时刻关注企业软件开发领域的变化与创新

架构师

3月 ARCHITECT

特别专题

新趋势新挑战

人人网移动开发架构

Silverlight for Windows Phone 7开发体验

Mobile 进行时： Dojo Mobile控件初探

虚拟研讨会：移动Web应用开发最新动态

Silverlight CoreCLR结构浅析

Hades--JPA的开源实现

访谈与书摘：
MasoudKalali的《GlassFish安全》

Nuxeo公司探秘：
从Python迁移到Java

Maven实战（四）--
基于Maven的持续集成实践

InfoQ
Enterprise Software Development Community

每月8号出版

篇首语 | Editor's letter

习惯与坚持

小时候，习惯是自然而然养成的，就在不经意之间慢慢形成，我们无需付出太多努力，它们就会伴随一生。长大之后，我却发现，想要再养成习惯就不是那么容易了，需要付出，需要努力，才能够特意地培养出来。

最近几年中，我真正成功养成的习惯有两个：

一是坚持运动。最早是从打篮球开始，而后开始坚持走路上班，到现在每周健身。方式虽然多次改变，但目标没有改变，那就是要让自己拥有健康。毕竟身体是革命的本钱，只有保持健康，才能更长久地做自己喜欢的程序员职业，才能够开开心心地接受各种挑战。

另一个习惯是写博客，包括技术博客和给自己女儿记录生活点滴的博客。目的很简单，写技术博客是为了把自己的一些心得体会和大家分享，进而更好地与天南地北的朋友交流；至于给女儿维护的博客，想的就是能够一直保持，等女儿出嫁的时候，把所有的内容装订成册，作为礼物送给她。

养成这两个习惯的过程中，我深切体会到做好这件事并非想象中的那么简单，重要的就在于坚持。这种坚持并非是一天两天，有个理论说，想要真正养成一个习惯，需要至少坚持 21 天才会成功。

我的体会真的是那样，在第三周左右的时候，之前的新鲜感已经褪去，惰性逐渐增加、累计，继续坚持会让自己很烦躁。如果就此放弃，之前的努力就付诸流水，相反，如果能够再坚持一周的话，似乎就可以完成从量变到质变的过程，从而真正让习惯成自然。

好的习惯会让我们一生受益，坚持运动不仅仅是让我可以保持健康的身体，而且让我的精力更加充沛，从而更好地保证工作的质量和连续性；写博客更是让我与全国乃至世界各地的程序员有更广泛的交流，进而不断提升自己。

对于从事 IT 这一行业的人来说，不管是程序员，还是架构师、项目经理，在工作中也有很多重要的习惯需要养成。对于程序员来说，要自觉地遵守代码规范，尽可能好地测试自己编写的程序；对于项目经理来说，可能是经常通过会议了解大家最近的情况，从而把握项目的进度；对于架构师来说，在设计架构的时候，不仅要关注应用的可用性，还要确保在可扩展性、可伸缩性、性能等各个方面都不会出问题。

总之，为了自己的健康，为了自己的心情，为了自己的代码，我们都需要养成一些良好的习惯，并努力坚持下去。在此与大家共勉。

本期主编：侯伯薇

目　录

[篇首语]

习惯与坚持 1

[人物专访]

REST 与 WEB 即平台，SUBBU ALLAMARAJU 访谈 4

[热点新闻]

ORACLE 提名 SOUJAVA 成为 JCP 执委以取代 APACHE 8

NOSQL 架构实践（一）--以 NOSQL 为辅 10

微软会统一他们的操作系统吗？ 15

ANDROID 3.0（HONEYCOMB）--新特性简介 17

云 API 纷争又起：RPC 还是 REST 22

[特别专题]

人人网移动开发架构 26

SILVERLIGHT FOR WINDOWS PHONE 7 开发体验 38

MOBILE 进行时： DOJO MOBILE 控件初探 49

虚拟研讨会：移动 WEB 应用开发最新动态 58

[推荐文章]

SILVERLIGHT CORECLR 结构浅析 73

HADES--JPA 的开源实现 85

访谈与书摘：MASOUDKALALI 的《GLASSFISH 安全》 91

NUXEO 公司探秘：从 PYTHON 迁移到 JAVA 94

MAVEN 实战（四）--基于 MAVEN 的持续集成实践 99

[新品推荐]

SPRING GEMFIRE 1.0.0 发布，面向 JAVA 与.NET 104

SPRINGSOURCE TOOL SUITE 2.6.0 M1 发布 104

JENKINS 首个版本发布及 HUDSON 支持 104

支持大数据的 JASPERSOFT 4 发布了 105

ORACLE 发布补丁，修复 DOUBLE.PARSEDOUBLEBUG 105

ORION--基于 WEB 的 ECLIPSE 105

AMAZON AWS 推出 CLOUDFORMATION 105

数据访问框架 METAMODEL 1.5 版带来更佳开发体验 106

更易于向 WINDOWS AZURE 部署 JAVA 应用的新方式 106

MOONLIGHT 4.0 的第一个预览版已经发布 106

[推荐编辑]

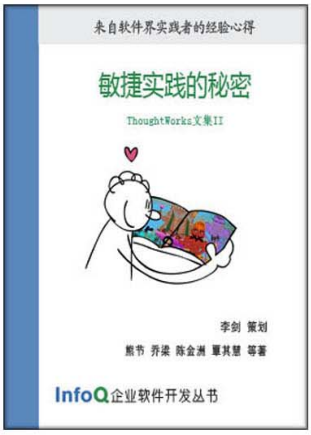
推荐编辑 | 原创团队编辑 张逸 107

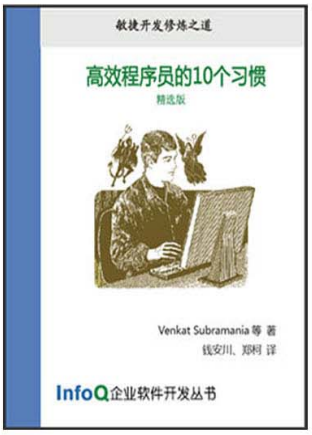
[封面植物]

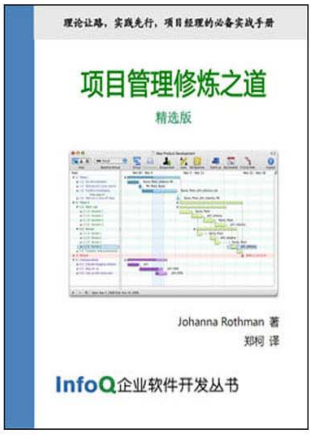
杏黄兜兰 108

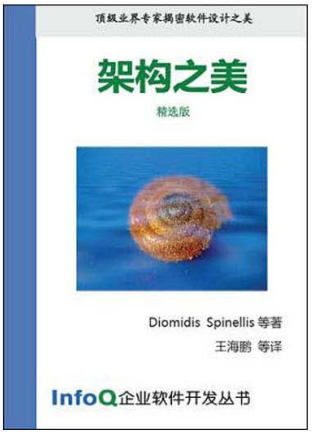
InfoQ企业软件开发丛书

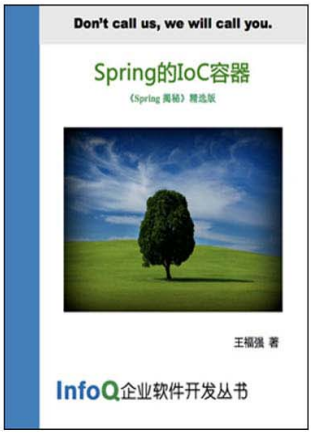
欢迎免费下载











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

人物专访 | Interview

REST 与 Web 即平台，Subbu Allamaraju 访谈

Subbu Allamaraju 是 Yahoo!的架构师，他负责监管以开发者为中心的应用程序平台，设计 RESTful Web 服务。多年来他一直投身于设计并开发基于 HTTP 之上的分布式系统的基础设施。InfoQ 就 RESTful 架构在现实中遇到的一些问题对他进行了采访。

Subbu Allamaraju 是 Yahoo!的架构师，他负责监管以开发者为中心的应用程序平台，设计 RESTful Web 服务。多年来他一直投身于设计并开发基于 HTTP 之上的分布式系统的基础设施。他参与了多本 JEE 相关书籍的编著，是《RESTful Web Services Cookbook》的合著者。

InfoQ： 嗨，Subbu，给大家做个自我介绍吧。

Subbu：我叫 Subbu，就职于 Yahoo!。我有两项日常工作：主要工作关注于提供大量用户数据，例如我管理提供用户内容的平台，其中提供用户产生内容、地点、身份信息等内容。我的第二个职责是去理解并找到让我们的平台具有互操作性的方法。

InfoQ： 过去几年里 RESTful 架构非常流行。团队犯的常见错误时什么，引起了什么问题？

Subbu：这是个好问题，不太好回答，原因有两个：从细节层面来看，很多的讨论是什么才是正确的东西，做什么才是正确的，某个特定的 HTTP 方法是否该这么用，或者一个 URI 是否该是这样的。我觉得最近几年里这类讨论正在逐渐减少，以前此类争论有很多。但我认为在宏观层面，问题是如何搞对系统的质量属性（quality attribute）。在学习 REST 时，我们中的绝大多数人是从约束开始的，而不是带来这些约束的质量属性。我发现这么做是很有用的，我鼓励从质量属性开始学起，再是约束，你会受益匪浅的。

我们要关注的第二个错误，或者说第二件事 ，是让东西能工作在一起。互操作性对我们来说是一个挑战。如果有很多基于 HTTP 的应用程序，它们由分散在全球各地的不同团队开发，让他们对某个东西达成共识对互操作性至关重要。那是一个挑战，当人们对 REST 的理解越来越好时，会有不一致的地方，我们需要能理解这些不一致。

InfoQ： 你谈到了质量属性，能不能详细说说？

Subbu：我们所说的质量属性就是那些你希望系统能做到的事，这样才能满足某些用户场景。举例来说，在出现一个错误或一类错误时，你的系统应该如何应对？这就是一个质量属性，你需要针对它来做设计，它不会自动冒出来。REST 对此不做保证，你要自己来考虑对策，让系统来做处理。

InfoQ： 理论上来说运用 RESTful 架构有很多好处。根据你的经验，相比其他集成方法，你从实际的 REST 集成项目中看到了什么具体的好处？

Subbu：我想最大的好处就是互操作性，并且你从缓存上获得了 70-80%的好处。如果你不知道如何正确运用 HTTP 的其他部分、超文本或别的东西，你还是可以从大规模缓存中获得 80%的好处。对我而言，这就像一个后场高远球得分（clear winner）。我开始意识到一件有趣的事，即这个意外新发现的概念的运用（原文：An interesting thing that I'm beginning to realize is this notion of serendipities to use.）。这句话是 4 年前 Steve Vinsoski 说过的，我想也是在这个大会上。当时他提到这是如何发生的，我还不太明白，但用 REST 和 HTTP 亲身实践后，我每天都在见证，这是最大的好处。

InfoQ： 关于 REST 相对于其他方式（尤其是 Web 服务）的好处的争论有很多，你对此的看法是什么？

Subbu：有了 Sam Ruby 和 Richardson 关于 REST 的经典之作，Ian Robison 和 Jim Webber 的演讲，还有我写的关于 REST 的书，我觉得这类争论已经少很多了。我认为我们正视图去搞清楚的唯一的东西是超文本约束，除此之外，我没看到最近有太多的争论。

InfoQ： 关于 REST，人们经常问的一个问题是“我该怎么处理事务？”对于这个问题，有很多观点，从“如果你需要事务，那你的架构就是非 RESTful 的”到多种事务协议和模式。你怎么看？

Subbu：我认为这是一个非常好的问题。在我的印象中，REST 是一种针对去中心化的相当解耦的系统的架构。当你引入事务这类东西时，你所讨论的就是两个解耦的应用程序之间的某类耦合和依赖。当你在两个你自己无法控制的系统上写东西时，让它们提交某些数据、维持一致性和其他质量属性是一件很困难的事。我觉得在尝试事务和 Web 即架构两者之间存在固有的冲突。

你不会用在受控环境（你可以赞成或期待协定）中使用事务的方式在 Web 上使用事务。此处你无法期待协定（agreement），这是 Web 即平台的一个有趣的方面，你不会就什么事达成一致，这些协议可能不会同意任何东西。这是去中心化 Web 的要点。

InfoQ： 你觉得什么是将 Web 作为集成平台的基本限制？比如，我们刚才讨论到的事务。

你持什么观点？HTTP 差不多有 20 年的历史了，尽管它是可扩展的，但创造它的初衷与我们现在的使用并不一样。

Subbu：即使我们用于 Web 的协议（比如 HTTP 和 URI）很古老了、不够现代，但在 Internet 时代，我们依然能从中受益匪浅，因为它们的设计采用了正确的抽象，设计者脑海中有正确的质量属性。有这些保证，我实际上没发现什么限制。事实上，我们还从这些技术中收获了不少东西。像 Tim Berners-Lee 这样的先锋为人类做了杰出贡献，我们至今还能从中受益。

当然，限制还是存在的，源自 HTTP 栈、HTML 等方面的不一致性、互操作性问题。这些方面要取得进展很困难，因为这些都是协议，而且需要时间才能搞定。我个人对目前的进展很满意，我相信专家们一直在努力，考虑当今 Web 的基本问题。

InfoQ： 你是 O'Reilly 的《RESTful Web Services Cookbook》的作者。能不能给我们描述一些最常用的技巧？

Subbu：据我所知最常用的技巧，源自那些与我交流过的人，就是寻找资源。我认为我要着重强调的一点是你的客户端应用程序需要更好的资源。另一个重要的技巧是控制并发，比如执行多种不同的写入，像那些有 SOA 背景的人问的事务问题，最近我都有看到。

InfoQ： 有没有一些模式你觉得接受程度还不够，应该被用得再广泛些？

Subbu：回到并发控制，这是最重要的一块，我觉得开发者应该更加关注它，我也会更强调它。当然，缓存也是大家需要的，要好好设计。经常有开发者设计不佳，没有发送正确的响应头，比如最后修改时间，正确使用 ETags 等等。

InfoQ： 有时开发者为了快速响应市场，走了一些捷径，比如穿隧（tunnel）动词和 URI，你怎么看？这些是无可避免的、合理的吗（至少在一个小范围内）？他们应该为此羞愧吗？

Subbu：回到质量属性：如果 Web 的好处不是你的目标，为了质量属性你不得不穿隧读和写，那就干吧，但你需要意识到这么做的后果。我认为关键是你要意识到你在做什么样的权衡。如果时间对市场很重要，那就为此做权衡。

InfoQ： 对于那些开发基于 REST 项目的开发者，工具支持目前是什么情况？为了让开发和调试更方便一些，有什么推荐的工具么？

Subbu：我想说，对于基于 HTTP 的应用程序，任何语言下都有很多工具。我可以用 JavaScript，用 Ruby，用 PHP 和 Python，它们都有完整的工具栈，那些语言的工具就和 Java、C#等语言一样成熟，但这些工具是由社区推动的，或者由市场推动。对我而言，我在自己的台式机上也有很多工具，比如 curl 和 wget，像 lighttpd 和 Apache 这样的 Web 服务器，这些就是我的

工具。

InfoQ： OAuth 很受关注，你好像支持最近发布的版本 2。能否给我们解释一下，是什么让 OAuth 成为保护资源的好方法的？

Subbu：在协议的进化过程中，我看到了两个主要的变化。第一个是密钥。作为协议，它更简单了，它解耦了访问令牌授予客户端的方式和与服务器交互的方式，这样 OAuth 能成为一个可扩展的协议，你会找到新的获取令牌的方法。它还尝试统一其他身份验证形式，比如 SAML，因此它是可扩展的。这是协议中的一个非常重要的特性。第二个变化是现在不再强制签名了，我不太清楚规范以后会如何发展，但目前对我而言这看上去还不错，我知道这两方面都有人关注。

我认为出于互操作性考虑，没有签名，像使用 Cookie 那样使用访问令牌，这让客户端更方便了。如果顺利的话，在客户端应用程序上设置好 TLS 就可以不用使用签名，同时获得相同的信任程度。

InfoQ： 你说过“前 30 分钟的体验是一个质量属性，任何一个 Web 平台要成功都必须考虑它”。你能再详细说说吗？这是什么意思？

Subbu：这个很有意思，我来自这样一家公司，在发布一个产品时要问一个问题——“客户在下 30 分钟或给定时间里能做什么？他能获得什么有意义的东西呢？” 我最近把这个观点提出来，是因为开发者在思考他们的平台、Web 服务、应用程序时，总觉得那是很特殊的。每个人都觉得自己的应用程序与众不同，有特别之处，某些方面独具匠心。实际上，从客户的观点来看，他感兴趣的是完成自己的任务，并不想区别对待各个应用程序。

对我而言这是一个重要的质量属性，因为客户认为他们需要的东西不必很特别，即使是在做一件很复杂的事。就算是在用 Google Maps，或者 Yahoo! Mail，你希望在使用它时和别的网站一样的。对最终用户而言，它们是一样的。对 Web 服务和任何大型系统而言都是如此。

InfoQ： 非常感谢，Subbu。

Subbu：谢谢，我很高兴。

原文链接：http://www.infoq.com/cn/interviews/jf-taobao-database

相关内容：在 RESTful 服务中实现部分更新、REST 反模式、针对 HTTP 数字签名协议和 API 的提议、OAuth 2.0 对 Web 有害吗？、OData Services 和客户端可用的验证机制综述

QCon 全球企业开发大会（北京站）

距离QCon北京大会召开在即

报名进入最后阶段，现在报名9折优惠

五场会前培训同时启动（4月6-7日）

领域驱动设计
Eric Evans

Android平台技术
高焕堂

打造更好的Backlog
Davind Hussman

敏捷开发Scrum
鲍央舟

精益软件开发与软件架构
黄邦伟

大会讲师确认工作完成，最新讲师及课程信息：

吴穹： IJI China董事总经理
杨海朝：新浪首席DBA
吴波：百度高级工程师
胡戊：腾讯广州研发中心
贺师俊：盛大创新院
余峰：淘宝网数据库专家
Nicolas Spiegelberg：FacebooK工程师
谢子斌：W3C中文小组主席
Eric Petersen：探索式测试专家

更多..

- 大会报名进入倒计时，现在报名9折优惠，5人以上团购可以享受更多优惠
- 五场会前培训全部确认，国内外知名机构讲师深度讲解软件架构、移动开发、敏捷话题。

优惠政策

时间：2011.4.8-2011.4.10
地点：北京-京仪大酒店
网站：www.qconbeijing.com
主办方：InfoQ
电话：13911797020
010-89880682
010-64738142
E-mail：qcon@cn.infoq.com

火热报名中

热点新闻 | News

Oracle 提名 SouJava 成为 JCP 执委以取代 Apache

作者 Charles Humble 译者 张龙

近日，Oracle 宣布将提名世界上规模最大的 Java 用户组之一的 SouJava 代表 Bruno Souza 加入 JCP 执委会。SouJava 在巴西拥有 20,000 到 40,000 名成员，还有几个分支机构；2005 年，它成为加入 JCP 的首个 Java 用户组（JUG）。Souza 是该用户组的创始人与上一届主席。

Oracle 的 SE 发言人 Henrik Ståhl 说到：

> 我们期待 Bruno 能够积极参与到 JCP 当中并帮助 Java 不断前进，同时也能帮助我们在接下来的几年内改革 JCP。

JCP 主席 Patrick Curran 说到：

> Bruno 从一开始就积极支持开源与 Java，他将为执委会增加新的活力，特别是在我们进入到第二个十年的时候，需要修改组织的进程之际他能加入进来真是太好了。

如果一切顺利，Souza 将接管 Apache 于去年 12 月所放弃的席位，Apache 与 Oracle 就 Java SE TCK 许可的问题已经争执很久了。Sun（在被 Oracle 收购前）与现在的 Oracle 都拒绝为 Apache 提供 Harmony（Java SE 的独立实现）的许可认证。

使用约束条款可以防止 Java SE 用于嵌入式系统中，特别是移动电话上，这是 Oracle 通过 OEM 许可发售 Java ME 的重要收入来源。Oracle 的考虑在于 OEM 可以凭借 Harmony 制造基于 Java 的手持设备而无需支付给 Oracle 任何的许可费用。事实上，Google 现在对 Android 所做的事情就是颇具争议的。Oracle 声称 Android 违背了大量的 Java 专利与版权。

Apache 离开 JCP 的决定发生在大多数 JCP 成员都对 Oracle 的 Java 7 与 8 路线图投赞成票之际。其中只有 3 个成员投了反对票，分别是 Apache、Google 与 Tim Peierls，但其他投了赞成票的成员也是迫于无奈而非自愿的。

Souza 在其博客上说到：

> 从我个人角度来说，我将这看作是一个机会，可以让 JCP 具备更好的开发者参与性，同时也可以让 JCP 尊重开源社区的需求。到目前为止，我与 Oracle 之间的讨论结果使我相信我们在一些问题上已经达成了一致，很明显我们会让其他人感到不爽，但这也是 JCP 的一部分。
>
> 未来还有很多事情要做：SouJava 需要进行选举并获得 JCP 成员的批准。如果 JCP 成员知道我们的加入将是有益的，那么我们就会加入到讨论当中并提出我们自己的提案。我们知道这不是一件轻松的事情，变化都不是那么容易的。但我们的小组是强大且独立的，不会因为讨论和决定很难而有所退缩。

目前执委会已经有两个席位空缺了：Peierls 与 Doug Lea。他们之所以选择离去，都是因为 Oracle 对待 JCP 的态度。Peierls 说“SE/EE 执委会永远也不应该成为 Oracle 的傀儡”。

去年 10 月和 11 月，Oracle 希望用 Hologic 公司替换掉 Lea 的尝试就得到了 JCP 成员的普遍反对。这次选举有 95%的投票赞成 Apache 重返执委会。

Joda-Time 创始人与 Apache 成员 Stephen Colebourne 说那时的 Hologic 表明了 Oracle 想要掌控 JCP 的欲望（Hologic 是 Oracle 的大客户）。InfoQ 有幸采访到了 Stephen Colebourne 以了解他对 Oracle 提名 SouJava 的看法。他说到：

> SouJava 表达出的是要重振 JCP 的愿望，方式就是增加 JCP 的透明度。他们需要做的就是将 JCP 分为两个独立的部分——核心规范（Java SE/ME），这也是 Oracle 想要封闭的部分，以及其他规范（Java EE 等），这是 Oracle 希望保持开源的部分。如果一切顺利，Apache 就可能会重新考虑加入到开源部分（我怀疑这取决于分割后 Oracle 到底会掌控多少内容）中。
>
> 当然了，我不反对任何人这么做（比如说 SouJava）。但我怀疑 Oracle 与 Google 之间的诉讼只要不结束，那么 Oracle 就会随意对其资产进行变更，这是因为我怀疑任何行动都会对可能的胜利作出妥协。
>
> 从 Oracle 的角度来看，SouJava 是个很不错的选择，因为他们采取了与 Apache 类似的方式表达出了人们的心声。我认为 SouJava 会有一定的风险，但成功的机会也很大。Bruno 当然完全有能力占据这个席位。作为一名 JCP 成员，我个人会对 SouJava 投赞成票。

原文链接：http://www.infoq.com/cn/news/2011/02/soujava_jcp

相关内容：Tony 谈 Java 发展以及数据库管理工具、Scala 与 Spring：强强联合

热点新闻 | News

NoSQL 架构实践（一）--以 NoSQL 为辅

作者 孙立

前面《为什么要使用 NoSQL》和《关系数据库还是 NoSQL 数据库》两 篇从大体上介绍了为什么要用 NoSQL，何时该用 NoSQL。经常有朋友遇到困惑，看到 NoSQL 的介绍，觉得很好，但是却不知道如何正式用到自己的项目中。很大的原因就是思维固定在 MySQL 中了，他们问得最多的问题就是用了 NoSQL，我如何做关系查询。那么接下来，我们看下怎么样在我们的系统中使用 NoSQL。

怎么样把 NoSQL 引入到我们的系统架构设计中，需要根据我们系统的业务场景来分析，什么样类型的数据适合存储在 NoSQL 数据库中，什么样类型的数据必须使用关系数据库存储。明确引入的 NoSQL 数据库带给系统的作用，它能解决什么问题，以及可能带来的新的问题。下面我们分析几种常见的 NoSQL 架构。

（一）NoSQL 作为镜像

不改变原有的以 MySQL 作为存储的架构，使用 NoSQL 作为辅助镜像存储，用 NoSQL 的优势辅助提升性能。

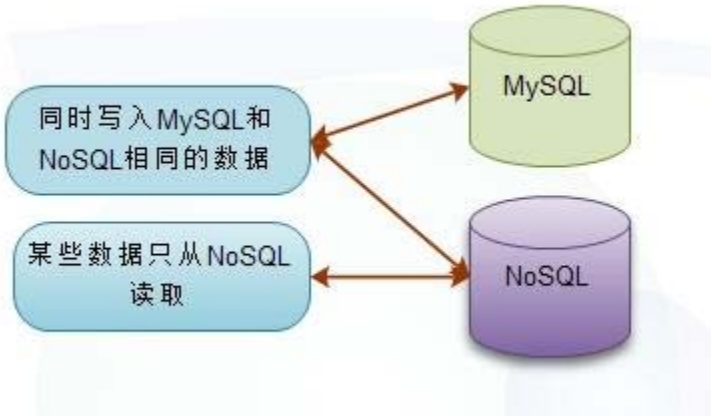


图 1 -NoSQL 为镜像（代码完成模式 ）

```
//写入数据的示例伪代码
```

```
//data为我们要存储的数据对象
data.title="title";
data.name="name";
data.time="2009-12-01 10:10:01";
data.from="1";
id=DB.Insert(data);//写入MySQL数据库
NoSQL.Add(id,data);//以写入MySQL产生的自增id为主键写入NoSQL数据库
```

如果有数据一致性要求，可以像如下的方式使用

```
//写入数据的示例伪代码
//data为我们要存储的数据对象
bool status=false;
DB.startTransaction();//开始事务
id=DB.Insert(data);//写入MySQL数据库
if(id>0){
    status=NoSQL.Add(id,data);//以写入MySQL产生的自增id为主键写入NoSQL数据库
}
if(id>0 && status==true){
    DB.commit();//提交事务
}else{
    DB.rollback();//不成功，进行回滚
}
```

上面的代码看起来可能觉得有点麻烦，但是只需要在 DB 类或者 ORM 层做一个统一的封装，就能实现重用了，其他代码都不用做任何的修改。

这种架构在原有基于 MySQL 数据库的架构上增加了一层辅助的 NoSQL 存储，代码量不大，技术难度小，却在可扩展性和性能上起到了非常大的作用。只需要程序在写入 MySQL 数据库后，同时写入到 NoSQL 数据库，让 MySQL 和 NoSQL 拥有相同的镜像数据，在某些可以根据主键查询的地方，使用高效的 NoSQL 数据库查询，这样就节省了 MySQL 的查询，用 NoSQL 的高性能来抵挡这些查询。

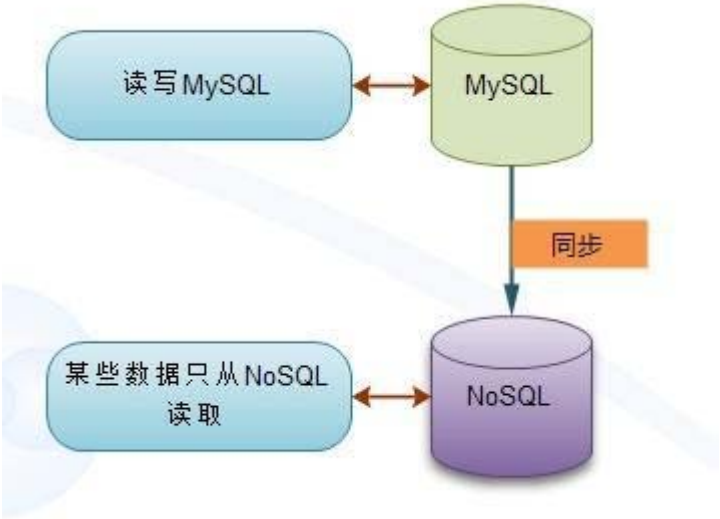


图 2 -NoSQL 为镜像（同步模式）

这种不通过程序代码，而是通过 MySQL 把数据同步到 NoSQL 中，这种模式是上面一种的变体，是一种对写入透明但是具有更高技术难度一种模式。这种模式适用于现有的比较复杂的老系统，通过修改代码不易实现，可能引起新的问题。同时也适用于需要把数据同步到多种类型的存储中。

MySQL 到 NoSQL 同步的实现可以使用 MySQL UDF 函数，MySQL binlog 的解析来实现。可以利用现有的开源项目来实现，比如：

- MySQL memcached UDFs：从通过 UDF 操作 Memcached 协议。
- 国内张宴开源的 mysql-udf-http：通过 UDF 操作 http 协议。

有了这两个 MySQL UDF 函数库，我们就能通过 MySQL 透明的处理 Memcached 或者 Http 协议，这样只要有兼容 Memcached 或者 Http 协议的 NoSQL 数据库，那么我们就能通过 MySQL 去操作以进行同步数据。再结合 lib_mysqludf_json，通过 UDF 和 MySQL 触发器功能的结合，就可以实现数据的自动同步。

（二）MySQL 和 NoSQL 组合

MySQL 中只存储需要查询的小字段，NoSQL 存储所有数据。

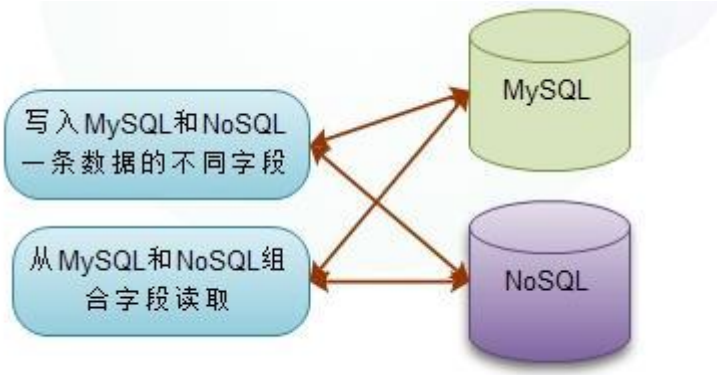


图 3 -MySQL 和 NoSQL 组合

```
//写入数据的示例伪代码

//data为我们要存储的数据对象
data.title="title";
data.name="name";
data.time="2009-12-01 10:10:01";
data.from="1";
bool status=false;
DB.startTransaction();//开始事务
id=DB.Insert("INSERT INTO table (from) VALUES(data.from)");//写入MySQL数据库,
只写from需要where查询的字段
if(id>0){
    status=NoSQL.Add(id,data);//以写入MySQL产生的自增id为主键写入NoSQL数据库
}
if(id>0 && status==true){
    DB.commit();//提交事务
}else{
    DB.rollback();//不成功，进行回滚
}
```

