时刻关注企业软件开发领域的变化与创新

架构师

1月 ARCHITECT

特别专题

2010年年终技术盘点

年度技术回顾之
数据库、NoSQL、开源软件

开放平台回顾与前景展望

2011互联网技术发展浅析

技术人员的产品观——暨2011年展望

百万级访问量网站的技术准备工作

采访开源Web框架
SimpleFramework开发团队

Java深度历险（一）
--Java字节代码的操纵

半静态语言-原理和价值分析

访谈与书摘：George Fairbanks与
《恰如其分的软件架构》

DOJO：不容忽视的RIA框架

InfoQueue
Enterprise Software Development Community

每月8号出版

篇首语 | **Editor's letter**

2010 这一年

时间过的好快，2010 已经渐行渐远，2012 也即将到来了：-)

在过去的这一年，业界发生了翻天覆地的变化：微博的持续火爆、NoSQL 热度的不断攀升、HTML 5 的崭露头角、可伸缩性架构的日渐升级、RIA 市场的激烈争夺、各种技术大会的层出不穷、开源领域的蓬勃发展。这让我们开发者不禁感慨：IT 界真是一日不见，如隔三秋兮。

在年终岁尾之际，《架构师》又与广大读者见面了。在本期《架构师》中，除去读者已经能够想象得到的精彩内容外，我们还邀请了各领域的业界专家畅谈过去一年来所发生的众多事件，并对未来一年的技术趋势和企业架构进行了展望。这里有丁香园冯大辉的年终总结与展望、腾讯潘少宁谈技术人员的产品观、新浪微博架构师杨卫华的 2011 互联网技术发展浅析、淘宝岑文初的开放平台回顾与前景展望。相信这些内容会让读者大呼过瘾。

在整个业界蓬勃发展之际，InfoQ 中文站当然也要与时俱进，不断前行了。细心的读者可能已经发现，现在 InfoQ 中文站原创内容的数量与质量都迈上了一个新台阶，新闻与深度内容的翻译速度也有了相当大的提高，我们所做的一切都是为了满足读者对高品质内容的追求，都是为了“时刻关注企业软件开发领域的变化与创新”这一理念。《阅读者》专栏的出现更是为广大从业者开启了智慧之门，帮助大家从浩如烟海的图书中寻觅到真正的优质好书，精彩的书评则让人拍案叫绝。这一切的一切都源于 InfoQ 中文站众编辑希望为社区贡献力量的愿景。目前的 InfoQ 中文站已经形成了两大阵营：原创团队与翻译团队。更加精细化的运作相信会为广大读者带来更加优质的内容。

即将到来的 QCon 2011 北京大会一定又是广大开发者的一次技术盛宴，过往的 QCon 北京大会已经证明了其价值所在，相信几个月后的 QCon 2011 北京大会会为广大从业者带来别样的感觉与收获，我们期待着你的参与。你的鼓励与建议将是我们不断前行的动力。

最后，真心祝愿广大读者在新的一年中家庭幸福，事业精进。

InfoQ 中文站翻译团队主编：张龙

目　录

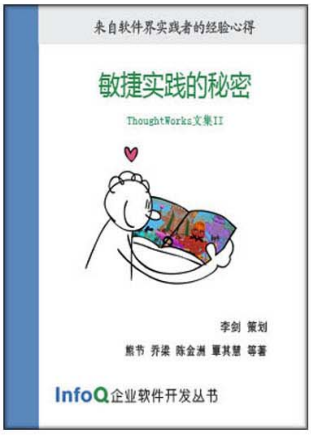
[新品推荐]

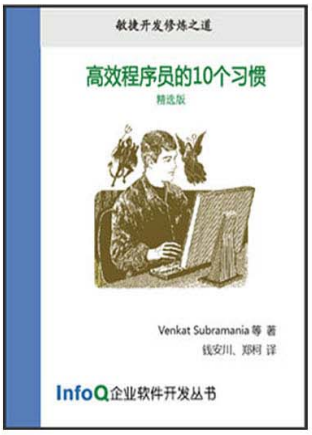
[推荐编辑]

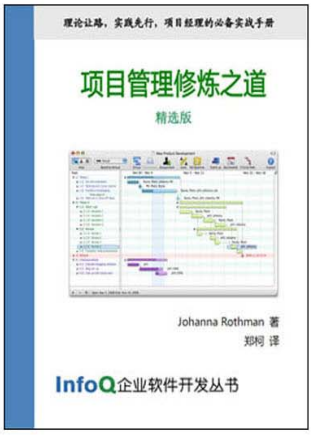
[封面植物]

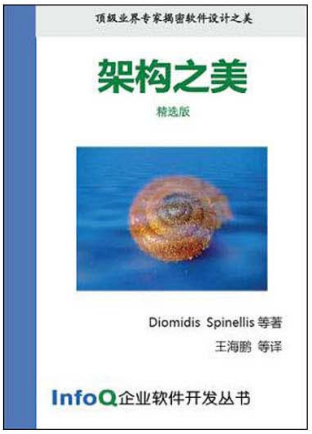
InfoQ企业软件开发丛书

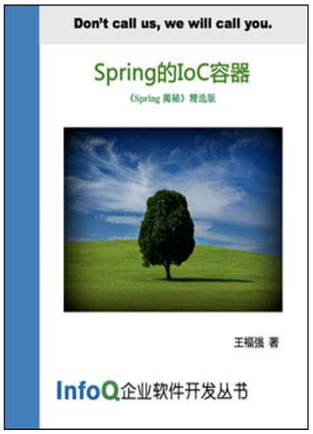
欢迎免费下载











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

杨卫华谈新浪微博架构

杨卫华，新浪产品部技术经理，目前工作以新浪微博技术平台为主，曾负责过新浪 IM 等通讯服务端架构设计。对互联网后端技术，分布式，网络编程，XMPP 即时通讯等领域感兴趣。InfoQ 中文站专门就新浪微博的技术支持以及维护等方面话题采访了杨卫华先生。

杨卫华，新浪产品部技术经理，目前工作以新浪微博技术平台为主，曾负责过新浪 IM 等通讯服务端架构设计。对互联网后端技术，分布式，网络编程，XMPP 即时通讯等领域感兴趣。曾组织多次广州及珠三角技术沙龙活动。个人 blog 为：http://timyang.net/。

InfoQ：大家都知道，在美国有一个非常有名的信息分享平台叫做 **Twitter**，而在中国，我们也有同样的方式，就是现在非常流行的新浪微博，它还有个非常温馨的名字，叫做围脖。而新浪微博的架构就是杨卫华先生主持开发的。

今天我有幸采访到杨卫华先生，让他来给大家谈一谈，在新浪微博的技术架构方面，他们是如何为用户提供更好的性能、更好的服务的。

卫华先生你好，我的第一个问题是，在新浪微博上有很多名人，名人的微博一般都是非常热的，对它们的访问量也特别高，那么对于这些微博，您采用了什么样的方式来支持这种大数据量的访问呢？

杨卫华（以下简称卫华）：对于这个问题，我们做过专门的分析。因为最近新浪微博有名人扎堆的现象，我们根据这个现象，从以下几个角度来进行解决。

首先根据中国的网络现状，比如说网通和电信，之间的网络访问速度会比较慢，我们考虑让用户能够访问就近的服务器，这样使用体验、速度都能达到要求。我们根据新浪以往的经验，在全国部署了大量服务器，这样就为微博提供了硬件上的保证。

第二个方面，在程序优化的方面，在产品上线之前，我们进行了全方面的压力测试，如果系统在某个方面可能会出现瓶颈，比如名人的访问量比较高的话，我们就从那个角度去优化。比如说 Cache 是否够用，数据库访问是不是瓶颈，这方面我们预先都有对压力的估计，然后

会针对那些方面去做优化。

第三个方面，对于那些静态资源，比如图片、视频、JS 脚本，我们有专业的 CDN 来解决的，这样就能够保证全国的用户在访问新浪微博时都能够得到比较好的体验。

InfoQ：现在的服务器大概都架设在哪几个部分？覆盖全国哪几个地区？

卫华：全国基本上大部分省份都有服务器，特别是一些比较核心的节点，比如北京、上海、广州，在这些核心的节点可能部署了更多的服务器，而在其它一些二线城市、其它省份也都有部署的。

InfoQ：您也是为这种大数据量做了充分的准备。最近大家都知道，玉树发生地震，对于这种突发事件，我们也会把微博作为一种信息交流、信息分享的平台，大家的访问也会造成大数据量访问，那么对于这种突发事件，您在技术架构上也做了相应的准备吗？

卫华：对，这种突发事件以及访问峰值，是微博上经常出现的现象。突发事件的访问峰值有两种，一种是可以预测的，比如说我们将来要搞的世界杯，比如春节，大家都相互拜年这种；另外一种是不可预测的，比如地震这种。对可以预测的这种，我们事先会做准备，比如说世界杯，我们要增加相关的服务器来完成。而面对这种不可预测的情况时，我们平时会有个数字，那就是我们平时的平均流量，硬件设备要比它高一定量，这样就能够应对这种峰值的请求。

另外从程序上来说，我们可能有一些专门的机制，比如说用户发表微博，并不是一发表就存到数据库中，简单地理解，他不是这样操作的。业界中微博之类的产品都有一种机制，叫做异步机制，也就是说，在发表的时候，我会把这个信息放到消息队列里面，然后再用另外一个专门的业务处理程序来处理它。当某一时刻发表量非常大，比如说地震了，很多人都会发表，那这个时候系统依然能够有条不紊的来处理这个业务，这样子就能让我们的系统稳定运行，并具有高可用性。

InfoQ：也就是要对整个事务的进行有效的控制？

卫华：对。

InfoQ：大家应该知道，因为有这么多的微博，有那么多名人，而且还有很多平民的、草根的微博，系统的数据量也是非常非常大的，而且还有很多很多的评论，很多很多的留言等等。那么对于这种海量存储，是不是也要做技术架构上的准备？

卫华：对，微博这个产品从技术上来说，有一个很大的特征，就是每天用户发表特别容易，这造成每天新增的数据量都是百万级的、上千万级的这样一个量。这样你经常要面对的一个

问题就是增加服务器，因为一般一台 mySQL 服务器，它可能支撑的规模也就是几千万，或者说复杂一点只有几百万，这样，你可能每天都要增加服务器，从而解决所你面对的这些问题。你要考虑，如果每天要加服务器，你的程序上、访问上会不会有问题，会不会间断。

我们其实有一些优化的方法，比如说我们会考虑热点数据和冷数据，如果经常要访问的这个数据，也就是热数据，而过几天才会访问的就是冷数据，我们会把它们合并，这样就可以按这个时间来分段，也就是把热数据放在一起，冷数据放在一起，这样可以解决这个访问热点的问题。

另外业界还有种思路，刚才说的用 MySQL，我们采用 Shade 的技术会按时间分片，这是一种解决思路；另外还有一种解决思路，业界特别现在国外流行的一种方法，也就是 NoSQL 的方法。有一种比较好的产品，现在大家比较关注，叫 Cassandra，就可以解决这个问题。如果我们每天要加一台服务器的话，那么我们程序、运维这些能不能跟上呢，是否有一种产品可以让你程序不需要做丝毫改动呢？Cassandra 这个产品就可以帮你来解决这个问题，你只需要把服务器插进去，那它马上可以使用，那个产品内部就有这样的机制。

InfoQ：那样的话对我们整个产品的维护就比较方便了？

卫华：对，这个可能就是说以后业界发展的一种方向，使用这种分布式的存储来解决这种海量增长的问题。

InfoQ：你觉得 **NoSQL** 的数据库和传统的关系型的数据库 那种更适合微博这种形式的网站？

卫华：从长远来说，NoSQL 这个更适合一些，特别是分布式的 NoSQL，刚才我也讲了，如果能全部下来的话，那可能经常要面对这种扩充的困扰，需要的干扰，可能是说，如果要保证服务不间断，可能就会面临一种很大的挑战，NoSQL，特别是这些分布式的 NoSQL 产品在内部就解决了这种问题，你不用停机，就可以加服务器，加设备。

InfoQ：这会对我们用户造成很大的方便？

卫华：对。

InfoQ：那么在性能方面，还有一种我们常采用的方式就是 **Cache** 的方式，那么在新浪微博系统里面，**Cache** 方式有什么样的特点？

卫华：在像微博这样的 Web2.0 产品里面，技术界有一种很重要的说法，Cache 就是 RAM，RAM 就是 Memory 的意思，RAM 也就是 New Disk，内存已经成为新的磁盘，代替磁盘的访问了。当我们大量使用 Cache 的时候，可能会存在很多问题，比如很多那种 Web2.0 的产品，它在 Cache 的数量已经不是 G 的概念了，不是几 G、4G、8G 的，可能达到一个 TB 的概念

了，一个 T 相当于 1024G，面对这样海量的数据，那我们访问的时候可能就会出现很多新的问题，比如我们的带宽，因为用户请求我的首页的时候，他会获取很多资源，比如有 50 个人关注你的微博，他需要从 Cache 里面把这 50 个人的数据都聚合起来，同时还会有很多人也在访问这个服务器，假如说，有一千个人访问，这一千个人里面，每个人都从五十个里面选，那么这个 Cache 的带宽将是一个比较大的问题，这是以前那种我们使用 Cache 时没有遇到过的。然后，为了解决这个带宽的问题，我们可以使用压缩的技术，我们保持 Cache 里面的数据，经过一种快速的压缩算法，比较传统的我们可以使用 GZip，那实际上在这种对时效性要求比较高的技术里面，我们是要求更快速的算法，比如说有一些 DOZO 算法，它对 CPU 消耗很小，但它压缩很快，效果也非常好。

另外的一个新问题，单点故障，我们非常依赖那个 Cache，假如某个时候它突然崩溃了，那么应用访问可能就会遇到很大的问题，也就是响应速度会出问题，为了解决这个问题，我推荐的做法是，使用一致性的哈希算法，就说送我一个业务，他可以用多个 Cache 服务器来完成，然后我们使用一致性的哈希算法，当一个 Cache 崩溃之后，它的请求就可以分散到其它的 Cache 来完成，总体的那个振荡不会太大，也就是说这个延迟会分散开来，让用户访问页面的时候感觉不到，实际上后台它可能有一台服务器，刚才经历一次 Crash，可能造成一次波动，经过我们这样改造之后，用户可能察觉不到这种变化。

