数据库处理队列

MySQL在很多地方都做得很好，但是在处理应用程序排队方面却并不理想。你的数据库中是不是有类似JOBS这样的表，其中有一个状态列，包括“queued”、“working”、“completed”这样的值。如果是，你可能把数据库来处理应用中的队列工作了。这样使用MySQL不好，因为会出现锁的问题，还有查找下一个任务时的搜索和扫描任务也会遇到麻烦。

建议使用RabbitMQ或者Amazon的SQS方案，因为这些外部服务更容易扩展。

第三宗：用数据库进行全文搜索

Oracle提供全文搜索支持，为什么MySQL却不能用呢？MySQL**确实**有这项功能，但是在很多版本中，只能配合老的MyISAM存储引擎使用。最好采用Apache Solr等经过验证的搜索解决方案，它专门用作搜索，有非常好的库，开发者可以使用多种现代web语言进行开发，并且非常容易扩展。只要在网络中增加服务器，或者做整体分布即可。

对于前沿技术感兴趣的同学，MySQL 5.6中提供了Innodb的崩溃安全和事务存储引擎。既便如此，还是建议使用外部解决方案，如Solr，或者Sphinx及MySQL Sphinx SE插件解决。

第四宗：每个层次的缓存不够

缓存，缓存，及更多的缓存。在web服务器和数据库之间可以使用memcache或者其他对象缓存机制。所有这些小结果集将会主流在内存中，等待未来需要它们的web页面。

也要考虑使用varnish这样的页面缓存，它位于web服务器之前，不妨将其看成一个迷你web服务器，处理很简单的页面，但是速度非常快。就像在堵塞的道路上开摩托车，让你的web服务器加速以完成更复杂的工作。

浏览器缓存也非常重要，但是你无法控制用户的浏览器，也许你可以？当然不是直接控制，但是你可以指导他们缓存哪些东西。还要使用合适的过期报头。让你的管理员配置Apache来支持这一点。

第五宗：太多技术欠债

技术欠债总有一天要还的。这是什么？探索新的点子，你会搭建一个原型。当这个原型已经部署给客户使用之后，要修改就变得更困难，因为某些问题可能有些东西还得掩盖起来。一个团队离开，另外的人接手这个应用，问题变得更复杂。随时间推移，你的技术债务不断积累，团队会花更多时间去维护老的代码以及修复bug，开发新功能的时间会越来越少。在某个时候就有必要重写代码了。

第六宗：对象关系映射器

ORM很受开发人员的欢迎，但是对于性能专家来说却不是这样。为什么会这样？原因主要是由于这两类人看待web应用视角不同。一种是功能开发，以满足业务需求作为考察的结果。性能和可扩展性在这种情况下往往位于较低优先级。ORM能让开发者更有效率，通过后端的数据存储将SQL的交互难度抽象出来，让开发者将注意力集中在功能的开发上。

从性能角度出发，看到的是另外的场景。如果用ORM实现SQL查询，数据库将无法优化复杂的查询。而且ORM不允许对查询的简单调整，拖慢调优过程。

第七宗：同步、串行、耦合或锁定进程

在web应用里，锁相当于现实世界中的交通信号灯。如果将交通灯换成环形交通枢纽可以显著提升流量。因为如果你去一个交通流量很小的地方，没有人会在交通灯下做无用的等待。更重要的是：如果有车流量很大，环形枢纽可以让车辆保持流动。如果需要锁，最好使用InnoDB数据表，因为它们能提供粒度更细的行级锁定，而不是MyISAM的表级锁定。

要避免出现等待其它节点消息而无法执行程序的半同步复制。在高事务并发的web应用中，成千上万的并发会话会导致这种等待的增加。

避免任何类型的两阶段提交机制。多阶段提交提供了一个串行入口，让多个节点投票决定数据应该是什么样子，但是它们是扩展性的毒药。最好使用采取最终一致性算法的技术。

第八宗：只有一份数据库拷贝

如果没有复制机制，那么你的数据库就只有一份。所有的web服务器只能使用唯一的后端数据存储，它就会成为漏斗或瓶颈。

第九宗：没有定量衡量标准

没有衡量标准是可扩展性的一剂毒药，因为你无法以可视化方式看见系统中发

生了什么。没有可视化的线索，业务部门、开发者和执行团队很难对扩展性方面的问题达成一致。如果团队认识不到这一点，要让大家理解：这些简单的工具可以提供基础设施的分析。