把需要查询的字段，一般都是数字，时间等类型的小字段存储于 MySQL 中，根据查询建立相应的索引，其他不需要的字段，包括大文本字段都存储在 NoSQL 中。在查询的时候，我们先从 MySQL 中查询出数据的主键，然后从 NoSQL 中直接取出对应的数据即可。

这种架构模式把 MySQL 和 NoSQL 的作用进行了融合，各司其职，让 MySQL 专门负责处理擅长的关系存储，NoSQL 作为数据的存储。它有以下优点：

- 节省 MySQL 的 IO 开销。由于 MySQL 只存储需要查询的小字段，不再负责存储大文本字段，这样就可以节省 MySQL 存储的空间开销，从而节省 MySQL 的磁盘 IO。我们曾经通过这

种优化，把 MySQL 一个 40G 的表缩减到几百 M。

- 提高 MySQl Query Cache 缓存命中率。我们知道 query cache 缓存失效是表级的，在 MySQL 表一旦被更新就会失效，经过这种字段的分离，更新的字段如果不是存储在 MySQL 中，那么对 query cache 就没有任何影响。而 NoSQL 的 Cache 往往都是行级别的，只对更新的记录的缓存失效。

- 提升 MySQL 主从同步效率。由于 MySQL 存储空间的减小，同步的数据记录也减小了，而部分数据的更新落在 NoSQL 而不是 MySQL，这样也减少了 MySQL 数据需要同步的次数。

- 提高 MySQL 数据备份和恢复的速度。由于 MySQL 数据库存储的数据的减小，很容易看到数据备份和恢复的速度也将极大的提高。

- 比以前更容易扩展。NoSQL 天生就容易扩展。经过这种优化，MySQL 性能也得到提高。

比如手机凤凰网就是这种架构 http://www.cnblogs.com/sunli/archive/2010/12/20/imcp.html

总结

以 NoSQL 为辅的架构还是以 MySQL 架构的思想为中心，只是在以前的架构上辅助增加了 NoSQL 来提高其性能和可扩展性。这种架构实现起来比较容易，却能取得不错的效果。如果正想在项目中引入 NoSQL，或者你的以 MySQL 架构的系统目前正出现相关的瓶颈，希望本文可以为你带来帮助。

原文链接：http://www.infoq.com/cn/news/2011/02/nosql-architecture-practice

相关内容：

- iDataForum2010 数据库技术论坛总结
- MySQL/HandlerSocket和VoltDB：NoSQL的竞争者
- MySQL变更：更加昂贵，且典藏版中没有InnoDB
- MemBase，NoSQL家族的一个新的重量级的成员
- 将CouchDB作为个人数据库

热点新闻 | News

微软会统一他们的操作系统吗？

作者 Abel Avram 译者 李永伦

英特尔 CEO Paul Otellini 暗示微软正在尝试把他们的操作系统统一成一个操作系统，运行环境从移动电话到桌面电脑，他的话为 Windows 和 Windows Phone 7 的将来留下了一个问号。

在 CES 2010 上，微软演示了运行在由 NVidia、Qualcomm 和 Texas Instruments 等制作的 System-on-a-Chip 上的下一代 Windows，所有都是基于 ARM 架构构建的，使用了适用于移动电话和平板设备等小型设备的低功耗处理器。在这样一个小型设备上运行全功能版本的 Windows 让人怀疑微软是否想把他们的旗舰操作系统用到移动电话上。微软 Windows 和 Windows Live 部门总裁 Steven Sinofsky 提到，Windows 的将来版本不会和 Windows Phone 7 在发展路线上出现交叉，这次移植到 ARM 上是为了让 Windows 运行在平板设备上，而不是移动电话上：

> 然而，他（Sinofsky）非常清楚 Windows Phone 7 不会出现跨越，排除了在平板设备上使用微软新的智能手机操作系统。“（Windows）Phone 只关注小型屏幕规格”，他说。“小屏幕使用 Windows Phone，而这些（平板设备）屏幕使用 Windows。”

但一个星期之后，英特尔总裁与首席执行官 Paul Otellini 提到微软“统一他们的操作系统”的可能性，让相同的操作系统能够运行在从智能手机到桌面电脑的设备上。他在与采访记者进行电话会议时提到：

> 对英特尔有利的是，他们（微软）统一他们的操作系统，我们将首次可以：第一，拥有一个运行在英特尔平台的重新设计的触摸操作系统，这就目前而言是还没有 的。第二，我们可以把运行 Windows 8（下一代 Windows）的超低功耗英特尔处理器用到移动电话上，因为这是相同的操作系统。我认为这对于我们来说是个好机会。

目前还不清楚微软的计划是什么。他们是否尝试统一他们不同版本的操作系统，尤其是 Windows CE 和桌面 Windows，既然 Windows 可以运行在低功耗设备上？（Windows Phone 7 是基于 Windows CE 的操作系统。)他们是否尝试跟随苹果的做法？iOS，适用于 iPhone、iPad、iPod Touch 和 Apple TV 的苹果操作系统，最初在 Mac OS X 的基础上构建，而后它的一些功

能又被整合回 Mac OS X 10.7（又名 Lion），比如说多点触摸、应用程序全屏运行模式、启动板（launchpad）以及其它功能，这会使人认为随着时间的推移 iOS 和 Mac OS X 最终会统一起来，合并完成之后 iOS 将会成为这个操作系统的名字。

微软很可能采取相同的措施，把移动电话和桌面的操作系统统一起来。其中，主要的问题是为移动电话创建特定的用户界面。虽然原本的 Windows 图形用户界面无需太多调整就能运行在平板设备上，但这个图形用户界面对于移动电话来说不太适合，尤其是智能手机。最简单的解决方案似乎是把 Windows Phone 7 的图形用户界面拿到 Windows 上使用。这意味着一个操作系统拥有两个用户界面，一个用于移动电话和平板设备，另一个用于上网本、笔记本电脑和桌面电脑。

微软及其重要合作伙伴英特尔在这个问题上的分歧导致了更多的困惑，削弱了 Windows Phone 在微软苦苦挣扎的移动市场上的地位。

原文链接：http://www.infoq.com/cn/news/2011/02/Will-Microsoft-Unify-OSes

相关内容：

- 三种流行操作系统的内核比较
- System-on-a-Chip上的Windows
- Windows 8 需要哪些特性呢？
- 微软的云经济学
- 雾里看花：微软的前端技术战略何去何从？

热点新闻 | News

Android 3.0（Honeycomb）--新特性简介

作者 Simon Guest 译者 张龙

上月 26 日，Google 发布了万众期待的 Android 3.0（Honeycomb）操作系统的开发者预览版。接下来，开发者就可以通过 AVD（Android Virtual Device）Manager 预览新的发布了，而 AVD 则是随着 SDK 一同发布并作为 SDK 一部分的。

Honeycomb 与 Android 之前的版本如 Froyo 和 Eclair 的不同之处在于它从一开始就是面向更大的显示设备的，特别是平板电脑。为了支持 这些大的显示设备，新的操作系统提供了一个名为“Holographic”的全新 UI 主题，它向操作系统增加了新的皮肤，同时保留了现有 Android 用 户所熟知的众多特性。

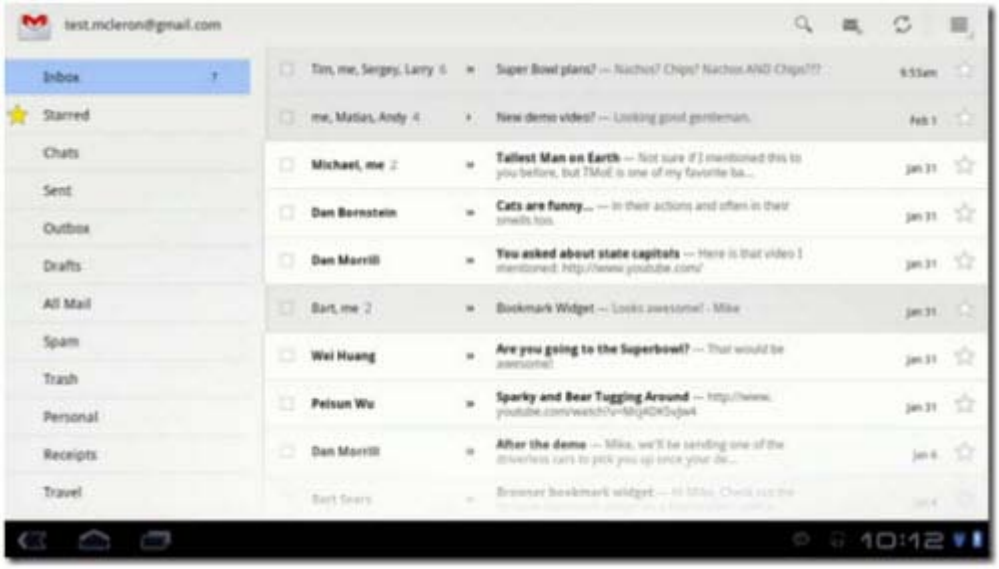
本月 2 日，Google 在其位于 Mountain View 的总部举办了一次大会，展示了新操作系统的诸多特性，包括在 Motorola 新的平板电脑 Xoom 上的一个演示。大会的视频已经更新到了 YouTube 上，但为了节省广大读者的时间，我们在这里总结出大家可能感兴趣的新操作系统的重要特性和声明。

Android 3.0 新的 UI 特性简介

Andy Rubin 首先登台，介绍了 Google 是如何通过开源过程“指导”操作系统的开发的，接下来登台的是 Android 移动产品领导 Hugo Barra。Hugo 首先展示了设备上一些新的 UI 特性，这些 UI 以一套全新的 widget 和一个新的、非侵入的通知系统为中心，他们都是面向平板电脑而设计的。

虽然新的 UI 主题看起来与 Android 2.x 版有所区别，但 Hugo 向在场观众演示了使用 Android 之前版本所编写的现有应用——甚至是图形密集型应用如 Fruit Ninja——都在 Honeycomb 上运行良好。

对于编写 iPhone 与 iPad 应用的开发者来说，这种向后兼容性是再熟悉不过的了。他们所熟知的是新的操作系统中使用某些新的 UI 原则所设计出来的一套模板。在演讲中，Hugo 将其称之为“fragments”，并演示了 fragments 如何在应用的不同部分之间进行变换，同时还介绍了如何使用不同的 fragments 为相同的应用进行换肤。比如说，某个应用可以拥有一套针对手持设备的 fragments、针对 portrait 模式下平板电脑的 fragments 以及针对 landscape 模式下平板电脑的 fragments。



Android 3.0 的性能

接下来讨论的是性能问题。新的 Android 3.0 包含了一个新的 3D 引擎和名为 Renderscript 的动画框架。在场观众看到了几个“让人眼花缭乱”的演示，主要介绍了如何使用 Renderscript 显示 YouTube 视频、书籍及相册的 3D 视频墙。

经历过其他图形框架如微软的 Windows Presentation Foundation（WPF）、Silverlight 或是 Adobe Air 的开发者们会认识到到这些低功率的元件都向高功率的设备提供了类似的图形处理能力。

会上还演示了设备新增的媒体功能，包括来自于 War Drum Studios 的两个多核游戏、从 PS3 平台迁移到了 Android 上，还有一个 Android 版本的 CNN 新闻阅读器，带有一个大页面视图和直播的 CNN 视频流。

但在所有这些媒体演示当中，最吸引观众眼球的还是对视频聊天的支持。Motorola Xoom 带有前后摄像头，这非常适合于视频会议。据推测，现场的演示使用的是无线网络而非 3G，过程非常平滑。这也得益于 Google 向视频应用中所添加的图像稳定化技术的支持。这不仅有助于图像的稳定，还通过忽略掉 side-shifted 帧以达到节省带宽的目的。

Android Market

最后登台演讲的是 Google 的新雇员 Chris Yerga，他谈到了对 Android Market 的新改进。Chris 演讲的主要内容都围绕在 Android Market Web Store 新的发布声明上。

到现在为止，Android 用户只能使用设备本身的应用。但借助于新版本的 Store，用户现在可以浏览 Store、寻找感兴趣的应用、根据类型和用户评论进行过滤，如果准备通过 Google Checkout 购买应用，只需将其发送到注册的设备上即可。

对于那些为 Android 平台编写应用的开发者来说，Store 还支持更长的描述、更大尺寸的图标、高分辨率的广告横幅，并且可以直接从页面上链接到 YouTube 的视频上。

最后，Chris 谈到了对 Android Market 财务交易的一些新支持，开发者可以查看不同货币的

价格，还支持在应用中购买。借助于 SDK，开发者可以凭借应用中购买这种方式直接在应用中向 最终用户收取费用。为了说明这一点，来自于 Disney Mobile 的总经理 Bart Decrem 演示了三款新的 Android 应用，其中包括流行的 Tap Tap Revenge，它通过新的 SDK 支持应用中购买。

Google 没有告诉我们什么

虽然新操作系统的演示主要面向的是消费者，但开发者可能对 Android 3.0 可插拔的 DRM 框架支持以及管理策略如加密存储和密码过期等内容感兴趣。在之前的 Apple 收益报告中，Tim Cook 提到很多企业都在评测并部署平板设备。虽然以上内容没有在这场演示中体现出来，但这里面的众多特性对于这些组织来说是极具吸引力的，特别是对那些需要管理用户设备的企业来说更是如此。

深入 Honeycomb

在 InfoQ 自己的测试中，我们发现虽然运行在硬件上的模拟器是用于开发当前版本的 Android 应用的，但虚拟机孱弱的性能清楚地表明这是个预览构建版。因此，开发者会发现这个构建主要用于探究新的特性，但很多人还是希望在下一版本发布后才开始为该平台编写应用。

Motorola Xoom 将于 2 月底发布，开发者则需要等上更长一段时间才能在完全支持的设备上体验到 Honeycomb。但不用担心，因为有 Barnes 和 Noble 的 Nook Color 设备的持有者——他们并不介意别人获取到设备的根权限并将内容擦除掉——我们现在就可以开始尝试 Honeycomb，有很多开源开发者将其移植到了设备之上，并且提供了下载指令。

大家的想法可能不同，但我们现在可以在发布前第一时间了解到新的 Android 平板操作系统。

原文链接：http://www.infoq.com/cn/news/2011/02/Honeycomb-Google-Announcement

相关内容：

- Android 3.0 针对平板电脑进行了优化
- 为网站和智能手机构建FlightCaster前台应用
- 使用FlashProfessionalCS5 开发AndroidAIR应用
- Amazon发布Appstore开发者门户
- 自由软件基金会谨慎支持Google应对专利官司

热点新闻 | News

云 API 纷争又起：RPC 还是 REST

作者 Mark Little 译者 马国耀

InfoQ 之前的一篇新闻报道了 William Vambenepe 的问题“如果云计算的先驱者们都不使用 REST，它对于云计算还重要吗？”报道发表之后，William 作出了进一步的反馈，试图阐明自己最初那篇博文的中心思想，而这中心思想却被别人忽视了，或者说视而不见。他的本意是：对于开发者而言，抛开语言结构（对象或 RPC）底层的交互机制，交互的方法是否通过 REST 实现并不重要。正如 William 所言：

> 方法（Method）调用就是普通的程序员选择怎样编写一段普通的代码。直接在线上调用远程 API 需要的改变是最少的。RPC 的复杂性从来就不是概念上的，而是存在于在实际操作中的。如何序列化方法调用，如何传输？CORBA、RMI、SOAP 都曾试图解决这个问题，但是没有一个能完美地保持简单并且能 在互联网上足够通用。在此过程中，XML-RPC 在一定程度上（不幸地）消亡了。

如他所指出的，AWS 使用了一种非常接近 XML-RPC 的方法。他们在此方法中作出了一些权衡，但是总体来说，AWS 采取了实用的方法，最终产生了一种概念上简单却又强大（并且可用）的方法。然而，很多人似乎仍然不同意这个看法，比如 Mike Pearce 提到：

> 我知道我是一个 REST 迷（人们这样称呼那些对 REST 痴迷的人）。但是事实上，Amazon 上并没有证明任何东西，除了傲慢和不尊重开发者之外，什么也没有。

William 试图在最近的博文中解释 Mike 所担心的安歇问题。譬如，其中 Mike 问到：

> AWS 使用自己实现的 API 而不采用标准，比如，哦，我还不知道，REST。对于那些不想学习 AWS 交互方式却不得不学习的开发者，这难道不是一种折磨么？

Mike 对此非常肯定，而 William 却不这么看……

> 我很疑惑。我看见过很多开发者艰难地理解 REST。我却从未见过哪个开发者被 RPC 所吓倒。至于 REST 是一个“标准”的说法，我得去看看规范。不要让我去读博士论文。

William 认为，关键的问题不是 REST 对于云计算的发展是否重要，而是云计算是否已经发展到了那个（开发者或实施者）必须（或应该）作出根本性变革的点。William 的观点是，单个提供商产品带来的 REST 的优势是非常有限的。

> 云 API 的使用模式可能会发展到那个点，在这个点上遵循 REST 规则所能带来的价值是非常诱人的。我只是认为我们还没有到这个点，而且老实说我也不为此激动。在这个热议的问题变成云服务提供商间交互时应考虑的问题之前，云服务还需要许多功能上的改进。而当共享的 RESTful API 成为最简单的协作 API 时，共享的 RPC API 也一定会变得非常容易管理。到那时，最热的话题可能会变成共享语义的问题，而不再是协议了。

还是有一些不同意 William 的提法的人认为，REST 是用在编程接口背后的，在这个层次上谈论它有必要吗？不过，William 最初的博文中的一个评论者说：

> 大多数 AWS（或其他云供应商的）用户从来看不到 API。他们是通过语言包、或 Web 客户端应用与这些云服务交互的。所以，真正关心此问题的人是 REST 迷们，以及库开发者们（比如我）。也许有那种差劲到人们无法使用的 API 存在，但是 AWS Query API 远不至于那么糟糕。它相当统一，而且很容易编程。只不过它不是 REST。另外，值得一提的是，并非所有 AWS 服务都是这样的。S3、 CloudFront 和 Route53 的 API 都比较倾向于 RESTful。

William 最近博文的整体主题是，好的开发者接口可以隐藏非 REST API 方法，而且还非常有价值，而反过来（译注：REST API 却无很好的开发者接口）并不一定是这样的。好接口与底层 REST 的结合也许是业界最终将会实现的理想世界，但是在现阶段，为了云的成功，既没有必要也没有足够的条件这么做。对于云开发者而言，确保资源语义和编程接口（包括错误处理）的正确性比最好的 RESTful 接口显得更加重要。

> 如果你的 RPC API 足够统一，以至于可使用 REST API 做其底层模型，那就很好了。

原文链接：http://www.infoq.com/cn/news/2011/02/rest-cloud-update

InfoQ中文站

www.infoq.com/cn

我们的**使命**：成为关注软件开发领域变化和创新的专业网站

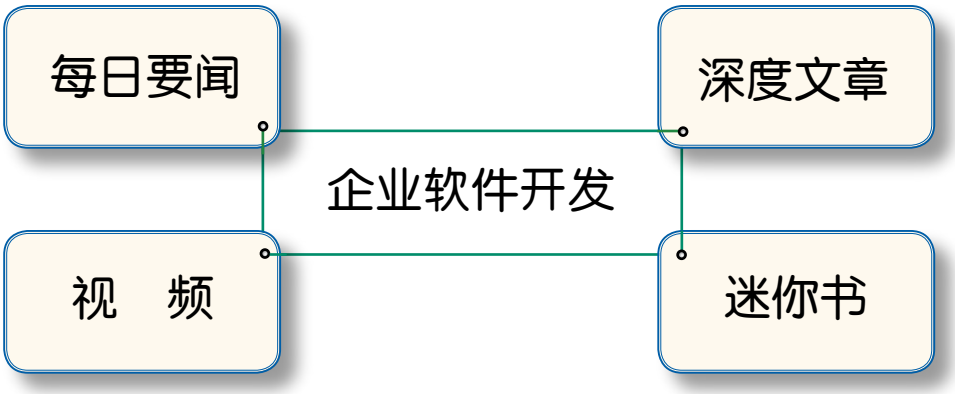
我们的**定位**：关注高级决策人员和大中型企业

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

我们的**特质**：个性化RSS定制、国际化内容同步更新

我们的**团队**：超过30位领域专家担当社区编辑

……



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

特别专题 策划：InfoQ 中文站； 执行：侯伯薇

在 IT 领域中，每年都会有新的趋势，最近几年以来，让整个业界保持关注的趋势包括云计算、虚拟化、NoSQL，当然还有本期《架构师》的主题——"移动开发"。

最早移动开发技术应该是 J2ME 和 Symbian，由于这两种技术和自己所从事的开发方式有很大的不同，所以一直采取的是旁观的态度，没有深入了解和学习的想法。然而最近不同了，移动开发技术领域风云突变，先是苹果的 iOS 在随着 iPad 的热卖让更多的人为其开发应用程序，放在 AppStore 上为自己赚到美元；接下来是 Android 的异军突起，并且在近期发布了 3.0 版本；还有微软携着 Windows Phone 7，也想一扫自己在移动领域的尴尬局面；此外还有黑莓、乐 Phone 等等我们了解和不了解的平台和技术，越来越多的厂商都把眼光放在移动开发上，希望能够在这个领域中有所作为。

随着这些软件平台的出现，移动开发不再仅仅限于手机上，平板电脑成为了另一兵家必争之地，我们从苹果和 Google 的战略中已经能够窥得一斑。而经过多家硬件厂商的努力，大量性能更高、功能更强大的手机和平板电脑的出现，更为平台提供了用武之地；广大开发者对于移动开发的热情和努力，更是让这个趋势愈演愈烈。

思考一下，这个趋势为我们带来了什么呢？很简单，就是让这个地球村更小了，每个人只要愿意，随时随地，通过各种移动设备，都可以连接到网络上，与自己的家人朋友随时保持联系。而通过这些设备访问 SNS 网站，更是缩短了人与人之间的距离，让彼此更容易了解和沟通。

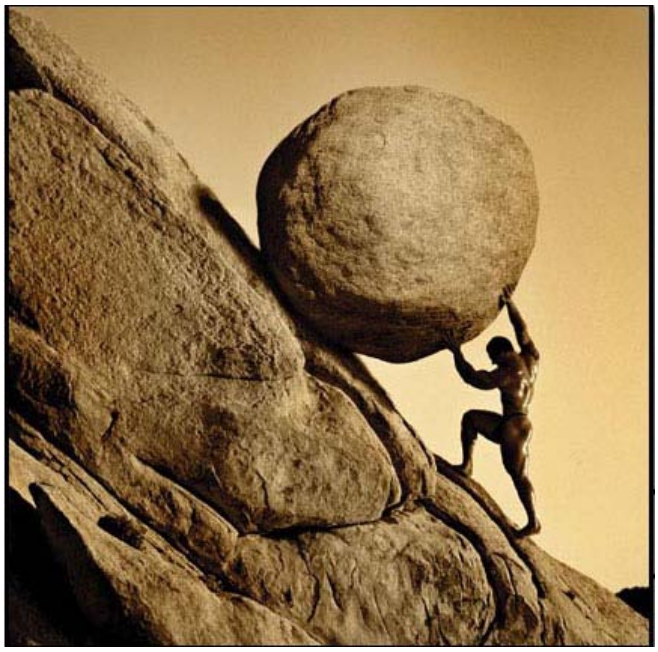
这种新趋势也给广大开发人员带来了很大的挑战，每种技术的更新速度都非常快，会根据具体的要求不断地做出改变，否则就很可能被其他竞争对手超过，在这场竞赛中，任何一家厂商都不敢掉以轻心。而作为开发人员，不得不试图了解各种技术，有些人会努力掌握所有技术，而更多人会由于时间和精力的不足，选择一种，这种选择很残酷，很大程度上是一种赌博，压上的是自己的时间和精力，但是否能够获得长久的回报还不得而知。

在前端开发之外，对于后端的架构也提出了一定的挑战，想要建立能够支持移动平台的服务器，所需要付出的努力要更大。最基本的就是面对的访问量问题，有了移动设备，用户可以随时随地访问，访问量会剧增，所以必须对这种日益巨大的挑战做出充分的准备。

另外，在移动设备上，如何才能够获得更好的用户体验，从而抓住用户，这也是对于前端的设计师提出了巨大的挑战，由于各种限制，在移动设备上无法达到和传统 PC 一样的用户体验，并且，很多情况下，在移动设备上的操作也和传统 PC 有很大的区别，那么我们应该如何做才能够让用户方便而开心地使用呢？

总之，新的趋势为这个世界带来了更多的新挑战，而挑战的存在就意味着机遇的存在，面对这一切，你做好准备了吗？

新趋势新挑战



人人网移动开发架构 — Silverlight for Windows Phone 7 开发体验

Mobile 进行时： Dojo Mobile 控件初探 — 虚拟研讨会：移动 Web 应用开发最新动态

特别专题

人人网移动开发架构

作者 闫志东、闻华强、马小东

前言

说起手机操作平台的发展先要说移动终端的发展，因为平台的发展离不开移动终端，近十年移动终端发展和未来移动终端趋势大体可分为以下四个个阶段：

第一个阶段：功能终端。满足用户基本通信需求，如发短信、打电话，附加些贪食蛇、推箱子小游戏。

第二个阶段：智能化的终端。可扩展第三方应用，实现上网浏览等互联网基础功能，以诺基亚 S60 手机为代表的。

第三个阶段：互联网和平台化的终端。手机和互联网更加紧密，浏览器、流媒体更加强大，互联网应用和手机系统特性结合的更加紧密；手机成为了一个平台，用户可以通过下载第三方应用来 DIY 这款终端，如偏好音乐，可以下载音乐类型的应用。代表为 iPhone、Android 和 Windows Phone 7。

第四个阶段（未来趋势）：物联网化的智能终端。此阶段的特点是现实生活和网络通过传感设备结合的更加紧密。

目前我们处于第三个阶段，对用户而言，由于收入不同、兴趣爱好不同、需求偏好的不同以及手机私人属性和随身性的特点，产生了不同的用户体验；对各个厂商而言，由于目标市场的定位不同、商业利益的不同、技术背景不同，造就了不同的手机操作系统。最终形成了手机操作平台多元化的局面。

目前主流手机操作平台可分为：Symbian、Android、iPhone OS 、MTK、Windows mobile、Wp7 六种。下面分别简述下这六个平台的情况。

Symbian：昔日王者，虽然眼下受到了 android 和 iPhone 的强势狙击，被其瓜分了部分市场，但是价格低，易用性强，应用程序多，加上诺基亚的品牌、渠道等优势，在短期内智能机霸主地位很难撼动。中期来看市场中心下移走中低端智能机路线，长期来看，有可能被 WP7

取代。如果它不革自己的命，那么很可能被别人革命。

Android：势如破竹，据国外媒体报道 Android 在去年第四季度已超过 Symbian 成为全球最大智能手机平台，结束了在 Symbian 在智能机领域长达 10 年的统治地位。作为后来者，Android 借鉴了 iPhone 的操作体验，但是由于 Android 完全开源，对于手机厂商和运营商来讲，很容易定制成自己特色和服务的手机，加上 Android 强大的互联网功能，因而获得二者的青睐。完全开源是把双刃剑，由于各厂商分别定义了各自的产品，这种不标准和不统一会给第三方软件适配带来门槛，会导致在单个某型号的移动终端 Android 应用偏少，所以 Android 有可能成为智能手机中的山寨机。

iPhone OS：神话缔造者，从热销的程度我们可以看出 iPhone 4 创造的奇迹。超炫的 UI 设计，良好的交互操作，海量的应用，牢牢占领高端市场。从短期来看，iPhone 4 突出的优势会让它再火一段时间。但是由于是自有系统，市场占有量取决于苹果手机终端用户认可情况，所以长期看，主要取决于苹果手机发展和竞争对手的变化。

Windows mobile：廉颇老矣，尚能饭否？无论从 UI 视觉效果，还是从易用性，还是第三方应用，Windows mobile 都完败 Iphone 和 Android。壮士暮年，该退隐江湖了。

MTK：山寨大王， MTK 是一个封闭的环境，不支持可扩展的应用，同时原功能也不完善，总之是个半成品。需要中间厂商来完善。相比较来讲，第三方程序少，易用性一般。山寨机的价格和功能形成的性价比优势，占据低端市场。

WP7：救世主，作为微软和诺基亚的救命稻草是值得期待的，笔者曾经体验过 WP7，采用卷轴式 UI 设计风格，使 UI 体验别具一格。系统和互联网应用的紧密结合，加上诺基亚和微软的强力支持。这个操作系统是值得期待的，有望在智能机领域形成 WP7、Android、iPhone 三足鼎立的局面。

上述六大平台分别对应不同的体验和功能实现。对产品设计人员和开发人员而言，它们通常会参照移动终端的 UI 设计规范。因为移动终端系统本身定义了一些常用的控件和响应方式。产品保持与终端系统的一致，不但可以降低开发成本，而且易于用户学习和使用。面对诸多平台，尤其是各个平台功能特点不尽相同，操作方式不同，屏幕大小不同，而每个主流平台又有相当规模的用户群，拥有众多不同的 UI 规范，这对于全平台的产品而言，无疑是具有灾难性的。？

本文就人人网移动开发中不同终端平台的差异和架构统一问题，以及相关服务器架构进行探讨。

移动终端之江山一统

历史历历在目

人人网 www.renren.com（原名:校内网），从０８年下半年开始手机软件的研发，当时国内一二线的互联网公司也已经开始了移动互联网的布局，但已发布并可供参考的产品并不多，尤其是 SNS 本身也还是一个新的互联业务，让我们的用户可以在手机上方便地访问 SNS，这可一下把我们难住了。为了可以快速推出第一个版本试水，我们先是选择 JavaME 平台来开发第一个人人的手机客户端。

人人网的主要业务包括新鲜事，个人主页（状态，日志，相册，留言），好友，站内信，聊天，游戏等等，这些业务都互相关联与衬托，并围绕好友关系（Social Graph），如果要把这些业务都搬到手机，短时间内根本无法完成，因为客户端类的软件与浏览器的网页在展现与交互上有非常大的差异，手机的屏幕大小限制也给设计带来了很大的困难，无疑是雪上加霜。当时我们挑选了用户常用的几个业务，新鲜事，个人主页（状态，日志，相册，留言），好友，站内信，按主站顶部导航的排版方式，设计了一多标签的导航界面，每个标签一个业务，页面跳转同主站，如下图：

图 1

看上去这个设计非常简约明到几乎完美，代码也非常好实现，大家激情澎湃，斗志昂扬准备迎接移动互联网的又一个奇迹，也许你和我以及我们的产品经理一样，低估了这一切，人人网的业务可不像聊天软件那样单纯，当我们的工程师各自完成自己的分配到的业务，并开始处理不同业务之间界面的跳转时，不详的预感笼罩了整个团队，当时的轻率导致了严重危机，大家知道，在我们通过浏览器访问网页，页面中超链接可以让你随意跳转到任何一个页面，且这些页面并不一定是同一个业务的相关页面，如我从个人主页也可以直接跳转到好友（跨标签），而且通过浏览器的后退按钮可以返回前面的页面，客户端类软件可不能做成这样的