InfoQ：用其它的服务器，同时来弥补这个地方的失误？

卫华：对，使用一致性的哈希算法，能够巧妙地达到这个目的。

InfoQ：您刚才提到了 **NoSQL**，另外在最近的业界还有一个流行的词就是 **Cloud**，云计算，我们是不是有计划以后会把微博系统推广到云平台上，或者说采用云计算的方式来处理呢？

卫华：没错，我们微博现在有一部分跟云计算结合比较密切，我们现在微博正打算推出一个开放平台，开放平台什么意思呢？就是说，第三方的开发者可以在我们上面写应用，可以连接到新浪微博，比如说可以获取信息，可以发表微博，而这些应用程序，可以放在我们的开发的另外一个服务上，叫新浪云。这个新浪云有什么好处呢？这些第三方开发的应用，可能他刚开发的时候，请求量不大，但有可能因为这个创意很好，忽然访问量大了。如果你用你自己的解决方案的话，可能就达不到这种要求。比如说最大的问题，可能就是全国访问不畅，或者访问量突然增长了，原来的服务器不够用，你要自己去加硬件，来不及处理。如果你放在那个新浪云上面的话，那我们系统自动会帮你解决这个问题，不管你的一个非常小的程序，比如一天只有几百个访问，还是一个海量的应用，我们都能够放在这个平台里面。在这个云应用里面，你不需要自己操心，系统自动会帮你把这个任务完成。另外它还有一个好处就是，这个云自动实现了全国分布，你只要 Host 在上面，全国的用户不管从哪里访问，他可以访

问一个就近的服务器，这在速度比自己部署都具有很大的优势。

InfoQ：那咱们新浪云现在已经正式推出来，还是正在计划中？

卫华：我们现在还处于测试阶段，我们采用一种邀请式，希望邀请更多的开发者来试用它，我们根据开发者的反馈来改善它，等到一定程度，我们再去大规模地推广。

InfoQ：以后对于大家来维护自己的微博、访问别人微博，是不是也更方便，不一定非要到各种各样的网页上，或者是手机等等，可以在自己开发的程序上就可以做这些事了，对吧？

卫华：对，以后结合这个微博的开放平台，结合新浪云，可以形成一个良好的生态圈，第三方的开发者要有一个很好的环境，给微博增加各种创意，增加各种应用。

InfoQ：这应该是对开发者带来的一个福音。

卫华：对。

InfoQ：感谢杨卫华先生接受我们的采访。谢谢！

原文链接：http://www.infoq.com/cn/articles/ywh-sina-mini-blog-arch

相关内容：

- 书摘和访谈：Open Source SOA
- Facebook 谈 Hadoop，Hive，HBase 和 A/B 测试
- 基于 Rails 的企业级应用剖析
- 黄晶谈人人网架构
- Joyent 首席代表张矩谈云计算

QCon

QCon全球企业开发大会（北京站）

大会分为十二个主题：

- 下一站：移动开发
- 技术热点是与非
- 可伸缩性架构设计
- 企业敏捷转型之路
- HTML 5开发平台
- 永不宕机的服务器
- 知名网站架构剖析
- 来自一线项目的经验
- 深入浅出NoSQL
- Web性能和扩展
- 设计优良的架构
- 更有效地做测试

专栏：代码之丑（七）--你的语言

作者 郑晔

这是一段用 C++编写的数据库访问代码：

```
int Comm::setIDBySevNum(const XString& servnumber) {
   DB db;
   db.setSQL("select id from users where servnumber=:servnumber");
   db.bind(":servnumber", servnumber.c_str());
   db.open();

   if  (!db.next()) {
      return -1;
   }
   setID(db.getString("id"));
   return 0;
}
```

它告诉我们，如果找不到需要的值，那么操作失败，返回-1，否则，返回 0，成功了。

显然，写下这段代码的人有着 C 语言的背景，因为在 C 语言里面，我们常常会用整数表示成功失败。我说过，这是一段 C++代码，而 C++里面有一种类型叫做 bool。

整数之所以能够占有本该属于布尔类型的舞台，很大程度上是受到 C 语言本身的限制。当然，C99 之后，C 程序员们终于有了属于自己的体面的布尔类型。

只是还有为数不少的 C 程序员依然生活在那个蛮荒年代。于是，很多人通过各种不尽如人意的方式模拟着布尔类型。不过，我们也看到了，偏偏就有这些生在福中不知福的程序员努力的重现着旧日时光。在我的职业生涯中，我见过许多用不同语法编写的 C 程序。

就个人学习语言经验而言，了解了基本的语法之后，如果有可能，我希望找到一本 Effective，寻求这门语言的编程之道。很多语言都有着自己的 Effective，比如《Effective C++》、《Effective Java》、《Effective C#》，等等。

不了解语言，也会给丑陋代码可乘之机。比如，下面这段 C++代码；

```
void CommCode::notifyCRM(XString* retparam) {
   if (NULL == retparam) {
     throw  IllegalArgumentsException(GetErrorMsg(" CommCode ::notifyCRM"));
   }
```

```
    ...
}
```

如果把指针换成引用，就可以省去参数为空的判断，因为在 C++里，引用不为空。这里选择了一个简单的例子，而在真实的代码里，这种检查漫天遍野，其丑陋可想而知。某些函数里面，检查甚至超过了真正的执行部分。

工欲善其事，必先利其器。有了铲子，就别再用手挖地了。

关于作者

郑晔，ThoughtWorks 公司咨询师，拥有多年企业级软件开发经验，热衷于探索各种程序设计语言在真实软件开发中所能发挥的威力，致力于探寻合理的 软件开发方式，加入 ThoughtWorks 公司后，投入到敏捷开发方法的实践之中，为其他公司提供敏捷开发方法方面的咨询服务。他的 blog 是梦想风暴。

原文链接：http://www.infoq.com/cn/news/2010/12/ugly-code-7

- 书评：《代码之道》
- 与 Patrick Smacchia 谈.NET 的代码分析
- 使用单实例类来处理对象元信息
- 基于 VS2010 的敏捷实践
- 代码之丑

阅读者开篇：软件架构师应该知道的 97 件事

作者 张逸

【编者按】《阅读者》专栏终于开张了。我惯常以蠹鱼自居，如今主持这个栏目，亦可以自封为“阅读者”了。

是的，我们是阅读者，不是书评家。一本好书不需要书评家的饶舌，阅读才是对它的最大尊重。之后呢？是束之高阁，藏于书山，还是不吝于分享与交流？真正的阅读者不屑于藏私，分享才是 Web 2.0 的王道。所以，有了豆瓣；现在，有了 InfoQ 的《阅读者》。

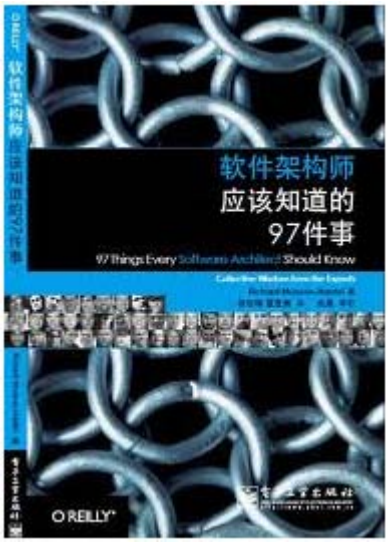
技术的阅读不同于文学的阅读。文学的阅读在于心灵的洗涤，以及那种追寻意境之美的沦浃肌髓；在于朦胧，在于玄妙，而所悟有所得。这表达了感性的一面。技术书籍的精要在于准确、简明、完整而实用，不需要引人遐想，更不需要堆砌朦胧之美。或许可以说，技术书籍传递的是钢铁的硬线条，展现了它的结构与质量，而不是闪动的金属辉光。技术书籍的阅读需要理性地分析，客观的评价。这正是《阅读者》专栏的最高原则。

软件业的特点是变化。若要提高软件开发的技能，就必须跟上技术发展的步伐。埋首醉心于项目开发与实战，固然能够锤炼自己的开发技巧，却难免受限于经验与学识。世界上并不存在速成的终南捷径，但阅读好的技术书籍，尤其是阅读大师们的经典著作，总能收到事半功倍之效。一位优秀的程序员，或许就是一名好的阅读者。好的阅读者，总是知道如何选择好的书籍。书海浩繁，良莠不齐。阅读技术好书，如与智者交谈，“与君一席话胜读十年书”；遭遇技术烂书，如被拐卖，“少小离家老大回，乡音无改鬓毛衰”。

《阅读者》专栏希望通过我们的阅读经验，去芜存菁。阅读者与好书是伙伴，与劣书是敌人；阅读者希望保持冷静的客观，不过偶尔也会表露个人色彩的爱憎分明。如何选择，最后还在于《阅读者》的阅读者。

本期《阅读者》介绍了一本小书。书虽然薄，却极有分量；内容不多，却有大智慧。

是否有程序员奢侈地想过，加入这样一个团队：那些世界上顶尖的架构师坐在你的身旁，与你结对设计；或者当你遭遇难题时，亲自为你指点迷津；或者在架构评审时，为你点评架构



的优劣。他们态度优雅，行动举止之间透露出 Geek 的风范；他们不厌其烦，展现出非凡的技艺。他们作为你的导师，言传身教传承着优良的软件工艺。这些顶尖的架构师或者声名显赫，或者技术超群。他们是 Neal Ford，Michael Nygard，Bill de hOra，Rebecca Parsons.....

我知道，你认为我在讲述天方夜谭。没有哪个程序员可以这样奢侈，可以这样幸运。

不错，这确乎是我的幻想。不可求，不可遇。然而，书却可以拥有这样的魔力，可以让这样的幻想得到实现。感谢 Richard Monson-Haefel 荟萃了这样一个团队，并将他们带到每一个读者身边，那就是这本薄薄的小书《软件架构师应该知道的 97 件事》。

我不想为这本书唱赞歌，然而阅读此书，确实让我受益匪浅。以下是我在阅读本书时摘要的笔记：

> 在大型软件项目中，关注根本复杂性，消除偶发复杂性，抽丝剥茧制订解决方案，才是真正的挑战。.....应该尽量选择源自实际项目的框架，警惕那些象牙塔里的产品；分析方案中有多少代码直接用来解决业务问题，有多少只是用来实现用户与应用的交互；谨慎使用软件厂商在幕后推动的方案，它们并非一无是处，但往往包含偶发复杂性；要量体裁衣，为问题制订“合身”的解决方案。
>
> 沟通必须简明清晰。没有人愿意阅读冗长的架构决策文档，架构师言简意赅地表达观点是项目成功的必要条件。项目启动之初，凡事能简则简，千万不要一头扎入冗长的 Word 文档里。可以借助工具，比如简单的 Visio 图表来表达你的想法，尽量画简单些，毕竟时过境迁，想法总会变化。非正式的白板会议是另一种有效的沟通手段，把开发人员召集起来，在白板上写下你的想法，比任何方法都来得有效。
>
> 假设有另外不同的团队打开了代码库，他们很容易便可了解到当前在做什么，这是优秀架构的基础。无需对架构进行过度的简化或为之准备面面俱到的记录文档；好的设计会以多种方式说明自身。

类似这样的架构箴言，在本书俯拾皆是。不要认为这些言论仅仅是泛泛而谈，仔细分析，你会发现其中蕴含的真理令人深省。不错，阅读本书无法让你成为一名优秀的架构师，毕竟本书并不是要传授架构的技巧，但它却能开拓你的视野，让你认识到软件架构以及架构师的诸多方面。正如本书的译者序写道：

> 全书由 97 篇格言式散文构成，没有高调的说教，没有抽象的术语，而是以平实、幽默、智慧的笔触，将他们认为对成为优秀软件架构师而言至为重要的精髓和盘托出。全书犹如一块玲珑剔透的水晶，97 个切面折射出来的都是出自一线软件架构师的专业智慧。

本书不是为初学者准备的，那些经验之谈对于初学者而言，略显隔靴搔痒；只有真正战斗在一线，并曾经为设计难题而头撞南墙，或者技术水平发展到了一个瓶颈期的架构师或者软件设计与开发人员，在阅读这些话语时，才会搔中他的痒处。就好比孙悟空听菩提祖师说法，手之舞之，足之蹈之，那是因为他听到妙处，体会了各种玄妙，所以才会喜不自胜，不觉作出踊跃之状。

原文链接： http://www.infoq.com/cn/news/2010/12/readers-begining

相关内容：

- 康伯谈淘宝双十一事件中的 CDN 优化
- 对业务架构的进行更合理抽象能成为企业架构的关键吗？
- ZapThink 探讨未来十年中企业 IT 的若干趋势
- 系统级复用的成功要素
- 书摘和访谈：《在企业中融合云计算和 SOA：循序渐进的指南》

LINQ to Z3，世界上最快的定理证明程序

作者 Jonathan Allen 译者 朱永光

微软研究院宣称，Z3 是世界上最快的定理证明程序。Z3 被设计作为其他应用程序的底层工具，它不适合单独使用。而嵌入到定理证明程序中的时候，在大量的项目中都有应用，包括 Spec#/Boogie、Pex、Yogi、 Vigilante、SLAM、F7、SAGE、VS3、FORMULA 和 HAVOC。

通过使用 Z3 的.NET 的 API，你可以使用面向对象的风格来编码定理。不过，在 Z3 练习指导中演示的一个非常小的问题，都用到了非常多的代码。Bart De Smet 编写了名为 LINQ to Z3 的包装器，把 OO 风格的 API 包装为查询风格的 API，极大地简化了 Z3 的使用。下面是 Bart De Smet 在做 Channel 9 访谈的时候演示的一个例子：

```
var theorem = from t in ctx.NewTheorem<Symbols<int, int, int, int, int>>()
   where t.X1 - t.X2 >= 1
   where t.X1 - t.X2 <= 3
   where t.X1 == (2 * t.X3) + t.X5
   where t.X3 == t.X5
   where t.X2 == 6 * t.X4
   select t;
 var solution = theorem.Solve();
 Console.WriteLine("X1 = {0}, X2 = {1}, X3 = {2}, X4 = {3}, X5 = {4}",
```

Z3 起初只提供了 Windows 平台的支持，不过现在也有 Linux 的版本。Z3 基于“微软研究院许可协议”发布，不能用于商业目的。你可以通过 Z3 on RiSE4Fun 来尝试一下在线版本。