第十宗：缺乏功能标志

应用程序如果缺乏功能标志，会很难实现优雅升级。如果你的网站流量猛增，而你不能施展魔法来扩展和提升流量，有内建的功能标志，运维团队就可以在不让网站宕机的前提下，降低服务器负载。你也得以有更多时间来扩展web服务器或是数据库，甚至是改进你的应用，允许同时读写多个数据库。

不提供这些开关，扩展性和可用性就受到了很大限制。

以上就是Sean Hull提出的：影响可扩展性的“十宗罪”。各位InfoQ的读者，以上这“十宗罪”，你在处理扩展性的过程中遇到过吗？你觉得还有哪些“罪”？又是怎么解决这些罪过的？欢迎在留言中说明。

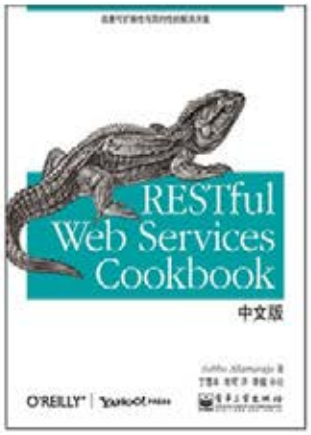
原文链接：http://www.infoq.com/cn/news/2013/06/10-sins-for-scalability

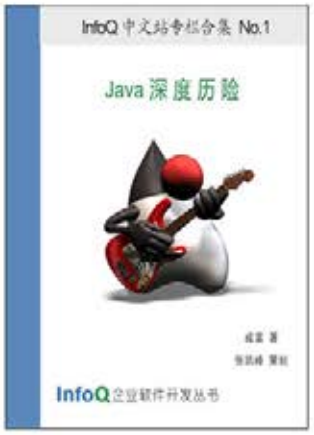
相关内容

- Google提供Cloud SQL，借此扩展云服务
- 曾宪杰谈Java在淘宝的应用
- Zend 开发者2013调查关注HTML 5、地理位置和Amazon Web Services
- 亚马逊面向托管和大数据处理的大动作
- 针对.NET开发者的NuoDB 1.1发布

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

esri China BEIJING

分 享 地 理 价 值

2013年Esri中国技术公开课

2013年的技术公开课也已经拉开了帷幕。全年的时间安排为2013年4月–12月，每月两场（分别是第二周和第四周的周二下午）。今年的课程采用视频录播的形式，您可以关注当日发布在gisall的最新课程，也可以随时回顾以往的课程。今年的课程在内容方面增加了行业应用、传统产品解析、IT 新技术方面的选题。通过网络投票的形式让观众来选择最想听的课程，入选课程将优先投放。

2013Esri技术公开课新举措：

1、课程内容选材更丰富
2、录制哪些课程由观众做主
3、课程网络观看更流畅
4、建立QQ群（178608911）让讲师与大家充分交流
5、向积极参与者颁发各种奖励

观众选中的课程将在
gisall网站视频频道投放
欢迎参加！

视频频道在线观看
www.gisall.com

Esri中国信息技术有限公司
电 话：010–57632000　传 真：010–57632299

新品推荐 | Product

Google 发布新一代Web UI库Polymer

作者 Zef Hemel ，译者 廖煜嵘

在今年的Google I/O大会上，Google发布了Polymer，它是一个使用Web组件构建Web应用的类库，同时也使用了为Web构建可重用组件的新的HTML 5标准。Polymer为大部分Web组件技术提供了polyfills功能，它能让开发者在所有的浏览器支持新特性前创建自己的可重用组件。

原文链接：http://www.infoq.com/cn/news/2013/06/webcomponents

Facebook发布用户界面库React，业界褒贬不一

作者 Zef Hemel ，译者 臧秀涛

Facebook开源了React，这是该公司用于构建反应式图形界面的JavaScript库，已经应用于构建Instagram网站及Facebook部分网站。最近出现了AngularJS、MeteorJS 和Polymer中实现的Model-Driven Views等框架，React也顺应了这种趋势。React基于在数据模型之上声明式指定用户界面的理念，用户界面会自动与底层数据保持同步。