自由，我们应该怎么处理不同业务界面的跳转呢？当时大家理解的跳转，根本没有考虑到不同业务之间后退的问题，而且浏览器的页面跳转，浏览器本身是不用知道下个界面是哪个业务，而客户端必须知道，否则根本无法处理事件交互。技术慌了，产品经理也慌了，眼看承诺的交付时间一天天临近，大家还是没有想出一个非常好的办法，多次尝试失败，有的方案，页面的跳转连我们自己都晕过去了，最后我们的第一个版本的 JavaME 客户端 1.0 以失败告终。

首战不利，大家心里都不是滋味，虽然通过后几个月的努力，最后我们决定将 1.2 版本的客户端以 beta 版本的形式发布，并公开提供了下载，除了拍照上传这个功能让我们值得高兴一下之外，其它功能也许只是让我们感觉能用而已。JavaME 的失利让我们从自负中清醒，骄兵必败。经过一段时间的讨论与分析，大家一致认为，我们需要一个手机浏览器，人人的业务不论在 PC 端还是手机端，最理想的展现方式还是浏览器，于是我们开始手机浏览器研发道路，我们花了 3-4 个月的时间，开发出了我们第一个基于 JavaME 的浏览器引擎，代号：Across（为什么起这个名，后面说）。浏览器引擎架构参考 Webkit，如下图：

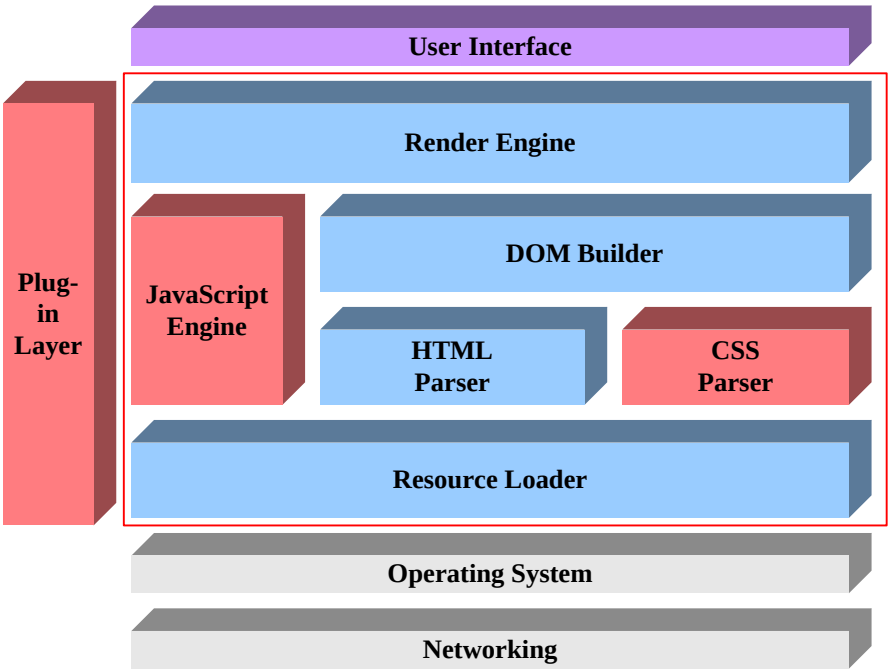


图 2 Across 浏览器结构图

上图中蓝色部分是我们引擎的实现部分，红色的 JS,CSS 及 Plug-in 是未来的计划，从技术角度看来，这个架构看上去非常的美，模块功能划分清晰且扩展性强，我们只要重写 Render Engine 可以把一个普通的页面渲染成我们想要的任何效果，遗憾的是最终我们还是没有把浏览器这个解决方案应用到正式发布的人人客户端版本，原因很简单，它还不是很完善。我们预先做了大量的测试用例，在完成开发以后，我们按测试用例逐条进行了测试，最终结论虽然在性能和 xhtml 标签支持上达到了我们的预期，但稳定性和包体积却并没有达到理想的状

态：运行时内存消耗在 xhtml 页面大小的 8-10 倍左右，如一个 30K 的 xhtml 页面完成解析，渲染必须保证有 300K 左右的空闲内存，如果渲染多个页面或频繁渲染新的页面，就很容易出现崩溃（后来我们发现代码中还有存在一些内存泄漏的地方）；另外，打包以后 200k 的体积对于 JavaME 手机来说已经大了。09 年上半年，Android 发布了 1.5 的 SDK,iPhone 入华的消息也是传得到处都是，显然，在这样的行业形势下，公司管理层已经对我们在 JavaME 上做浏览器引擎的计划失去兴趣，我们的工程师被重新安排了新的任务，Across 没有见到用户就成了历史，浏览器没有救得了我们。

痛苦中反思

09 年上半年是我们最痛苦的一段时间，折腾了大半个年头，结果我们没有发布一个值得骄傲的产品，市场却快速变化着，iPhone，Android 带着闪耀的光芒进入了大家的视野，我们也不得不做出调整，开始分兵投入 iPhone 及 Android 的阵营。我们开始反省前面的失败，我们似乎走了两个极端，第一次在考虑产品及技术的架构时过于的简单草率，以致后期面临强大的心理压力，第二次却是一个典型的过度设计案例，虽然从技术角度看这是一个非常有挑战且非常有意思的项目，当时我们的设想是先完成 JavaME 平台的浏览器，然后移植 Symbian 等其它平台，统一架构。但是移动互联网市场近几年的急速变化，不论从人力和时间上都不允许我们再把浏览器项目继续下去。

为什么我们要统一架构？

人人网的业务种类非常多，而且 PC 端都基于浏览器网页的模式，不论内容还是排版都经常需要优化变更，如果我们通过纯客户端的形式把全部现有业务迁移到到手机端，那么，当我们完成第 5 个业务的迁移时，可以前两个业务主站已经发生了变更，或者客户端刚刚上线之前的某个业务已经需要兼容运行了，在这种情况下，要么我们能快速迭代客户端版本，赶上主站的业务的迭代速度，要么我们使用浏览器或类似浏览器的模式，所有业务放在服务器做，这就是我们为什么考虑开发 Across，名字意在横跨所有手机终端平台。

真的可以统一架构吗？

可以，浏览器本身就是一个非常优秀的跨平台解决方案，但这个方案的前期投入非常大，且项目执行风险过高，人人网的业务大都是基本动态网页实现，使用了大量的 AJAX 及 Flash 技术，最终我们放弃了浏览器方案，我们要的统一架构肯定不是这个。

山穷水尽,柳暗花明

放弃了浏览器的方案，我们可谓山穷水尽，难道还有第三个方案？Facebook 的 iPhone 应用

给了我们很大灵感，看图：

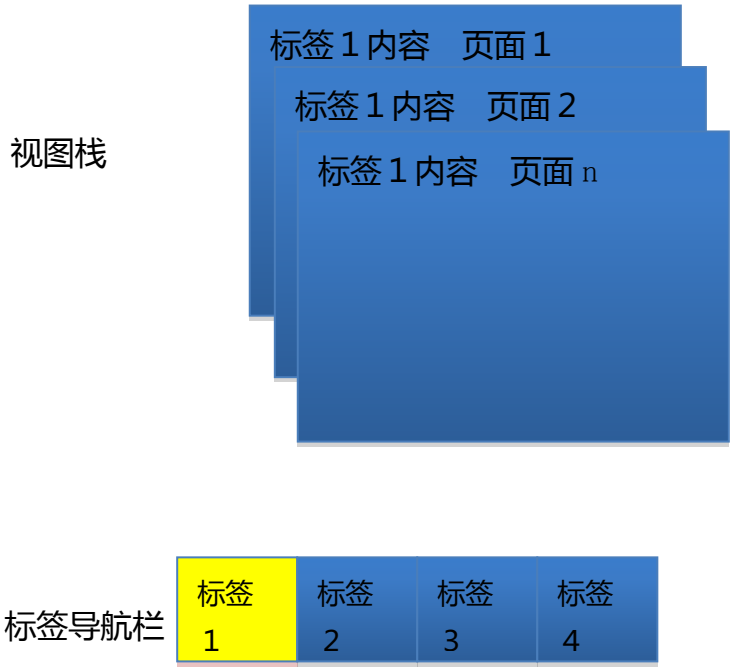


图 3

从产品的角度看，这个图显示的布局与我们第一次尝试的设计没什么区别，深入一点我们会发现，这个设计与之前的设计最大区别在于页面跳转，每个标签都有独立的一个视图栈，理论上无限大，通过当前栈顶视图可以打开新的视图后自动压栈保存，当前视图如果要后退默认退回视图栈里保存的上一个视图的内容。那如果是标签 1 的页面需要跳转到标签 2 对应的页面怎么办呢，是否自动切标签？答案是不切，标签只是用于业务导航，且有独立的视图栈，视图栈中的页面可以与业务无关，打个很好理解的比方：当我们在使用 Chrome 的浏览器时，我们同时在多个标签分别打开多个不同网站或页面，也可以打开同样的网站或页面，每个标签都有一个独立的后退的记录，这种设计非常有规律，用户容易理解不容易晕，现在页面跳转及后退的问题很好的解决了。不论 JavaME，还是 iPhone 或者 Android 的客户端，我们都使用了同样的设计。

数风流人物，还看今朝

当我们客户端都使用了这种标签+视图栈的方案以后，我们的各平台在设计上基本达到了统一，并在现有设计上快速迭代演进。大家可能想了，在代码层统一这才叫本事，也许你没错，但是我们不会轻意再做这样高风险的尝试，如今手机平台的差异相当的大，就从主流平台的

开发语言看就够你折腾了，JavaME 及 Android 是使用的 Java，iPhone 使用的是 Objective-C，Symbian 是纯 C++，现在诺基亚与微软联姻 WP7，可 WP7 将不再支持 C/C++的开发，主推 C# + Silverlight，好吧，我们只能再观察一下了。

在接下来的一到两年，移动互联网将以前所未有的速度发展，大部分互联网公司都开始了或已经推出了较成熟的移动终端的解决方案，创业公司也会层出不穷，推出各种优秀的移动终端应用：移动支付，LBS，基本通讯簿的即时通信，手机音乐，手机视频，手机阅读等等，iPad 点燃平板电脑的硝烟，平板的设计再次给了我们很大的挑战，数风流人物，还看今朝。

高可靠性和可扩展的服务

现在移动互联网拼的都是服务，客户端良好的用户体验背后，都有强大的服务器技术支撑，人人网也不外如是。

业务层次模型

人人网采用 JavaEE 技术作为主要的业务解决方案，基本按照通用的 JavaEE 模型进行架构设计，如下图：

图 4 人人网业务层次模型

- WEB 层基于 REST 风格和 MVC 设计模式，为用户提供基于 WEB 的访问接口

人人网采用的是自己开发的 WEB 框架 Rose，该框架基于 Spring Framework，类似 RoR 框架，增强了对 Controller 编码部分的默认约定和 REST 风格 URL 的支持，该项目前已开源，下载地址为 http://code.google.com/p/paoding-rose/

- 业务层封装业务逻辑，为 WEB 层提供业务接口，操作数据访问层提供的数据。

  人人网开发了自己的 SOA 框架 XOA 以支持业务层抽象，该框架结合 Rose 框架，以 REST 风格对业务进行分类、消息格式封装和路由，如以下 URL：

  ```
  xoa://blog.xoa.renren.com/photo/{user-id}/{photo-id}
  ```

  该 URL 代表某用户的某个照片，操作 GET/PUT/POST/DELETE 分别对应对应照片资源的读、修改、新增、删除。即通过资源+操作的方式对外提供 Service。

  XOA 支持远程调用，并可以通过简单添加服务器的方式进行横向扩展。该框架目前准备开源，敬请关注。

- 数据访问层提供对数据库访问的封装

  人人网使用 Java 语言开发了自己的 Object-Relation Mapping 框架 JADE（Java Database Engine），并支持数据库的水平横向切分。该框架和 Rose 框架一体开源，下载地址相同。

- 数据持久层数据的持久存储，主要采用 MySQL 数据库，并且开发了自己的海量存储系统 Nuclear。

  Rose、Jade、XOA 作为集成度很高的一整套解决方案，在人人网大量采用，大大降低了开发成本，并在框架级解决了服务于企业解决方案的 JavaEE 技术在互联网领域的适用性。

可扩展的高性能系统

人人网每天都要承受亿级 PV 海量用户的并发压力，和其它大型互联网站点类似，服务器架构方面做了很多工作。

高性能数据存储系统

在数据存储方面，人人网做了以下工作：

- 和其它互联网大型站点相同，MySQL 数据库做水平拆分以支持横向扩展
- 人人网作为国内第一大 SNS 网站，欲存海量 UGC 数据，必有海量存储系统。Nuclear 存储系统在高性能、高可靠、可扩展的海量数据存储需求下横空出世。

Nuclear 本身的数据存储基于 Key-Value 形式，底层可以使用 MySQL/Memory, Cassandra, TC, Redis 等存储引擎，提供弱结构化的查询功能。

1. 高扩展性

一个 Nuclear 集群支持 1 到 n(n<264)个节点(Node)的规模

2. 高可靠性

单个节点的 crash 永远对系统的运行造成影响，不存在单点风险。系统永远可写入。

3. 高性能

在 4 个节点、一般服务器配置情况下有测试数据表明单节点访问可达 15862 req/s, 平均单次请求耗时仅 5ms。

本文不欲详细介绍 Nuclear, 有兴趣的读者请参考我们的技术站点。

可扩展的高性能业务服务系统

人人网的业务层是支持分布式、可横向扩展的。

- 人人网主要使用 JavaEE 架构进行业务开发，其中 Spring 提供的 IoC 和 AOP 功能分别起到了业务对象装配和横向关注点分离的良好抽象。XOA 框架基于 Spring 和 netty，使用 google 的 spdy 协议作为网络传输协议，除享受到 Spring 红利之外，也提供了基于 Java NIO 的网络高性能服务器环境。单个 XOA 服务是无状态的，具有幂等性，XOA 客户端使用 Java NIO、通过 Keep alive 保持对各个后台服务器的 TCP 长连接，可实时监测后台服务器的健康状况，并把用户请求负载分散到各个后台服务器上，在单个节点失效的情况下不影响服务，如图 5 所示：

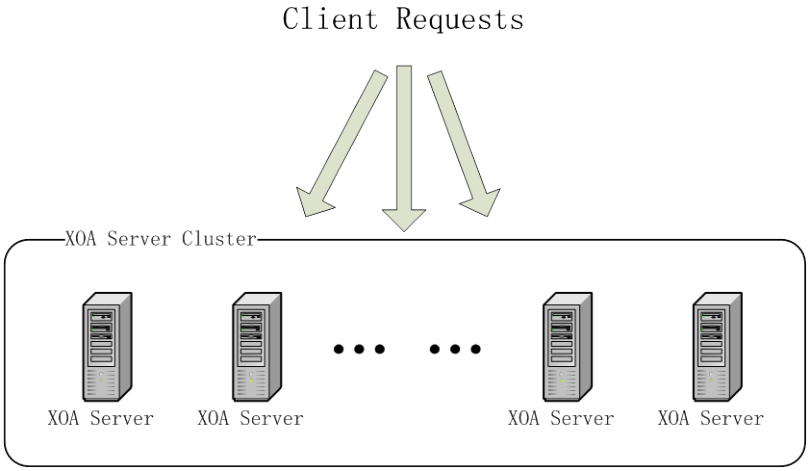


图 5 XOA 负载均衡

- 很多关键性的业务对性能要求特别高，也需要借助很多 Linux 操作系统的特性，这时 Java 的优化已经不能满足需求，需要使用 C++语言进行开发。人人网采用 ICE 框架，进行这部分业务的开发，它解决了 Java、C++等多种语言开发的框架和通讯问题。人人网目前使用 ICE 进行开发的业务层称之为中间层，主要解决类似搜索、用户好友关系计算等性能要求苛刻的底层关键性业务问题。其运维模式和 XOA 类似。

  和其它大型网站一样，业务层使用 Memcached 作为业务层的分布式数据缓存，且根据业务将缓存集群划分为多个池，集中进行管理，如图 6 所示：

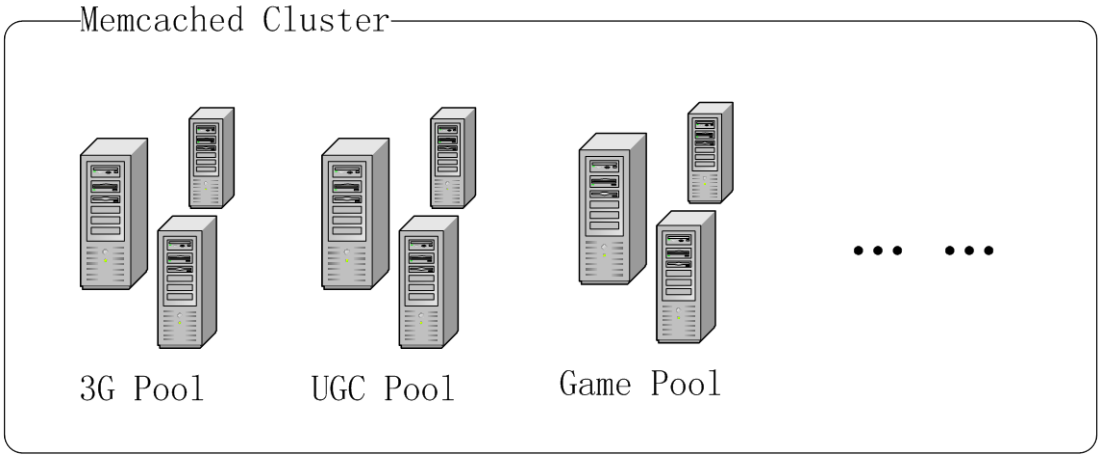


图 6　Memcached Pool

- 在 WEB 层，使用目前性能最高的 Servlet 容器产品 Resin 作为 HTTP Server，使用自行开发的 Java WEB MVC 框架 Rose 对用户请求请求分发。负载均衡方面使用了 F5 或者 nginx。为了减少 Session 复制同步的开销，每台 WEB Server 都禁用 Servlet Session，即每台服务器都是无状态的，单个 Web Server 失效后，不会发生用户状态丢失的问题。用户状态的跟踪由中间层统一处理。

对移动终端的支持

服务器对移动终端的支持主要是通过 HTTP 协议提供 json 数据接口实现的，服务器基本采用了人人网共同的架构：

- HTTP WEB 层做了更多的 MVC 抽象，除了提供基于 html 的 Page View 之外，还生成只提供 json 或者其他数据格式的 Feed View。
- 服务器通过 gzip 压缩减少流量，节省用户资费。
- 客户端构建 REST 风格的 HTTP 请求，WEB 服务器下发数据完成远程调用。

3G 的 API 层直接面向移动终端，提供基于 HTTP 和其他 Socket 协议的服务器访问接口，并在业务层抽象出 3G 部门自己的公共平台，如图 4 所示：

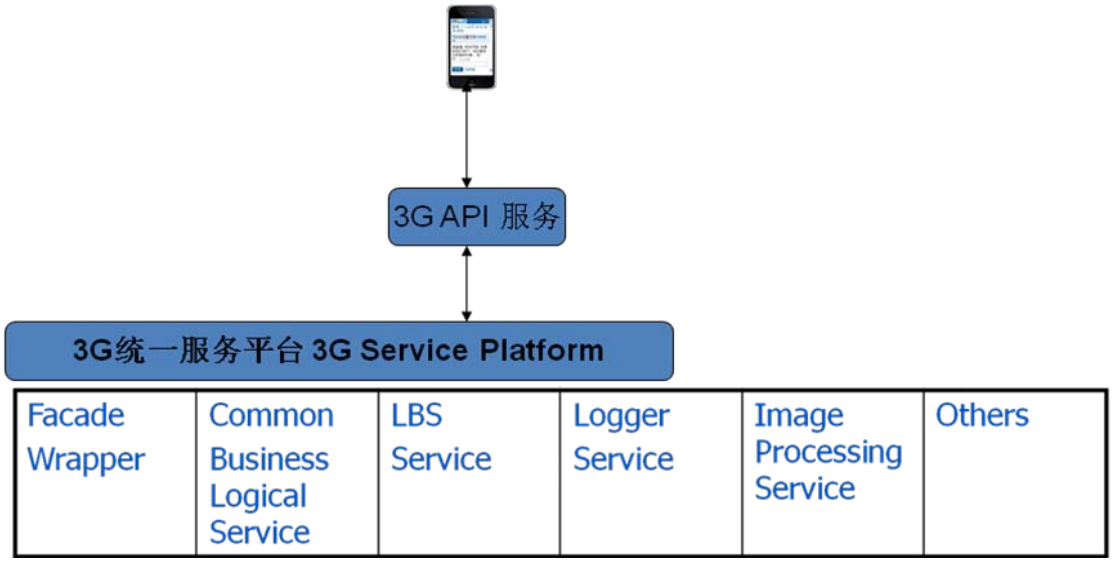


图 7 3G API 架构

除此之外，针对移动终端的特点和中国特色的移动互联网环境，做了一些特殊的工作：

- 低端机型运算能力和内存资源都有限，API 平台做了相应优化
  - 复杂的运算服务器完成，减少移动终端负载。如图片的缩放、裁剪运算、图片质量就是服务器进行的。
  - 客户端和服务器进行的数据输入输出尽量减小，这样可以节省内存存储。在表示相同数据的情况下，尽量采用数据量小且易于解析的格式。API 平台使用了 JSON，没有使用 XML
- 为了减少用户资费，传输流量进行了控制
  - nginx 开启 gzip 压缩，减少网络传输流量。
  - 中国移动的 cmwap 网关会自动对服务器输出的 gzip 流进行解压缩，从而使 nginx 的 zip 压缩失去了意义。这种情况下，api 服务器对输出进行 gzip 压缩后，作为普通二进制流进行响应输出。
  - 除了纯文本的 json 之外，api 服务器也支持基于 google protocol buffers 格式的输出，从而提供更紧凑的格式输出。
- 提高客户端访问速度

- Api 平台支持基于逻辑时序的批处理操作，将多个 api 的网络接口调用合并为一个，减少多次 tcp 握手造成的巨大时间损耗，提高客户端每个 UI 界面的响应和显示速度

作者简介

闫志东，人人网 3G 事业部高级技术经理，资深工程师，人人网技术委员会委员，目前负责人人网 3G 部门服务器架构方面的工作，他在基于 C++、Java 的服务器技术和架构方面拥有多年经验。个人非常喜欢阅读计算机技术、科幻、历史类书籍。

闻华强，人人网 3G 事业部高级技术经理，资深工程师，是人人网 3G 部门客户端技术负责人，他有长达七年的移动终端开发和管理经验。阳光男孩，热爱体育运动，对《明朝那些事儿》爱不释手，是忠实的明矾。

马小东，人人网 3G 事业部产品经理，在移动互联网业界浸淫七年，对很多产品大趋势的把握和细节设计有非常独到的看法。爱手机、爱生活、爱人人。

相关内容：

- 黄晶谈人人网架构
- NEC提出统一交互与协作的新软件架构
- HTML5 在豆瓣中的应用
- Google Android SDK舍弃Java ME，支持Java Lite和Apache Harmony
- Java EE 6 Web Services：JAX-RS 1.1 提供了基于注解的REST支持

特别专题

Silverlight for Windows Phone 7 开发体验

作者 李永伦

直接下水

正如学习游泳应该下水尝试一下，而不是坐在教室里看书，学习 Windows Phone 7 开发技术也应该亲身实践一下，而不是仅仅翻阅资料。那么，怎样实践呢？不必想得太复杂，这里所说的实践就是找个需求实现一下。那么，找个什么需求来实现呢？也不必想得太复杂，作为一名程序员，我们每天都在帮助别人实现需求，可我们自己的需求呢？千万别告诉我你没有需求哦！想象一下，当你拿到一个 Windows Phone 7 手机，你会希望它为你做点什么特别的事？从这点出发，发挥你的想象力，你应该能找到一个起点的，比如说，写个课程表应用。

有了需求，很自然就想寻找实现需求的工具，说到这里，你可能马上想到价格不菲的 Visual Studio 2010 和 Expression Studio 4，有的话当然是好，没有的话也不用担心，因为微软提供了免费的 Expression Blend 4 for Windows Phone 和 Visual Studio 2010 Express for Windows Phone，足以应付常见的 Windows Phone 7 开发需求了。装好开发环境之后，可以做个 Hello World 试一下，就像图 1 这样：

图 1

很多人觉得 Hello World 太过简单以至于不屑一顾，我不是这么想的，我认为 Hello World 的价值之一在于用最简单的方式验证开发环境是否成功安装和正确设置，而这正是后续故事得以展开的基准点。

在开发 Windows Phone 7 的应用时，我喜欢用 Expression Blend 来设计用户界面，然后用 Visual Studio 来编写内部逻辑。Expression Blend 提供了丰富的设计体验，其中一个我最喜欢的特性是设计时数据的支持，你可以按照图 2 所示的方式从 XML 文件里导入现有的数据：

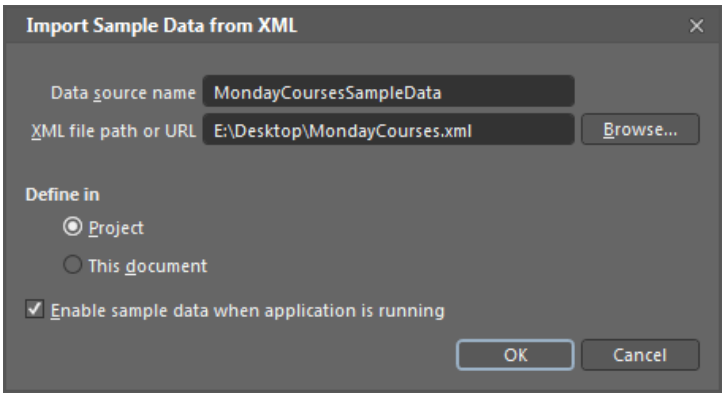


图 2

然后通过拖放把数据绑定到控件上，如图 3 所示：

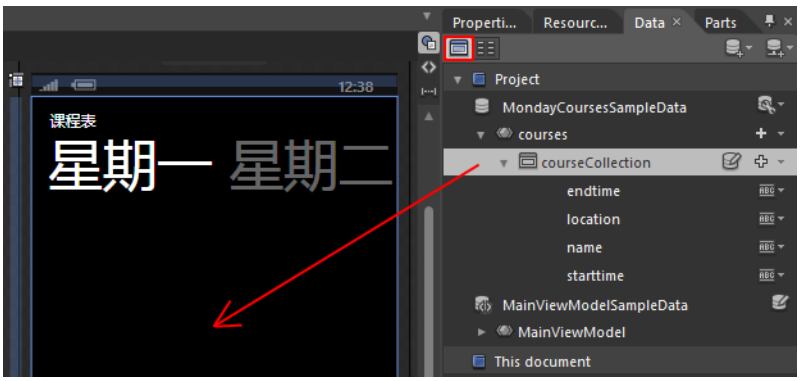


图 3

此外，你也可以让 Expression Blend 为你自动生成设计时数据。当你在 Expression Blend 里设计好用户界面之后，就可以在 Visual Studio 里为控件编写事件处理程序了。如果你在 Expression Blend 的 Options 里设置了"使用 Visual Studio 创建和编辑事件处理程序的代码"，如图 4 所示：

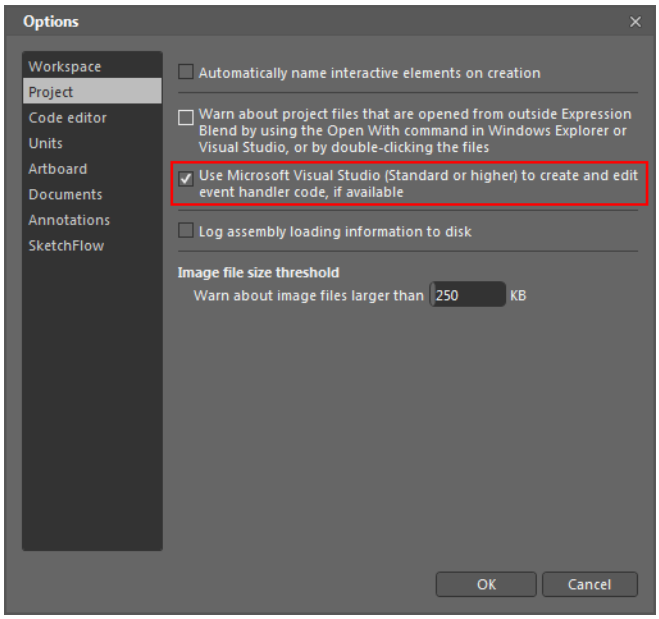


图 4

那么当你在图 5 所示 Expression Blend 的 Properties 面板上双击控件事件旁边的编辑框时，Visual Studio 会打开对应的代码文件，并定位到对应的事件处理程序，如果该事件处理程序还没有，Visual Studio 会为你创建一个。当然，如果你比较极客，喜欢手工编写 XAML，那么 Visual Studio 基本上已经可以满足你的所有需求了。

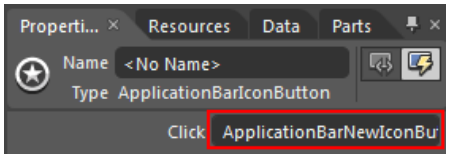


图 5

应用的开发和测试是密不可分的，当你想运行并测试应用时，就轮到 Windows Phone 7 模拟器出场了，图 6 像你展示了我们的程序在模拟器中运行的样子：

图 6

你还可以通过 Visual Studio 在模拟器里调试你的应用。不过，这个模拟器只支持系统自带的软键盘输入，你无法通过电脑的键盘进行输入，这会为输入测试数据造成一些不便。

善用资源

虽然 Expression Blend 和 Visual Studio 已经提供了很多 Windows Phone 7 控件，但这显然无法满足日益增长的需求，于是微软发布了 Silverlight for Windows Phone Toolkit，通过附带的示范程序（如图 7 所示），你可以感受一下这些组件能够实现什么功能和效果：