原文链接：http://www.infoq.com/cn/news/2010/12/LINQ-Z3

相关内容：表达式即编译器、Aaron Erickson 谈论 LINQ 和 i4o、Jimmy Nilsson 谈 LINQ to SQL

使用 Brahma 在 GPU 上执行 LINQ、微软回答"关于数据的 10 个问题"

可伸缩系统的设计模式

作者 Jean-Jacques Dubray 译者 张龙

过去十年所取得的一个主要成就就是面向大众的可伸缩系统的广泛应用，尤其是云系统和某些高可伸缩的 Web 应用。比如说，Facebook 平均每秒可以处理 1300 万个请求，峰值达到了 450 M/s。即便如此，可伸缩系统背后的概念与架构仍然在快速发展着。大约 3 年前，来自加利福尼亚洲的软件架构师 Ricky Ho 曾撰写博文详细分析了可伸缩系统的现状。3 年后，他认为是时候重新谈谈这个话题了。

Ricky 将可伸缩性定义为：

> 可伸缩性解决的是在持续增长的性能、花费、维护代价以及众多其他因素的情况下如何降低系统的负面影响。

在其最新的博文中，他列举了如下模式：

- 负载平衡
- 分散与聚集
- 结果缓存
- 共享空间（又叫做 Blackboard）
- 管道与过滤
- Map Reduce
- 大块的同步并行
- 执行编排

如果说负载平衡、结果缓存和 Map Reduce 已经得到了广泛应用，那么某些模式现在正面临着社会化媒体所带来的新问题。比如说，上个世纪 80 年代所提出的大块同步并行现在就作

为 Google Pregel Graph Processing 项目的一部分，支持 3 种常见的处理模式：

> - 捕获（比如说 John 通过社交网络联系到了 Peter，那么在这两个 Person 结点间就会建立一个连接）
> - 查询（比如说找到 John 的朋友当中年龄小于 30 且已婚的那些朋友）
> - 挖掘（比如说找到硅谷中最有影响力的人）

Ricky 还介绍了执行编排模式：

> 该模型基于智能的调度者/编排者，用于跨越集群调度准备运行的任务（基于依赖图）。

他说该模式已经在微软的 Dryad 项目中得到了应用，程序员可以“使用成千上万台机器而无需了解并发编程”。

> Dryad 程序员会编写几个顺序程序，然后使用单向通道将其连接起来。计算是结构化的，以有向图的方式进行：程序是图形顶点，而通道则作为图的边。Dryad job 是个图形生成器，可以合成任意方向的无圈图。这些图甚至可以在执行期间改变来响应计算中的重要事件。

我们今天所使用的可伸缩性模式仅有 10 年的历史。接下来会有什么限制呢？你有构建可伸缩系统的经历么？忽略了哪些东西呢？

原文链接：http://www.infoq.com/cn/news/2010/12/scalable-design-patterns

相关内容：

- Oracle SOA 产品战略揭幕
- 使用 Spring AOP 和 AspectJ 编排工作流
- 在 ESB 中选择路由还是编配？
- 使用 JBoss ESB 和 JBPM 实现垂直市场解决方案（VMS）
- 组件组合的策略与技巧

Hudson 逃离 Oracle

作者 Alex Blewitt 译者 张龙

近日，开源的持续构建项目 Hudson 计划另开分支并更换名称，这缘起它与 Oracle 硬件套件之间在 java.net 上的各种摩擦。

在 Oracle 收购 Sun 之前，Sun 在 java.net 上管理着所有代码。但 java.net 在基础设施上存在着问题，有人则提议迁移到 Kenai 上。收购延缓了这种转换，但人们总是希望转换能够尽快进行。

> java.net 非常不可靠已经成为尽人皆知的事情了，这对 Hudson 的开发与使用造成了诸多问题——随着 java.net 的下线，开发人员将无法解决 这些问题，用户也没法下载 Hudson、插件也无法更新。在这之上，java.net 的问题系统烂到了极点——根本就没法用。

那时，问题追踪系统已经迁移到了 http://issues.hudson-ci.org/，而源代码则落后了。但正是由于 java.net 基础设施被锁定，我们没法进行开发，也没法在邮件列表上讨论，这反而加快了迁移的步伐。邮件列表迁移到了 Google 上，有人提议将代码迁移到 GitHub 上，这个提议最终被采纳，原因在于 GitHub 使用了 Git 这个流行且大家都很熟悉的 DVCS。

关于到底是谁推进了整个过程人们众说纷纭。但它与 Oracle 最近所采取的一系列变革有着千丝万缕的关系。他还提醒团队说 Oracle 现在拥有 Hudson 的名字，因此如果要对项目另开分支就必须得换个名字才行。这个线索上的诸多评论都强烈建议迁移到 GitHub 上，Hudson 已经将代码迁移到了 GitHub 上了，身份是 HudsonLabs。

> 你需要另开分支。你（Hudson 开发者社区）并不拥有自己的商标。今天，Hudson 还用作基础设施；明天你就没法使用 Hudson Barcamp 了，因为它与 Oracle 形成了竞争关系，如果制作 HudsonT 恤，你可能就会收到 Oracle 发来的法律意见书。
>
> 广大社区的言论让人觉得实在是太屈尊府就了。对于任何开源项目来说，被动用户、活跃用户、贡献者之间的比率应该是个递减的关系（比如 95:4:1），这已经成为一个标准了，但 Ted 告诉我们一个好消息。Oracle 并不是代表着那 95%的被动用户，很可

能比这个数字要小得多，因为活跃的 Hudson 社区洋溢着多样的风采。

无论结果如何，Oracle 再一次误判了开源社区，他可能已经着手准备彻底抛弃 Hudson 和它的开发者了。

原文链接：http://www.infoq.com/cn/news/2010/12/hudson-moves

相关内容：

- 豆瓣数据存储实践
- 使用面向本地移动 Web 应用的构建服务避免 SDK 紊乱：RhoHub 与 Apparat.io 简介
- 用 UML 做好系统分析
- 代码规范的自动化监管
- 企业级 Java 软件构建系统 EL4Ant

HTML5 Labs - 新技术原型实验场

作者 Abel Avram 译者 杨晨

微软决定不会在 IE9 的开发中加入实验型的早期 web 技术，这个角色由 HTML5 labs 来担任，这是一个测试例如 IndexedDB 和 WebSockets 例如 IndexedDB 和 WebSockets 这类技术原型的网站。

Christopher Blizzard，一个 Mozilla 的开源理念传播网站，宣布 WebSockets 的早期实现版本在 Firefox 4 Beta 8 中被禁用，这是由于一个协议层的安全问题隐患可能危及浏览器和 Internet 之间的通信。Adam Barth 解释了这样的溢出漏洞是如何生成的。Firefox 4 Beta 7 中搭载的是 WebSockets 草案 76 版的实现，Blizzard 提到了在这个安全隐患消除之后，Firefox 将会继续使用这项技术。

看起来 WebSockets 的问题并不是个例，这种情况发生在每一项不断变化的早期标准中，直到这些标准变得稳定。但是这种情况会使得开发者感到困惑并且不满，尤其是当他们认识到这项标准是如此的善变，而且也会考虑当前的实现版本也许并不会用在最终的标准实现中。为了避免这种问题，微软决定 IE9 中仅仅在包 含那些已经足够成熟足够稳定的 web 技术，而且创建了 HTML5 Labs 这样测试技术原型的网站。这次，微软不再将它在新标准上的工作完全地隐藏起来，而是让任何一名感兴趣的开发者都可以看到微软正在做些什么，目前为止进度如何，是否能够提供反馈并且影响浏览器的开发，Dean Hachamovitch，IE 小组的经理，解释了创建这样一个原型网站的原因：

> 在 IE9 中，开发者希望的是已经稳定的 HTML5 技术，这样他们不仅能够更好地利用已经成熟的 HTML5，而且能够在 HTML5 Labs 中使用早期的 HTML5 技术。将这些技术分开，开发者可以在同一浏览器中混合使用不同的东西，而不会产生副作用。

Hachamovitch 强调提供一个稳定的产品并且有一个渠道供开发者尝试新技术是多么的重要：

> 问题是这些正在构建的技术实现版本如何在开发者需求（他们不希望一遍又一遍地重写代码来获取新的功能）和客户需求（他们不希望站点和浏览器仅仅能够工作）中取得平衡。现在，iPhone 和 iPad 4.2 支持 WebSockets。而 Firefox 和 Opera 出于安

全和兼容性的考虑，禁用了这个实现。

现在，HTML5 Labs 包含了 IndexedDB 和 WebSockets。IndexeDB 现在仍是一个 Web 标准草案，这是一个面向仅存储简单数据和层级对象的数据库的 API，这种数据库实现上来看即是 key-value 数据库。IndexedDB 希望能够为 JavaScript 提供本地浏览器存储。另外一个原型是 WebSockets，这是一个浏览器和服务器之间提供双工通信的协议。这个协议旨在提供 HTTP 作为通信信道缺失的功能，以及一些需要替代的技术，例如 long polling（Comet）或者 AJAX。

如果需要使用原型技术，开发者需要手动下载并且安装它，可以通过注册 DLL 文件或者运行 MSI 文件。WebSockets 原型有一个简单的聊天 demo，能够在 IE9 Beta 和 Chrome 下面运行的很好，它支持在不同的标签和窗口中畅聊。

原文链接：http://www.infoq.com/cn/news/2010/12/HTML5-Labs

相关内容：

- Internet Explorer 9 Preview 3 提供了更好的 HTML 5 支持
- Google、Mozilla 和 Opera 质疑微软的 HTML 5 兼容性测试结果
- 微软声明将支持 HTML5 和 H.264 视频标准
- IE 9 最新的预览版本提升了性能，改善了对标准的兼容性
- Chrome 4 现已支持 HTML 5 Web SQL Database API

灰狐十年，中国开源稳步前进

作者 崔康

2000 年 12 月 28 日，Huihoo.com 上线。从此，灰狐（Huihoo）开始了它的互联网之旅。2010 年末，灰狐迎来了 10 岁生日。在献上衷心祝福的同时，我们也应该看到中国开源事业正在蓬勃发展。

Huihoo 是一个自由开放（Free&Open）的社区，也是一个自由开放的服务公 司，Huihoo 将所有的研究、开发、服务都放在与自由开放相关的技术和项目上。Huihoo 重点关注开源企业软件、开源客户端软件、开源软件分发三个主要发展方向，帮助企业和个人更好的使用自由和开源软件，提高生产力。

今年 5 月 19 日，灰狐动力和开源力量宣布合并，两位开源社区领袖龙辉和程旭文重新走在一起，推进中国的开源事业：

> 这（合并）是国人探索开源道路的一次自我救赎。
>
> 中国是开源的消费大国，相信没有程序员否认自己使用过开源软件。但是，长期以来， 国内的开源环境一直为人诟病。“只知道索取，不知道奉献”，“践踏开源版权”，“看不起国内的开源软件” 等现实让国人享受国外开源成果的时候不由得沉默。
>
> 国人一直信奉“洋为中用”，借鉴国外先进经验，这本无可厚非。互联网使得人们可以轻易的获取到国外的开源成果，于是沾沾自喜，以为捡到了大便宜，认为“老外真是太傻了”。但如若不遵守国际规则，实则损害了国人的信誉。
>
> 我们的 IT 进程的起步比国外晚，导致 IT 的创新中心都是在国外，如果习惯了跟随，并且心生惰性，这样一定会自信心不足，也就很容易理解“看不起国内的开源软件”的心态。这些问题无法回避，也不可能在短时间内解决。好在中国的开源环境逐渐成熟。君不见，每家 IT 公司里都有众多的开源高手，他们有着对开源的使用经验的见解，他们有激情、有创造力，往往是公司的技术骨干。君还不见，世界上几乎每一个开源产品，都有中国工程师的身影。另外，据统计，托管在 sourceforge 和 google code

> 上的由中国人主导的开源项目有好几千之多。
>
> 这些开源的力量如果能汇聚起来，将会产生巨大的社会价值。立足现实，开源人唯有以自己的实际行动付诸实践，不断前行。 灰狐动力与开源力量的结合，就是开源力量的一次汇集。

目前，灰狐社区维护的项目很多，包括 Open Download Manager、Open Browser、Open IM、Open Media Player、JFox 等，最近还相继推出了 Huihoo API 和基于微博开放平台和 Xweibo 的 Huihoo 微博系统。而开源力量维护的项目则更多，包括 EasyJWeb、微博爬虫 Sinawler、Openbravo 等。

对于十岁生日，灰狐在微博上对大家的支持表示了感谢：

> 灰狐十岁了。十年的风雨，在坚持里正迎向彩虹般的明天。我们相信坚持就是胜利，攻到底就升变。感谢所有朋友对我们的祝福，让灰狐在鼓舞中望见未来，感谢一路有你，我们将更加勇敢。

国内社区纷纷通过各种形式对灰狐十周岁表示了支持和祝福。

InfoQ 中文站创始人、总编辑霍泰稳：

> 十年在时间的长河里只是一瞬间，但是对于一个开源社区却意味着难得的坚持，愿灰狐社区继续努力，为国内的技术社区介绍和创作更多有价值的开源软件！十周年快乐。

W3China 创始人、InfoQ 中文站成员徐涵：

> 灰狐十年了，伴随着中国开源事业的启蒙。但革命尚未成功，同志仍须努力，中国软件需要灰狐这样的开源精神传道者与先行者。愿与灰狐共同见证“源”机降临的时刻！

为了庆祝十周岁生日，灰狐近期将会在上海、广州、成都组织一系列聚会，请灰狐的新老朋友关注。

在欢庆灰狐十周年的同时，我们应该看到，目前国内的开源事业日渐起色，或者说，开源正在得到越来越多国内企业的认可，包括对开源平台和开源技术的支持，以 InfoQ 中文站近期报道的社区开源新闻为佐证。