原文链接：http://www.infoq.com/cn/news/2013/06/facebook-react

Yahoo!开源运行在Hadoop上的Storm——Storm-YARN

作者 Boris Lublinsky ，译者 臧秀涛

上周，Yahoo!宣布开放运行在Hadoop集群上的Storm（即Storm-YARN）的源代码。该实现支持Storm应用利用Hadoop集群的计算资源，还支持Storm应用直接访问存储在HDFS和Hbase上的Hadoop数据。

原文链接：http://www.infoq.com/cn/news/2013/06/StormHadoop

PhoneGap 2.7发布，官方弃用1.9版本

作者 Jonathan Allen，译者 臧秀涛

Adobe发布PhoneGap 2.7，同时宣布不再支持PhoneGap 2.0之前的版本，其中包括发布还不到一年的1.9版本。

原文链接：http://www.infoq.com/cn/news/2013/06/PhoneGap-2-7

Visual Studio 2013浮出水面

作者 Jeff Martin，译者 李彬

Visual Studio 2012发布之后不到9个月，微软就已经公布了年内发布Visual Studio 2013的计划。InfoQ为大家整理了目前已经公布的消息。

原文链接：http://www.infoq.com/cn/news/2013/06/vs2013_announced

Entity Framework 6的重大变革

作者 Jonathan Allen，译者 姚琪琳

Entity Framework为了支持新的开源模型，需要切断与.NET Framework核心发布版的联系。但这需要对EF 6中ORM已暴露的命名空间进行重大的修改。

原文链接：http://www.infoq.com/cn/news/2013/06/EF6-Breaking-Changes

Oracle官方发布Java EE 7和Glassfish 4

作者 Charles Humble，译者 臧秀涛

6月12日，Oracle官方以在线研讨会的形式发布了Java EE 7。在Java EE 7发布的同时，Oracle还以GlassFish 4.0的形式提供了该规范的一个参考实现。我们期望看到官方在适当的时候发布Netbeans 7.3.1，作为兼容Java EE 7的IDE。

原文链接：http://www.infoq.com/cn/news/2013/06/ee7-launch

使用Scala.js将Scala编译为JavaScript

作者 Zef Hemel，译者 臧秀涛

在上周的Scala Day上，Sébastien Doeraene演示了Scala.js，这是一款将Scala编译为JavaScript的编译器。该编译器支持全部Scala语言特性，允许开发者用Scala开发从前端到后端的Web应用，而且也潜在地带来了在服务器和客户端之间复用代码的可能性。

原文链接：http://www.infoq.com/cn/news/2013/06/scalajs

TURING

微信账号：
图灵教育

图灵教育推荐

书名：推荐系统
作者：Dietmar Jannach
Markus Zanker
Alexander Felfernig
Gerhard Friedrich
译者：蒋凡
书号：978-7-115-31069-9
定价：59.00 元

- 推荐系统的入门佳作
- 多位著名学者联手打造
- 全面讲解，透彻分析

百度技术委员会主席廖若雪、新浪微博数据挖掘技术专家张俊林、人民搜索商务搜索部总监常兴龙、百分点信息科技有限公司首席运营官张韶峰联袂推荐！

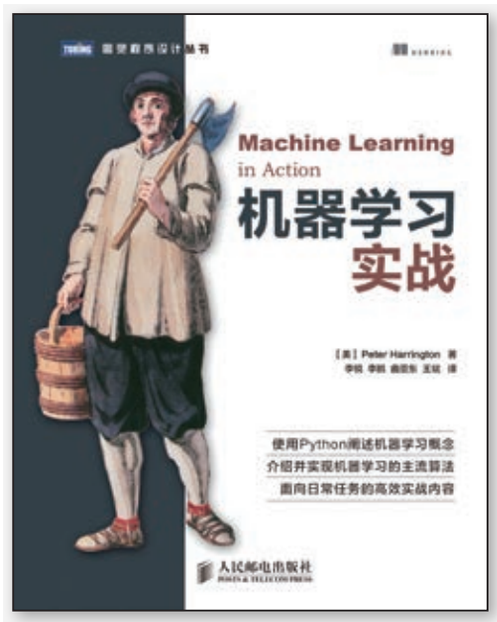
本书全面阐述了开发最新水平推荐系统的方法，其中呈现了许多经典算法，并讨论了如何衡量推荐系统的有效性。书中内容包括基本概念和最新进展两部分：前者涉及协同推荐、基于内容的推荐、基于知识的推荐、混合推荐方法，推荐系统的解释、评估推荐系统和实例分析；后者包括针对推荐系统的攻击、在线消费决策、推荐系统和下一代互联网以及普适环境中的推荐。此外，本书还包含大量的图、表和示例，有助于读者对于知识的理解和把握。