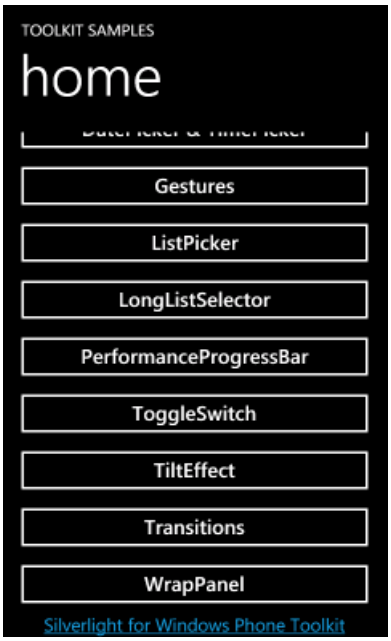


图 7

最新版本是 Feb 2011，里面包含了以下组件：

1. AutoCompleteBox
2. ContextMenu
3. DatePicker
4. GestureService/GestureListener
5. ListPicker
6. LongListSelector
7. Page Transitions
8. PerformanceProgressBar
9. TiltEffect
10. TimePicker
11. ToggleSwitch
12. WrapPanel

如果你在使用期间遇到一些奇怪的问题，请别急着自己解决，先看看别人是否也遇到这些问题，另外，你也可以看看是否有人提交了相关的补丁，除非你一眼就能看出问题所在，或者

你享受解决问题的过程。我在使用 LongListSelector 控件时就遇到一个奇怪的问题，我在一个页面上使用这个控件，里面显示一组作业，然后通过新的页面来新建和编辑作业，每当我从新的页面返回时，LongListSelector 控件都无法显示刚才新建的作业或者更新的内容，经过一番调查，我最后发现 LongListSelector 控件没有考虑到打开新的页面这种情况。当我们打开新的页面时，Windows Phone 7 为了节约资源会把原来那个页面卸载（unload）；而当我们从新的页面返回时，布局系统会重新计算并构建原来那个页面，这会导致 LongListSelector 控件在子元素的处理上出现错误（这个问题在最新的 Feb 2011 版本里也没得到修复）。从这里我们不难看出，虽然 Windows Phone 7 的应用属于本地应用，但它和我们平时接触到的 Windows 应用程序是不同的，你需要搞清楚页面的生命周期以及页面的导航是如何工作的，除非你的应用只有一个页面。

此外，微软也提供了适用于 Windows Phone 7 的 Prism，如果你打算在 Windows Phone 7 应用里使用 MVVM 模式，那么 Prism 里面提供的一些类型就会帮上忙。当然，你也可以选择其它 MVVM 框架，比如说，MVVM Light Toolkit，还有 Caliburn Micro。类库和框架固然能够节省一定的工作量，但如果你想切实地提高自己的能力，那么需要的就是知识和技术，你可以读一读微软发布的《Windows Phone 7 Developer Guide》（英文）和《Windows Phone 7 开发者培训包》（中文），你也可以上 Windows Phone 7 Development Best Practices Wiki 看看别人有什么最佳实践，吸取一下别人的经验。如果你有些问题确实不知道怎么处理，还可以试着到 App Hub 的论坛或者博客园的博问上问一问。

技术之外

如果你希望写个好的应用，那么仅仅关注技术方面的内容是不够的，你需要把应用看做一组完整的用户体验，然后站在用户的角度考虑每个细节。

举个例子吧，你认为图 8 中的页面设计有没有问题？

图 8

没有，一切看起来都是那么正常，至少表面上确实如此，用户可以完成相关的操作，页面的相关功能也没有 bug，可是，在图 9 中你会看到，每次输入完毕之后都不能直接关闭页面，因为确定按钮被软键盘挡住了：

图 9

要想关闭页面，用户就需要先单击页面空白处关闭软键盘，然后才能单击确定按钮关闭页面。你觉得这是不是一个问题呢？我在测试这个应用的时候，一开始只是觉得有点儿麻烦罢了，但当我重复这个过程大概十次之后，我就开始痛恨这个多余的步骤，并且深深地体会到这个

小小的细节为用户体验带来了多么糟糕的影响！于是，在开发后续的功能时，我尝试了新的设计，用 Application Bar 按钮替代传统按钮，就像图 10 中显示的这样：

图 10

这样，用户在输入完毕之后马上可以单击确定，这个小小的改动使得整个操作体验变得更加平滑、流畅。

再举一个例子吧，你觉得图 11 中的页面设计有没有问题？

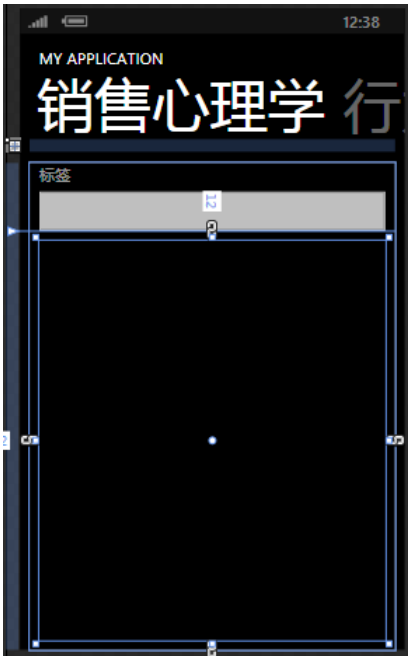


图 11

这是一个支持通过标签进行筛选的笔记应用，上面显示当前标签，单击这个控件将会打开一个新的页面，用户可以在新的页面里选择其它标签，下面显示根据当前标签筛选的笔记。还是那句老话，这个设计很正常，用户照样可以完成相关的操作，那么，还有什么需要修改的吗？试想一下，既然这是一个关于笔记的应用，那么它的主要任务自然就是显示笔记内容，而笔记标签作为一种辅助的手段，不应该占用本来就不多的手机屏幕空间，它应该在用户需要的时候才出现。为此，我修改了页面的设计，使显示笔记内容的 ListBox 占满页面可用空间，然后把显示笔记标签的 ListBox 隐藏在屏幕下方，修改后的程序运行效果如图 12 所示。

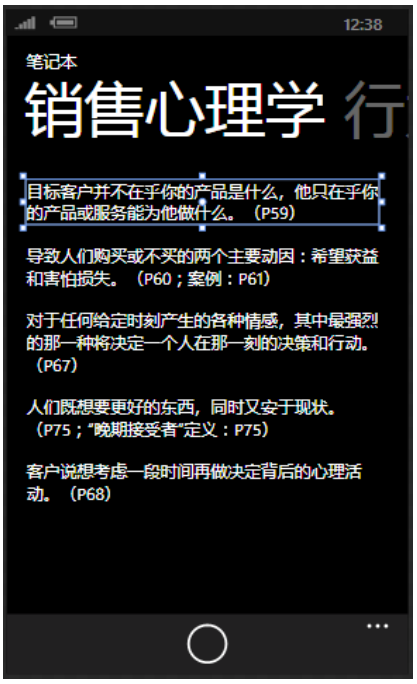


图 12

当用户单击 Application Bar 上的按钮时，显示笔记标签的 ListBox 将会从屏幕下方外面向上平移。这样，应用就可以最大限度地利用有限的屏幕空间来完成最重要的任务了。

很久之前，我看过一部电影，名字叫做《和你在一起》，里面有个情节至今仍然历历在目，余教授在听完他的学生拉的小提琴之后对这个学生说："都对，就是不好，你的音乐没有感情，没有感情的音乐怎么感染听众？"现在的用户对于品质的要求越来越高，他们不再仅仅因为你正确地实现了他们的需求就选择你的应用，他们更加重视你是否切身处地的为他们设想、考虑他们的感受，因为选择是有成本的，他们一旦选择了你的应用，他们就需要尝试适应你的应用，因此他们会评估这样的付出是否值得。下次，当你打算开发一个应用时，不妨问一下自己：凭什么你的应用值得用户选择？

着眼未来

去年 Windows Phone 7 首次发布时，有人戏说它是 2007 年水平的 iPhone，潜台词当然是指它和当年的 iPhone 一样退化为单任务方式，比较讽刺的是，当微软信心满满地解释不支持多任务是为了保证电池的续航性时，苹果宣布 iOS 支持多任务了，尽管那还不是真正的多任务。然而，正当 Windows Phone 7 的开发者学习如何处理单任务方式导致的各种问题并为之而发愁时，微软又宣布 Windows Phone 的后续版本将会支持真正的多任务。如果你希望你的应用在支持多任务的 Windows Phone 手机上有出色的表现，你现在就要把它考虑进你的设计里了。

为 Windows Phone 7 开发应用还有一个广受关注的问题--应用的部署。在 Windows Phone 7 之前，我们可以自行制作 CAB 安装包，然后把它复制到手机上安装，可惜，这种情况在 Windows Phone 7 上不复存在，因为微软决定 Windows Phone 7 的应用必须通过 Marketplace 部署。此外，如果你想在真机上测试你的应用，你需要注册成为 Windows Phone 7 的开发者并交纳 99 美元的会费（目前中国大陆的开发者还不能直接注册），否则你无法把你的应用部署到真机上。为了解决这个问题，Long Zheng、Rafael Rivera 和 Chris Walsh 一同发布了一款名为"ChevronWP7"的 Windows Phone 7 手机开发者解锁工具：

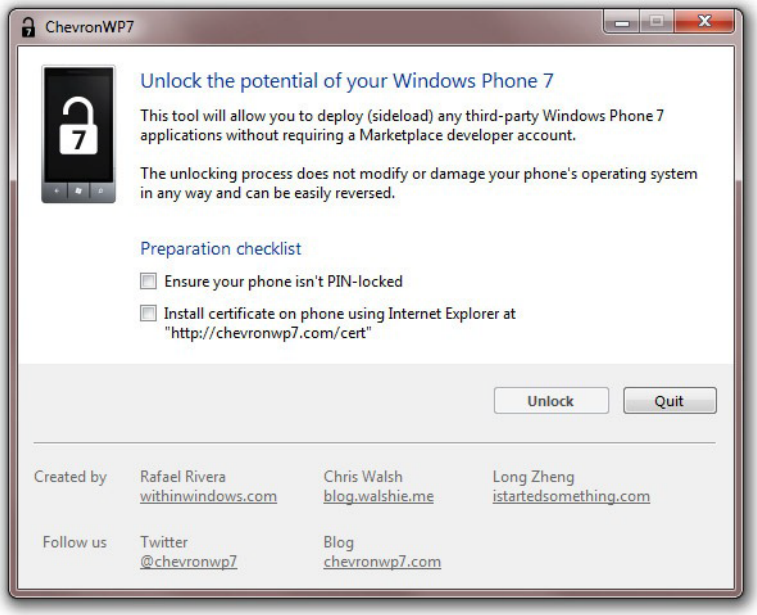


图 13

随后，微软与 Windows Phone 7 破解团队 ChevronWP7 进行了会晤，共同探讨 ChevronWP7 解锁工具的官方化，我相信他们的合作会为我们开发者提供更多部署的选择。

经常留意业界动态，及时学习所需知识，合理利用各种资源，综合考虑不同因素，这些都是

开发一个好的应用必不可少的，然而，如果没有踏踏实实地踏出第一步，一切都是空谈，因此，最终我们又回到了最初的问题：学习 Windows Phone 7 开发技术应该亲身实践一下。

相关内容：

- Silverlight面向客户端，HTML5 面向Web
- Silverlight作为Web应用程序技术的角色引发争论
- Windows Phone 7：突破过去，奔向未来
- Nokia宣布支持Silverlight技术
- Prism 4.0 发布最终版，支持Windows Phone 7

特别专题

Mobile 进行时： Dojo Mobile 控件初探

作者 阮奇

你有想过在 iPhone 或是 Android 设备浏览器上运行与手机原生程序界面相同的 Web 程序吗？ 你有想过让你的 Web 应用更适合在手持设备上交互使用吗？如果是，那么你有必要读一下这篇文章。

移动开发是时下最炙手可热的话题，各种应用程序如雨后春笋出现在应用商店里。大家的智能手机里也多多少少安装了各种应用，然而问题也随之出现：手机的储存空间限制，手机应用的升级维护成本以及多处访问的需求，曾经桌面应用程序也遇到过类似问题，最终导致了 Web 应用的崛起。这是不是预示着手持设备上基于移动浏览器的 Web 应用的发展前景呢？事实上，基于移动浏览器的 Web 应用正在快速的发展，来自 RWW 的调查结果如下图所示，移动浏览器的 Web 应用（Mobile Touch Web sties）的增长速度远高于 iPhone 与 Android 应用。

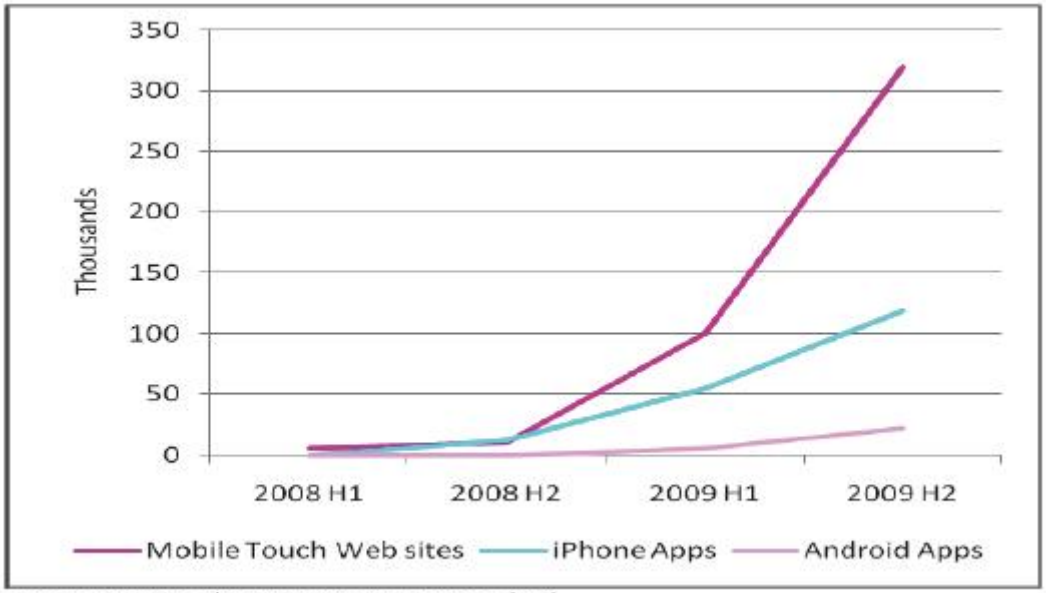


Figure 1: Growth of Mobile Touch Content

基于移动浏览器的 Web 应用的迅猛发展也给 Web 开发者带来了不小的难题， 首先就是桌面浏览器与移动浏览器的事件机制的巨大区别——事件机制： 触摸事件与鼠标事件，单点

事件与多点事件等等， 开发者需要针对手持设备浏览器设计的开发包。 同时，开发者也需要针对手机的小屏幕量身定制的界面控件库，搭建适合在手机上交互的 Web 应用。

针对开发者的这些需求，针对移动浏览器的 Javascript 开发包纷纷登场， 比如 jQuery 的 jQTouch、Sencha（原 Ext）的 Sencha Touch。 Dojo 也在 1.5 版本中推出了 Dojox.mobile 模块，Dojo Mobile 框架是一套移动终端的 Web 应用开发框架，主要面向手持设备上的 Web 富客服端应用开发，提供了 iPhone 和 Android 两套主题，使得基于 iPhone 或者 Android 的 Web 应用具有手机本地应用的外观和效果，而同时也给了开发者更多的主导权，下面将基于 dojo1.5 版本，重点介绍 dojo mobile 框架。

Dojo Mobile 功能结构介绍

以功能性来说，Dojo Mobile 开发框架包含以下内容：

- 一个非常轻量级的控件解释器，仅针对 mobile 控件。
- 在以 WebKit 为内核的浏览器上，大量使用 CSS3 实现与 iPhone 和 Android 本地程序类似的特效;同时也支持非 WebKit 内核的手机浏览器，使用 dojo 自带的 dojo.animateProperty 与 dojox.gfx 代替 CSS3 动画模拟特效。
- 模拟 iPhone 和 Android 本地程序外观的 CSS 主题。
- 一系列模拟 iPhone 或 Android 界面控件行为的控件， 移动设备上的界面控件外观和行为与传 统控件具有很大的不同，下一节会着重介绍 Dojo Mobile 中的控件库。

Dojo Mobile 控件库

在使用控件库之前，首先需要载入 Dojo Mobile 代码环境和相应的 CSS 主题。

加载 Javascript

```
dojo.require("dojox.mobile");
//加载为dojox.mobile定制的解析器，这里也可以载入dojo.parser，不过后者会载入更多的代码，所以在这里推荐使用dojox.mobile.parser.
dojo.require("dojox.mobile.parser");
//为内核不是WebKit的浏览器，加载兼容代码，使用Javascript代替实现CSS3的动画特效。
dojo.requireIf(!dojo.isWebKit, "dojox.mobile.compat");
```

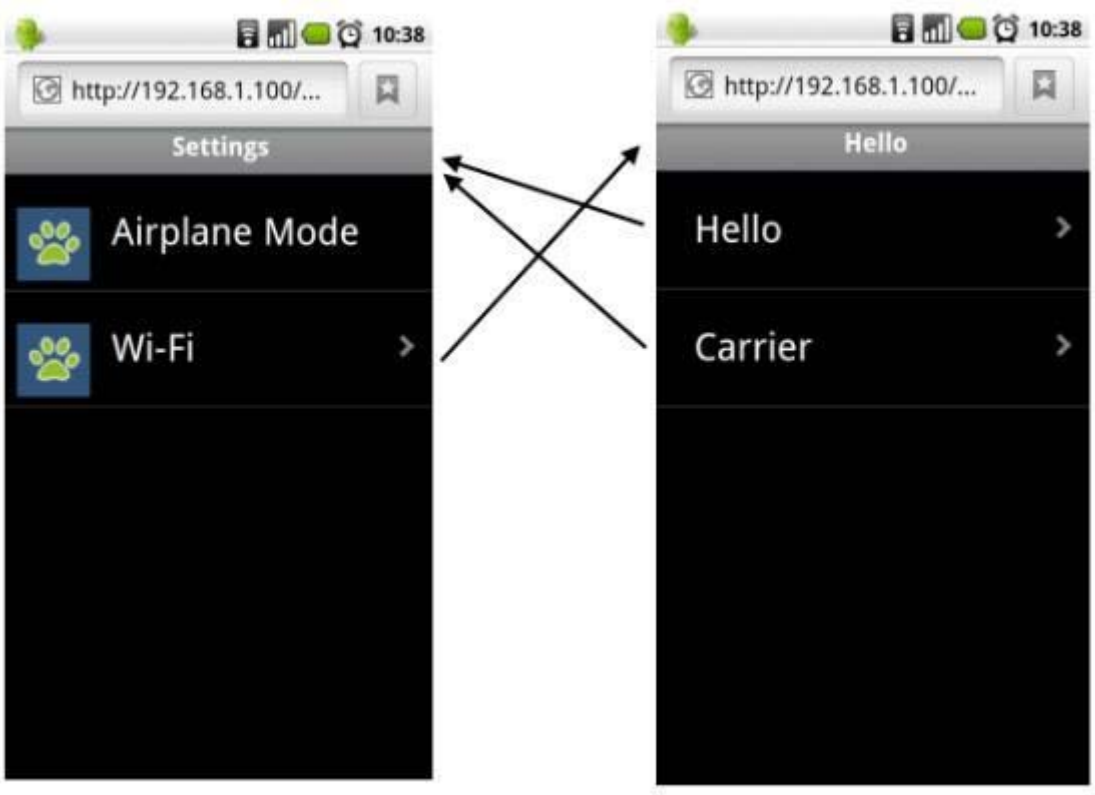
选择模拟手机平台外观的 CSS 主题，这里选择的是 iPhone 的界面风格：

```
<style>
   <link rel="stylesheet" type="text/css"
```

```
    href="dojox/mobile/themes/iphone/iphone.css">
</style>
```

View

View(视图)是占据手机全屏页面的容器控件，一个 HTML 页面可以包含多个 View，但是只有被标记为 selected 的 View 才会被显示。各个 View 之间的 导航只需要通过 moveTo 参数就可以实现。请看下面的“setting”视图与“hello”视图建立导航的例子:



关键代码如下：

```
<!-- "setting"视图, 当前为被选中显示状态 -->
<div id="settings" dojoType="dojox.mobile.View" selected="true">
    <h1 dojoType="dojox.mobile.Heading">Settings</h1>
    <ul dojoType="dojox.mobile.EdgeToEdgeList">
        <li dojoType="dojox.mobile.ListItem">Airplane Mode</li>
        <li dojoType="dojox.mobile.ListItem" moveTo="hello">Wi-Fi</li>
    </ul>
</div>
<!-- "hello"视图，当前为隐藏视图 -->
<div id="hello" dojoType="dojox.mobile.View">
    <h1 dojoType="dojox.mobile.Heading">Hello</h1>
    <ul dojoType="dojox.mobile.EdgeToEdgeList">
        <li dojoType="dojox.mobile.ListItem" moveTo="settings">Hello</li>
```

```
        <li dojoType="dojox.mobile.ListItem" moveTo="settings">Carrier</li>
    </ul>
</div>
```

同一个 HTML 中的 View 之间，通过 moveTo 参数建立了导航链接，十分方便直观。

Heading

Heading(标题栏)是移动应用中非常典型的导航栏控件，一般来说，Heading 位于当前视图的上方，显示视图的名称，在 iPhone 应用中还常常带有 ToolBarButton（工具栏按钮）. 下面的例子是一个包含导航键和工具栏按钮的典型 iPhone 风格的 Heading：

代码如下，点击 Bookmarks 将导航到 id 为 bookmarks 的视图：

```
<h1     dojoType="dojox.mobile.Heading"     label="News"     back="Bookmarks"
moveTo="bookmarks">
    <div          dojoType="dojox.mobile.ToolBarButton"          label="Done"
class="mblColorBlue" style="width:45px;float:right;"></div>
</h1>
```

除了 moveTo 属性可以完成同一 HTML 中不同视图的跳转，Heading 还提供了 goTo(moveTo, href) 控制视图的跳转，goTo 不但可以在本 HTML 页面中完成图间的切换，而且还可以在不同的页面间进行跳转。

这里特别需要指出的是 iPhone 与 Android 应用中的标题栏是有比较大的区别的：iPhone 应用中的 Heading 跟接近于工具栏，会包含很多按钮；而传统的 Android 应用中 Heading 一般只包含当前视图的标题。下图是两者的区别：

iPhone：

Android：

Switch

Switch(开关)也是一个在 iPhone 应用中常见的控件，而在 Android 应用中一般使用 Check Box 来表示开关状态。在状态改变时将触发 onClick 与 onStateChanged 事件。

代码示例如下：

```
<div id="switch" dojoType="dojox.mobile.Switch" value="on" leftLabel="Start"
rightLabel="Stop"></div>
...
<script>
    dojo.addOnLoad(function(){
        dojo.connect(dijit.byId("switch"),"onStateChanged",function(state){
            // "Switch is now on/off"
            console.log("Switch is now " + state);
        }
    });
</script>
```

根据手机 UI 的设计模式，一般来说 Switch 控件和 Check Box 在列表中使用比较得当。

List

在手机屏幕尺寸的限制下，列表已经成为手机应用中最常见的布局方式，dojo mobile 框架中当然也引入了列表控件：EdgeToEdge List 和 RoundRect List。两种列表控件都使用想同的列表项（dojox.mobile.ListItem），可能大家看到这两种列表的名称感觉有点诧异，不过相信看了下面的例子就容易理解了。

EdgeToEdget List： 每条列表项都是从左到右占满了屏幕

EdgeToEdge List 声明方式：

```
<ul dojoType="dojox.mobile.EdgeToEdgeList">
    <li dojoType="dojox.mobile.ListItem">......</li>
    <li dojoType="dojox.mobile.ListItem">......</li>
    ......
</ul>
```

RoundRect List： 一组列表项被包含在一个圆角矩形容器中

RoundRect List 的声明方法也与 EdgeToEdge List 如出一辙：

```
<ul dojoType="dojox.mobile.RoundRectList">
    <li dojoType="dojox.mobile.ListItem">......</li>
```

```
    <li dojoType="dojox.mobile.ListItem">......</li>
    ......
</ul>
```

ListItem

作为列表的单位，声明方式已在上文提及，这里主要介绍一下 ListItem 的比较重要的属性：

- rightText：在列表项的右侧显示文字也属于 iPhone 应用程序的风格，rightText 一般与跳转箭头同是出现，表示该列表项的子列表或者下一个视图。
- btnClass：列表项中不同的 Button 是由 btnClass 来决定的，在 dojox/mobile/themes /buttons.css 中，声明了添加按钮（mblPlusButton)、删除按钮(mblMinusButton)，以及勾选框按钮(mblCheckButton).

Category

Category(目录)也是手机应用中十分常见的内容分类控件，在同一个视图包含多个列表， 在手机本地应用中的联系人列表、手机功能设置等视图都用到了目录控件。下面是 dojo mobile 框架中的列表目录控件：

Category 作为列表的标题， 用法也很简单，只需要在列表之前声明即可：

```
<h2 dojoType="dojox.mobile.EdgeToEdgeCategory">List One</h2>
<ul dojoType="dojox.mobile.EdgeToEdgeList">......</ul>
<h2 dojoType="dojox.mobile.EdgeToEdgeCategory">List Two</h2>
<ul dojoType="dojox.mobile.EdgeToEdgeList">......</ul>
```

TabContainer，TabPane 与 TabBar

TabContainer(选 项卡容器)我们应该已经很熟悉了，与列表的搭配使用成为了手机应用程序的主流布局方式，不过在最新的 dojo1.6 中 TabBar 取代了 TabContainer 与 TabPane 的组合，dojo mobile 的开发者不建议继续使用 TabContainer 与 TabPane。当然，在界面和行为上来看 TabBar 与 TabContainer 并无二 致，如下图所示：

那么来看看两者的声明结构：

TabContainer：

```
<div dojoType="dojox.mobile.View" id="general" selected="true">
    <div dojoType="dojox.mobile.TabContainer">
        <div dojoType="dojox.mobile.TabPane" label="New">
            <ul dojoType="dojox.mobile.EdgeToEdgeList" id="list1">
        </div>
        <div dojoType="dojox.mobile.TabPane" label="What's Hot">
        <ul dojoType="dojox.mobile.EdgeToEdgeList" id="list2">
        </div>
        <div dojoType="dojox.mobile.TabPane" label="Genius">
            <ul dojoType="dojox.mobile.EdgeToEdgeList" id="list3">
        </div>
    </div>
</div>
```

原先的 TabContainer 的结构与 dojo 控件库 dijit 中的 TabContainer 如出一辙：TabContainer 下

包含多个 TabPane，在 TabPane 中包含列表。而在手机中，这些列表作为内容主体被 TabContainer 所封闭，不能被 TabContainer 以 外的控件连接导航。这与 dojo mobile 提倡的视图切换思想不符，再来看看 TabBar 的声明结构：

```
<ul dojoType="dojox.mobile.TabBar" barType="segmentedControl">
   <li dojoType="dojox.mobile.TabBarButton"
       moveTo="view1" selected="true">New</li>
   <li dojoType="dojox.mobile.TabBarButton" moveTo="view2">What's Hot</li>
   <li dojoType="dojox.mobile.TabBarButton" moveTo="view3">Genius</li>
</ul>
<div id="view1" dojoType="dojox.mobile.View" selected="true">
   <ul dojoType="dojox.mobile.EdgeToEdgeList" id="list1">......</ul>
</div>
<div id="view2" dojoType="dojox.mobile.View">
   <ul dojoType="dojox.mobile.EdgeToEdgeList" id="list2">......</ul>
</div>
<div id="view3" dojoType="dojox.mobile.View">
   <ul dojoType="dojox.mobile.EdgeToEdgeList" id="list3">......</ul>
</div>
```

各个列表都被放在视图之中，TabBar 仅仅作为一个导航栏放在页面顶端，使用 moveTo 来切换对应的视图，外部的控件仍然可以导航到这些视图。从控件间松耦合的角度，来看 TabBar 的确是更适合 dojo mobile 设计风格的控件。

结语

移动网络正处在一个高速发展的时期，Web 应用也开始在手持设备上占据主流。开发者需要一个提供标准界面、兼容大部分移动浏览器的开发框架来加速开发工 作，Dojo Mobile 是一个很好的选择，而且 Dojo Mobile 正在不断地完善，在最新发布的 Dojo1.6 中，Mobile 框架又有了大幅改进： 增强了对于手机触摸事件支持，添加加了新的控件。本文介绍了 Dojo Mobile 框架中比较常用的控件，更多的内容有待大家深入挖掘发现。

参考资料：

- Dojo 中文博客
- Dojo 开发文档

原文链接：http://www.infoq.com/cn/articles/dojo-mobile

相关内容：微软计划在 MIX 上宣布 WindowsPhone 7 的多任务处理功能、Mobl——用于创建 HTML 移动应用程序的新 DSL、ScottGu 澄清 Silverlight 发展策略、

特别专题

虚拟研讨会：移动 Web 应用开发最新动态

作者 Dionysios G. Synodinos 译者 胡键

为了掌握移动 Web 应用开发的最新动态，InfoQ 邀请一些该领域最流行的库、工具和框架的缔造者，组织了一场虚拟研讨会。

参与者及其框架分别是：

- Michael Mullany，Sencha Touch
- Brian LeRoux，PhoneGap
- Christopher Plieger，iWebKit
- Chris Apers，WebApp.Net
- Christoph Pojer，mootools-mobile

InfoQ：你能给大家描述一下使用你项目进行开发的过程吗？如何开发和测试 Web 应用？

> **Michael （Sencha Touch）：**我们这个 JavaScript 框架针对的是触控式（Touch-based）移动 Web 应用，开发者在使用 Sencha Touch 时可以继续使用他们喜欢的 IDE 和选择的服务器端。这跟开发其他 Web 应用没任何区别，但是移动设备肯定会给测试方面带来些挑战，因为移动 Web 浏览器不具备插件功能，而该功能给桌面浏览器提供了丰富的调试功能。

> **Brian （PhoneGap）：**开发者在开发 PhoneGap 应用时首先要做的一件事就是决定他们目标用户需要的平台。PhoneGap 支持 Apple iPhone、Google Android、RIM Blackberry、Palm webOS、Symbian、Meego 和 Windows Phone 7。一旦确定目标用户和平台，开发者就需要花点时间搞清楚输入方式和屏幕设备尺寸。比如，iPad UI 就不能工作在 Blackberry 上。Palm 设备有极好的手势（Gestural）支持，Androids 使用物理按键作为主要的交互手段，而 iPhones 则用虚拟按键完成除了电源和音量之外的任何操作。一种 UX（用户体验）模式不能简单的照搬到所有平台。
>
> PhoneGap 项目有一个单独的 WWW 文件夹，开发者可以在其中放置他们的 HTML、CSS

和 JavaScript。PhoneGap 的唯一限制是该 WWW 文件夹必须包含一个 index.html 文件。Symbian 和 webOS 要求应用是单页的，我们认为这是个好的范式。（注，其他 PhoneGap 平台没有该限制，因此，若你不支持 Symbian 或 webOS，你*可以*创建多页应用.....但是页面间迁移会很难看。）