淘宝推出开源平台——淘蝌蚪

淘宝开源平台自 6 月底上线 以来，引起了国内社区的广泛关注。目前，平台已经发布了若

干开源项目，其中不乏来自于淘宝之外的项目在此落户。平台的负责人残剑（全佳营）在接受 InfoQ 中文站专访时，对开源事业表示了支持：

> 我们更希望聚集起开源爱好者，为国内技术领域提供一个良好的沟通交流平台，而淘宝的技术也将先开源，带头做好开源意识。
>
> 淘宝开源平台目前刚起步，未来我们会推出更多的功能，我们希望平台不仅是代码开源平台，更是开源思想的平台，有交流，有热情，体验开源的过程，同时享受开源的结果，有产出。所以，在未来的规划上，我们希望平台在技术支持上提供更多开发、测试工具，在沟通上，有更好的交流方式，能沉淀下很多很好的知识体系，同时，我们也会积极与国内的开源组织开展各种线下活动，拉近用户距离。
>
> 我们希望国内的开源能发展的更好些，有更多的人参与，有更好的平台支持，淘宝本身也会大力支持开源项目，如果有好的项目上来，通过淘宝技术委员会的审核，淘宝可以资助其团队开发，我们相信国内的开源也会发展的如火如荼。

淘宝在开源平台上先后推出了 Key/Value 结构数据存储系统——Tair 和自主研发的文件系统——TFS，InfoQ 中文站都做过深入报道。

天涯推出开源 **key-list** 类型内存数据引擎——**Memlink**

天涯社区最近开发了一款数据引擎——Memlink，并将其开源。天涯社区在北京研发中心的技术负责人冯勇先生在接受 InfoQ 中文站专访时指出开源是一种义务：

> （为什么开源？）一方面，我们很多工作都得益于已有的开源系统，所以回馈开源社区是我们应做的义务；另一方面，技术分享也有利于公司本身技术的成长，并吸引更多的技术人才。

数据级权限管理中间件 **Ralasafe** 开源

Ralasafe 是一款国产开源数据级权限管理中间件，使用 MIT 协议，项目负责人汪金保在接受 InfoQ 中文站专访时，谈到了开源的体会：

> Ralasafe 从 2004 年开始研发，2009 年正式向市场发布。当时我们使用闭源模式，商业推广。也做了几个单子，但我们也听到很多市场的声音：
>
> 1. 很多企业认为 Ralasafe 很好，但商业的软件，不愿意使用。
> 2. 有些企业尤其是大企业，像 Ralasafe 这样的安全软件，采取闭源模式不放心。
> 3. 很多开发人员呼吁我们，希望我们开源。

> 鉴于这些情况，我们团队慎重考虑后，决定开源免费。开源后，我们看到软件下载量不断增加，很多网友通过 email/社区/QQ/gtalk 等方式，与我们互动沟通。不仅找出我们软件的一些 BUG，也让我们学习到很多东西，让我们不断提高，软件品质不断提高。开源并不像我们原来所想，只是付出。开源更是双赢。在地域方面，我们已经推出中英文软件和中英文网站，为全球用户无偿服务。在商业方面，Ralasafe 的商业模式主要是赞助商和咨询定制服务。我们也期望中国开源软件，越来越多，更多软件走向国际，形成大气候。

我们欣喜的看到，中国的开源事业正在稳步前进，InfoQ 中文站一直坚定地支持国内社区的开源活动，将继续关注和报道开源新闻。

原文链接：http://www.infoq.com/cn/news/2010/12/huihoo-ten-years

相关内容：

- 与苏哲谈开源以及基于 Linux 的软件开发
- 2010 年 InfoQ 中文站开源技术总结：Made in China
- Node.js 开源项目获得 Joyent 资助，发展进入快车道
- Yahoo 推出开源 YUI 跨浏览器测试工具 Yeti
- 开源 HTML 解析工具包 jsoup 1.3.1 发布

腾讯微博开放平台正式上线，微博应用市场发力

作者 崔康

12 月 16 日，腾讯微博开放平台正式上线，它基于腾讯微博系统，为广大开发者和用户提供开放数据分享与传播平台。广大开发者和用户登录平台后，就可以使用平台提供的开放 API 接口，创建应用从微博系统获取信息，或将新的信息传播到整个微博系统中。而就在几天前，新浪微博创新基金刚刚开始接受申请，主要针对新浪微博应用的创业者。由此可以看出，国内微博应用市场逐渐展开角逐，竞争越发激烈。

据腾讯微博开放平台网站的介绍，开放平台是基于腾讯微博系统，定位于开放数据分享与传播平台。登录平台并创建应用后， 你就能通过从平台获取到的应用 App key 和 App Secret，使用各种 API 实现丰富的应用功能。应用凭借用户的帐号，访问其微博帐户进行内容读写，都需要在初次使用时，得到用户本人授权，授权系统使 用 Oauth 机制。得到用户授权后，应用就可以获取当前用户的用户名、头像图片、当前用户的听众和及收听列表等信息，并在用户操作后，完成微博信息的读写。

获得用户帐号授权后，使用平台提供的 API，可以创造出以下功能：

- 用户动态绑定并发送到微博将授权用户在网站/软件中新动态，通过应用即时发送到腾讯微博。使用微博平台提供的发送微博信息 API，可将用户在你的网站/软件上产生的新动态，生成一条微博信息，作为一条微博信息同步发送到腾讯微博网站中。
  - ✧ 对用户，能够向他的听众查看到自己最新的动态。
  - ✧ 对网站/软件，能通过微博信息中的链接，吸引用户的听众打开指定网页。
  - ✧ 对已通过来源字段审核的应用，产生的微博信息在腾讯微博网站，或其他应用中将展示来源，用户点击后可打开指定开发者网页。
- 快速分享内容到微博。腾讯微博中的用户关系，开放且有强烈的互动性，通过用户间的互动和传播，能实现信息几何级数传播。传播的内容也会在微博网站上展现，用户点击链接后可直接进入指定页面，起到为网站吸引流量的效果。

- 建立基于微博互动的用户关系。通过展现用户最新微博信息，并提供立即收听功能，帮助用户建立互动关系。平台提供的微博秀展现应用，让你通过几行 HTML 代码就能够在你的网站上展示自己的微博新动态。
- 获得微博信息作为内容。可获得整个微博平台最新微博信息，或指定用户发布/收听的微博信息，作为内容在你的网站/软件中展示。腾讯微博中活跃着不少名人和专业人才，在微博中发送最新最真实的消息，都能作为内容。

开发者在构建微博应用之后，可以向腾讯提交应用来源字段审核。提交应用审核并通过后，从应用发出的微博信息，在腾讯微博和其他应用中显示时，将带上应用的来源字段和来源网址信息，用户点击，就会打开指向的网站，为网站带来流量。

来源字段的审核内容包括：

1. 来源字段——希望从你的应用发布的微博，在腾讯微博及同其他微博中显示的来源名称“来自***”
2. 来源链接—— 希望用户从腾讯微博或其他微博中，点击后打开的网页地址

来源字段的审核提交标准：

1. 创建时间超过 15 天
2. 接口调用量累计超过 10000 次
3. 用户授权数量超过 100 人

腾讯微博开放平台还提供了一些官方微博应用，供开发者借鉴和利用，其中包括：

1. 一键转播——嵌入一键转播到你的网站里，访客便能将网页信息直接传播至腾讯微博。分享资讯的同时，用户通过来源链接可进入你的网站，从而提升访问流量。
2. 微博秀——使用微博秀，将生成的代码放置到你的博客、网站或是其它支持 html 代码的位置，就能向网页访问者展示你在腾讯微博的最新广播和听众。
3. 微博广播站——使用微博广播站，将生成的代码放置到你的博客、网站或其他支持 html 代码的位置，不仅可以展示你的最新 20 条广播，登录腾讯微博后，你还可以直接在这里写微博，让更多人了解和收听你。
4. 微博签名档——使用微博签名档，在你的博客、论坛签名、电子邮件签名或其他可以引用网络图片链接的位置使用，就能同步显示你最新的微博信息。

对于应用开发者最关心的开发平台 API，网站上列出了详细的说明：

1. API 文档——提供 API 接口 url、格式、http 请求方式等说明
2. Oauth 授权说明——介绍 Oauth 认证机制，及请求授权方式
3. API 调用权限——说明不同权限开发者可调用的 API 范围，及调用次数频率限制

其中值得关注的是开放平台对 API 读写频率的要求：

> 初级授权
>
> 读请求：每 API 单用户每小时 150 次（包括获取公共时间轴，首页时间轴等）
>
> 写请求：不可用
>
> 中级授权
>
> 读请求：每 API 单用户每小时 150 次
>
> 写请求：每用户每小时最大 100 次（包括发微博、转播、对话、私信、收听等）
>
> 腾讯合作方授权
>
> 读请求：每 API 单用户每小时 5000 次
>
> 写请求：每用户每小时最大 400 次（包括发微博、转播、对话、私信、收听等）

如果开发者认为 API 调用次数不够，那么腾讯对此做了解释：

> 首先微博 API 技术作为是一种 HTTP 轮询(POLLING) 协议 ,并非通过即时推送(realtime push)获取信息，为保证用户获取到最新微博信息，需要调用 API 定时读取，根据微博信息实际更新统计，我们建议将读取频率控制在 1 分钟以上/次，也可通过：
>
> - 智能频率控制，根据当前小时内剩余可用次数，灵活变更更新频率，同时需注意为用户手动操作留有 API 使用剩 余次数。
> - 提供手工刷新功能，根据用户需要手工读取新微博信息。
>
> 如用户量过大，当前 API 调用频率限制已不敷使用，可在你的应用数据达到相应标准后，通过邮件申请更高级别的调用权限，审核通过后，官方将进行调整，在应用详情页面可查看到应用当前 API 调用权限级别。

最近微博应用市场火爆。就在几天前，InfoQ 中文站刚刚报道新浪微博开发者创新基金向微

博应用创业者敞开大门：

> 规模为 2 亿元的中国微博开发者创新基金正式开始接受申请，为国内的中小型创业公司或者个人创业者提供了一个大展身手的平台……（基金）旨在为专注于新浪微博平台的第三方应用开发者提供创业投资，同时为开发者提供与创业相关的必要辅导，帮助开发者加速实现创业梦想。
>
> 基金投资项目单笔投资额以 300 万以下为主，但并不限制更高的投资额度。针对应用开发项目的要求是：已经在新浪微博开放平台上申请了 APP key 的开发者，应用本身已经可以为用户提供服务，即该应用已经被放置到微博应用列表中。

由此可以看出，随着微博的普及，互联网企业开始重视这块市场，鼓励第三方开发和部署微博应用，一方面可以提高自身微博的人气，另一方面可以和第三方分享微博市场预期的利润“蛋糕”。 不管是腾讯微博开放平台，还是新浪微博创新基金，都昭示着微博应用开发者们迎来了大发展的好时机。InfoQ 中文站也会持续关注相关动态。

原文链接：http://www.infoq.com/cn/news/2010/12/tencent-open-microblog

相关内容：

- Dino Chiesa 谈微软的 SOA 策略
- 企业级开发，PHP 准备好了吗？
- 沙龙记事：新浪微博架构师杨卫华分享微博应用开发技巧
- Silverlight 之轻

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

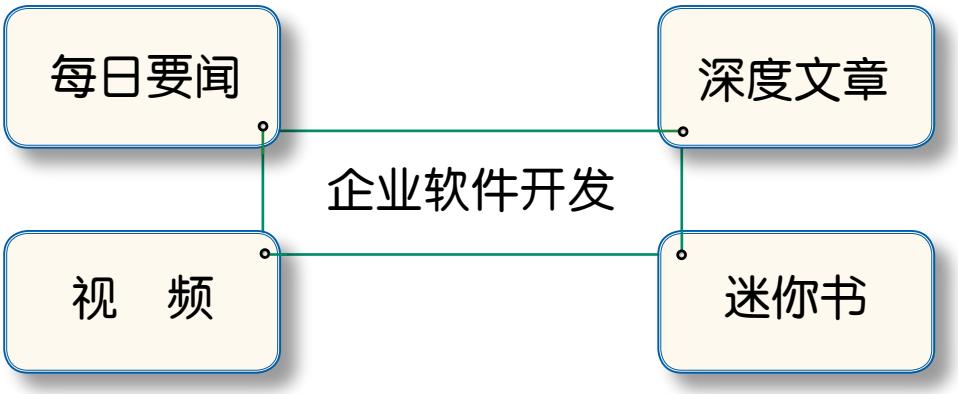
我们的**定位**：关注高级决策人员和大中型企业

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：霍泰稳　葛明见

一年将尽，繁忙之中又增加了浓浓的总结展望的气氛。一年以来，技术热点更新迭换，有的就此沉寂，有沉淀下来成为新的经典。我们本期架构师的特别专题邀请各个领域的的专家权威，根据各自的经历和看法为大家总结这一年来各自领域的变迁与改革。

2010 年年终技术盘点

年度技术回顾之数据库、NoSQL、开源软件 —开放平台回顾与前景展望

2011 互联网技术发展浅析—技术人员的产品观

特别专题

年度技术回顾之数据库、NoSQL、开源软件

作者 冯大辉

年终岁尾，做个总结吧。要说过去的这一年，起码国内的技术会议多了很多，甚至是几千块的门票也有市场了，可能也是物价上涨的副作用？像 QCon（Beijing）、SD 2.0 、微博开发者大会、TUP、UCD 年会、D2 年会、Velocity（Beijing）等会议，参会人都非常踊跃甚至有些会议一票难求，这是好现象，相信 2011 年有更多有价值的会议值得我们参加。再说说技术方面的事儿吧，下面是我的几个关注点。

数据库

Oracle RDBMS 、SQL Server 、DB2 等几大商业化产品似乎没什么值得一说的事件。Oracle 公司收购 Sun 之后，MySQL 前途曾一度堪忧，现在看起来 MySQL 生命力依旧顽强，只是在今年开发节奏明显慢了不少，也或许是 Oracle 在调整节奏，不过 5.5 版本的发布还是让不少 DBA 颇为惊喜，除了 InnoDB 成为默认的存储引擎之外，其他的一些特性倒是差不多都来自技术社区的反馈或是驱动，比如来自 Google、Facebook 的改进，多少对新的 MySQL 特性产生了一定影响。值得注意的是，这一年中 PostgreSQL 发展相当的迅猛，随着 9.0 的发布，引入了更为高级的复制技术，弥补了功能上的一个短板，MySQL 的命运多舛给 PostgreSQL 带来了契机，令人感慨。以前我期待的 SSD 虽说已经逐渐成熟，但似乎没有像预期的那样对数据库软件带来更大的影响。