详情请点击：http://www.ituring.com.cn/book/1078

- 使用 Python 阐述机器学习概念
- 介绍并实现机器学习的主流算法
- 面向日常任务的高效实战内容

本书共四部分，讲述监督学习和无监督学习，涉及 *k* 近邻算法、决策树、朴素贝叶斯算法、支持向量机、AdaBoost 算法、*k*- 均值、分类回归树等十几种算法，讲述每种算法时都配有具体示例。

详情请点击：http://www.ituring.com.cn/book/1021



书名：机器学习实战
作者：Peter Harrington
译者：王斌
书号：978-7-115-31795-7
定价：69.00 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

JavaOne

MAKE THE FUTURE JAVA

中国 • 上海

CHINA SHANGHAI

07/22~07/25

立即报名

JavaOne 2013

2013年7月22-25日

上海世博中心

凭InfoQ13注册码 报名即享 75折

特别惊喜

QCon上海同期折扣基础上

再减1折

限7月21日前

精彩看点：

- JCP.next：复兴中的Java标准
- 性能实践：理解你的应用程序的真实性能
- NetBeans IDE 7.3中的新特性
- 透过Nashorn项目看服务器端JavaScript
- 用Java开发Web游戏中的经验教训
- 高效的Scala
- 基于MySQL 5.6版本的数据库开发运维
- 通往Lambda之路
- 为什么需要切换到Java SE 7？
- Taobao GCIH项目：直接分享Java对象，提醒垃圾收集性能
- 用HTML5解禁Java EE平台
- 用JavaFX构建一个平台独立的交互式电子书编译器
- 打开那扇隐藏的门：随处部署JavaFX
- 当Groovy、Clojure、Scala和JavaFX在一起
- 高效JavaScript单元测试
- 转向客户端：JavaFX和HTML5
- 迁移Spring应用到Java EE 6
- Java EE 7：生产效率高速提升和HTML5
- Java PaaS和Oracle Java云服务
- 淘宝的鹰眼分布式日志系统
- 便于移动支付的新型Java Card平台
- 用于嵌入式设备的简易中间件
- 基于Raspberry Pi的Java时间

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
Enterprise Software Development Community