我们让开发者去想办法解决每种平台的构建主要是因为我们无法再分发不同的移动 SDK。此外，要记住平台跟操作系统的对应关系（iOS 需要 Apple，Windows Mobile/Phone 7 要求 Windows）。在 Nitobi，我们一般可能使用下面三种技术之一，取决于要支持的平台。

- 把单独的 WWW 文件夹符号链接（symlink）进每个特定 PhoneGap 平台的代码库
- 同上，但使用 git 子模块或外部 svn
- 手动将 WWW 复制到每个项目的构建脚本

这些技术的任何一种都很适合工作于单个平台的小团队或分散在所有平台的大型团队。之后，就是普通的 Web 开发了，很多 PhoneGap 开发者都尽量在构建步骤节约时间，好让他们集中工作在 Chrome 或 Safari（有时是 Firefox）上。

> **Christopher （iWebKit）**：对最终用户来讲，开发过程相当简单。 用户指南给他提供了所有可用的元素，其初始风格跟 iOS 界面看起来非常像。因此，他要做的全部事情就是把一部分代码复制到一个 HTML 文件中，改变它的内容，然后他就可以继续前进了！要想对其进行测试，只需使用任何 Webkit 浏览器即可，如 Chrome 或 Safari，但我推荐用 iPhone 或 iPod touch（哪个版本都行），因为 CSS 文件产生的布局也是基于检测到的屏幕大小。

> **Chris （WebApp.Net）**： 使用 WebApp.Net，你只需要下载安装包，将其解压到开发文件夹中即可。这时，你可以阅读文档来开始使用。在一个静态 Web 站点上使用 WebApp.Net 相当简单，但要想在动态环境中使用它，可能需要很好地理解 Ajax 和基本的 XML 知识。WebApp.Net 使用 XML 来实现多个部分的异步展现和内容更新。例如，你可以创建完整的层级内容或是只更新某特殊标签内的文本。它还支持从异步响应中执行脚本。

写文档得花点时间，我也正全力以赴地做这件事，但总的来说它仍处于未完成状态。支持论坛是最好的文档，其中有些讨论非常有趣。示例的源代码也非常有帮助，因为它实现了一些非常有趣的框架特性。论坛还有一个板块专门用于发布哪些 Web 应用使用了 WebApp.Net，它们也都是理解该框架的不错起点。

开发者可以轻易地在使用 WebKit 引擎的桌面浏览器内测试自己的 Web 应用。Firefox 也同样得到了支持，另外还提供了一个特殊的 CSS，用来调整特定于厂商的属性。

Christoph （mootools-mobile）： 对于什么是 Web 应用，存在着很多不同的定义。在我看来，Web 应用不仅意味着“丰富的交互”，而且还涉及利用和转换用户数据或设备数据，同时还包括利用用户相关的信息。我相信每个人都有不同的定义和需求。这就是我为何要把东西都尽量以模块形式发布的原因。我发布在 GitHub 上的项目都是独立的，只需要某些 MooTools 的核心组件。MooTools，其本身作为一个模块库，是 Web 应用开发的理想之选，因为你可以对其进行定制，构建自己的版本。

mootools-mobile 旨在提供低层次的功能，如自定义 swipe 和 pinch 事件。它并没打算成为一个全面的移动 Web 应用框架，但它可以成为其基础。模块化是关键，只需在其上进行构建即可。模块化暗示着抽象。这样，开发过程就具备了非常好的可扩展性，因为其他开发者可以更灵活地选择适合（而且只适合）其项目的工具。

我坚信 JavaScript 应该独立于所有内容和形式。我的开发方式通常是不依赖任何 CSS 或 HTML——大多数时候，我也“只”是个 JavaScript 开发者。我甚至不碰任何其他东西。我专注于模块化——这样，每个浏览器和平台都有其自己的一套模块。这也让在旧浏览器或其他不支持移动设备上关闭某些特性变得非常简单。

测试移动应用最好的方式是为尽量多的代码写测试代码，尽可能最小化手工测试的需要。

InfoQ：很多移动设备都有数种输入方式，包括触摸屏，有时还需要多个手指进行操作。你的框架支持这些输入方式吗？你认为它提供了其他额外的好处吗？

Michael （Sencha Touch）：Sencha Touch 是针对触控设备的应用框架，因此我们给开发者提供了一整套触控管理器。我们将基础的 JavaScript 触控事件抽象成开发者用来编程的基础设施（如 doubletap、tap/hold、swipes、pinches 等）。我们认为触控界面提供了移动设备上最佳的交互模型，因为它们能让你解决大量烦人的 UI 问题。我们认为触控/手势接口将在所有移动平台得到支持。拥有虚拟控制可以让你在不需要它们的时候给内容提供更多的空间，支持更丰富的交互，更符合用户的预期。要是不支持手势，一些常做的简单事情（如放大内容，重定位，然后再缩小回原来大小），都会是件复杂的过程。这也是在我们有了手势支持之前，移动应用没有真正腾飞的原因之一。我们认为对于移动应用，按按钮，使用跟踪球（Trackball）等，正慢慢地在移动设备上消亡。毕竟，在桌面浏览器里，你从未见过浏览器应用会在用户按 Windows 的启动按钮时做某些特别

的事情。

Brian （PhoneGap）：如果设备支持触控事件，那么针对该设备的 PhoneGap 也支持。触控体验绝对非常重要，我们才刚刚开始摸到手势交互设计的门道。Palm 的 webOS 在这方面是个非常先进的平台，我们也期望能把很多这方面的手势支持带到其他平台。

然而，触控并不总是有意义。虚拟键盘就相当难触击，常用按键应该能随时可访问，同时又不侵占有限的屏幕大小。Palm、Blackberry、 Android 在这方面也都进行了相当好的探索。Blackberry 使用者常常期望类似上下文菜单的“Blackberry 按钮”。Android 的伙计们则想要菜单、搜索、回退和 Home 按钮。满足用户体验要比满足其他花里胡哨的兼容性重要得多。在对触摸屏的痴迷退热之后，Android 上的虚拟回退按钮只不过是个烦人的组件。

Christopher （iWebKit）：由于 iWebKit 复制了 iOS 界面的所有元素，因此它绝对是"可触控的"。它们能很好地适合你的手指，而不是让你必须去适应站点，这是使用这个框架的主要原因。其他多指手势并没有缺省内置，但开发者在读过 Apple 的在线指南后可以轻松地添加进来。在我看来，这些手势并不是真的不可或缺，因为目前的互联网只是分发诸如文本、图像或视频这类媒体的一种简单方式。简单的体验对用户来讲是最好的。

但是，随着 Chrome OS 和 Html 5 的到来，我认为我们很快就会看到新的、更像“应用”的网站，而且这类站点的一小部分会被触摸屏设备接纳。它们将和本地应用一样好，尽管技术上是可能的，但现在时机未到。市场还没有形成。

Chris （WebApp.Net）：很多时候，手势只是使用一个手指完成诸如 swipe-to-delete, pan up/down.....多触控手势通常用于缩放和旋转特性，此外就没有了。在 iPad 相册应用（它可以让你收集大量的图片）中的发现效果使用了跟 pinch 手势相同的行为（缩放），但无法重新伸缩图像。

WebApp.Net 目前还没有手势事件，但在未来会有完全可配置的手势及相关事件。手势对用户体验非常有益，提供了一种简单自然操作 iPhone 用户界面的方法。只要手势得到了很好地设计，它们就会非常高效，并且提供非常好的生产力。

Christoph （mootools-mobile）： 尽管这是对的，但并不是很多移动浏览器都确实支持触控事件。目前只有移动 Safari 和 Android 浏览器支持触控事件，即便是后者也还不支持多触控（最近的模型终于开始支持它们了）。这意味着 mootools-mobile 主要针对这些设备，直到其他设备上的浏览器提供这样的特性。mootools- mobile 提供了像 swipe、pinch 和 touchhold 这样富交互的自定义事件，马上还会有更多！此外，还为所有点击事件提

供了一种自动的替换机制，以克服 iOS 上的触控延时。这让移动 Web 应用响应更迅速——简单但高效！

InfoQ：在遇到 HTML 和 JavaScript 组件时，客户端的功能测试尤其困难，特别是还涉及多个目标平台的时候。开发者该如何应对这个问题？

Michael （Sencha Touch）：我认为现在还没有真正好的答案。传统的功能测试不足以应对移动平台：仅仅因为某些测试正确并不意味着它的速度就满足使用要求了。所有平台都有模拟器，但是同样存在着限制。iOS 模拟器只能工作在 Mac 上；Blackberry Torch 模拟器只能在 Windows 运行。iOS 模拟器可能在复制设备体验方面做得最好，但是你需要在每个设备上都进行测试，以确保你用到的所有东西都有合理的性能。在 Blackberry Torch 上有很多 Web 特性，如好看的 CheckBox，但一旦你要想做点复杂的事情就会有效率问题。Android 有时会具有挑战性，因为电话 OEM 厂商加入的某些特性可能会让你的应用出问题——相比起 Android Web 应用来讲，原生 Android 应用会更糟。每种设备都有一些特殊的东西，应用开发者必须依赖它才能进行测试。要想保证一个特别的用户体验，真的没有其他办法。

Brian （PhoneGap）：我们使用由 John Resig 开发的优秀的 qUnit 库对我们的 JavaScript 库尽可能进行单元测试。它工作在所有的 PhoneGap 平台上。不幸的是，对于手势界面无法进行可靠的自动化测试，像 CSS 动画/变形/过渡这类主观性更强的体验，自动化测试尤其困难。对应用有个好‘感觉’是它成功的关键，测试它的最佳途径就是把它构建到一个设备中去，然后不断把玩。幸运的是，由于设备的性能特征和实际可用屏幕的大小，大部分移动应用关注于很好地实现一个简单功能，完全不同于我们在桌面应用中已经习以为常的单块应用。因此，这种手工测试的方式实际上并不太难或耗费资源。

Christopher（iWebKit）:就 iWebKit 来讲，这实际不是太大的问题。它可以在任何 WebKit 的浏览器上做到开箱即用，这指的是：iOS、blackberry os6、google android，基本上是除了 Windows Mobile 之外的所有主要智能手机平台。一旦 Windows Phone 7（或 8）内置了 css3 和 html5 支持，对于像 iWebKit 这样的现有框架会需要做一些调整，但理论上它是可以工作的。现在唯一的限制是，对于 iWebKit 来讲，它使用的是 iPhone 布局，但给框架换个皮肤很容易，因此，你可以给它提供一个适合所有平台的自定义外观。

Chris （WebApp.Net）：在进行 iPhone、Android 或 webOS 开发时，开发者可以方便地依赖所提供的模拟器，但它们的调试选项相当有限。作为一种最后的手段，他们可以使用经典的基于 WebKit 的浏览器，除了最新的 WebKit Web 检查器，还可以其他搭配提供的强大工具。当然，桌面版通常提供了比其移动版更多的 API，后者并没有实现所有相同

的特性。

大多数时候，这些特性是针对非常先进的用途的，这算不上是个问题，但我记得，在未实现 CSS 灰度的 webOS 上工作的时候，我不得不依赖画布来渲染灰度。我认为总有办法绕过兼容性问题。

如今，大多数浏览器都实现了强大的调试和性能监视工具。对于 Ajax 应用来讲，这些工具更有用，但是开发者必须尽量复现目标环境，以免在部署时被跨域问题搞得焦头烂额。实际上，跨域限制对于 file:// URL 模式来讲并不存在，我知道有时（经常）前端开发者将他们的代码弄成这样子，假如要做的事情仅仅是直接从 JavaScript 渲染 JSON 内容的话。当然，JSONP 对于绕开跨域限制很有帮助，尤其是在服务器端代理无法使用的情况下。

因此，这一问题的最短答案是：使用所提供的开发者工具:）

Christoph （mootools-mobile）：前面已经提到，应该尽量最小化手工测试的需要。Arian Stolwijk，MooTools 团队的成员之一，和我最近重写了 MooTools Spec 引擎，它是我们的单元测试库。它是独立的工具，基于 Jasmine，可用于任何项目。它可以在浏览器、node.js 中，以及通过 JSTestDriver（正在做）运行代码。你可以在 GitHub 上找到它：http://github.com/mootools/mootools-runner——它甚至可以让你模拟鼠标和键盘事件，通过一个叫做 Syn.js 的优秀库。更多信息和亲手尝试，可以参见这个说明：http://github.com/mootools/mootools-core-specs。

基本上，如果把 DOM 交互从业务逻辑中解耦出来，你就已经可以自动化测试应用的大部分代码了（可能甚至只是通过 node.js 运行它）。

InfoQ：自从早期的 WML 和只接受 XHTML 的移动浏览器出现，已经过了不少时间。这段时间以来，你认为哪些事情已经改变了？对于我们已经有了更强大的设备，但需要更多应用的今天，我们有没有学到点教训，可供我们参考？

Michael （Sencha Touch）：发生的最大事情就是摩尔定律继续地快速推进，携其余威继续丰富处理器的能力。当今的移动设备现在看起来就像是 2000 年左右的桌面，这意味着它第一次可以支持丰富的应用。发生的另一件事是 HTML5 处理给我们提供了基于标准的 Web 技术，这对一般开发人员来讲更实际、有用和容易理解。每个人都希望浏览器能变回 90 年底所有事情都偏离轨道之前的那个样子。

我们认为第一个教训是开发者需要评估他们是否应该支持无法提供良好应用体验的旧设备。想当初，WML/WAP 应用就很难使用，功能也很有限。只是创建一堆快捷、整洁和直观的用户界面还不够，你需要创造一种参与体验。我们内部有一份审视新手机用的

最简手机规范。当今的 Android 2、iOS 和 RIM OS6 都达标了，我们期望 Nokia 手机也可以很快出线。我们从来没有指望过让一个 Sencha 应用运行在 Motoroal 的 Razr 手机上。

我们的另一个想法则是不要过多担心利用操作系统特定的功能。肯定有一撮移动开发者会有意地使用每个移动平台的原生部件或控件——即便他们开发的是 Web 应用。但我们认为，Web 自己本身就是一个平台。桌面应用的外观在 Mac、Windows 和 Linux 上没什么不同，那只要是它们都有相同的输入机制可以利用的话，移动应用为什么要在 Android 和 iPhone 上显得不一样呢？

Brian （PhoneGap）： 设备 API 是我们在现代 Web 中看到的最大、最有趣的变化。每个人都在谈论 HTML5，一旦 Web 成为访问设备传感器（如照相机）和设备数据（如相片）的首选平台，革新就会发生。今天，你可以使用 PhoneGap 书写提供设备 API 的 Web 应用。

不幸的是，桌面编程中已经学到的教训无法很好的转化到 Web 的小屏幕和语言（即 JavaScript）上。许多开发者把大型系统的设计哲学（这里为大家都钟爱的“四人帮”的设计模式做下广告）应用到非常简短、函数式和动态的 JavaScript 语言上。使 JS 成为一种 Java 或 C 的变体，在使代码内存(这对移动设备来讲是必须考虑的事情)膨胀的同时，损害了 JavaScript 最好的特性。革新的空间很大；那些敢于使用*现有*工具而不是使用*以前*工具的开发者正在创造未来的 Web。

Christopher （iWebKit）：严格的讲，“移动浏览器”根本就不存在。现在唯一的区别就是屏幕大小和使用触摸屏输入。引擎跟桌面浏览器完全一样。我们是否学到了什么教训？很难讲。当时的手机非常慢，用户体验也很糟糕。硬件不足以有效地运行大而重的网站，直到 Apple 决定重新设计“移动互联网”，给 iPhone 提供一个大屏幕，一种简单的导航方式，以及最重要的运行“桌面”浏览器的能力。

因此，我说我们没有学到任何东西是指，我们没有改进当时的移动互联网，而是抛弃了它，转而进入了一个全新的领域。当今可用的这些应用就是明证。它不是之前相同的“东西”。手机真正的概念已经消失了，现在我们拥有的是智能机。

Chris （WebApp.Net）：WAP 本身是一个非常有趣的标准，具有很好的脚本编程功能，将网络带入了移动世界。但问题在于大多数规范由于设备的限制没有在移动浏览器里实现，同时没有讨论仅限于几行文本的小屏幕。同时，WML 对 XML 的依赖非常强，单单一个错误就能让整个页面无法访问。假若不是所有浏览器都支持相同的特性，那么使用一个简单的<b>都会导致错误。这迫使开发者为几乎每个不同的设备都创建一个版本。

WAP 2.0 朝着用户更友好的移动 Web 又迈出了一步。这个新版本使用具有 CSS 能力的 XHTML 子集（移动概要，Mobile Profile），但实现又一次很不均衡，屏幕分辨率的巨大差异同样也没有减轻开发者的工作。

如今，大多数智能机都有不错的大屏幕，同时包含能够正确展示桌面网页的高级浏览器。制造商似乎明白了移动 Web 的挑战，在他们的移动浏览器上投入了大量时间。更好的例子是 iOS 4.x 上的移动 Safari。这个浏览器包含了大量难以克服的特性。你也可以发现某些只在每日构建的 WebKit 版本上可用的特性。至于 iOS 的新版本，它带来了几乎真正的多任务，可以使用 Web Worker，同时非常不错而且几乎完全实现了 HTML5 API。

应用开发的兴趣现在似乎集中在本地开发，但 Web 应用正越来越接近底层 API，如地理位置 API、触控/手势事件或非常有趣的文件 API，虽然仍处于草案阶段，但已经完全在移动 Safari 中实现了。更别说 webOS 本来就是只基于 Web 的。在我看来，Web 应用的开发更快，而且更容易发布和维护。当然，由于设备限制你很难用 JavaScript 和 Canvas 或 SVG 创建一个 3D 游戏，但大多数应用可以用 HTML/JS/CSS 完成，webOS 已经很好展示了这一点。此外，Google 将大部分它的桌面 Web 应用都发布了移动版，越来越多的编辑器也踏上了这条道路。

很多本机应用都使用 HTTP Feed 给应用提供内容，大部分时候只用到了少许操作系统特性。所有这些应用都可以轻易地使用 HTML/CSS 展示，因此本机和 Web 的差距并不是非常大。

Christoph （mootools-mobile）： 老实讲，直到最近，我还没有做过任何移动应用的开发。我想，对大多数人来说有件事情很明显，那就是世界已经跟早些时候不同了。人们最终发现了移动 Web 应用的潜力。只开发一个 Web 应用而不是多个本机应用会节省大量的资源。鉴于此时桌面上的革新正在浏览器中发生，而不是在平台本身，我认为移动设备没有理由会与众不同。Web 是一种通用平台。

我们目前最大的问题是，移动 Web 开发就像是在一块小区域内玩有 99 个地雷的挖雷游戏，而不像是桌面浏览器中在一个大区域内玩同样规模的游戏。我认为这是非常恰当的类比。我们几乎没有移动浏览器调试工具，也几乎没有浏览器特性的文档，不同平台导致的问题和差异程度会让开发者发疯。我们才开始，还需要些时间。

InfoQ：你认为当前 Web 应用开发实践在多大程度上可以为移动 Web 应用所用？

Michael （Sencha Touch）：主要差异在于大多数移动应用都很小，因此今天的移动开发所需的组织规模远小于桌面开发。对于消费类的 iPhone 应用来讲，用户会话平均时间是 4 到 5 分钟。我们会猜测如果你排斥 Email，你会想在协同移动应用上得到类似结果。在对我们的开发者群进行的一次调查中显示，屏幕个数为 10-20 的设计大致占 80%。手机屏幕真是太小了，无法完成重量级的工作。

我认为，随着我们向平板计算前进，这会发生巨大变化，你将拥有更复杂的应用，而且用户会话的时间也更长，开发过程和团队将更像传统的桌面开发。

Brian （PhoneGap）： 所有专业的编程实践都可以很好的发生作用。把代码放进版本控制系统。架设自动化构建。书写单元测试。没有采用这三项实践的项目不见得会失败，但它们肯定不会有可以预测的发布时间或稳定的质量改进。

大多数 Web 应用实践都围绕着好的语义标签，清晰的 css 和优雅的解决跨浏览器问题的 DOM 库。在移动世界，我们有设备要求我们挑战这些实践。好的标签，尽管是个好主意，但增加了大量标记，导致延迟。深度嵌套的元素增加了渲染复杂度，让 CSS 查找变慢。基本上，要是你以性能的名义少做点事，你可能是对的。你可能没得选。我们都喜欢在我们的 PhoneGap 项目里使用 jQuery，但由于这个库实在是太大了，导致初始化时间非常长。我们最终完成了针对移动设备的自己的库，叫做 XUI，来解决这个问题。我们已经在所有 Nitobi 的 PhoneGap 项目中使用它有两年了。作为支撑 PhoneGap 平台的微 DOM 库，它工作得非常好，但它显然没有提供插件框架。

如果你只是面向最新的 iPhone，而且你想画出 GUI，那 jqTouch 是一个不错的选择。如果你要构建 iPad 应用，想要 iPad 风格的 GUI 那么 Sencha Touch 绝对是你的最佳之选。这些选择方案中的任何一种结合 PhoneGap 都将让你不必亲自动手去做本地应用，说的白一点，就是 5 分钟。没有一种本地平台给予了开发者这种生产力，放手让他们能够在多个平台间移植。但这个故事的真正意思是要理解你的用户，然后构建他们期望的体验。

每个移动项目都应从受众研究开始。许多接触 PhoneGap 的人仅仅是出于对 iPhone 感兴趣，但一旦我们去深入了解他们的受众，我们几乎总是发现他们大多数的用户都有 Blackberry。交互设计师比以往更重要，并且他们的职责是确保他们的研究不仅仅是从 Smashing 杂志上下载 iPhone 模板。他们需要知道所有移动设备的维度和象素密度。他们需要知道每种平台的常见 UI 范式。他们需要理解手机不仅仅意味着电话，而且也不只是意味着触控界面。没有一种平台要比另一种更“正确”。成功的移动应用将适应每种平台，不错，这意味着开发者要进行条件构建/编码。让其用户感到愉悦的超级成功的应用和平台间特性公共差异最小的平庸应用之间的区别可能就是一种伟大的平台体

验。你打算构建哪一种？

移动设备上的富客户端正展现出一种与桌面完全不同的新的令人着迷的问题空间。例如，数据复制和同步就是一个大问题。CouchDB 的伙计们正在这上面工作，我们也投入到了其中。这是一个超级难的问题。要是你的数据库有上 T 的数据，客户端只能保存 20M，会怎样？某种分页的分片？要是客户端现在是离线的呢？这其中充满了陷阱。安全性和隐私则开启了另一个满是蠕虫的盒子。

Christopher （iWebKit）：当前只有寥寥几个非常特殊的工具，如 iWebKit 或其他框架可以使用，但几个月之后，情况会发生变化。例如，你会看到大量更加重量级的 JavaScript 库内置触控手势。我怀疑明年移动和常规开发工具将会合并。唯一的区别就是屏幕尺寸，但这不是问题。

Chris （WebApp.Net）：移动 Web 应用需要针对用户输入有效快速地装入和反应。在移动情形下，网络质量是决定性的。这就是为何你必须关心你手头在做的事情的原因。Ajax 是减少不必要流量的关键，因为你只需要加载请求的内容，而不用重新加载整个页面。某些框架密集地使用 Ajax，这是其优点，但是这往往造就非常巨大的操作多种特性（而且大部分根本就从未被用过）的 JS 代码，使得加载应用的时间变得很长，至少头次加载如此，让使用者感到沮丧。这些大文件可以利用离线 Web 应用特性或本地存储 API 轻易地被缓存起来，支持 HTTP 压缩虽然可能会有所帮助，但是解决之道应该依赖插件，以便开发者可以选择使用和抽取哪些文件，打包进一个文件以改善加载时间。例如，jQuery 非常有趣，但是重新了发明轮子，很多流行的移动操作系统给浏览器提供了令人印象深刻的对于 W3C 和 WHATWG 标准的支持，这使得大多数 jQuery 特性没有什么用。同样 Apple 的 iAd 框架令人震惊的巨大（超过 100k）。这似乎是因为框架里使用了 MVC 模式，这迫使代码非常结构化。

Christoph （mootools-mobile）：我认为 Web 应用和移动 Web 应用二者非常相似。只是它们要实现的目标不同罢了。普通桌面 Web 应用可能把焦点放在传统功能上，而移动应用可能使用设备提供的关于当前它所处环境的任何信息。关于跟当前用户所处位置相关的任何环境。地理定位就是个绝佳的例子。移动设备变得对它们的环境越来越敏感，同时我们也会有 API 来访问硬件必须提供的任何东西。

我个人只要看到新事物就会想去找一种它的使用案例。最近我对本地存储、地理位置和 history.pushState 的兴趣渐增（参见 http://github.com/cpojer/mootools-history ）。

InfoQ：你的项目在利用 HTML 5 API 吗？你认为 HTML 5 的哪个特性对于移动设备最有价值？

你能给我们举一些例子吗？

Michael （Sencha Touch）：我们启动了所有跟 HTML5 有关的有趣技术。地理位置绝对是移动 HTML5 API 的头名。地理感知对移动设备的意义重大，原因在于它可以让你持续简化应用。例如，http://www.geocongress.us 是个告诉你国会代表团最近完成了哪些事情的 Sencha 应用——引进的法案和进行的投票。我们不需要向使用者询问他们想跟踪哪些国会代表和参议员，我们只要查看位置 AD 就能立刻显示跟这个使用者有关的代表团。它简化了界面，立刻提供了相关信息。

排在地理位置后面的是离线应用功能，包括缓存 manifest 和本地存储，因为移动应用在网络不可用时也需要工作。Touchsolitaire.mobi/是一个可以将你的游戏状态保存到本地存储的 Sencha 应用，这样你就可以离线使用了。

除此之外，我们还在主题中大量使用了 CSS3 的功能，包括过渡、动画和造型。我们认为 CSS3 动画广告对移动领域的广告业有重大影响，因为有如此多的手机无法用 Flash。我们甚至完成了一个它们的演示，地址在：http://dev.sencha.com/deploy/css3-ads/ 。

Brian （PhoneGap）： PhoneGap 扩展了给定平台上可用的本地浏览器的所有功能，在很大程度上，这意味着我们是基于 WebKit 的，它绝对是 HTML5 的典范。PhoneGap 工程可以利用在 WebKit 上实现的*全部*HTML5 api，但是，一如既往，这同样伴随着一些警告。比如，CSS 过渡/变形/动画和画布元素仅在 iPhone 和 webOS 上有硬件加速。未来，我相信画布将对创建丰富和高性能的界面至关重要，但我们现在离那儿还有相当的距离。

另一个重要特性主要是围绕离线存储的。我个人非常高兴 WebKit 选择了 SQLite，可并不是所有的移动设备都支持它，因此我们写了一个叫 Lawnchair 的库来隐藏内部的存储。对于简单地保存客户端的 JSON 文档，它工作真的非常好。PhoneGap 目前的另一个目标是尽快完成一个一致的 FileReader/FileWriter 实现。

Web Worker 是移动的另一个让人激动的领域，因为它可以极大改进应用的性能。最后，我们*真的*对可能将 Web Socket 用于实时应用感到激动（我们已经给 iPhone 和 Android 提供了 XMPP 支持）。想想：地理位置、实时消息传输、联系人 API、照相存储.....有多少很酷的应用会因此孕育而生！

Christopher （iWebKit）：iWebKit 完全是基于 css3 的，它是 HTML5 引入的最重要的新特性。这样做的主要目的是为了让包尽可能的轻量级。只需使用非常少的图片，其余的由 iPhone 自己产生。利用 css3，我还可以创建圆角边、阴影、自动伸缩的背景图以及自

定义的表单元素。它还有一些超炫的特性，如根据元素在页面上的位置来检测某个元素（例如奇偶元素）以及根据屏幕宽度自定义风格。

Chris （WebApp.Net）： WebApp.Net 目前还没有使用任何 HTML5 API。不管怎样，这些 API 已经是 API 了，没有必要去封装，并且可能更像被作为插件而不是框架的一部分去使用，如视频播放器。一些 HTML5 API 非常特殊，如 Selection API 就跟文本编辑有关。至于跨域的限制，跨文档进行消息传输的 API 非常有趣的允许跨源（cross-origin）通信，但是开发者需要非常小心的使用它，在任何时候确保安全性。它已经在 iOS 3.x 中实现了，在基本的表单中只允许基于字符串的通信，在 iOS 4.x 中将会改进，实现全部的 API 规范。但是最有趣的 API 应该是画布 API，它可以实现非常高级的用途（利用 WebKit 特有的 -webkit- canvas（ ） CSS 功能）。实际上，混合画布和视频在移动 Safari 上还不支持，会导致安全错误，希望它能早一点被修复。对于那些不属于 HTML5 规范的部分，地理位置 API 和所有的存储 API 允许更大的灵活性，并且可以涉足那些原来仅限于本地应用的领域。

InfoQ：使用 HTML/JavaScript 技术栈对于移动设备有重大意义，你认为像 Flash 这样基于插件的富互联网应用技术的前景如何？它们是互补还是会最终被淘汰？

Michael （Sencha Touch）:我认为说它们过时还言之过早，但是就跨设备覆盖来说，Flash 和 Silverlight 简单说都存在问题。Flash 可能在 Android 上找到归宿，尤其是由于 Android 依旧缺乏对 SVG（Web 矢量图形标准）的支持。要是 Silverlight 出现在 iPhone 或 Android 上我会很惊讶。我们期望 HTML5 以及基于 HTML5 的框架能尽快开发出真正与 Flash 和 Silverlight 目前提供功能相当的能力，如设备访问和内容保护。

Brian （PhoneGap）： Flash 跟 PhoneGap 会一起工作得很好，实际上，你可以随时随地地使用它。现在就已经有一个桥接 Flash/PhoneGap 的示例存在了。我认为开发者可能会把焦点过多的放在“哪些已死”和其他趋势上。Flash 是一个工具，由此，只要合适，我们就应该使用它。记住*所有技术*都会过时是有好处的。

Christopher （iWebKit）：出于 SEO 的原因，我反对像 flash 这样的技术，假如用它来交付内容的话。在 iOS 中不包含 flash 的决定是 apple 的选择，但这并不能说它就是个差劲的技术。对于特殊的用途，如站点的动态图形部分，flash 或 silverlight 是目前实现很酷外观的最简单方式。HTML5 也能够做到，但未必就更强或是更易用。同样，对于视频来讲，flash 是保护你媒体的唯一方法。就目前来说，flash 不会消亡，但是几年后它的市场可能会萎缩。最终它会像其他任何技术一样消失，可能到时会使用 html6？