NoSQL

在去年的回顾文章中我说到 "就数据管理方式的趋势来看，NoSQL 在将来会成为一个非常重要的数据解决方案"。一年之后，NoSQL 的确已经成为网络架构中一个基础的组成部分了。涌现出来的 NoSQL 相关的产品，最成功的要数 MongoDB，在新型 Startup 中颇为流行，赢得了不少创业技术团队的青睐（比如，引领创新潮流的 LBS 先驱 FourSquare 就是采用的 MongoDB，尽管为此吃了不小的亏），创建 MongoDB 的 10gen 技术团队甚至在年底拿到了

红杉的风险投资。除了 MongoDB 之外，Redis 的发展也不错。来自名门大厂的 Cassandra、Dynamo、CouchDB 等产品的发展倒是稍显平淡。作为 MySQL 的 NoSQL 插件出现的 HandlerSocket 的让人感到惊喜。这个技术方案会给很多应用场景带来新的契机，相信新的一年会有很多技术团队大胆的采用 HandlerSocket。其它几个 DB，似乎到现在仍没有类似的解决方案出现。

我有一个猜测是 Redis 从 VM 转向 Diskstore 模式后，有可能超越 MongoDB 么？

开源试水

Yahoo! 发布的 S4 不出意外的话，极有可能成为 Hadoop 那样有影响力的项目，对于实时计算领域会带来极大的冲击。相信今年国内会有用户进行尝试。LinkedIn 开源的 Kafka 也有必要关注一下。针对招聘类网站会有一定的借鉴意义。

2010 或许可以称之为中国互联网企业回馈开源领域的试水之年。先是淘宝网开源平台，淘蝌蚪（code.taobao.org）的上线并且推出分布式 Key-Value 存储及高性能缓存系统----TAIR，随后开放了淘宝文件系统以及 WebX 框架，足见诚意。說起 WebX，人人网也发布了自己的开源 Web 开发框架 Rose。然后有盛大创新院开源哼唱检索引擎，随后在互联网口水大战尘埃落定之后，金山的启动金山卫士开源计划，甚至百度也发布了 JavaScript 开发框架 Tangram --喊了一年终于开源了一个产品出来，颇为不易阿。而淘宝系的前端工程师们的开源项目 KISSY 发展也颇为迅猛，推荐关注。更早一些的开源项目，豆瓣的 BeansDB 在年底进行了大幅度更新，再次引起技术社区的注意。此外，射手播放器作者沈晟发布的基于 MongoDB 的短网址分支项目 SESO 也很有意思，希望能继续发展下去。基于 Key-Value 的开源产品多了不少，天涯也开放了一个 Memlink。

以团队为单位进行的产品开源，很容易变成一个只是"公开代码"的项目，开源，还应积极鼓励技术团队成员积极的与技术社区互动，输出更多文档，用更多的案例支撑，这样才能相辅相成，才能取得真正的收益。否则的话，容易被看成为了开源这个"名"而开源，有始无终。

期待在 2011 年 ,腾讯能在开源领域做点表率？还是网易开源一个游戏引擎呢？只有拭目以待了。也期待国内互联网企业能积极支持开源社区，不要只顾着开源自己的那几个产品。开源比封闭更值得欣赏，心态也比姿态更为重要。

说到开源，顺便说一下"开放平台"，2009 年喊着做开放平台的各大网站，现在已基本偃旗息鼓 ,国内这一年中也没有一家将所谓的"开放平台"真正的做起来 ,倒是经过一年多的铺垫，新浪以微博为基础的的应用平台已经具备了一定的潜力和规模，2011 年值得期待。如果说

开源，看的是心态，那么，开放平台，则看的是企业的心胸。

2011 做点什么？

眼看着越来越多的解决方案，越来越开放的技术分享，不由得让人生疑：架构是否已经不再重要？其实，构建一般中到大型的站点，已经没什么秘密技术可言（比如，还有人一度放出来"腾讯大讲堂"这样的内部信息资料，颇为戏剧性，但大家看了之后也就是新鲜几天而已，网络中更有价值的信息已经是比比皆是了）。重要的是如何用成熟的技术将产品做好，加快开发节奏，更快改进产品质量。

所以，对我自己而言，新的一年重要的还是回归基本技术，和团队一起将丁香园（http://dxy.com）的产品做好，"望着天上的星，也要看着脚下的坑"，关注新东西，更要避免因为技术冒进造成不必要的人力物力浪费，说起来容易，真的做起来，怕是也没那么简单。

关于作者

冯大辉，现任丁香园（http://www.dxy.com）网站 CTO。此前曾在阿里巴巴集团工作 5 年，历任支付宝首席 DBA、数据架构师等职，曾为支付宝的技术发展做出过重要贡献，是支付宝技术发展的见证人之一。知名技术 Blogger（http://dbanotes.net），狂热的 Twitter 用户，网名 Fenng。

相关内容：

- 图形数据库、NOSQL 和 Neo4j
- 走近淘宝开源平台
- Apache Nuvem 将带来更多的开源云？
- 与冯大辉谈数据库架构
- 一个技术观察者的年度展望

特别专题

开放平台回顾与前景展望

作者 岑文初

由于时间比较仓促，写的有些凌乱，有兴趣的同学可以更多的线下沟通。

回顾

淘宝的开放算上 2010 年已经走了快三个年头了，从服务提供者的角色转变为开放平台的角色，从 30 多个服务到 300 多个服务，从 2000w 的日调用量到 8 亿的日调用量，在技术和产品上都在经历着不断地蜕变。

开放平台产品的几个关键词：快，稳，安全，易用，透明。这些关键词构成了开放平台第一阶段的产品需求。

快：速度是服务调用的第一感知。在一次服务调用过程中，最多拆分的步骤：DNS 解析，TCP 连接的建立，数据传输（包括上下行），数据解析，平台安全校验，后台业务处理。因此如何优化这些步骤就成为提速的关键，优化有几种思路：

1) 直接优化处理过程，减少时间消耗。

2) 间接优化处理过程，提高处理过程中资源利用率，增加并发处理效率。

3) 合并处理过程，减少重复处理。

4) 处理逻辑外移，合理利用外部计算能力。

这里分别举几个例子来说明一下上述的概念：

1）在年初时一直采用 apache+jboss 的结构 ,但是发现在数据解析部分用容器处理会比较慢，其实是因为外部 http 请求数据传输质量有好有坏，当大数据量请求服务到来时，容器需要多次 IO 很多小数据包后才能够继续后面的业务处理，本身过多的 IO 处理直接增加了数据解析阶段的时间，也增加了整体业务处理时间。通过采用 nginx+jetty NIO 的模式，利用 nginx 的数据堆积转发可以大大降低后端容器在数据解析上的消耗时间（nginx 消耗

也不大），另一方面 Jetty 的 NIO 通过修改仿 BIO 的模式也能够提高数据处理性能。

2）对于开放平台来说，由于存在大量的上行大数据量服务的存在，因此上行带宽其实也是比较宝贵的资源，同时也是整体业务处理快慢的一种约束，如何提高带宽的利用率，通过 Lazy 的方式解析请求字节流，按需读取数据，遇到业务判断不满足立刻结束请求返回错误，可以大大的降低对无用的大数据量请求的解析成本，从而提高整体的带宽利用率，提高整体的请求处理能力。

3）批量服务处理模式及 QL 的实现能够极大地提高多次无关性请求或者串联请求的处理效率，其实在 Facebook,twitter 都有现成的样板，但到了淘宝这种电子商务复杂度高，安全性高的体系中，有很多细节需要去关注。

4）平台各种访问控制策略计算都外移到外部分析集群中，虽然牺牲了部分的即时性，但是减少了平台计算成本（特别是全局性的计算需要依赖外部 kv 系统来统一控制），同时策略可灵活运行期配置，业务相关多变的内容被移出，防止经常发布导致的不稳定性。另外对于后端服务结果的统一格式化操作也交由各个业务提供者自己处理（以类库方式依赖平台提供的格式化工具包，但计算却分散在各个应用提供者集群而不是开放平台集群）。

稳：稳定是服务最基本的需求。如何做到稳定主要从几个不稳定的角度去看：外部应用的不稳定，平台依赖的不稳定，后端服务提供者的不稳定。外部应用常常会由于自己的异常发布或者爬虫等其他原因突然大量的调用平台，此时将会给平台和后端服务造成很大的威胁。因此需要对应用有访问频率控制，访问频率控制粒度最小到服务级别，控制目标可以是应用级别，用户&应用级别，应用&IP 级别，甚至是应用&Agent 级别。对于平台依赖的不稳定，当前通过平台整体的管道化作降级实现，整体流程被拆分成很多管道，多个管道之间业务关联不强，当其中一个管道依赖资源出现问题时，可以在主流程中“卸载”该管道，同时告警，当异常处理完毕后，可以再将管道人为增加上去。（当然如果这个管道是关键不可缺的情况，就无法简单降级处理）。后端服务提供者的能力及稳定性都参差不齐，但是由于同样接入在开放平台上，加之传统容器阻塞模式的处理业务请求，使得后端某一个服务处理能力出现问题时，将会消耗大量的开放平台前端容器线程，最终影响其他后端正常的请求无法进入被处理。因此，采用 Jetty 的类似于 Servlet3 的异步化请求处理方式，能够将容器线程池与业务线程池分离，同时业务线程池可以根据业务规则和后端服务处理能力来选择是否处理请求，还是排队等候处理或者直接丢弃请求，最终实现服务隔离和服务的差异化处理。

安全：是开放双刃剑最危险的一面。安全主要有三方面的考虑：用户信息安全，淘宝数据安全，ISV 身份安全。用户信息安全在很多开放平台上都采用 OAuth 的方式，而淘宝开放平台

则简化了 OAuth 的模式，便于开发者简单开发。但还是有些问题需要进一步去优化和解决，例如手机端和桌面端的授权无法走 OAuth 的回调返回结果的模式，因此现在采用应用轮询的方式和手工拷贝授权码的解决方案，这需要有更好的优化措施（OAuth2 的解决方案中有些可以尝试）。另一方面，淘宝电子商务体系的差异性，使得授权可能涉及到两方甚至多方（例如交易服务，团购服务），因此用户授权策略也不能简单的使用 OAuth 的方式来解决。淘宝数据安全性主要体现在对于服务调用的监控，通过服务调用分析来判断服务调用的合法性（应用授权情况及服务调用情况比对等），同时通过不同的 ISV 体系来限制服务的访问范围和访问频率，间接地控制数据非法获取的问题。ISV 身份安全主要是考虑高级别的 ISV 的访问权限很高，APPKey 和 Secretcode 被盗用的风险很大，因此通过 IP 控制和动态密钥的方式来保证高级别应用的身份安全性。

易用：与人方便自己方便。在开放平台团队中易用也是大家最关注的一个产品特性。如何降低服务升级对于外部开发者的影响，如何降低后端服务发布的接入成本，成为开放平台工具化的最初需求驱动。开放平台升级一次服务，影响之广，周期之长是有目共睹的（twitter 的 auth 升级，foursquare 服务升级），降低服务升级影响最有效的方式就是提供被广泛使用的各个语言版本的 SDK，同时数据类服务的 SDK 是最细粒度的 SDK，后续的流程化组合化的 SDK 将会根据领域的不同而不同，更加便于开发者开发，一方面简化了开发者的开发成本，另一方面升级过程往往可以在发布新的 SDK 中顺利得到过度。（顺带对于应用端到服务端真实的业务请求处理时间和成功率统计也依赖于 SDK）。另一方面，从开放平台初期的手把手接入开放平台，手工编写服务定义和返回结果格式文件，手动编写 SDK 和发布 wiki 文档，到现在填写表单，自动创建定义文件和结果格式文件，自动生成多版本 SDK 和上线 wiki 文档，都为服务提供者减轻了发布服务的负担，同时也减少由于人为失误导致的问题。其实回过头来看，不管改善了那一方的体验，最终节省了开放平台的支持资源，也提高了服务质量。而 SDK 的垂直化封装，改善用户体验将会是明年的一个重点，因为淘宝的服务和普通的 SNS 及地理等服务不同，有很强的业务性和规则性，因此单独的数据类服务对于开发者来说门槛太高，需要根据不同领域的应用有不同封装和指导。（商城，大卖家，买家营销，江湖，淘客，行业分析等等）

透明：就像一切归于尘土一样，上述一切归于系统透明。不论是开放平台还是其他什么系统，最终要的是系统数据透明化。快的优化基于优化前对于系统瓶颈的查找和优化后系统处理能力的对比，这些需要数据来说明。稳定的基础需要海量的数据加上业务规则做决策，在问题出现前防范于未然，在问题出现后最小代价快速解决问题。安全的深度挖掘需要对数据作关联分析，最终找到不合理性的服务调用。易用也需要根据数据作指导，确定易用不是平台设计人员的易用，而是开发者在乎的易用。

在开放平台体系中，针对数据量在 T 级以下，分析业务规则复杂多变，即时性要求高的前提下，采用松散的分布式处理协作结构，抽象统计分析模型作为分析引擎，满足上述的业务需求，具体的结构如下：

具体的文档还在编写中，一些集成测试还需要做的更好一些，后续考虑开放出来更多人可以参与贡献（有太多可以优化的地方），代码可以直接在 google code（http://code.google.com/p/cheep-worker/）上 check out 下来：

```
http://cheep-worker.googlecode.com/svn/trunk/ cheep-worker-read-only
```

技术展望

外部通知服务（类似于 twitter 的 streaming 服务）的推广，推送数据变更通知到应用，将采用 comet streaming 模式来即时推送，节省短链接带来的性能损耗（网络连接消耗和对方服务能力不确定性，改主动多次连接为被动长连接），同时简化消息推送到达率策略。

今年更多的网站开始开放服务，因此会考虑在平台间为了合作而作部分的兼容，例如在授权模式上除了使用简化的 OAuth 以外，也兼容 OAuth1 和 OAuth2（这部分将可能成为数据类服务的标准，也就是安全级别较高的服务将会迁移到这种模式下）。

QL 和 Batch api 的模式将会考虑从内部走向外部，QL 和 Batch api 的在内部垂直化开发过程中得到业务的反馈不断完善后，考虑将这部分机制堆外，提高开发者开发效率，同时也为将来可能的外部服务简单的 Mashup 提供技术上的可行。