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

云计算成就

创业梦想

阿里云开发者大赛2013

阿里云携手开发者
共同促进云生态系统的繁荣发展
欢迎广大开发者参加 2013 阿里云开发者大赛
让云计算成就你的创业梦想

5月全面开启
dasai.aliyun.com/2013

权威专家指导

创业梦想套餐

百万丰厚奖金

知名风投聚焦

精品沙龙交流

2013.5.7-9.6
大赛启动、报名至作品提交截至

2013.6-8月
"凌云"沙龙

2013.7-8月
高校训练营

2013.9.9-9.20
初赛

2013.9.23-10.11
复赛

2013.10月
阿里云开发者大会现场决赛

2013.10月
大师课

关注［阿里云开发者社区］官方微博、微信

阿里云 aliyun.com

http://e.weibo.com/aliyunopen

本大赛详情请见官方网站，解释权归阿里云计算有限公司。

推荐编辑 | 臧秀涛

各位读者朋友，大家好。很荣幸受邀成为本期《架构师》的推荐编辑。和InfoQ网站结缘已久，因为对Java和架构的爱好，每每被该网站的高质量内容所吸引，第一手的热点新闻和业界报道也让我受益匪浅。 机缘巧合，在新浪微博上看到InfoQ中文站主编贾国清（@frankjiagq）发布的译者招募信息，我便跃跃欲试了。因为之前翻译过一些技术文章，也跟别人合作翻译了一本技术书籍，所以希望能在InfoQ中文站上为技术传播贡献一份自己的力量。 InfoQ中文站对翻译的质量要求比较严格，新人加入要经过试译，试译通过之后才专为译者。而译者提交的译稿也要经过老编辑审校通过之后才能发布。前辈们的审校让我在准确表达、遣词造句和流畅表达方面获益良多。 加入之后，才发现翻译新闻和文章远远不是英文转换成中文这么简单。作为技术翻译，原文上的各种资料链接、技术背景都要了解一下，这是准确传达的前提。此后还要考虑遣词造句和中文习惯等问题。在自己可控的范围内，尽量做到最好，所以一篇稿件前前后后可能要处理四五次。发布之后也是诚惶诚恐，一方面自己会再读一遍，发现问题马上反馈；另一方面期待读者的评论，尽量和读者互动。有时候发布之后要修改感觉不太舒服的地方，可给水羽哲等编辑添了不少麻烦。 谈到与读者的互动，这里补充几句。非常希望能跟读者进行高质量的互动，但是有时候看到“像机器翻译的一样”之类的留言，心里很不是滋味。当然，问题肯定在译者身上，存在理解不到位和表达不到位等问题，这时候就要再回过头来核对，甚至找其他编辑帮忙看看，借以改进质量。但是，如果读者能详细指出一些具体问题，那真是求之不得，万分感谢的。希望能和读者共同进步。 InfoQ编辑团队非常务实，大家经常为一些翻译或技术细节探讨半天，群策群力，争取准确地传达信息，“促进软件开发领域知识和创新的传播”。跟大家的合作非常开心。我的新浪微博是@臧秀涛，希望有机会跟读者进行更多交流。如果您有意加入我们，做技术的传播者，请邮件联系editors@cn.infoq.com。

封面植物

紫罗兰

紫罗兰，原产地中海沿岸。目前我国南部地区广泛栽培。为十字花科紫罗兰属下的各个种的植物的统称。多年生草本植物，叶子长圆形或倒披针形，花紫红色，也有淡红、淡黄或白色的，有香气，果实细长。供观赏。 紫罗兰原产地中海沿岸。目前我国南部地区广泛栽培。2年生或多年生草本花卉，一般在头年秋季播种，翌年春季开花，株高30~60cm，全株被灰色星状柔毛覆盖，茎直立，基部稍木质化。叶面宽大，长椭圆形或倒披针形，先端圆钝。总状花序顶生和腋生，花梗粗壮，花有紫红、淡红、淡黄、白等颜色，单瓣花能结籽，重瓣花不结籽，果实为长角果圆柱形，种子有翅。花期12~4月，果熟期6~7月。功效 淡紫色的紫罗兰花神秘而优雅，由于颜色鲜艳、花瓣薄，多褶且透光，因此即使以冷水冲之，精华一样可以释出。能够清热解毒，美白祛斑，滋润皮肤，除皱消斑，清除口腔异味，增强光泽，防紫外线照射，紫罗兰对呼吸道的帮助很大，对支气管炎也有调理之效。可以润喉，以及解决因蛀牙引起的口腔异味。紫罗兰的不仅赏心悦目，还可以制成花茶，沁心沁脾。

InfoQ
Enterprise Software Development Community

架构师 7 月刊

每月 8 日出版

本期主编：杨赛

美术/流程编辑：水羽哲

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：杨赛，InfoQ 高级策划编辑

对软件开发、系统运维、互联网产品、移动技术、项目管理等方向感兴趣。写过一点 Flash 和前端，现在只是个伪码农。在 51CTO 做过三年多，创办了《Linux 运维趋势》电子杂志，也做过一些失败的互联网项目。最近关注如何做好的采访，以及中文界技术作者生态圈的发展。曾在新加坡、英国求学，在大学读的物理，在天幻网做过一些专题和游戏汉化，练过点长拳，玩过足球、篮球、羽毛球等各类运动和若干乐器。KK 的粉丝。

《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。