Chris （WebApp.Net）: Flash 是个好东西，但是我们必须承认它让很多网站和计算机变得很慢，主要是因为如今的很多广告都在使用这项技术，而且还因为一些开发者并没有真正掌握 Action Script 和某些 Flash 特性的影响，导致几乎全部的 CPU 使用率都在执行他们的代码！不管怎样，Web 似乎越来越朝着无插件浏览器的方向发展。最佳的例子就是 HTML5 视频和音频 API 的演变，以及主要视频网站如 YouTube 或 Vimeo 对这一特性的采纳。尽管画布和 SVG 很强大，但它们都需要手工编写几乎所有的代码，而这方面 Flash 则已经有一个功能齐全的创作工具。因此，暂时来讲，我认为它们是互补的，而且由于 Flash 被广泛的使用，它仍然会保留一段时间。但是软件厂商正在开发简化画布使用的解决方案或是渲染成 Flash，例如，或是作为画布和 SVG 的混合体。这主要是因为 Flash 不被 iPhone 和 iPad 支持，而且还因为插件已经被证明了具有某些不稳定性，而使用浏览器内置的功能则要热门得多。

Christoph （mootools-mobile）: 开放性决定了 Web。它需要不惜一切代价保持开放，这里没有专有软件的容身之地。这就是我想说的全部内容。

InfoQ:对于目前准备开发移动 Web 应用的团队，你有何建议？哪些应该是他们的主要关注点，有哪些常见的陷阱是他们需要小心的？

Michael （Sencha Touch）: 对于用户体验要非常小心——商业应用更应如此。不要试图给应用包装太多功能。保持界面的简单以及屏幕个数可管理。及早进行非正式的性能测试。就富体验来讲，对于真的“必须支持的”移动设备的数量要现实点，但是对于不支持富体验的设备要能退回到使用普通的 HTML。

不要试图去摆弄自己的 UI 组件和低层次抽象，尤其是在已经有像 Sencha 和 jQTouch 这样的工具来帮助你完成这些工作的时候，你这种做法就是重新发明轮子。

此外，专注于移动 Web 应用开发，并不意味着你就不能利用本地应用的分发渠道。Phonegap 在允许你把 Web 应用打包成本地应用进入 Android 市场或 Apple App Store 方面做了很好的工作。

Brian （PhoneGap）: 我们认为你能做得最好的就是去准备移动 Web。从一个移动 Web 开始，只把 PhoneGap 视为一个积极的增强技术，用于应用商店的分发和早期介入设备 API。浏览器正迎来这种东西。（而且随便说一句，还没有其他跨平台的移动解决方案适合这类风格的开发）。想想你的用户是谁，不要让你的个人设备喜好影响你。

想想临界速度和大致的连通性。不是每个人都钟爱 iPhone 4，事实上，你的很大一部分

潜在用户可能是在一个很差网络上的 Blackberry 4.6。

Christopher （iWebKit）：除了用来构建 Web 站点的常用建议，还需要记住 3 件事。首先是绝大多数目前的智能手机用起来都很直观，而且提供了非常棒的用户体验。这也必须是你应用中的一部分，通过采用你自己的屏幕大小和设备的触控控制去体验。第二件事就是不要把用户局限于站点的最小版本。如今的用户需要在他们的手机上完全体验你的站点。最后一个就是要记住手机是移动的。它不会跟计算机一样快，而且网络速度也有限制。让你的 Web 应用变得轻快，在 3G 和临界情况下加载都不是太耗处理器的资源。

Chris （WebApp.Net）：我的建议是深刻了解 HTML5 提供的新能力以及新的 W3C 规范，但也要留意 ECMAScript 规范，好很好的理解 JavaScript。大多数 API 已经在移动浏览器中实现了，将会在他们的移动应用开发过程中发挥很大作用。开发者必须了解他使用的平台以及常见的用户使用习惯和行为。例如，使用操作系统定义的导航区域，使用用户为了得到所需动作而期望的手势。不要重新发明轮子，但要有创意和革新。你不需要模拟操作系统的界面，这主要是因为你的 Web 应用可以在不同的平台上浏览。

看看现有的框架，即使你不打算使用它们，了解它们如何实现跟你项目有关的特殊特性。使用 Ajax 时要考虑搜索引擎和用户书签。取决于你的 Web 站点，用户可能想要用书签保存链接——假设文章是通过 Ajax 加载的——而后可能会失望的看到无法访问你的内容。讨论搜索引擎会更兼容和高效的使用 HTML5 提供的新媒体特性。

但是最重要的一点在于：总是考虑用户。在设计 Web 应用时，用户体验必须是你考虑的核心。不要试图把所有的东西都塞到同一个页面，关注一个想法，分拆你的内容。记住，你不是在开发一个网站，而是在使用 Web 技术开发一个应用。行为和期望是不同的。Web 应用更简单，它就会越成功。

Christoph （mootools-mobile）：首先，要尽可能让你的手碰到真正的设备。模拟器并不能代替它。了解不同移动浏览器的区别。考虑使用大屏幕设备，因为这样测试要更容易。选一个目标平台，试着为老的移动浏览器提供一个单独的视图。对于高级移动浏览器，你可以（也应该）增加模拟本地应用的富交互。简单而强大的技巧：使用 css transitions 和 translate3d 来得到硬件加速。

InfoQ：你的项目未来计划是什么？

Michael （Sencha Touch）：HTML5 和 CSS3 已经提供大把的技术来让我们可以开始完全利用移动设备。存在着大量我们关注的运行时革新和工具支持领域。这里的挑战是选哪一个先开始解决！更多扩展的预定义 UI 组件、更好的图形化支持和富数据管理可能是

Sencha Touch 下一版的 3 个主题。

> **Brian （PhoneGap）**：我们目前主要的重点是让人们可以更容易的上手。我们的文档非常丰富，你可以在这里浏览 http://docs.phonegap.com ，而且围绕测试和构建 PhoneGap 项目有更好的工具支持。我们一直在增加新的平台，而且测试套件也变得越来越全面。这是个大项目，没有人能够完成所有的 事情，因此我们需要*你的*帮助来构建未来的 PhoneGap。加入 PhoneGap 邮件列表、IRC 频道或即便只是查看一下 Twitter 上的 @phonegap，如果想帮忙的话。可以从很多简单的 Bug 开始入手。

> **Christopher （iWebKit）**：我已经开始进行 iWebKit 的下一版更新。它的结构将会改变，而且会使用最新的 HTML5 元素好让用户编辑更简单，结构更容易理解，一眼就能看明白。此外，我打算植入一个全新的主题系统，通过隔离 CSS 文件，让改变主题变得更简单。最后的一点是它将缺省就兼容 iPad 和 iphone 4。你的站点完成后，它就能在任何 Apple 的设备上工作。

> **Chris （WebApp.Net）**：我正忙于写本关于在 iPhone 和 iPad 上开发 Web 应用的书。它应该在未来几月出版。因此，我真的没有时间去做 WebApp.Net 和它的路线图。但是我正在写的书让我创建一些有趣的特性来阐述某些章节，当然它会成为 WebApp.Net 的一部分。最重要的工作是翻新当前 WebApp.Net 的某些功能，好让它们更符合标准。接下来，我将把重心放在对 iPad 的支持上，但是这将要求对框架的核心做一些改动。

> **Christoph（mootools-mobile）**：我将对移动 Web 应用提供更有用的补充，我会在我的 GitHub 上发布更通用的插件（参见 http://cpojer.net ）。一旦 MooTools 团队发布 MooTools Core 1.3（很快，我保证！【译注：已经发布了。】）我将开始写点文章介绍如何使用我的代码。我做的所有事情都是针对 MooTools Core 1.3 的，所以耐心点，除非你想活在刀锋上。

你可以在 InfoQ 上找到更多关于**移动开发**的内容！

原文链接：http://www.infoq.com/cn/articles/mobile-web-development

相关内容：

- Google发布移动Web性能工具PCAPWebPerformanceAnalyzer
- 虚拟研讨会：HTML5 的新JavaScript框架、诺基亚就WindowsPhone 7 与微软结成联盟、
- 详细解析—SUP移动应用开发平台、开发者眼中的Android手机平台

开始报名啦！

Adobe在线课堂第五期

Adobe

随着AdobeFlash平台技术的不断发展及开发工具的不断更新，Flash 平台应用的开发变得越来越轻松和高效，这些应用也被越来越多的行业及企业广泛地采用。为达到帮助开发者理解Flash平台技和技术以及Flash应用的开发技巧的目的，特邀请来自Adobe的技术专家在线讲解Adobe平台技术。

董龙飞：Adobe平台技术推广经理

2005年加入 Adobe公司，任职企业解决方案销售工程师，是 Adobe 企业级产品 LiveCycle,Connect 等的技术专家。目前是 Adobe 平台技术传教士，致力于推广 Flash 平台技术在企业级解决方案中的应用。他相信 Flash 平台技术是企业应用展现层的最佳技术方案。董龙飞也是 《Flex 大师之路》的作者和多本 Flex 书籍的中文版译者。

课程：Flex Hero开发移动应用101

本讲座将介绍如何使用Flash Builder Burrito 4.5 （预览版），基于下一代代Flex SDK （代码名称：HERO ）开发移动设备应用。本讲座为入门讲座，您可以在 http://labs.adobe.com/technologies/flashbuilder_burrito/下载Burrito。我们将介绍下一代FlashBuilder4.5，以及使用Flex开发移动应用的基本知识，并基于Android平台开发一个示例应用。

本次课堂共五期 点击浏览相关课程

1月11日	李鹏
1月18日	马鉴
2月22日	黄竣
3月 1 日	董龙飞
3月 9 日	董龙飞

分享知识！
相聚Adobe在线课堂！

推荐文章 | Articles

Silverlight CoreCLR 结构浅析

作者 吴磊

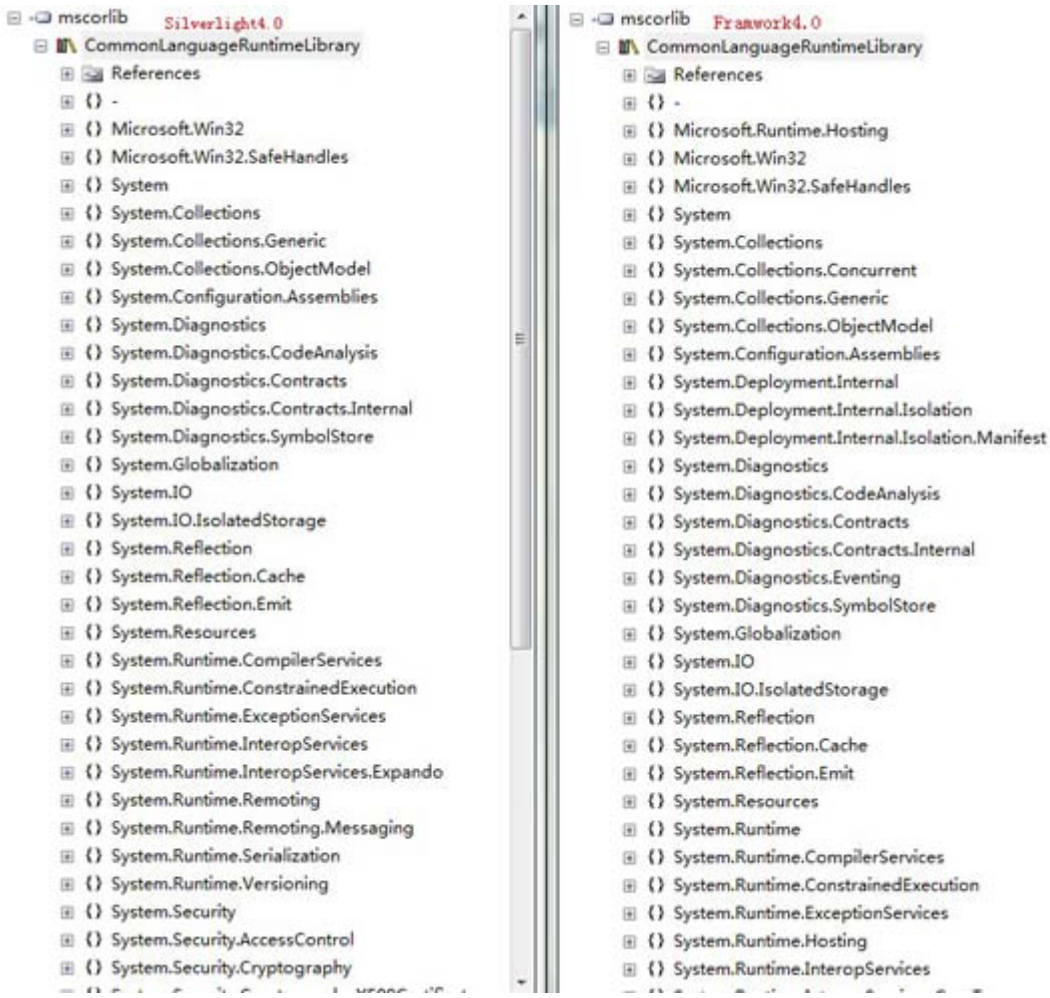
Silverlight 背后有强大的.Net 平台支持，自 Silverlight1.1 开始它就使用 CLR 运行时环境，而 Silverlight4 更是将运行时环境升级为最新的.Net4.0 版本。下面我们就来看看 Silverlight CoreCLR 里面究竟有什么？

我通过解压 Silverlight 安装程序了解其.Net 的身份与丰富的基础类库。下图是笔者解压 Silverlight4 安装程序后的文件列表截图：

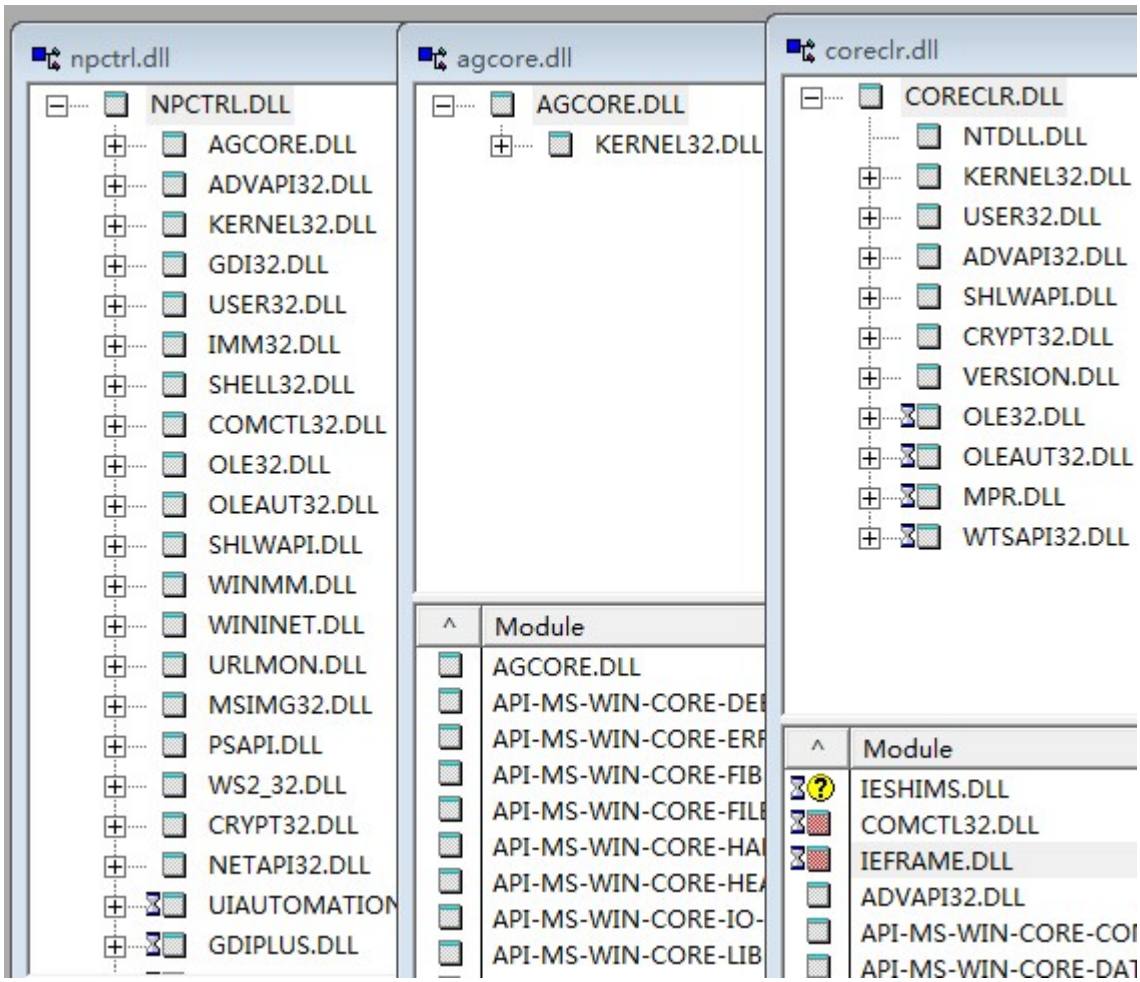
manifest.XML	2008/7/10 12:32	XML 文档	6 KB
agcp.exe	2010/12/4 0:23	应用程序	16 KB
coregen.exe	2010/12/4 0:23	应用程序	72 KB
silverlight.configuration.exe	2010/12/4 0:23	应用程序	342 KB
sllauncher.exe	2010/12/4 0:23	应用程序	478 KB
core.dll	2010/12/4 0:23	应用程序扩展	5,822 KB
coreclr.dll	2010/12/4 0:23	应用程序扩展	3,442 KB
ctrl.dll	2010/12/4 0:23	应用程序扩展	1,001 KB
ctrl.Resource.dll	2010/12/4 0:23	应用程序扩展	749 KB
microsoftvisualbasic.dll	2010/12/4 0:23	应用程序扩展	254 KB
mscorlib.dll	2010/12/4 0:23	应用程序扩展	1,558 KB
mscorrc.dll	2010/12/4 0:23	应用程序扩展	10 KB
silverlight.configuratio.nui.dll	2010/12/4 0:23	应用程序扩展	736 KB
SLMSPRBootstrap.dll	2010/12/4 0:23	应用程序扩展	417 KB
system.dll	2010/12/4 0:23	应用程序扩展	234 KB
system.runtime.serialization.dll	2010/12/4 0:23	应用程序扩展	410 KB
system.service.model.dll	2010/12/4 0:23	应用程序扩展	514 KB
system.service.model.web.dll	2010/12/4 0:23	应用程序扩展	78 KB
system.windows.browser.dll	2010/12/4 0:23	应用程序扩展	146 KB
System.Windows.dll	2010/12/4 0:23	应用程序扩展	1,442 KB
systemcore.dll	2010/12/4 0:23	应用程序扩展	530 KB
systemnet.dll	2010/12/4 0:23	应用程序扩展	226 KB
systemxml.dll	2010/12/4 0:23	应用程序扩展	318 KB
XAPAuthenticodeSIP.dll	2010/12/4 0:23	应用程序扩展	20 KB

其中 mainfest.XML 是程序集清单文件，而 sllauncher.exe 是 Silverlight 脱浏览器应用（OOB）的宿主启动程序，而其余动态链接库文件就是 Silverlight4 的运行时环境。如果你安装了

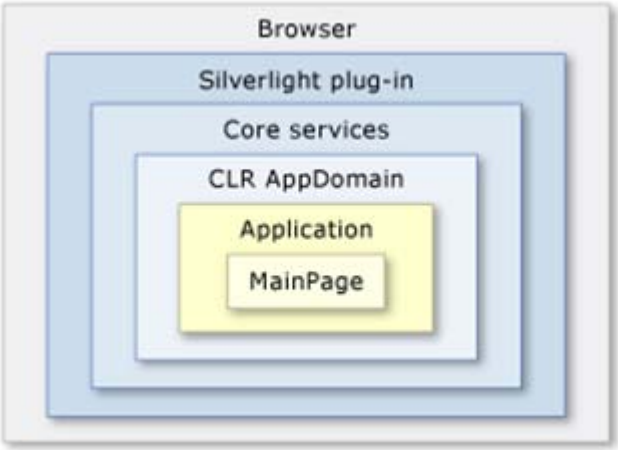
Silverlight 运行时环境 （当然还包括语言资源文件），也可以在 C:\Program Files\Microsoft Silverlight\下的对应版本文件夹发现它们。而有心的读者或许会意识到为什么 Silverlight 不使用与.Net Framework 一致的程序集文件，这一问题笔者在《Silverlight 之轻》中已经给出了答案，现在我们就来看看 C:\Windows \Microsoft.NET\Framework 下对应主版本下的程序集文件是否存在不同，以 CLR 核心程序集文件 mscorlib.dll 为例，在 Framework4.0 下为 5075KB，而在 Silverlight4.0 下则只有 1558KB 大小。如果我们使用 Reflector 比较一下会发 现，Silverlight 类库的精简情况：



Silverlight 安装文件夹下 npctrl.dll 为浏览器插件，为宿主浏览器提供了 Silverlight 的插件式集成，同时也避免了 JavaScript 等对不同浏览器的兼容问题。如果使用 depends 工具查看 Silverlight 安装文件夹下的 npctrl.dll、 coreclr.dll、agcore.dll 的话，我们可以发现它们没有通过.Net 总入口 MSCOREE.DLL 程序集启动，它们并不是 Silverlight 的.Net 基础类库：



如上图所示，这些动态链接库并不是托管程序，而是引用 Windows 的内核与 API 的底层接口，是负责与操作系统交互的核心运行文件。现在我们来看看 MSDN 关于 Silverlight 运行环境的图例：



其实 npctrl.dll 就是 Silverlight plug-in，这一点可以从安装时的注册表监控中找到答案：

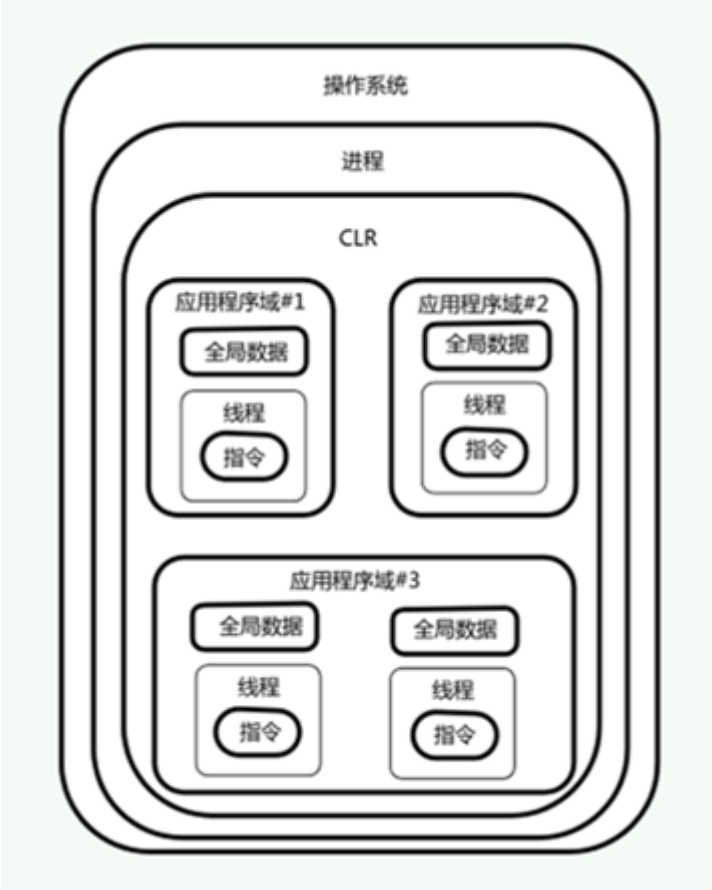
```
HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\InternetExplorer\ActiveX
Compatibility\{32C73088-76AE-40F7-AC40-81F62CB2C1DA}
```

```
HKEY_LOCAL_MACHINE\SOFTWARE\Classes\CLSID\{DFEAF541-F3E1-4c24-ACAC-99C30715
084A}
HKEY_LOCAL_MACHINE\SOFTWARE\Classes\CLSID\{DFEAF541-F3E1-4c24-ACAC-99C30715
084A}\InprocServer32
```

其键值是: C:\Program Files\Microsoft Silverlight\<版本号>\npctrl.dll

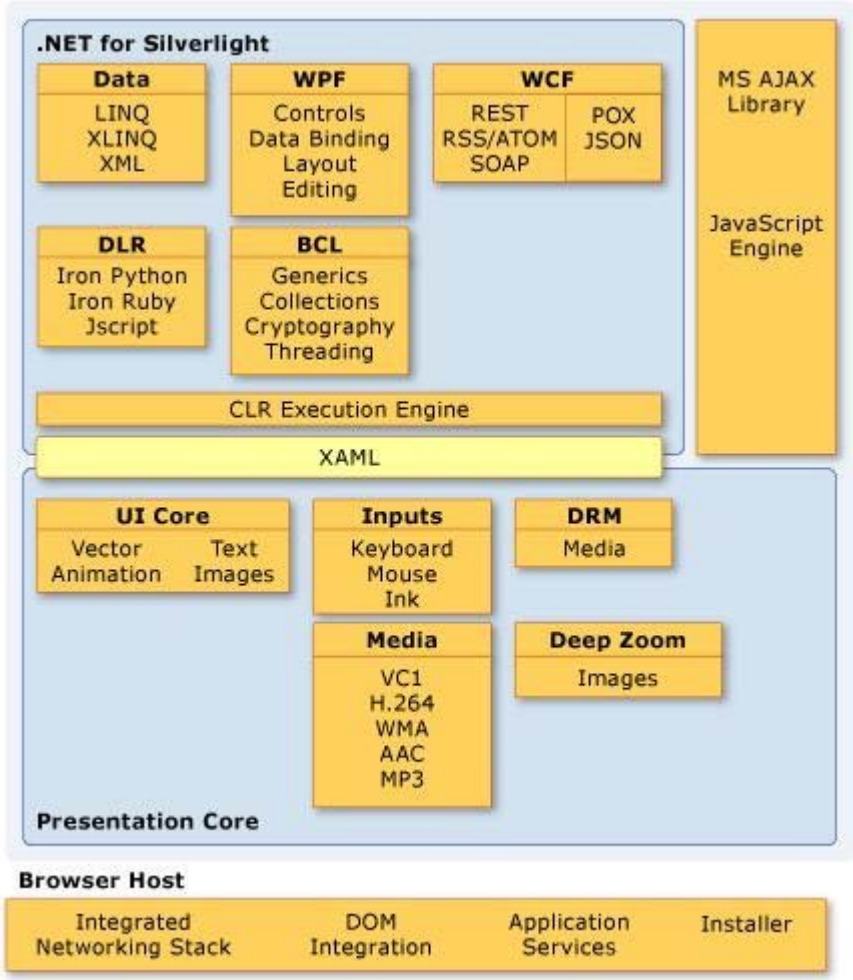
npctrl.dll 本身是个 ActiveX 控件与宿主浏览器交互，而 npctrl.dll 会调用 agcore.dll 来处理一些非托管的 API，它具有 NP_GetEntryPoints、NP_Initialize、DllGetClassObject、InstallOfflineApp、RemoveOfflineApp 等调用函数。npctrl.dll 会调用 agcore.dll 来进行绘制，以及 coreclr.dll 来创建 CLR，Silverlight 应用程序就是依赖于这个 CLR 中而被运行的。

agcore.dll 与 coreclr.dll 共同构成了图中的 Core Services ,agcore.dll 是 Silverlight 的核心展现部件 Core presentation framework 包括了 XAML parser, UI Core, Inputs, DRM(digital rights management), Media, Deep Zoom 等，多数是需直接与具体的操作系统 API 打交道的功能。coreclr.dll 是用来创建并执行 Silverlight 托管代码的总控程序 ,图中的 CLR App Domain 就是由 coreclr.dll 来创建，App Domain 应用程序域是介于 CLR 进程与线程之间的逻辑概念，在一个应用程序域中可以共享全局数据，如下图所示：



而 coreclr.dll 负责各 Silverlight 应用程序的内存管理（包括自动内存管理、垃圾回收等）、运

行引擎（包括 JIT 编译器、异常捕获管 理、安全管理、类型安全校验等）。按照 MSDN 的 Silverlight 整体架构图所示 agcore.dll 所负责的展现核心部件 presentation Core 与 coreclr.dll 所负责的 CLR 执行引擎(CLR Execution Engine)起到了 Silverlight 应用中极其重要的引擎作用。

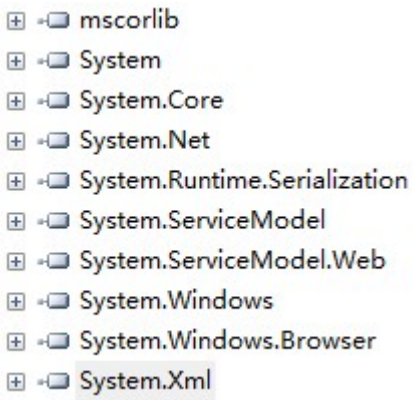


其余动态链接库作用如下：

- mscorrc.dll：运行时资源集
- mscordaccore.dll：外部数据访问支持库，包括对跨进程的数据访问
- Microsoft.VisualBasic.dll（托管程序集）：VB 运行库支持
- Silverlight.ConfigurationUI.dll：Silverlight 右键中的配置界面程序集
- mscordbi.dll（属于 Silverlight 开发运行时）：运行时环境 Debug 调试服务
- dbgshim.dll（属于 Silverlight 开发运行时）：多 CLR 调试助手

注：以上两个动态链接库文件是加载到 Visual Studio 中，作为 Silverlight 程序调试所用。

最后我们就来看看 mscorlib.dll 与 System.*.dll ,这两类托管程序集文件构成了 Silverlight 运行时的基础类库，通过 Reflector 我们可以了解到该基类库的全貌：



下面是对各个程序集文件的分析：

Mscorlib.dll：CLR 的核心基础类库，包含以下常用命名空间

命名空间	命名空间简介
System	包含基本数据结构和基类，运行时管理及委托类，如 Object 基类；String，Int，Double, DateTime，Array 等基础数据结构；Convert 类型转换类；Math 数学函数类；Random 随机数生成类；Activator 等用于通过创建实例的类；Type 类型声明类，用于反射；以及 GC 垃圾回收类等
System.Collections.Generic	包含定义泛型集合的接口和类，如 Dictionary，List 等
System.Collections.ObjectModel	包含可在可重用库的对象模型中用作集合的类，如 Collection<T>，ObservableCollection<T>，ReadOnlyCollection<T>
System.Globalization	包含定义区域性相关信息的类，这些信息包括语言、国家/地区、使用的日历、日期、货币和数字的格式模式以及字符串的排序顺序
System.IO	包含文件模式、文件访问、文件共享和类用于路径操作和操作流的类及枚举。如 Directory，File，StreamReader，StreamWriter 等类及 FileAccess 等枚举
System.IO.IsolatedStorage	包含用于管理独立存储的类，在 Silverlight 中，所有 I/O 操作都被限制为独立存储，不使用操作系统的文件系统，如 IsolatedStorageFile
System.Reflection	包含获取托管代码中程序集、类、属性、方法等的反射类，此类为运行时动态调用提供了可能，如 Assembly，PropertyInfo ，MethodInfo