Hosting 服务的提供及应用行为监控。应用的用户增长不确定性需要有弹性的后端提供服务能力扩展支持，因此 Hosting 将会成为应用发展的很重要一环，同时在 hosting 上增加应用发布控制和应用数据访问控制，可以更好的保证数据安全性和应用稳定性的监控。

如何将业务开放不再独立于后台系统，实现开放更为直接，将成为很多从已有系统开放的网

站较大的挑战，前期出于快速响应的需求，采用独立开放部门去封装开放服务的方式，而到了初中期由于业务变更的复杂性，更希望能够由服务提供者直接开放服务，内部服务调用体系和外部服务调用体系统一（facebook 这样的网站内部服务体系和外部保持一致化，极大降低了服务内外两套系统的磨合成本）。

在海量请求和大数据量上下行请求的场景下（特别是在网络数据传输不好的情况下或者在手机终端的应用），如何保持快速，稳定将会成为底层系统的最大挑战，一则优化现有 Web 容器（Jetty NIO 和事件驱动模型），二则重新采用新的实现方式简化 Web 容器实现（API 请求对于 Http 协议中的很多规范实现要求不是很高）。

关于作者

岑文初（花名：放翁），2006 年加入阿里巴巴，2007 年初开始负责阿里软件平台架构设计，2007 年底开始进入开放平台领域，2009 年 8 月加入淘宝开放平台，现在是淘宝开放平台主架构师.个人没有什么特别擅长的，只是能够在学习情况下较快进入角色。

相关内容：

- 互联网开放平台技术趋势和讨论
- 开放平台技术架构剖析
- 腾讯微博开放平台正式上线，微博应用市场开始发力
- 基于 SAE 的微博应用开发
- 虚拟座谈会：Oracle 和 Java 的发展

特别专题

2011 互联网技术发展浅析

作者 杨卫华

编程语言

由于 Apple 的魅力，Objective-C 获得了飞速发展。其他主流语言变化基本不大。
从 5 月的 Google I/O 大会来看，Go 语言在 Google 内部得到不少应用，但是社区关注点成功案例，在 2010 尚未明显突破。

函数式编程语言在分布式及互联网领域依然非常受重视，但是 Haskell, Erlang, Scala 等语言都缺少一个契机走向主流。

在 TIOBE 排行中，Java 依旧是第一语言，但是由于 Oracle 收购 Sun 及 Oracle 与 Google 的 Android 官司事件给 Java 发展蒙上一层阴影。在年底，IBM 和 Apple 都先后加入了 OpenJDK 项目，统一化了后续 Java 的方向。

数据及存储

根据国外知名技术站点 HackerNews 上半年前的一个投票"初创公司用什么数据库"，在 1044 个结果中，排行前 4 位是 MySQL 433,PostgreSQL 249, MongoDB 138, Redis59。

从中看到 MongoDB 及 Redis 取得了众多初创公司的青睐。其中推荐关注 Redis，在大量的 benchmark 测试中 Redis 基本战胜了 Memcached。Redis 是什么？如果你认为 Redis 是一个 keyvalue store，那可能会用它来代替 MySQL；如果认为它是一个可以持久化的 cache,可能只是它保存一些频繁访问的临时数据。Redis 是 REmote DictionaryServer 的缩写，在 Redis 在官方网站的的副标题是 A persistent key-value database withbuilt-in net interface written in ANSI-C for Posix systems，这个定义偏向 keyvalue store。还有一些看法则认为 Redis 是一个 memorydatabase，因为它的高性能都是基于内存操作的基础。另外一些人则认为 Redis 是一个 data structureserver，因为 Redis 支持复杂的数据

特性，比如 List, Set 等。对 Redis 的作用的不同解读决定了对 Redis 的使用方式。

在分布式存储领域，在 2010 年，Cassandra 在年初的火爆没有持久，下半年 Twitter 暂停在主业务后 Cassandra 逐渐在业界淡出。到年底时，Facebook 新的统一通讯产品突然宣布使用 HBase，随后其他一些产品如淘宝的一淘也宣称使用了 HBase，因此建议大型存储尤其是对 Hadoop 已有技术投入的公司可更投入适当力量研究 HBase。

平台及应用

随着云计算及开放平台的发展，软件开发模式已经发生了很大的变化，传统的信息系统需要走向开放及社交平台化，需要连接 Amazon、Facebook 等平台。更多新的软件机会在 Facebook，AppStore 等社交及移动平台浮现。在平台上开发软件和传统方式有很大差异，需具备快速开发能力，以及产品上线后应对用户急速增长的压力。很多初创公司并不具备大规模服务系统开发经验，因此使用云存储及云计算是在平台上开发应用最好的选择。在国外 Amazon 等厂商的服务已经非常成熟，从新开发的应用到上百万用户的系统都可以使用。初创项目尽量利用已有资源，切忌一切从头开发。

技术动向

年初在北京举办了 QCon 2010 大会, Facebook 及 Twitter 都分享了相关技术架构，从中可以学习到大型 Web2.0 系统的架构设计经验，到年底以关注 Web 性能为中心的 Velocity 大会也来到了北京，Facebook 分享的 BigPipe 等前端加速技术相信又会在业界带来前端优化的新思路。Web 及前端开发不再是以 Web 页面开发为主，前端脚本优化为辅的思路，使用前端框架为中心驱动 Web 页面开发的思路才能满足动态应用速度及用户体验的要求。

关于作者

杨卫华，新浪微博架构师，对互联网后端技术及大规模分布式系统架构有浓厚兴趣，经常通过微博及博客发表技术观点。

相关内容：Goat Rodeo：面向 Web 应用的统一数据模型、GigaOm 结构会议关注新兴的数据架构、Facebook 谈 Hadoop，Hive，HBase 和 A/B 测试、JavaFX 的未来在哪里？、Sequel：Ruby 的数据库工具包

特别专题

技术人员的产品观——暨 2011 年展望

作者 潘少宁

我勒个去，时间转瞬即逝，已悄然步入不知会是杯具或洗具的 2011 年。神马都是浮云，2010 已经过去。让我们把更多的精力放到 2011，顺便迎接 2012。言归正传，不瞎扯了。今天想与大家分享和讨论下"技术人员的产品观"。

曾经，在我的周围经常发生着这样一幕：

> “这里用户体验不好，麻烦修改下。“P 说。
>
> “这里也不不好，麻烦修改下。” P 说。
>
> “还有这里...这里...这里...这里.....” P 说。
>
> 产品同事总有一堆堆的问题找我们修改。头都爆炸了。

这只是一个例子，实际中，还有很多。对于这一幕，我们不用纠结产品同事是否如何，我们站在技术人员的角度来看看这个问题。

在我们开发产品的过程中，我们是否考虑过产品是否易用呢？是否考虑过用户群是谁呢？是否考虑过产品目标是什么呢？是否考虑过产品的商业模式？是否考虑过产品的运营？是否考虑过产品的优化？是否考虑过产品数据的深度挖掘？...

站在肯定的立场，我们如何思考和实施呢？

首先我们来看看做产品前，都要考虑的几个问题。

(一) 产品定位

我们要做什么产品？核心竞争力是什么？

(二) 行业分析

该行业有没有比较成功的案例？是否有竞争对手？竞争对手的商业模式是怎样的？

(三) 用户分析

我们的用户群都有哪些？分别都有什么特性？

(四) 产品分析

产品有什么核心功能？分别可以满足用户的哪些核心需求？产品有哪些内容？产品的哪些内容最能吸引用户？

(五) 商业模式

产品如何盈利？如何长远发展？

在这五个方面之前，还要考虑用户需求调研，可行性分析等。

我们再来看下，产品开发过程中，主要都关注些什么？

(一) 交互设计

产品所要传达的信息，如何能更好的传递给用户？不同的用户群是否要需要不同的展现和交互方式？

(二) 用户体验

如何给用户以最好的 feeling？

当然，在开发过程中，还会关注开发过程，测试等等。

最后我们来看下，开发完成后，都关注些什么呢？

(一) 产品推广

产品如何推广？广告？联盟？微博？SEO ……

(二) 产品运营

产品如何运营？如何拉来用户？如何留住用户……

(三) 数据挖掘

运营数据需要监控哪些？如何分析？根据分析结果如何优化产品……

在这些过程中，我们技术人员，需要关注和考虑什么呢？如果你全部关注，最好不过了。

为什么需要关注这些呢？

技术固然重要，但体现技术价值的，将技术转化为生产力的产品也同样重要。在关注技术的同时，关注产品，创造更好的产品，创造更高的价值。

Google wave 、google TV 从技术角度而言，很给力。但从产品角度而言，却 Ungelivable，最终走向失败。

不要把我们的视野仅仅停留在 coding，不要两耳不闻窗外事，一心只做 coding 活。

腾讯 CEO 马化腾，是个技术达人，但也非常关注产品。几乎每一款产品出来第一个体验的就是他。

百度 CEO 李彦宏，也是个技术牛人。其也非常之关注产品。百度的诸多产品也都经历其手。

关注这些有什么用呢？

也有很多技术人员在迷茫，难道要 coding 到七老八十？于是大家都在讨论发展方向和转型的问题。至于具体转型到哪个方向？是否要转型，暂不做讨论。但有一点是要提及的，就是可转型方向之一便是产品。

当你站在用户使用的角度去开发产品，那必定是较为好用的产品；当你站在老板的角度开发产品，那必定是较为成功的产品。

再说的实在一些，关注产品，对于技术人员的职业素质的提高，大有裨益。关注产品，来年的加薪可能就是你，来年的升职的可能就是你。

2011 新年伊始，抬起头看看，做个不是非常艰难的决定，关注产品，培养你的产品观，拓展你的职业发展之路，成就你的职业旅程。

关于作者

潘少宁，腾讯网高级软件工程师。

相关内容：

- 品牌推广应用设计的最佳实践、布局和展现相关的建议——给用户体验设计者
- 画好 Web 流程图、Richard Durnall 谈系统管理和从外向内的组织结构
- 王坚谈如何设计成功的互联网产品

开始报名啦！

Adobe在线课堂第二期

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。

马鉴：Adobe平台技术推广经理

99年开始接触 Flash，2004年加入 Macromedia 公司任职解决方案工程师，2006至2008年，在 Adobe 公司担任过解决方案工程师和大客户技术经理，目前是 Adobe 平台技术传教士。平时最喜欢研究有关于 Flash 平台技术的新应用，新发现和新创意，同时也喜欢学习其他 Web 前端技术。

课程：全新Flash 3D APIs

3D开发者们已经从Flash播放器5.0或6.0的时代就开始构造一些伪3D的动画效果，开发者和设计师们一直都在不停的创新以挑战Flash播放器的功能极限，而Adobe也将持续维持功能上的更多革命以保证这种创新的趋势。Adobe将为Flash播放器引入一个全新的内置3D编程接口来增强对3D开发者的支持，这个全新的3D编程接口允许开发者们构建真正基于硬件加速的3D绘图功能上的Z-Buffer，Shaders和更多特效应用。这个演讲将在Flash 3D编程接口于明年早些时候被Adobe正

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 朱建华

分享知识！
相聚Adobe在线课堂！

百万级访问量网站的技术准备工作

作者 刘志一

当今从纯网站技术上来说，因为开源模式的发展，现在建一个小网站已经很简单也很便宜，所以很多人都把创业方向定位在互联网应用。这些人里大多数不是很懂技术，或者不是那么精通，而网站开发维护方面的知识又很分散，学习成本太高，所以这篇文章将这些知识点结合起来，系统的来说，一个从日几千访问的小小网站，到日访问一两百万的小网站，中间可能会产生什么问题，以及怎么才能在一开始做足工作尽量避免这些问题。

你的网站因为努力经营，访问量逐渐升高，在升高的过程中，问题也可能开始显现了。因为带宽的增加、硬件的扩展、人员的扩张所带来的成本提高是显而易见的，而还有相当大的一部分成本是因为代码重构、架构重构，甚至底层开发语言更换引起的，最坏的情况就是数据丢失，所有努力付之一炬。这类成本支出大多数在一开始就可以避免，先打好基础，往后可以省很多精力，少操很多心。

对于不同的初期投资成本，技术路线的选择是不同的。这里假设网站刚刚只是一个构想，计划第一年服务器硬件带宽投入 5 万左右。对于这个资金额度，有很多种方案可选择，例如租用虚拟主机、租用单独服务器，或者流行的私有云，或者托管服务器。前两种选择，网站发展到一定规模时需迁移，那时再重做规划显然影响更大。服务器托管因为配置自主、能完全掌握控制权，所以有一定规模的网站基本都是这种模式。采用自己托管服务器的网站，一开始要注意以下几点——

一、开发语言

一般来说，技术人员（程序员）都是根据自己技术背景选择自己最熟悉的语言，不过不可能永远是一个人写程序，所以在语言的选择上还要是要费些心思。首先明确 一点，无论用什么语言，最终代码质量是看管理，因此我们从前期开发成本分析。现在国内流行的适用于网站的语言，大概有 java、php、.net、 python、ruby 这五大阵营。python 和 ruby 因为在国内流行的比较晚，现在人员还是相对难招一些。.net 平台的人相对多，但是到后期需要 解决性能问题时，对人员技能的要求比较高。剩余的 java、php 用人可以说是最多的。java 和 php

无法从语言层面做比较，但对于初期，应用几乎都是 靠前端支撑的网站来说，php 入门简单、编写快速，优势相对大一点。至于后端例如行为分析、银行接口、异步消息处理等，等真正需要时，就要根据不同业务需 求来选择不同语言了。

二、代码版本管理

稍微有点规模的网站就需要使用代码版本管理了。代码版本管理两点最大的好处，一是方便协同工作，二是有历史记录可查询比较。代码版本管理软件有很多，vss/cvs/svn/hg 等，目前国内都比较流行，其中 svn 的普及度还是很高的。

假设选了 svn，那么有几点考虑。一是采用什么树结构。初期可能只有一条主干，往后就需要建立分支，例如一条开发分支，一条上线分支，再往后，可能要每个小组一个分支。建议一开始人少时选择两条分支，开发和线上，每个功能本地测试无误后提交到开发分支，最后统一测试，可以上线时合并到上线分支。如果每人都建自己的分支，合并时会浪费很大精力，对于几乎每天都要修改几次的 WEB 应用来说，所费时间太多。