	等
System.Reflection.Emit	包含允许编译器或工具发出元数据和 Microsoft 中间语言（MSIL）的类，该类可以向编译器顶端堆栈写入 IL 语言，可实现更加灵活的动态化支持，如 ILGenerator，OpCodes 等
System.Runtime.InteropServices	提供各种支持 COM 互操作及平台调用服务的 .NET Framework 基础结构
System.Runtime.Serialization	包含可用于将对象序列化和反序列化的类，如 DataContractSerializer，XmlObjectSerializer 等
System.Security	提供公共语言运行时安全系统的基础结构。包括 Cryptography 的加密类库，支持 AES、SHA1 及 SHA256 哈希函数等加密算法与 X.509 证书
System.Text	包含表示操作和格式化 String 的类，如 Encoder ，Decoder，StringBuilder
System.Threading	包含进行多线程编程的类，如 Timer，WaitHandle，Thread，ThreadPool 等

System.dll：主要是对 Mscorlib.dll 基类的扩展，包括 Uri 支持、组件模型、正则式等

命名空间	命名空间简介
System	包含 Uri 相关类，如 Uri，UriBuilder 等
System.Collections.Generic	对 Mscorlib.dll 的扩展，增加 LinkedList，Queue，Stack 等
System.ComponentModel	提供用于实现组件和控件运行时和设计时行为的类，包括用于 Xaml 绑定及执行异步操作的类，如 BackgroundWorker，AsyncOperation 等
System.Text.RegularExpressions	包括正则表达式引擎的类，如 Regex，Match 等

System.Core.dll：主要扩展委托、Linq 及动态语言等

命名空间	命名空间简介
System	包含委托方法类，如带返参的委托方法类 Func 与不带返参的委托方法类 Action，目前最多支持 16 个泛型参数；扩展了 Lazy<T>的延迟泛型对象。
System.Linq	提供支持使用语言集成查询（LINQ）进行查询的类和接口，如 Enumerable，Queryable

System.Linq.Expressions	包含 LINQ 的基础表达式树的类、接口和枚举，与.Net4.0 中的表达式树完全一致，提供申明式编程方式
System.Dynamic	提供支持动态语言运行时的类和接口，支持以 dynamic 声明的动态变量

System.Net.dll：负责网络通信的程序集

命名空间	命名空间简介
System.Net	提供多种网络协议的简单编程接口，如 WebRequest 和 WebResponse 类封装不同协议下异步通讯的差异，使开发者不必考虑各种不同协议的具体细节，而 HttpWebRequest 和 HttpWebResponse 是上述两类的的 HTTP 特定实现；WebClient 类提供了上下行数据的工具
System.Net.Sockets	提供 Socket 网络协议的托管实现，在 Windows 下是 Winsock 接口的托管实现，而在 Mac OS X 上 BSD UNIX Socket 的托管实现

System.ServiceModel.dll：负责 WCF 分布式通信的程序集

命名空间	命名空间简介
System.ServiceModel	提供 WCF 的编程接口，支持 WebService 调用等
System.ServiceModel.Channels	包含构造和修改客户端用于与服务进行通信的消息时所需的类型、枚举和接口、用于交换消息的通道的类型、用于构造这些通道的通道工厂，以及用于配置这些通道的绑定元素，如 BufferManager，BodyWriter，Message 等
System.ServiceModel.Syndication	包含构成 Silverlight 联合对象模型的所有类，如 RSS 2.0 来回序列化类、Atom 1.0 来回序列化类

System.ServiceModel.Web.dll：扩展了 JSON 序列化类

命名空间	命名空间简介
System.Runtime.Serialization.Json	用于将对象序列化为 JavaScript 对象表示法（JSON），并可用于从 JSON 反序列化对象，如 DataContractJsonSerializer 和 JsonReaderWriterFactory 类

System.Windows.dll：Silverlight 控件类库，与 Windows Presentation Foundation (WPF) 中存在的类相似

命名空间	命名空间简介
MS.Internal（对外不可见）	包含 Silverlight 内部控制组件与枚举，以内部密封类形式存在，如负责下载的 Downloader 类，负责错误的 Error 类以及负责 XAML 信息的 XamlMemberInfo 类等
System.Collections.ObjectModel	扩展了 mscorlib.dll 的集合类，使其可用于一般用于 UI 控件数据源绑定，如 ObservableCollection<T>等
System.Collections.Specialized	包含用于集合类中的事件处理的专用类，提供数据更改通知
System.ComponentModel	扩展了 System.dll 的组件和控件运行时和设计时行为的类，专门负责数据更改通知
System.IO.IsolatedStorage	扩展了 mscorlib.dll 的独立存储操作类，加入 IsolatedStorageSettings 类以提供一个在独立存储中存储键/值对的 Dictionary<TKey, TValue>
System.Runtime.InteropServices.Automation	提供可与已注册 COM 进行交互操作的类，如 AutomationFactory 类允许在 OOB 模式下创建 COM 组件，并与 Silverlight 程序实现互操作
System.Windows	包括了 Silverlight UI 控件及属性的基础类、结构及枚举，如程序总框架 Application 类，程序集部件 AssemblyPart 类，控件基类 DependencyObject、UIElement、FrameworkElement，样式类 Style 与 Setter，UI 状态管理类 VisualStateManager、VisualState、VisualStateGroup，事件路由类 RoutedEvent，消息控件 MessageBox 等
System.Windows.Automation	提供对 Silverlight UI 自动化客户端的支持
System.Windows.Controls	包括 Silverlight UI 基础控件，如 Canvas 、Button、TreeView 、DataGrid、WebBrowser 等近百种控件
System.Windows.Controls.Primitives	提供 Silverlight UI 基础控件的局部供开发者控制，如 ButtonBase、CalendarDayButton、ScrollBar 等
System.Windows.Data	包含用于将属性绑定到数据源的类，如 Binding 等
System.Windows.Documents	包含支持 Silverlight 中基本文档对象的类，如 TextElement、Italic、Bold、Run 等
System.Windows.Ink	提供用于手写笔交互的类，如 Stroke 等

System.Windows.Input	提供支持 Silverlight 客户端输入的类，如 Cursor、Keyboard、FocusManager、TouchPoint（多点触摸）、InputScope、MouseEventArgs 等
System.Windows.Interop	为 Silverlight 插件的属性提供托管代码公开的类，这些属性也存在于宿主浏览器的 HTML DOM 中，可以被 JavaScript 访问，如 SilverlightHost，Content 等
System.Windows.Markup	提供对 XAML 处理的类，包括 XAML 读取器 XamlReader 等
System.Windows.Media	提供集成富媒体的类，包括绘图、文本和音频/视频内容等，如 Brush、ImageBrush、VideoBrush、Geometry、AudioCaptureDevice 以及可视化树助手类 VisualTreeHelper 等
System.Windows.Media.Animation	提供动画支持的类，包括时间线、故事板和关键帧等，如 Storyboard，Timeline，SineEase（正弦缓动）等
System.Windows.Media.Effects	提供视觉效果的类，可用于将视觉效果应用到位图图像，如 BlurEffect，DropShadowEffect，PixelShader 等
System.Windows.Media.Imaging	提供图像处理的类，如 BitmapImage 等
System.Windows.Media.Media3D	提供 3D 的类，支持 Silverlight 中矩阵/透视演示,如 Matrix3D 等
System.Windows.Messaging	提供支持在多个基于 Silverlight 的应用程序之间创建本地消息传递通道的类，如 LocalMessageReceiver，LocalMessageSender 等
System.Windows.Navigation	提供支持导航的类，如 UriMapper，NavigationService 等
System.Windows.Printing	提供打印服务的类，如 PrintDocument 等
System.Windows.Resources	提供对通过 Silverlight 应用程序模型检索资源的支持类，如 StreamResourceInfo
System.Windows.Shapes	Silverlight 基本形状库，如 Line，Ellipse，Rectangle，同时还包括 Path 类，该类可以加载路径数据，以便通过对象模型或内联格式描述复合几何图形
System.Windows.Threading	提供支持 Silverlight 线程处理系统的类，但只包括 Dispatcher 方式的多线程类，如 Dispatcher 和 DispatcherTimer

System.Windows.Browser.dll：与宿主浏览器的交互程序集

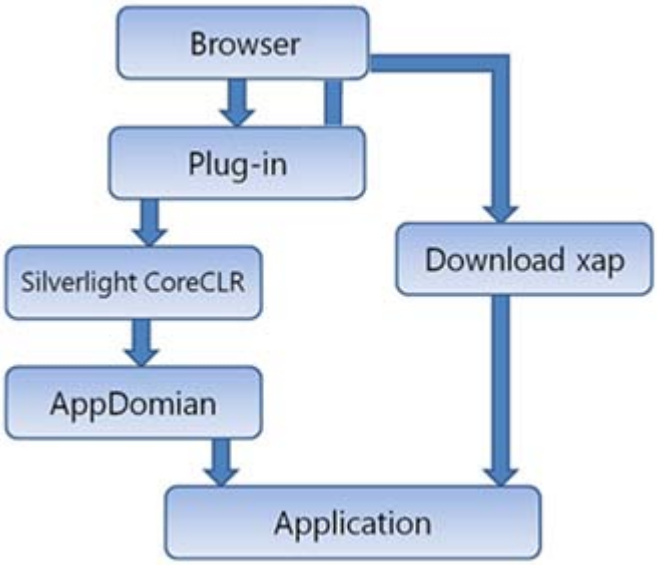
命名空间	命名空间简介
System.Windows.Browser	提供用于在基于 Silverlight 的应用程序中的托管代码与 JavaScript 之间实现交互的类，此功能也称为“HTML 桥”，如 BrowserInformation ，HtmlDocument，HtmlElement，ScriptObject 等

System.Xml.dll：处理 XML 操作的程序集

命名空间	命名空间简介
System.Xml	提供基于 XML 标准处理的类，如 XmlReader、XmlWriter、XmlConvert、XmlDictionary 等
System.Xml.Schema	包含为 XML 架构定义语言 (XSD)提供支持的类，如 XmlSchema 等
System.Xml.Serialization	包含用于将对象序列化为 XML 格式的文档或流的类，如 XmlSerializer，XmlRootAttribute 等

至此，我们已经对 Silverlight 运行环境中所有的类进行了分析，但在实际应用中我们还可以通过对 Silverlight 工具或第三方工具程序集的引用来扩展应用框架，从而使前端程序具有更加强大的展现与处理能力。在开发环境中，我们还可以引用 C:\Program Files\Microsoft SDKs\Silverlight\v4.0\Libraries 文件夹下的 SDK 程序集，譬如 System.Json ,System.Xml.Linq 等。如果我们使用 Silverlight ToolKit 工具包来扩展 Silverlight 框架，那么我们将在 C:\Program Files\Microsoft SDKs\Silverlight\v4.0\Toolkit 文件夹下找到对应的工具程序集文件，如 System.Windows.Controls.Toolkit.dll 就提供了如 WrapPanel 、DockPanel、DragDropTarget 等 UI 控件扩展。

最后，我们来回顾一下 Silverlight 应用程序的启动过程：



当用户浏览某个带有 Silverlight 的网页时，浏览器加载 Silverlight Plug-in 插件（npctrl.dll），npctrl.dll 启动 Silverlight 核心 CLR 服务（coreclr.dll、 agcore.dll）,同时浏览器将下载指定的 xap 文件。coreclr.dll 在浏览器宿主进程中启动一个 AppDomain 来容纳 Silverlight 的 UI 与工作线程，并调用 agcore.dll 绘制加载界面，待加载完成后实例化 Application 类，并通过启动事件（Startup）的委托（默认为 Application_Startup）对主页面进行加载，至此客户端就可以在浏览器中看到 Silverlight 程序了。

以上就是笔者针对 Silverlight CoreCLR 结构进行的简单分析 希望对正在实施或关注 Silverlight 项目的读者能有所帮助。下一期我们将通过一个 Demo 了解一下 Silverlight 在多线程方面的性能优势。

原文链接：http://www.infoq.com/cn/articles/silverlight-coreclr

相关内容：

- Silverlight 4 更新：改进了网络延迟与内存使用
- 从Silverlight访问Windows 7 的特性
- 使用Silverlight构建企业级RIA经验分享
- WPF与Silverlight的关键区别？
- Silverlight和Java的互操作

推荐文章 | Articles

Hades--JPA 的开源实现

作者 Oliver Gierke 译者 池建强

简介

几乎每个应用系统都需要通过访问数据来完成工作。要想使用领域设计方法，你就需要为实体类定义和构建资源库来实现领域对象的持久化。目前开发人员经常使用 JPA 来实现持久化库。JPA 让持久化变得非常容易，但是仍然需要编写很多模板代码。Hades 是一个开源库，基于 JPA 和 Spring 构建，通过减少开发工作量显著的改进了数据访问层的实现。本文是 Hades 的向导性教程，会带你走进 Hades 的世界，帮助你了解 Hades 在数据持久化方面能为你做些什么，并简要介绍了最新的 2.0 版本的新特性。

Hades 的核心部分由通用的持久化库实现构成，不仅提供了基本的 CRUD（增删改读）方法，而且增加了一些通用场景功能的实现，类似分页这样的功能你就不用自己实现了。还包括查询实体页、动态增加排序定义等能力。

查询

查询是使用最多的数据访问操作，如果你了解了 Hades，就会意识到查询作为 Hades 的核心功能之一，定义和执行查询是多么容易，大大减少了工作量。一般完成一个查询操作包括以下三个步骤：

1. 增加特定实体的库接口
2. 增加查询方法
3. 微调查询

首先你需要声明一个继承了 GenericDao<Entity, Id>的接口，该类保证在 Hades 接口中定义的 CRUD 方法在你的接口中是有效的，如下：

```
public interface UserRepository extends GenericDao<User, Long> { … }
```

其次是根据业务需求为你的接口增加查询方法：

```
List findByUsername(String username);
```

这又会引出问题，那就是如何实现查询方法。如果你没有增加任何附加的元数据，Hades 会尝试查找格式为{entity}. {method-name}的 JPA 查询方法，如果有效的话就使用该方法。如果没有找到任何可以解析的方法名，那就会创建一个查询方法。因此，之前的例子就会产生这样一样查询：select u from User u where u.username=?。查询解析支持更多的关键字，你可以随时在参考手册文档中找到它们。

如果想定义手工查询，你可以使用 Hades 的@Query 注释功能，让该查询直接在方法中执行：

```
@Query("select from User u where u.lastname = ?");
List someReallyStrangeName(String lastname);
```

这样你可以自由的进行方法命名，既可以对查询定义实现完全控制，又不必担心那些查询模板代码。围绕着执行查询还有很多方式，例如使用分页查询、执行编辑查询等。

引导 Hades

到目前为止我们已经讨论了如何创建带有查询方法的库接口，接下来我们看一下如何使用这些接口。首先创建实体类的实例，然后根据该实例创建 GenericDaoFactory 的实例，通过工厂实例的 getDao 方法创建你需要的资源库，如下代码所示：

```
EntityManager em = … // Get access to EntityManager
GenericDaoFactory factory = GenericDaoFactory.create(em);
UserRepository userRepository = factory.getDao(UserRepository.class);
```

由于资源库经常通过依赖注入的方式被其客户端持有，所以 Hades 提供了优雅的方式与与 Spring 应用集成。在 Spring 的应用中引导 Hades 非常简单，只需使用 Hades 的命名空间，并声明可以被匹配到的基础包即可：

```
<hades:dao-config base-package="com.acme.*.repository" />
```

以上配置方式会选中所有继承了 GenericDao 的资源库接口，并为每个接口创建 Spring 的 Bean。当然，命名空间支持更细粒度的配置，以实现精确控制。Hades 的 Eclipse 插件实现了与 Spring IDE 和 SpringSource 工具套件的无缝集成，这样你就可以从其它 Bean 引用 Hades 的资源库，并在你的工作空间把它们标记为 Spring 的 Bean。想了解详细信息请查看其参考手册。

审计

在系统中记录实体类的创建者、修改者和相关日期等信息是一个很常见的需求。Hades 提供

了监听器 EntityListener，可以帮你透明的实现这些功能。要启用 Hades 的审计功能，只需要在 orm.xml 文件中定义 AuditingEntityListener 即可：

```
<persistence-unit-metadata>
    <persistence-unit-defaults>
        <entity-listeners>
            <entity-listener
class="org.synyx.hades.domain.auditing.support.AuditingEntityListener" />
        </entity-listeners>
    </persistence-unit-defaults>
</persistence-unit-metadata>
```

并在 Spring 的配置中启动审计功能，如下：

```
<hades:auditing auditor-aware-ref="auditorAware" />
```

auditorAware 是引用的 Spring Bean，需要实现 AuditorAware 接口，通常是实现针对当前用户的安全查询功能。如果你只想跟踪创建和编辑日期，只需忽略该属性即可。想了解更多审计特性，请参考相关文档。

JPA 2.0 和事务性资源库

Hades2.0 是基于 JPA2.0 以及相关的 Hibernate、EclipseLink 和 OpenJPA 版本构建的。基于该版本的 CRUD 事务操作使用非常简单，开箱即用，对于相对简单的场景基本不需要再进行事务性的封装。未来结合资源库接口可以让事务管理变得更加简单。

```
public interface UserRepository extends GenericDao<User, Long> {
    @Transactional(readOnly = true);
    List<User> findByUsername();
    @Override
    @Transactional(readOnly = true, timeout = 60);
    List<User> readAll();
}
```

正如你看到的，你可以简单的通过为查询方法增加注释@Transactional 的方式把它们加入事务。当然，也可以不用注释的方式，基于 XML 的事务配置也能运行的很好。在 GenericDao 中的 CRUD 操作具备默认的事务性（对只读操作来说就是把 readOnly 设置为 true）。如果你想为这些方 法重新配置事务，简单的为这些方法声明自定义的@Transactional 即可实现。想了解更多详细信息，请参考事务相关的参考文档。

规范

Hades 最新版本的一个非常酷的特性是基于 GenericDao 的扩展即可符合规范。这里提到的

规范指领域驱动设计的概念，DDD 是由 Eric Evans 和 Martin Fowler 首次提出，从本质上捕捉实体的业务规则的一种开发方法。Hades 提供的抽象方式可以很容易的基于 JPA2.0 的标准 API 实现领域驱动设计。在这里我们假设你已经读过上面提到的两位作者写的 DDD 相关的书籍。Hades 提供了如下方式定义规范：

```
class BookSpecifications {
    public   Specification<Book>   hasAuthorWithFirstnameLike(final   String
firstname) {
        return new Specification<Book>() {
            public Predicate toPredicate(Root<Book> root, CriteriaQuery<?> cq,
                CriteriaBuilder cb) {
                return cb.like(root.join("author").get("firstname"),
                  firstname);
            }
        }
    }
    public Specification<Book> hasGenre(final Genre genre) {
        return new Specification<Book>() {
            public Predicate toPredicate(Root<Book> root, CriteriaQuery<?> cq,
                CriteriaBuilder cb) {
                return cb.equal(root.get( "genre"), genre);
            }
        }
    }
}
```

当然，这并不是最优雅的代码片段（如果未来与 Java 8 里的单抽象方法（SAM）特性结果结合的话，效果会更好）。从积极的方面考虑，我们在资源库层面获得了执行规范的可能性：

```
bookRepository.readAll(hasAuthorWithFirstnameLike("Oliv*"));
```

好了，我们在声明式查询方面已经做了很多，不是吗？规范的能力真正闪光的时候是把它们整合为新的规范的时候。Hades 提供了规范助手类，可以进行任意的合并：

```
bookRepository.readAll(where(hasAuthorWithFirstnameLike( "Oliv*").or(hasGen
re(Genres.IT)));
```

通过这种方式，扩展资源库就变成了只是增加新的规范。少量的规范就可以帮助你使用类似 DSL 的柔性 API 查询资源库，而不是针对每次外部查询请求都使用查询方法访问资源库。想了解更多详细信息，请参考相关文档。

扩展

2.0 版本的另一部分内容是扩展模块，可以实现 Hades 与展现层技术框架（例如 Spring 的 MVC）的无缝集成。它提供了属性编辑器和 Spring 3.0 转换器，可以透明的把实体类与 MVC

控制器方法的绑定在一起，同时 MVC 扩展还可以动态的从 HTTP 请求中提取分页信息。一个展示用户列表的页面，在控制层的写法就是这样：

```
@Controllerclass UserController {
    @Autowired
    UserRepository userRepository;
    @RequestMapping("/users")
    public void showUsers(Pageable pageable, Model model) {
        model.addAttribute("users", userRepository.readAll(pageable));
    }
    @RequestMapping("/users/{id}")
    public String showUser(@PathVariable("id") User user, Model model) {
        model.addAttribute("user", user);
        return"user";
    }
}
```

就像你看到的，showUsers 方法不需要解析 HttpServletRequest 获取分页信息。另外请注意，在 showUser(...)方法中绑定的 id 路径变量的参数已经是实体类了。由于 Spring MVC 基础框架的配置内容已经超出了本文要讨论的范围，如果想了解更多信息，请参考扩展模块配置信息的相关章节，其中提供了非常容易的设置样例。

下一步？

展望未来，2.1.x 版本之后，Hades 可能会成为 Spring Data 项目的一部分，其核心将作为实现其它数据存储资源库的实现基础。Spring Data 是 SpringSource 的项目，其目标是为新兴的数据库、特定的数据存储提供 Spring 的方言支持，包括 NoSQL 数据库。

总结

Hades 大大简化了使用 JPA 执行数据层的访问过程。你可以自由的进行精确的 CRUD 操作、执行查询和规范等。Hades 既可以独立使用，可以方便的集成到 Spring 中。除此之外，Eclipse 的插件 Spring IDE/STS 和附加的 Spring Roo 都可以非常容易的创建资源库。想了解更多信息，请参考项目网站。

关于作者

Oliver Gierke 是 VMware 公司的 SpringSource 部门的高级顾问，是 Hades 开源项目的项目主管。大约在两年前他在原公司 Synyx 启动了该项目。

原文链接：http://www.infoq.com/cn/articles/hades_jpa_repositories_done_right

推荐文章 | Articles

访谈与书摘：MasoudKalali 的《GlassFish 安全》

作者 Srini Penchikala 译者 杨娟

Masoud Kalali编著的《GlassFish安全》涵盖的内容有：Java EE安全模型；如何在Java EE应用中设计开发安全的Web、EJB模块，并将它们部署到GlassFish服务器环境中。Masoud在书里还详细介绍了GlassFish应用服务器和OpenDS目录服务器的安装及配置，OpenDS主要用来实现基于轻量级目录访问协议（LDAP）的用户认证和授权。

本书还介绍了用OpenSSO框架实现单点登录（SSO）的解决方案，其中包括用RESTful Web Services对用户进行认证和授权、通过OpenSSO解决方案确保Java EE应用和Web Services的安全。

InfoQ就此书对Masoud进行了采访，和他讨论了写作动机、Java EE 6 里的安全新特性及其他一些话题。此外，InfoQ还为读者准备了本书的书摘（第三章：设计、开发安全的Java EE应用；778 KB的PDF）。

InfoQ：你为什么编写这本书呢？

Masoud Kalali：写这本书的原因有很多，但最主要的还是以下几点：

开发人员清晰地理解安全、尤其是 Java EE 安全是很重要的，这就是写这本书最重要的原因。我认为对 Java EE 领域的开发人员来说，掌握 Java EE 安全的基础知识至关重要，所以在书里我用简单却详尽的方式对安全进行了阐述。

另一方面，GlassFish 是一款优秀的应用服务器产品，要在生产环境中使用 GlassFish，理解它的安全机制是很有必要的。

最后就是想用比参考文档更容易理解的方式，向那些对安全感兴趣的开发人员介绍 Java EE 6 引入的安全新特性。

InfoQ：对Java EE 6引入的安全新特性，你是怎么看的？

Masoud：和 Java EE 的其他服务、组件相比，安全并没有太多的变化。EJB 和 Web 层在 Java

EE 5 和 Java EE 6 里都有非常大的变化，再看看安全，尽管 Java EE 6 在平台安全方面添加了一些新鲜气息，但我觉得我们仍然有进步的空间。

InfoQ：Web 层组件 Servlet 现在可以通过声明（@ServletSecurity 注解）和编程方式（HTTPServletRequest 接口增加的 login 和 authenticate 方法）支持认证。你如何看待 Web 层提供的这种安全支持？

Masoud：现在的开发和部署都从 XML 描述符转向了注解，增加注解符合这一趋势。但编程方式的安全增强则更有意思一些，login、logout 这些方法能让开发人员更好地控制认证和授权过程。很高兴 Java EE 平台能有更好的安全特性。

InfoQ：和 Spring安全 3 等其他安全框架相比，Java EE 6 里的安全新特性怎么样？

Masoud：我觉得不能简单地把 Java EE 平台安全和 Spring 安全放在一起比较，顾名思义，前者是平台级的，后者只是基于平台构建、处理更细粒度需求的框架。当我们通盘使用 Spring 框架，或者软件架构师确信系统不需要十分细致的安全集成方案时，使用 Spring 安全就可以了。

Java EE 平台提供的安全机制无法满足需求的时候，我倾向于使用 OpenSSO 之类的产品，OpenSSO 不仅能非侵入式地和 Java 应用集成，还可以让 Java 应用从本地或通过 RESTful 接口调用其他编程语言，从而使用相同的基础设施。

InfoQ：安全相关的其他JSR是怎样和Java安全模型保持一致的呢？比如JASPIC（Java Authentication Service Provider Interface for Containers，JSR 196）和JACC（Java Authorization Contract for Containers，JSR 115）。

Masoud：众所周知，产品的可插拔性和扩展性在产品需要添加新特性的时候非常有用。我们发现，当产品 A 支持的契约接口同样被其他类似产品支持的时候，可插拔性会更加重要。

JSR-196 和 JSR-115 这两个规范提供的大纲和契约接口统一了 Java EE 应用服务器的认证和授权模型。

JSR-196 的目标对象不是应用开发人员，而是框架和安全产品的开发人员，因为它能简化不同认证模型和 Java EE 应用服务器之间的集成。使用 JSR-196 SPI，开发人员可以开发自己的认证模块、把该模块插入应用服务器、对认证模块进行配置，并拦截服务器和客户端之间传输的消息。可以在四个点进行消息拦截：

1. 客户端，请求发送给服务器之前。

2. 服务器端，目标服务接受客户端请求之前。

3. 服务器端，给客户端返回响应之前。

4. 客户端，处理服务器响应之前。

消息拦截的使用示例可以参看GlassFish应用服务器的消息安全，Metro Web Services框架就是利用消息拦截提供了容易配置、容易管理的Web Services安全。

JSR-196 的另一个应用场景是把 OpenID 等“none-supported”的认证模型集成到应用服务器当中，然后在部署到应用服务器里的应用中使用这些认证模型。

我们一提到安全，首先想到的两件事情就是认证和授权，正如我前面所说的，JSR-196 为 Java EE 应用服务器提供了统一的验证机制。JSR-115 提出的 SPI 则允许模块开发人员用自己的逻辑定义主体、定义角色、检查主体是否具有特定的角色，从而为支持 JSR 的 Java EE 应用服务器增加新的授权模型。

InfoQ：你希望 Java EE 后续的版本中能增加哪些安全特性呢？

Masoud：我希望能让开发人员更自由地控制整个认证和授权过程，不论使用缺省配置还是使用自己的配置和定制。目前，有些框架有专门的安全层，比如 Seam； 还有些框架给开发人员提供了很不错的安全方法定制，比如Spring安全；我期望能有一些基于角色的访问控制，也期望SSO能更加广泛地应用。 OpenSSO、Oracle访问管理器、IBM Tivoli访问管理器等访问管理产品现在都开始支持刚刚提到的这些功能、以及很多其他的安全集成特性了。

InfoQ：谢谢你接受我们的采访。最后一个问题，你喜欢的 IT 类图书和非 IT 类图书分别是什么？

Masoud：说到IT类图书，我喜欢《大型软件体系结构：使用UML实践指南》，最喜欢的非IT类图书则是《魔戒》系列。

原文链接：http://www.infoq.com/cn/articles/kalali-glassfish

相关内容：

- GlassFishGem前途未卜，谁将取而代之？
- Oracle宣布GlassFish路线图
- JSF在GlassFish管理控制台中的应用
- JMX在GlassFish中的应用、虚拟座谈会：Oracle和Java的发展

推荐文章 | Articles

Nuxeo 公司探秘：从 Python 迁移到 Java

作者 Charles Humble 译者 张国春

成立于 2000 年的 Nuxeo 公司是一家开源的企业内容管理领域的专业公司。2006 年，他们宣布正在进行从 Python 到 Java 的核心技术平台的迁移。四年之后，InfoQ 再次采访了 Nuxeo 的 CEO Eric Barroc，以了解这种技术转换的进展情况，以及他们新的技术栈和在 ECM 领域所处的地位。同时，我们也对动态和静态类型语言各自的优点进行了一些探讨。

ECM 市场的发展遵循这样一个常见的模式：起初市场由专业的企业软件提供商如 Documentum、Interwoven、Vignette 和 Stellent 所主宰。随后，IBM 带着 FileNet，微软带着 Sharepoint 也进入了这个市场。随着这些大型的通用软件提供商进入这个市场，开源软件公司也逐渐成为令人瞩目的参与者。这些开源软件公司包括 Nuxeo 和 Alfresco。按照 Barroca 的说法，在更普遍的意义上，ECM 工具也逐渐成为了一种日用品软件：

> 大家每天都面临着这样一个问题：如何管理范围日益增加的电子化存储信息，如何识别可能泄露的信息，并避免法律纠纷。理所当然地，随着微软的 Sharepoint 把基本的内容管理特性和概念交付到主流商业用户的手中，使得 ECM 产品逐渐成为一种日常能见的技术。我们相信开源软件的成熟和增长，以及近来认可的工业标准都会促使用户将人力物力投入到这个领域中。
>
> 与此同时，业务也在寻求实现新系统，更新技术的机会。90 年代构建的产品已经走近产品生命周期的终点。现在创建的内容，是移动、社交、开放和互操作的世界，这正是 Nuxeo 平台架构孜孜以求的目标。

Nuxeo 的核心架构构建在 Java EE 之上，使用 Java EE 作为主要技术栈并通过 OSGi 提供的模块系统进行构建。Nuxeo 构建了自己的运行库作为其使用的组件模型，例如，它允许在服务端和客户端之间交换调入或调出的 EJB 服务。

产品大量使用了 Java EE 的标准，包括：

- 数据存储抽象层使用 JDBC、JPA 和 JSR-170

- 事务管理使用 JTA (XA)，连接池则使用 JCA 技术（若未提供应用服务器，则使用 Apache Geronimo）
- Web 引擎和 REST API 使用 JAX-RS
- UI 层使用 JSF(与 Seam 和 RichFaces 配合使用)

Barroca 告诉我们："这些标准实施良好，虽然组件并非标准，但功能却更加强大，几乎能解决所有问题"。"Lucene、Hibernate、 Eclipse 以及许多不很知名的组件，使得应用系统的构建变得轻而易举。我们认为 Python 能很好地运用于许多应用场景，但对于我们的目标市场而言， 它并不是合适的技术平台。

Nuxeo 产品总共使用了大约 100 个库。除了 Barroca 在前面提到的外，关键的库还包括：以 jBPM 作为工作流引擎、OpenSocial 和 OAuth 的内嵌控件和社交特性（Apache Shindig），以及 Apache Chemistry 提供的 CMIS（内容管理互操作服务）。

系统的构建基于 Maven 和 Hudson，并以 JUnit 作为单元测试框架。目前，产品支持 JBoss、Tomcat 和 Jetty，而且平台的大部分内容也能够运行于纯粹的 OSGi 之上。要增添一个新的支持 Nuxeo 运行的应用服务器还是比较快的，几天就能完成相应的工作。

Java 库所具有的深度和广度是转换到 Java 平台的一个主要动力，但 Barroca 还提到了其他原因。

> **市场**：Java 应用系统的市场非常庞大。所有的公司都知道 Java 技术，大多数公司也在使用 Java 应用系统。很多公司在 IT 系统中使用了 Java 技术。所有的系统集成商都具有广博的 Java 知识。
>
> **社区**：同样庞大。例如，Java Apache 社区就非常了得。每年都有一些关于 Java 技术的大型会议（JavaOne、ApacheCon、Devoxx、Jazoon 等等），以及成百上千的小型会议。
>
> **技术规范**：在 Java 世界中，很多技术都有基于已发表的技术规范的实现与参考实现。这有助于规范的清晰化，代码的互操作性，同时也推动了规范的实现者去做到极致。
>
> 我们还受益于许多高水平的工具。这些工具包括虚拟机、调试工具、集成开发环境以及监控和性能定基等。

Barroca 谈到了重新构建的平台架构与开源开发模型在 ECM 领域开辟了新的市场，使得 Nuxeo 占据了有利的位置。

> 新型的独立软件提供商（ISV）正在使用 ECM 平台，将业务知识打包到软件中，以创建和销售这些内容应用。从建筑项目管理和生物科技与生命科学的临床试验管理，到州和地方政府控制指挥中心所使用的软件，这些应用跨越了不同行业和职能部门。

我们平台的灵活性和融合了开源技术的软件特性为开发内容应用的架构师和开发者提供了便利。这种开发模式使得新的构建内容应用方式变得简单，清晰，快速。

开源 ECM 也带动了内容管理市场发展出新的需求。之前，企业之所以没有部署 ECM，是因为高昂的前期费用，或者缺乏对所构建和定制的应用系统的控制。现在，这些问题有了新的解决方案。

一个备选方案是在 JVM 上运行 Jython 编写的程序，因为 Jython 是用 Java 编写，以 JVM 作为目标运行平台的 Python 实现。但 Nuxeo 没有采用这个方案。

> 我们不惜时间，希望能够实实在在地利用在 ECM 解决方案积累的六年经验，按照我们的想法去从头构建一个完整的平台。这不仅是编程语言的转换，也是平台的转换。我们要创建一个完整的 ECM 平台，由开发者所设计，为开发者而设计，那些 ECM 厂商还停留在口头上的想法，我们却在技术上将他们变成了现实。在这个行当里，很多商业软件厂商的扩张策略是通过并购来构建自己的软件产品线。但他们只有一套软件，却没有一个技术平台。在过去的四年里，我们已经可以构建完整的技术平台，并通过它来创建内容应用。

考虑到 Nuxeo 系统的项目被重新设计和重写了，我们该如何比较两个系统之间的差异。

> 在性能方面，取决于你的度量值，新的系统在规模上至少提升了 10 到 100 倍。功能特性的数量增加了大约 4 倍。在开发者的兼容性和易用性方面，不仅保持了先前的优点，还提供了更多的工具和体验。新平台的整体表现比旧平台翻了一番，但离翻两番的提升还有些差距。

起初的转化工作花费了超过 55 个人年的工作——而 Barroca 之前粗略的估计是 10-20 个人年的开发代价。

> 如果算上其他贡献者和持续的开发，我们花费了 150——200 个人年。但是我不是很喜欢用人年来衡量软件开发，因为对于开发者而言，都没有谁有固定的人年值，更何况公司呢。我们有一个非常资深的开发团队。

那么，Nuxeo 的老用户对于这个改变的反应如何呢？Barroa 告诉我们：

> 从安装数量上看，约 80%的客户可以接受这个改变。对于那些不愿采用新平台的机构，可由社区的力量来保证对旧平台的继续支持。这就是开源开发模式的迷人之处。一种技术能够存活依赖于用户和开发者群，我们把这看作一种“有未来”的优势。

既然 Nuxeo 同时使用了 Python 和 Java，因此有必要谈谈这两种语言的优点。通常的主张是动态语言有着更快的编码速度，因为他们天生具有更强的表现力。静态语言则可以在测试阶段节省更多的时间，因为缺乏类型信息使得用动态语言编写的代码更难测试，而且动态语言也比静态语言运行速度慢，扩展时所需代价更大。Barroca 告诉我们：

> 在研发 Nuxeo 的实践中，我们同时使用了动态类型语言和静态类型语言（Python 和 Java）来编写大规模的应用系统，其中有成千上万个类，数以百万计的代码行规模，我们可以肯定地说静态类型语言更适合编写此类应用系统。
>
> 使用静态类型语言编程，我们可以对编写的代码进行有效的静态分析，这会带来如下的好处：
>
> - 集成开发环境可以提供正确的代码自动补全
> - 集成开发环境可以知道所有类中所有方法和字段的用法，便于搜索和重构
> - 更多的编译时检查
> - 发现代码中存在的模式和缺陷的工具
> - 根据代码使用的实际类型生成文档的工具
>
> 在绝大多数的情况下，动态类型语言的方法总是接受同一种类型的参数，所以除了少输入一下参数的类型外，动态语言不会带来任何好处。但静态类型语言使用显式类型，这不但没有任何阻碍，并且它本身也是好的文档。
>
> 另一种情形是，当需要真正的多态时，现代的静态语言提供了泛型，以此获得一部分（并非全部）动态类型语言的优势。
>
> 有些时候，例如 Python、Javascript、Ruby 和其他动态类型语言，体现出来的另一种优势是，可以在运行时修补一个对象实例，给它添加行为（在 Python 领域，这常被称作 Monkey Patching）。这好像不错，但它对理解代码，代码静态分析，调试等来说就太糟了。它可能在修修补补时很方便，但从长期来看，并不是个好办法。
>
> 最后，动态类型语言也提供了反射工具来找出一个未知类型对象的属性和方法，但是 Java 从 Java 5 开始早就有这些特性了，所以动态类型语言在这方面也没有什么优势。

Barroca 希望深入探索最近在 Java EE 6 和 Java SE 7 中发布或者即将要发布的技术。这些技术包括：模块系统、JSF 2、Bean 验证、媒体组件和新的 NIO 库。他仍然相信 Java 对他们的产品而言是最好的技术。

> 我们也为我们所创造的技术而感到自豪，从运行时到我们在高层（包括 UI）所创建的模块。它并没有被公开，因为我们完全将其看作是 ECM 的一部分。但平台中仍有一些好东西有助于改善 Java 应用程序。扩展系统、基于 Seam/JSF 的模块化 UI、分布式汇编引擎和许多组件都是我们在创建模块化 Java 应用系统时，频繁用到的重要技术。在定制基于 Nuxeo 的应用系统时，不需要分离出 UI 来单独定制，你可以通过插件来完成。这是一个非常开放和可扩展的模型。

原文链接：http://www.infoq.com/cn/articles/nuxeo_python_to_java

相关内容：

- Python赢得Tiobe 2010 年度语言大奖
- JetBrains开发者工具
- 半静态语言–原理和价值分析
- Nuxeo推出了自己的插件商店Marketplace，企业内容管理产品支持CMIS
- Nuxeo推出Fise语义引擎

推荐文章 | Articles

Maven 实战（四）--基于 Maven 的持续集成实践

作者 许晓斌

Martin 的《持续集成》

相信很多读者和我一样，最早接触到持续集成的概念是来自Martin的著名文章《持续集成》，该文最早发布于 2000 年 9 月，之后在 2006 年进行了一次修订，它清晰地解释了持续集成的概念，并总结了 10 条实践，它们分别为：

- 只维护一个源码仓库
- 自动化构建
- 让构建自行测试
- 每人每天向主干提交代码
- 每次提交都应在持续集成机器上构建主干
- 保持快速的构建
- 在模拟生产环境中测试
- 让每个人都能轻易获得最新的可执行文件
- 每个人都能看到进度
- 自动化部署

原始文章距今已 10 年有余，这在软件行业中算是很长的时间了，但我们都能看到 Martin 总结的这些实践依旧闪耀着光芒，依旧有很多团队在努力实践它们并得到了丰厚的回报，当然也有很多团队因为各种原因拒绝实践持续集成从而无法体会到个中好处。

从这 10 条实践中我们能找到很多流行开源工具的影子，例如版本控制工具cvs、svn、git，自动化构建工具Maven、Ant，自动化测试框架 JUnit、TestNG，以及持续集成服务器

CruiseControl和Hudson等等。其实不论你是否实践持续集成，单独使用这其中的很多工具都能发挥极大的价值，持续集成的一大意义在于它引入了一个有效的流程，能让这些工具有机融合，并相互促进。 关于持续集成还有一本获得Jolt大奖的图书，名为《持续集成——软件质量改进和风险降低之道》。但无论是Martin的文章，还是这本图书，都没有阐述使用Maven作为自动化构建工具实施持续集成的细节。本文旨在介绍一些基于Maven实施持续集成的实践，希望这些经验能从具体处帮助到读者。

架设私有 Maven 仓库

Martin的文章并没有涉及到依赖管理的内容，但在Java的世界中，依赖管理是开发人员不得不面对的问题。无论是外部的开源类库依赖，还是项目内部的模块间依赖，都需要有效地管理。可以说依赖管理是持续集成核心的内容之一。Maven通过其依赖管理机制和随处可用的中央仓库 有效地解决了这个问题，用户只需要在POM中声明项目所需要的依赖，Maven就能在构建的时候自动从仓库解析依赖。

不过仅仅这样是不够的，我们知道，持续集成的最大好处在于降低风险，简单地来说就是尽早暴露问题，能让开发人员及早发现并修复，从而降低修复成本。可是，如果每个人都从中央仓库重复下载依赖，这是非常耗时的，集成的反馈周期肯定会延长。我已经无数次听到有人抱怨“Maven 在下载整个 Internet！”。构建要快！持续集成反馈要快！Maven 你不能拖慢这个流程。

幸运的是开源世界有很好的解决方案，只要使用Maven仓库管理器软件如 Nexus建立一个私有的Maven仓库，问题就能迎刃而解。原理很简单，这个位于局域网内的Maven仓库能够代理所有外部仓库，从而避免所有人从Internet重复下载依赖文件。这样Maven解析依赖的时候仅限于局域网，构建速度就大大地加快了。例如大家都需要使用 junit-4.8.2.jar，当第一个人向私有仓库请求的时候，私有仓库从中央库下载并缓存下来，假设耗时 10s，之后其他人需要 junit-4.8.2.jar的时候，私有仓库直接使用缓存的文件，这个耗时可能就是 1s。如果有 100 个开发人员使用该文件，那节省的时间就是 100 * 10 - (10 + 99 * 1) = 891s ，实际情况中依赖的数量可能会是成百上千，那节省的时间就变得非常的可观。

也许有人会说，我也完全可以将项目依赖加入到版本控制中，这一点甚至在 《卓有成效的程序员》中 都被明确提及，在该书第 5 章的"DRY版本控制"一节中，Neal Ford有这么一段话：“所有用来构建项目的东西都应该被放入版本控制，包括二进制文件（类库，框架，JAR文件，构建脚本等等）”。作者进一步解释了其目的，这么做能够保证项目不受外部因素影响（如依赖版本变化，甚至丢失），保证构建的稳定，作者也同时提及了一般版本控制工具处理二

进制文件的性能问题。抛开这条结论性的实践，仔细考虑其目的，我们就能发现，私有Maven仓库同样能保证构建的稳定，而且能避免版本控制工具处理二进制文件而造成的潜在性能问题。所以，我斗胆说一句，Neal Ford所提的这条实践OUT了！

私有 Maven 的仓库的意义还不仅限于此，结合自动化部署和 Maven 的 SNAPSHOT 机制，它能大大促进项目集成的效率。

在模块化的开发环境中，大家各司其职，专注于自己所负责的模块，持续集成的规则是，在往版本控制提交代码前，需要先保证本地构建没有问题，那一般的做法就是更新所有模块的代码并构建。可是，真的需要构建那些其实你并不怎么关心的模块么？且不谈一旦构建他人代码时出错，你往往会不知所措，这种做法同时也增加了本地构建的时间。

Maven 有 SNAPSHOT 版本的概念，其目的就是让你能够构建一个临时的版本，供团队他人使用，这样他们就不必在代码的层次关心自己的依赖。于是私有 Maven 仓库就充当了一个中介的作用，而持续集成服务器就多了一个职责，每次它成功构建一个模块，都应该将该模块的 SNAPSHOT 版本发布到 Maven 仓库中。现在，大家就不用去构建别人的代码了，Maven 能自动帮你从私有仓库解析下载依赖的最新 SNAPSHOT（使用 **mvn** 命令的**-U** 参数强制更新）。注意，除了持续集成服务器外，任何其他人都不应该发布 SNAPSHOT 版本到 Maven 仓库，因为只有持续集成服务器的环境是可信任的，你能在本地成功执行 **mvn clean install** 并不代表持续集成服务器上该命令能成功，由于每个人的本地环境各有差异，因此集成的成功与否应当以持续集成服务器为准，而只有集成成功后，SNAPSHOT 才可以被部署到私有仓库供他人使用。

鉴于上述的原因分析，我认为在基于 Maven 的持续集成环境中，再怎么强调私有 Maven 仓库的重要性都是不为过的。

正确的集成命令

在持续集成服务器上使用怎样的 mvn 命令集成项目，这个问题乍一看答案很显然，不就是 **mvn clean install** 么？事实上比较好的集成命令会稍微复杂些，下面是一些总结：

- **不要忘了 clean**：clean 能够保证上一次构建的输出不会影响到本次构建。
- **使用 deploy 而不是 install**：构建的 SNAPSHOT 输出应当被自动部署到私有 Maven 仓库供他人使用，这一点在前面已经详细论述。
- **使用-U 参数**：该参数能强制让 Maven 检查所有 SNAPSHOT 依赖更新，确保集成基于最新的状态，如果没有该参数，Maven 默认以天为单位检查更新，而持续集成的频率应该

比这高很多。

- **使用-e 参数**：如果构建出现异常，该参数能让 Maven 打印完整的 stack trace，以方便分析错误原因。
- **使用-Dmaven.repo.local 参数**：如果持续集成服务器有很多任务，每个任务都会使用本地仓库，下载依赖至本地仓库，为了避免这种多线程使用本地仓库可能会引起的冲突，可以使用**-Dmaven.repo.local=/home/juven/ci/foo-repo/**这样的参数为每个任务分配本地仓库。
- **使用-B 参数**：该参数表示让 Maven 使用批处理模式构建项目，能够避免一些需要人工参与交互而造成的挂起状态。

综上，持续集成服务器上的集成命令应该为 **mvn clean deploy -B -e -U -Dmaven.repo.local=xxx** 。此外，定期清理持续集成服务器的本地 Maven 仓库也是个很好的习惯，这样可以避免浪费磁盘资源，几乎所有的持续集成服务器软件都支持本地的脚本任务，你可以写一行简单的 shell 或 bat 脚本，然后配置以天为单位自动清理仓库。需要注意的是，这么做的前提是你有私有 Maven 仓库，否则每次都从 Internet 下载所有依赖会是一场噩梦。

用好 Profile

如果不需要考虑各种不同的环境， 而且你的自动测试（包括集成测试）跑得飞快，那你就不用为项目建立多个集成任务。但实际的情况是，集成的时候可能要考虑各种环境，例如开发环境、测试环 境、产品环境。而当项目越来越大，测试越来越多，控制构建时间在一个可接受的范围内（例如 10 分钟）变得越来越不现实。《持续集成——软件质量改进和风险降低之道》中介绍了一种名为 **分阶段构建（staged build）**的解决方案，例如你可以将构建分为两个部分，第一部分包括了编译和单元测试等能够快速结束的任务，第二个部分包括集成测试等耗时较长的任务，只有第一部分成功完成后，才触发第二部分集成。这么做的意义在于让持续集成的反馈尽可能的快。

Maven的Profile机制能够很好的支持分阶段构建。例如，借助 Maven Surefire Plugin，你可以统一单元测试命名为 **UT，统一集成测试命名为 **IT，然后配置Maven Surefire Plugin默认只运行单元测试，然后再编写一个名为integrationTest的Profile，在其中配置Maven Surefire Plugin运行集成测试。然后再以此为基础分阶段构建项目，第一个构建为 **mvn clean install -B -e -U** ，第二个构建任务为 **mvn clean deploy -B -e -U -PintegrationTest** 。前一个构建成功后再触发第二个构建，然后才部署至Maven仓库。值得一提的是，Maven Surefire Plugin能够很好支持JUnit 3、JUnit 4 和TestNG，你可以按照最适合自己的方式来划分单元测试和集成测试。

另一个常见的分阶段构建案例是生成 Maven 站点，使用 **mvn clean site** 生成站点往往比较耗时且耗资源，这样的任务对应的持续集成中的**持续审查**阶段，该阶段往往不需要很高的集成频率。你会希望每 10 分钟就检查源代码变更并编译测试，但很少有人会希望每 10 分钟让系统生成一次测试覆盖率报告、CheckStyle 报告等内容，因此合理的做法是使用一个较低的频率，例如每天，这样可以避免无谓的资源消耗，更重要的是，这样不会拖慢本该很快的编译 和单元测试等反馈内容。

还有一些情况是系统需要基于不同环境进行集成，这时候就需要用到Maven的属性机制、资源过滤、以及前面提到的Profile。篇幅原因，这里不再展开。

小结

持续集成是敏捷最重要的实践之一，但如何在基于Maven的环境下实践持续集成却鲜有文章详述，本文介绍了一些该主题的最佳实践，包括架设私有仓库、使用正确的集成命令、利用Profile等技术处理分阶段构建等等。本文旨在让广大Maven用户认识到这些实践的存在及重要性，并没有详细解释一些诸如 Nexus安装配置、Maven Surefire Plugin配置、或者说Profile配置使用方面的细节，如果你希望看到更细节的介绍，可以参考我的《Maven实战》一书。除了上面的内容之外，该书还详细解释了如何使用Hudson（也许该 改称Jenkins了）这一最流行的开源持续集成服务器。当然，如果你有关于Maven和持续集成方面的经验，也请不吝分享。

关于作者

许晓斌（Juven Xu），国内社区公认的Maven技术专家、Maven中文用户组创始人、Maven技术的先驱和积极推动者。对Maven有深刻的认识 实战经验丰富，不仅撰写了大量关于Maven的技术文章，而且还翻译了开源书籍《Maven权威指南》，对Maven技术在国内的普及和发展做出了很大的贡献。就职于 Maven之父的公司，负责维护Maven中央仓库，是Maven仓库管理器Nexus（著名开源软件）的核心开发者之一，曾多次受邀到淘宝等大型企业开 展Maven方面的培训。此外，他还是开源技术的积极倡导者和推动者，擅长Java开发和敏捷开发实践。他的个人网站是：http://www.juvenxu.com。

原文链接：http://www.infoq.com/cn/articles/xxb-maven-4-ci

相关内容：创建并扩展 Apache Wicket Web 应用、Maven 实战（一）——坐标规划

时刻关注企业软件开发领域的变化与创新

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

Java社区：企业Java社区的变化与创新

.NET社区：.NET和微软的其它企业软件开发解决方案

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

SOA社区：关于大中型企业内面向服务架构的一切

Agile社区：敏捷软件开发和项目经理

Architecture社区：设计、技术趋势及架构师所感兴趣的话题

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

新品推荐 | New Products

Spring GemFire 1.0.0 发布，面向 Java 与.NET

作者 张龙

近日，SpringSource 发布了 Spring GemFire 1.0 的首个 GA 版，面向 Java 与.NET。Spring GemFire 项目是个分布式的数据管理平台，旨在简化基于 Spring 的高可伸缩应用的构建。

原文链接：http://www.infoq.com/cn/news/2011/02/spring-gemfire-1.0.0-release

SpringSource Tool Suite 2.6.0 M1 发布

作者 张龙

近日，SpringSource 发布了下一版 SpringSource Tool Suite（STS）的首个里程碑版本，增加了不少新特性，同时还更新了很多功能。

原文链接：http://www.infoq.com/cn/news/2011/02/springsource-toolsuite-release

Jenkins 首个版本发布及 Hudson 支持

作者 Alex Blewitt 译者 张龙

Jenkins 的首个版本业已发布，同时还为那些想从 Hudson 迁移到 Jenkins 上的用户提供了一个迁移文档。 由于它是以 WAR 文件的形式发布的，因此最值得关注的变化就是 WAR 文件的名字已经不再是 hudson.war，而变成了 jenkins.war；在 UNIX 系统上，运行库的位置位于/var/lib/jenkins 而非/var/lib/hudson。run-as 用户也被重命名为了 jenkins。

原文链接：http://www.infoq.com/cn/news/2011/02/jenkins-hudson

支持大数据的 JasperSoft 4 发布了

作者 Ron Bodkin **译者** 张龙

近日，JasperSoft 宣布提供对大数据系统报表的支持，包括 Hadoop、几种流行的 NoSQL 数据库以及 3 个 MPP 分析关系数据库上的各种模型报表。

原文链接：http://www.infoq.com/cn/news/2011/02/jaspersoftbigdata

Oracle 发布补丁，修复 Double.parseDoubleBug

作者 Charles Humble **译者** 张龙

近日，Oracle 发布了一个名为 FPUpdater 的补丁程序，用于修复 Java 平台上存在了 10 年之久的 Bug，恶意用户可以利用该 Bug 向服务器发起拒绝服务攻击。时隔这么多年才发布，该补丁程序可谓是打破了记录。

原文链接：http://www.infoq.com/cn/news/2011/02/double_bug

Orion--基于 Web 的 Eclipse

作者 Michael Stal **译者** 沙晓兰

Eclipse 基金会的执行董事 Mike Milinkovich 在一月份宣布了一个名为“Orion”的新工具。Mike 把它称作“Eclipse 的新冒险”，这个工具将为开源工具的集成提供一个浏览器环境。二月初的时候，开发团队发布了 Orion 的里程碑版本 M5。

原文链接：http://www.infoq.com/cn/news/2011/02/orion-eclise-web

Amazon AWS 推出 CloudFormation

作者 James Vastbinder **译者** 丁雪丰

今天 Amazon AWS 团队推出了一个新的工具——AWS CloudFormation，它使用模板来

简化 AWS 资源的准备与部署。AWS 用户无需额外费用即可获得该工具。此外，Amazon 还为开发者和系统管理员提供了一些范例模板。

原文链接：http://www.infoq.com/cn/news/2011/02/aws-cloudformation

数据访问框架 MetaModel 1.5 版带来更佳开发体验

作者 Michael Stal **译者** 马国耀

Eobject.org 的开源软件 Java 框架 MetaModel 针对不同数据源实现了统一访问、查询和挖掘 API。它支持广泛的数据持久技术，如关系型数据库、XML 文件和 Excel 表格等。

原文链接：http://www.infoq.com/cn/news/2011/02/Data-Access-Framework-MetaModel

更易于向 Windows Azure 部署 Java 应用的新方式

作者 Abel Avram **译者** 侯伯薇

微软最近发布了一种工具，它简化了向 Windows Azure 部署 Java 应用程序的过程。

原文链接：http://www.infoq.com/cn/news/2011/02/Deploy-Java-Apps-Azure

Moonlight 4.0 的第一个预览版已经发布

作者 Abel Avram **译者** 朱永光

Moonlight 4.0 的第一个预览版包含了所有 Silverlight 3.0 的 API 和一部分 Silverlight 4.0 的 API。新特性包括：浏览器外支持（Out-of-Browser）、GPU 加速的图形处理、3D 变换、投影渲染、V4L2 视频捕获、H.264 和 AAC 支持，以及更好的流媒体平滑处理。

原文链接：http://www.infoq.com/cn/news/2011/02/Moonlight-4

推荐编辑 | 原创团队编辑 张逸

第一次认识 InfoQ 中文站，大约是在 2008 年初，中文站.NET 社区的主编朱永光邀请我担任.NET 社区的编辑。对于这一邀请，我倍感荣幸。"乐莫乐兮新相知"，InfoQ 给我的正是这样让人欣喜而又耳目一新的感觉。在众多的技术网站中，InfoQ 的风格是如此的别具一格，尤其是中文站对于国外先进技术的介绍与剖析，更是为国内的技术人士打开了一扇开放之窗。InfoQ 面向的是软件行业的中高端群体，限定这样的受众不可避免地会在访问量上受到影响，但却营造了一种专业、严谨而高端的技术文化，可谓"失之东隅收之桑榆"。InfoQ 新闻的新与广，文章的深与精，是其最大的特色。这种广博与专业，正是我喜爱 InfoQ 的缘由。与 InfoQ 的初识，绝对是我的一笔宝贵财富。

第二次认识 InfoQ 中文站，却是在 2009 年 4 月的北京，QCon 在清华科技园国际会议中心那次震撼的开幕。正是这一次，我开始为 InfoQ 的能量感到惊讶。如此众多的大师云集北京，布道清华，剑指企业开发的巅峰体验。很难想象那种一票难求的境况会出现在技术大会之上。被众多粉丝围住的大师们，开始有了灿烂的星味。而作为 InfoQ 编辑的我，则是近水楼台先得月，有了与大师面对面的难得机会。正是在这一次 QCon 北京，我采访了 Martin Fowler，又与 EBay 的首席架构师 Randy Shoup 相谈甚欢。虽然，我的技术水平并未因为聆听大师演讲而突飞猛进，但自我感觉境界却有了质的提升。这是一种只可意会的感悟。因为 QCon，InfoQ 在我眼中开始变得强大。

第三次认识 InfoQ 中文站，却源于一次尴尬的经历。当我兴致勃勃将悉心写就的文章发给 InfoQ 的另外一位编辑时，却被委婉地告知文章内容不够深，不建议录用。我被彻底打倒了，原来自己人也没有后门可开啊！失落之余，回味再三，我开始为跻身这样的团队感到自豪了。回想之前我与几位编辑共同审校业界专家的大作，不是同样追求内容的严谨与精深吗？我们在邮件组中畅所欲言，提出自己的看法。言辞犀利却真诚，反驳辛辣却没有火药味，这是一种坦诚相待的真友谊。技术团队，就要这样才能进步，才能出精品。因为这次失败的投稿，InfoQ 让我感到了一种前所未有的荣誉感。忽然觉得脚下的路虽然漫长，自己却走得如此的踏实。

不知不觉，三个年头过去了，从最初认识 InfoQ 到现在，我们似乎都有了长足的进步。通过 InfoQ，我了解到了最新的技术，前沿的动态，以及最专业深度的文章；最关键的是，它还给了我展示的舞台，能够与那些叱咤风云的人物站在一起，享受技术给我们带来的乐趣。最可贵的是我拥有了一帮真诚相待的战友，即使有的素昧平生，或者萍水相逢，但在 InfoQ 这个大家庭中，我们携手进步，共同期望掀起技术大潮中的一点浪花。

封面植物

杏黄兜兰

杏黄兜兰，我国特有珍稀植物，产于云南碧江，生于岩壁上。我国植物学家张敖罗 1979 年初次采集；1982 年经陈心启、刘方媛定名的新种。杏黄兜兰花形与硬叶兜兰相似，不同之处是它的整个花朵为金黄色，香港称之为“金兜”。

生长分布：产于中国云南省西部（碧江、泸水）。生于海拔 1400-2100 米的石灰岩壁积土处或多石而排水良好的草坡上。缅甸可能也有分布。杏黄兜兰模式标本采自云南省碧江。

植物形态：地生或半附生植物，地下具细长而横走的根状茎；根状茎直径 2-3 毫米，有少数稍肉质而被毛的纤维根。叶基生，二列，5-7 枚；叶片长圆形，坚革质，长 6-12 厘米，宽 1.8-2.3 厘米，先端急尖或有时具弯缺与细尖，上面有深浅绿色相间的网格斑，背面有密集的紫色斑点并具龙骨状突起，边缘有细齿，基部收狭成叶柄状并对折而套叠。

花瓣大，宽卵状椭圆形、宽卵形或近圆形，长 2.8-5.3 厘米，宽 2.5-4.8 厘米，先端急尖或近浑圆，内表面基部具白色长柔毛，边缘具缘毛； 唇瓣深囊状，近椭圆状球形或宽椭圆形，长 4-5 厘米，宽 3.5-4 厘米，基部具短爪，囊口近圆形，整个边缘内折，但先端边缘较狭窄，囊底有白色长柔毛和紫 色斑点；退化雄蕊宽卵形或卵圆形，长 1-2 厘米，宽 1-1.5 厘米，先端急尖，背面具钝的龙骨状突起。花期 2-4 月。

种植培育：植比较困难，需要较高的昼夜温差，喜欢带有石灰质的介质，病虫害较少。杏黄兜兰在开花结束后，会在根部形成多条走茎，走茎的顶端，会长出新的小苗，可以待小苗成长成熟后剪下繁殖。

用途：杏黄兜兰的花、叶均有较高的观赏价值，室内盆栽，用以装点书房、客厅，格外高雅。

1kg.org
多背一公斤
爱自然 | 更爱孩子

架构师 3 月刊

每月 8 日出版

本期主编：侯伯薇

总编辑：霍泰稳

流程编辑：葛明见

美术编辑：胡伟红

读者反馈：editors@cn.infoq.com

投稿：editors@cn.infoq.com

InfoQ 中文站新浪微博：
http://t.sina.com.cn/infoqchina

商务合作：sales@cn.infoq.com 13581658359

本期主编：侯伯薇，InfoQ 中文站翻译团队编辑

侯伯薇，生于丹东凤城，学在春城长春，工作在滨城大连；虽已年过而立，但自问童心未泯；对代码热情不减，愿与天下程序员共同修炼，不断提升。译有《学习 WCF》、《Expert C# 2008 Business Objets》。其在 InfoQ 的编辑特辑为：侯伯薇。

InfoQ
Enterprise Software Development Community

《架构师》月刊由 InfoQ 中文站出品。

所有内容版权均属 C4Media Inc.所有，未经许可不得转载。

时刻关注企业软件开发领域的变化与创新

架构师

www.infoq.com/cn/architect

每月8号出版

InfoQ neue
Enterprise Software Development Community

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