向服务器部署代码，可以手工部署也可以自动部署。手工部署相对简单，一般可直接在服务器上 svn update，或者找个新目录 svn checkout，再把 web root 给 ln -s 过去。应用越复杂，部署越复杂，没有什么统一标准，只是别再用 ftp 上传那种形式，一是上传时文件引用不一致错误率增加，二是很容易出现开发人员的版本跟线上版本不一致，导致本来想改个错字结果变成回滚。如果有多台服务器还是建议自动部署，更换代码的机器从当前服务池中临时撤出，更新完毕后再重新加入。

三、服务器硬件

在各个机房里，靠一台服务器孤独支撑的网站数不清，但如果资金稍微充足，建议至少三台的标准配置，分别用作 web 处理、数据库、备份。web 服务器至少要 8G 内存，双 sata raid1，如果经济稍微宽松，或静态文件或图片多，则 15k sas raid10。数据库至少 16G 内存，15k sas raid 10。备份服务器最好跟数据库服务器同等配置。硬件可以上整套品牌，也可以兼容机，也可以半品牌半组装，取决于经济能力。当然，这是典型的搭配，有些类型应用的性能瓶颈首先出现在 web 上，那种情况就要单独分析了。

web 服务器可以既跑程序又当内存缓存，数据库服务器则只跑主数据库（假如是 MySQL 的话），备份服务器所承担就相对多一些，web 配置、缓存配置、数据库配置都要跟前两台一致，这样 WEB 和数据库任意一台出问题，很容易就可以将备份服务器切换过去临时顶替，直到解决完问题。要注意，硬件是随时可能坏掉的，特别是硬盘，所以宁可 WEB 服务器跟

数据库服务器放在一起，也一定不能省掉备份，备份一定要异机，并且有异步，电力故障、误操作都可能导致一台机器上的所有数据丢失。很多的开源备份方案可选择，最简单的就是 rsync，写 crontab 里，定时同步。备份和切换，建议多做测试，选最安全最适合业务的，并且尽可能异地备份。

四、机房

三种机房尽量不要选：联通访问特别慢的电信机房、电信访问特别慢的联通机房、电信联通访问特别慢的移动或铁通机房。机房要尽可能多的实地参观，多测试，找个网络质量好，管理严格的机房。机房可以说是非常重要，直接关系到网站访问速度，网站访问速度直接关系到用户体验，访问速度很慢的网站，很难获得用户青睐。

五、架构

在大方向上，被熟知的架构是 web 负载均衡+数据库主从+缓存+分布式存储+队列。在一开始，按照可扩展的原则设计和编程就可以。只是要多考虑缓存失效时的雪崩效应、主从同步的数据一致性和时间差、队列的稳定性和失败后的重试策略、文件存储的效率和备份方式等等意外情况。缓存失效、数据库复制中断、队列写入错误、电源损坏，在实际运维中经常发生，如果不注意这些，出现问题时恢复期可能会超出预期很长时间。

六、服务器软件

操作系统 Linux 很流行。在没有专业运维人员的情况下，应倾向于择使用的人多、社区活跃、配置方便、升级方便的发行版，例如 RH 系列、debian、ubuntu server 等，硬件和操作系统要一起选择，看是否有适合的驱动，如果确定用某种商业软件或解决方案，也要提前知晓其对哪种操作系统支持最佳。web 服务器方面，apache、nginx、lighttpd 三大系列中，apache 占有量还是最大，但是想把性能调教好还是需要很专业的，nginx 和 lighttpd 在不需要太多调整的情况下可以达到一个比较不错的性能。无论选择什么软件，除非改过这些软件或你的程序真的不兼容新版本，否则尽量版本越新越好，版本新，意味着新特性增多、BUG 减少、性能增加。一个典型的 php 网站，基本上大多数人都没改过任何服务器软件源代码，绝大多数情况是能平稳的升级到新版本的。类似于 jdk5 到 jdk6，python2 到 python3 这类变动比较大的升级还是比较少见的。看看 ChangeLog，看看升级说明，结合自己情况评估测试一下，越早升级越好，升级的越晚，所花费的成本越高。对于软件包，尽量使用发行版内置的包管理工具，没有特殊要求时不建议自己编译，那样对将来运维不利。

七、数据库

几乎所有操作最后都要落到数据库身上，它又最难扩展（存储也挺难）。数据库常见的扩展方法有复制、分片，设计时要考虑到每种应用的数据如何复制、分片，当然这种考虑一般会推迟到技术设计时期。在初期进行数据库结构设计时，要根据不同的业务类型和增长量预期来考虑是否要分库、分区，并且尽量不要使用联合查询、不使用自增 ID 以方便分片。复制延时问题、主从数据库数据一致性问题，可以自己写或者用已有的运维工具进行检测。

用存储过程是比较难扩展的，这种情形多发生于传统 C/S，特别是 OA 系统转换过来的开发人员。低成本网站不是一两台小型机跑一个数据库处理所有业务的模式，是机海作战。方便水平扩展比那点预分析时间和网络传输流量要重要的多的多。

另外，现在流行一种概念叫 NoSQL，可以理解为非传统关系型数据库。实际应用中，网站有着越来越多的密集写操作、上亿的简单关系数据读取、热备等，这都不是传统关系数据库所擅长的，于是就产生了很多非关系型数据库，比如 Redis/TC&TT/MongoDB/Memcachedb 等，在测试中，这些几乎都达到了每秒至少一万次的写操作，内存型的甚至 5 万以上。在设计时，可根据业务特点和性能要求来选择是否使用这类数据库。例如 MongoDB，几句配置就可以组建一个复制+自动分片+failover 的环境，文档化的存储也简化了传统设计库结构再开发的模式。但是当你决定采用一项技术时，一定要真正了解其优劣，例如可能你所选择的技术并不能支持你所需要的事务和数据一致性要求。

八、文件存储

存储的分布几乎跟数据库扩展一样困难，不过只有百万的 PV 的情况下，磁盘 IO 方面一般不会成大问题，一两台采用 SATA 做条带 RAID 的机器可以应付，反而是自己做异步备份比较复杂，因为小文件多。如果只有一台机器做存储，可以做简单的优化，例如放最小缩略图的分区和放中等缩略图的分区，根据平均大小调整一下块大小。存储要规划好目录结构，否则文件增多后维护起来复杂，也不利于扩展。同时还要考虑将来扩容，例如采用 LVM，或者把文件根据不同规则散列到不同机器。磁盘 IO 繁重的情况下更容易出现故障，所以要做好备份，若发现有盘坏掉，要马上行动更换，很多人的硬盘都是坏了一块之后，接二连三的坏下去。

为了将来图片走 cdn 做准备，一开始最好就将图片的域名分开，且不用主域名。因为很多网站都将 cookie 设置到了.domain.ltd，如果图片也在这个域名下，很可能因为 cookie 而造成缓存失效，并且占多余流量，还可能因为浏览器并发线程限制造成访问缓慢。

九、程序

一定硬件条件下，应用能承载多少访问量，很大一部分也取决于程序如何写。程序写的不好，可能一万的访问都承载不了，写的好，可能一两台机器就能承担几百万 PV。越是复杂、数据实时性要求越高的应用，优化起来越难，但对普通网站有一个统一的思路，就是尽量向前端优化、减少数据库操作、减少磁盘 IO。向前端优化指的是，在不影响功能和体验的情况下，能在浏览器执行的不要在服务端执行，能在缓存服务器上直接返回的不要到应用服务器，程序能直接取得的结果不要到外部取得，本机内能取得的数据不要到远程取，内存能取到的不要到磁盘取，缓存中有的不要去数据库查询。减少数据库操作指减少更新次数、缓存结果减少查询次数、将数据库执行的操作尽可能的让你的程序完成（例如 join 查询），减少磁盘 IO 指尽量不使用文件系统作为缓存、减少读写文件次数等。程序优化永远要优化慢的部分，换语法是无法“优化”的。

然而编程时不应该把重点放在优化上，应该关注扩展性。当今的 WEB 应用，需求变化非常之快，适应多种需求的架构是不存在的，我们的扩展性就要把要点放在跟底层交互的架构上，例如持久化数据的存取规则、缓存的存取规则等，还有一些共用服务，例如用户信息等。先把不变的部分做完善，剩下的部分就很容易将精力放在业务逻辑上面了。

关于作者

刘志一，从 1999 年做个人网站开始一直专注于互联网，目前就职于一家垂直行业 C2C 网站，做产品和开发方面工作。新浪微博：http://t.sina.com.cn/liuzhiyi。

原文链接：http://www.infoq.com/cn/articles/lzy-million-visits-site-technical-preparations

相关内容：

- 数据库在现代搜索技术中的应用
- 建设全功能团队——实践篇
- 王坚谈如何设计成功的互联网产品
- 伏威谈淘宝网的高并发处理与压力测试
- 重构 TekPub——从 ASP.NET MVC 框架迁移到 Ruby on Rails

采访开源 Web 框架 SimpleFramework 开发团队

作者 马国耀

在技术的道路上，永远不缺乏充满激情、勇于创新的人们。只要我们善于思考、发现、总结与提升，就能创造出造福社区、为他人所用的好东西。而 SimpleFramework 就是一个很好的例子。

在其网站开放一个月之后，InfoQ 有幸采访了 SimpleFramework 的开发团队（SD：SimpleFramework 开发团队）：

InfoQ：目前市场上的 **Web** 框架有很多，譬如 **Struts**、**Spring** 等，为什么还要创建 **SimpleFramework** 呢，能够为 **InfoQ** 的读者简单介绍一下该框架的由来、发展及现状吗？

SD：Simpleframework 起始并非一独立框架，而是对现有框架（如 Struts）的业务性补充。最初，利用 Filter 机制和 XML 格式的业务描述来实现请求的可配置扩展处理，将 Struts 等框架中的一些公共功能以组件（如 AjaxRequest、Submit 等）的形式提供，基于组件来实现 UI 和业务（逻辑）功能。随着这一套思路在很多大型企业级应用中使用，一套基于 Filter "后处理" 技术的、完整的 MVC2 和组件化的 WEB 应用开发模式也日渐成熟。它利用 MVC2 实现关注点分离，利用组件复用实现缩短周期、节约成本、提高质量等目的。SimpleFrameworkV3.0 已经发布，它包含核心组件库、基本组件库、业务组件库和扩展组件库。

InfoQ：从其功能上看，它与 **Struts** 具有最多的相似点，请问 **SimpleFramework** 较之 **Struts** 的优势和劣势各是什么？

SD：与 Struts2 相较，SimpleFramework 的优势和劣势简要如下：

优势：

1. 组件体系：声明式组件定义；开放的组件体系，丰富的组件（库）
2. 可扩展性：可自定义或扩展组件，可按照给定规范整合第三方组件资源。

3. 可积累性：可将模块/某类应用封装为组件，可构建业务型组件
4. 可整合性：基于过滤器机制可整合既有应用资源。
5. 实现模式：SimpleFramework 不仅仅是 MVC2 的实现，还具有完整的组件化的 Web 业务支撑能力。

劣势：

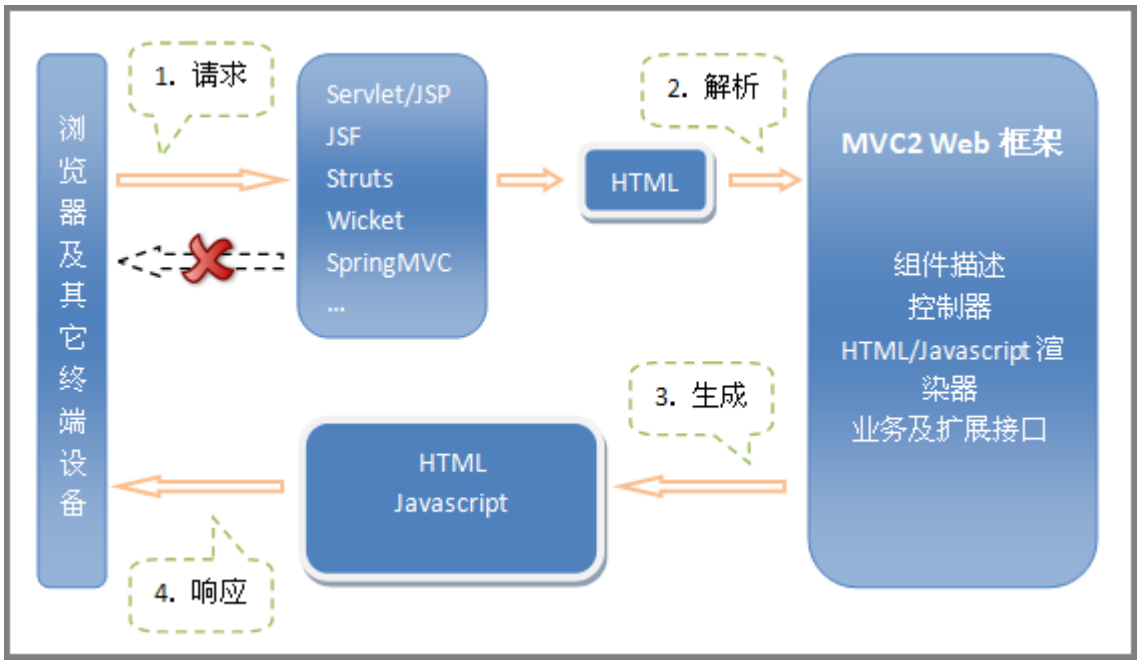
1. SimpleFramework 没有 Struts 的成熟度及用户群。
2. SimpleFramework 暂时还没有丰富的第三方资源。

InfoQ：“后处理” 是 **SimpleFramework** 的技术基础，请您解释一下何为 “后处理”，**Simpleframework** 在这个过程中都做了些什么？

SD：在介绍“后处理”之前，有必要先介绍一下过滤器。许多因素决定了过滤器的重要性：首先，它提供了将公用任务封装成可重用单元的能力；其次，过滤器可用于转换来自 **Servlet** 或 **JSP** 页面响应。Web 应用的基本任务是格式化数据后返回客户端。将过滤器作为 Java Servlet 规范的一部分，为开发人员提供了实现可重用跨容器转换组件的机会。过滤器可以实现多种不同类型的功能，诸如：基于用户身份的阻断式认证请求、Web 应用的用户日志和审计、图像转换及地图缩放、减小下载体积的数据压缩、特定区域的请求-响应本地化实现、适应多类客户端之 WEB 应用的 XML 内容 XSLT 转换、标记化、触发资源访问事件、MIME 类型链接、缓存处理等。这些仅是过滤器应用的很小部分而已，过滤器的应用还有更多。

而 SimpleFramework 用过滤器来做“后处理”——拦截 HttpResponse 并导向 SimpleFramework 作“二次处理”：分派请求逻辑；依据“组件声明”生成或渲染组件、生成代码、处理业务规则、重构响应内容等；其并结合其封装可重用单元和转换页面响应的能力来实现组件处理机制，构建可重用的组件。据此，**SimpleFramework** 已具有完整的 **MVC2** 特征。

其工作机制如下图所示：

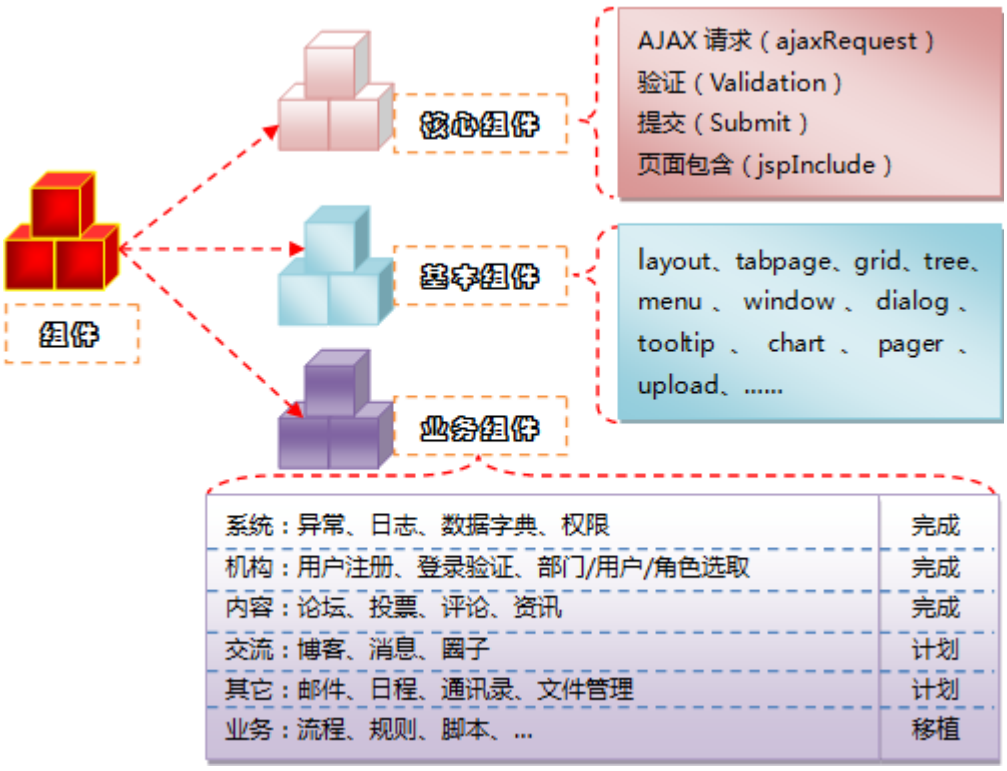


InfoQ：**SimpleFramework** 的另一重要特点是“组件”的概念，组件能够大大减少程序员的开发工作量。在 **SimpleFramework** 中，组件能做什么，**V3** 中已提供的组件有哪些？

SD：SimpleFramework 提供了大量可重用组件，并以一定的接口/API 方式暴露给用户，供应用层调用。

- 核心组件：提供业务无关的功能性处理组件，如 AJAX 请求、表单提交、页面包含、验证等。
- 基本组件：提供 Web 应用开发所需的 UI 组件，如 Layout、Tree、Menu、Window、Grid、Upload 等。
- 业务组件：提供业务相关的模块或公共功能级组件，如工作流、论坛、投票、评论、字典选取、异常、系统日志等。

其组成如下图所示：



InfoQ：在您提到的核心组件中包含 **AJAX** 组件。它能够自动生成 **AJAX** 客户端代码，自动加载到浏览器并在浏览器中执行。那么，当框架提供的 **AJAX** 组件不能满足用户所有的 **AJAX** 需求时，**V3** 提供何种扩展机制供用户来扩展组件的功能？

SD：普通 AJAX 从技术上讲就是 AjaxRequest 完成基于 Ajax 的请求及响应过程，通过 XMLHttpRequest 对象可以很容易实现其基本功能，但 AJAX 组件通过封装 AjaxRequest 可以做更多的事情，在 Simpleframework 中 AJAX 组件有以下特点：

1. 不需要或少量的 js 代码，可能在回调函数中配置。
2. 后台提供了 Handle 接口，开发人员可以专注处理业务逻辑。
3. 自动处理返回的内容，其中采用动态 js、css 装载技术，不需要关心返回的内容中是否含有 js 或 css。

Simpleframework 提供组件扩展机制：其一，每一种组件都提供了相应的业务接口，用户可以继承并覆盖相应的方法来实现自己的业务需求；其二，如果现有的组件和当前的业务需求确实存在巨大的差异，可以按着组件规范定义一个新的组件，并注册到 Simple 系统中来。

一般而言，这两种扩展机制已能满足绝大多数的用户扩展需求，另外 SimpleFramework 是开源产品，用户还可以在遵循协议的前提下基于源码扩展新的组件实现机制。

InfoQ：据我所知，**SSH**（**Struts+Spring+Hiberate**）是目前应用极为广泛的一套框架组合，而

且很多程序在这方面都有很深的积累，那么 **SimpleFramework** 能够与之兼容吗？如果一起使用，您建议二者如何配合使用，即推荐使用 **SSH** 中的哪些功能、 **SimpleFramework** 的哪些功能？

SD：SimpleFramework 完全可以和 SSH 兼容使用，因为 SimpleFramework 的基于过滤器的“后处理”机制决定了这一点，完全可以将 SimpleFramework 看做 SSH 或 Struts 的一个“过滤器”来应用。

SimpleFramework 与 SSH 配合使用时，建议如下：

SSH	SimpleFramework
Spring：Bean Manager、IoC Hibernate：ORM Struts2：FormBean、Action	1） 配置为 Struts 的过滤器。 2） 可以使用 Simpleframework 的核心组件、基本组件和业务组件。 3） 可用 Hibernate 替代 SimpleFramework 的 DAO 层。 4） Struts 的 Action 继承 SimpleFramework 的相关抽象类或实现 SimpleFramework 的接口作为 SimpleFramework 的业务类（Handle Class）。

InfoQ：**SimpleFramework** 已有哪些成功案例，它最适合于构建哪一类应用系统？

SD：在电力和远洋系统的办公自动化、电信的无线网络优化系统中经过大数据量和大并发用户的实战考验。官方网站即基于 SimpleFramework 组件搭建，目前已经提供：文件存储、个性页面、新闻资讯、社区交流等应用。

以 SimpleFramework 的内容组件为基础，可作为构建互联网社区应用的支撑框架。以 SimpleFramework 的基本和业务组件为支撑，可作为构建企业应用的 Web 框架。

InfoQ：目前有没有该产品的路线图，近期会对 **V3** 做哪些方面的增强，能否跟我们的读者先透露一下？

SD: SimpleFramework 下一步的 RoadMap 规划如下：

- 文档补充：对现有 SimpleFramework 文档做进一步的补充和完善
- 内核优化：优化 SimpleFramework 内核代码，进一步优化 HTML 的解析速度和执行效率。
- 组件优化：优化核心组件、基本组件，诸如组件代码生成和渲染效能，并添加一些常用的基本组件。

- 应用服务：以业务组件中的内容组件为基础，为小型互联网社区应用提供快速解决方案（当前主要任务）。
- 组件完善：完善机构组件、系统组件；将在后续的阶段中择机升级流程组件，以提供完整的流程业务支撑组件。

欲了解更多信息，请访问 SimpleFramework 网站。

时至今日，SimpleFramework 的核心团队成员有四人，以下是他们的介绍：

陈侃，是 simpleframework.net 的首席架构师和共同创始人。在 2001 年秋季全面转向对 Java，并对 Java 情有独钟，发起了 simpleframework 开源项目，并参与了许多前沿技术的开发（如工作流、AOP、脚本语言和 Web 等），技术之外，还有一手飞车绝技，酷爱打羽毛球。

赵贵根，系统过程博士，团队的精神领袖。为团队把控工作重点、难点和方向，曾参与过大型的 HIS 系统、电子病历、零售企业 POS 系统、中小企业 ERP 系统、华北电力和国家知识产权局 OA 系统、国家电网电能采集系统、联通资源管理系统的分析设计和建设过程。目前正从事 SimpleFramework 的开源项目并致力于 SimpleFramework 技术与思想的推广。

陈圩贤，中科院硕士，多年从事安全领域的研究和技术工作。目前致力于 simple 的宣传和推广工作。爱好打球、登山、游泳。

刘彬，目前从事企业信息化方面的研发及架构工作，有 Java 大型开发项目经验，兴趣的技术领域是 SOA、Workflow、J2EE 编程，Simpleframework 的开源项目建设，并致力于 Simpleframework 问题的解决和技术推广。

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

相关内容：

- Spring 专家 Isvy 北京再访：Spring 3.1 近况
- 访谈：Paul Fremantle 谈 WSO2 Stratos
- Yehuda Katz 谈 Rails 3.x
- OSGi 原理与最佳实践（精选版）
- Scala 与 Spring：强强联合

Java 深度历险（一）--Java 字节代码的操纵

作者 成富

【编者按】Java 作为业界应用最为广泛的语言之一，深得众多软件厂商和开发者的推崇，更是被包括 Oracle 在内的众多 JCP 成员积极地推动发展。但是 对于 Java 语言的深度理解和运用，毕竟是很少会有人涉及的话题。InfoQ 中文站特地邀请 IBM 高级工程师成富为大家撰写这个《Java 深度历险》专 栏，旨在就 Java 的一些深度和高级特性分享他的经验。

在一般的 Java 应用开发过程中，开发人员使用 Java 的方式比较简单。打开惯用的 IDE，编写 Java 源代码，再利用 IDE 提供的功能直接运行 Java 程序就可以了。这种开发模式背后的过程是：开发人员编写的是 Java 源代码文件（.java），IDE 会负责调用 Java 的编译器把 Java 源代码编译成 平台无关的字节代码（byte code），以类文件的形式保存在磁盘上（.class）。Java 虚拟机（JVM）会负责把 Java 字节代码加载并执行。Java 通过这种方式来实现其“编写一次，到处运行（Write once, run anywhere）”的目标。Java 类文件中包含的字节代码可以被不同平台上的 JVM 所使用。Java 字节代码不仅可以以文件形式存在于磁盘上，也可以通过网络方式来下载， 还可以只存在于内存中。JVM 中的类加载器会负责从包含字节代码的字节数组（byte[]）中定义出 Java 类。在某些情况下，可能会需要动态的生成 Java 字节代码，或是对已有的 Java 字节代码进行修改。这个时候就需要用到本文中将要介绍的相关技术。首先介绍一下如何动态编译 Java 源文件。

动态编译 Java 源文件

在一般情况下，开发人员都是在程序运行之前就编写完成了全部的 Java 源代码并且成功编译。对有些应用来说，Java 源代码的内容在运行时刻才能确定。这 个时候就需要动态编译源代码来生成 Java 字节代码，再由 JVM 来加载执行。典型的场景是很多算法竞赛的在线评测系统（如 PKU JudgeOnline），允许用户上传 Java 代码，由系统在后台编译、运行并进行判定。在动态编译 Java 源文件时，使用的做法是直接在程序中调用 Java 编译器。

JSR 199 引入了 Java 编译器 API。如果使用 JDK 6 的话，可以通过此 API 来动态编译 Java 代码。比如下面的代码用来动态编译最简单的 Hello World 类。该 Java 类的代码是保存在一个字符串中的。

```
public class CompilerTest {
   public static void main(String[] args) throws Exception {
      String source = "public class Main { public static void main(String[] args)
{System.out.println(\"Hello World!\");} }";
      JavaCompiler compiler = ToolProvider.getSystemJavaCompiler();
      StandardJavaFileManager fileManager =
          compiler.getStandardFileManager(null, null, null);
      StringSourceJavaObject sourceObject = new
          CompilerTest.StringSourceJavaObject("Main", source);
      Iterable< extends JavaFileObject> fileObjects =
          Arrays.asList(sourceObject);
      CompilationTask task = compiler.getTask(null, fileManager, null, null,
          null, fileObjects);
      boolean result = task.call();
      if (result) {
         System.out.println("编译成功。");
      }
   }

   static class StringSourceJavaObject extends SimpleJavaFileObject {

      private String content = null;
      public  StringSourceJavaObject(String  name,  String  content)  throws
URISyntaxException {
         super(URI.create("string:///" + name.replace('.','/') +
            Kind.SOURCE.extension), Kind.SOURCE);
         this.content = content;
      }

      public CharSequence getCharContent(boolean
         ignoreEncodingErrors) throws IOException {
         return content;
      }
   }
}
```

如果不能使用 JDK 6 提供的 Java 编译器 API 的话，可以使用 JDK 中的工具类 com.sun.tools.javac.Main 不过该工具类只能编译存放在磁盘上的文件，类似于直接使用 javac 命令。

另外一个可用的工具是 Eclipse JDT Core 提供的编译器。这是 Eclipse Java 开发环境使用的增量式 Java 编译器，支持运行和调试有错误的代码。该编译器也可以单独使用。Play 框架在 内部使用了 JDT 的编译器来动态编译 Java 源代码。在开发模式下，Play 框架会定期扫描项目中的 Java 源代码文件，一旦发现有修改，会自动编译 Java 源代码。因此在修改代码之后，刷新页面就可以看到变化。使用这些动态编译的方式的时候，需要确保 JDK 中的 tools.jar 在应

用的 CLASSPATH 中。

下面介绍一个例子，是关于如何在 Java 里面做四则运算，比如求出来(3+4)*7-10 的值。一般的做法是分析输入的运算表达式，自己来模拟计算过程。考虑到括号的存在和运算符的优先级等问题，这样的计算过程会比较复杂，而且容易出错。另外一种做法是可以用 JSR 223 引入的脚本语言支持，直接把输入的表达式当做 JavaScript 或是 JavaFX 脚本来执行，得到结果。下面的代码使用的做法是动态生成 Java 源代码并编译，接着加载 Java 类来执行并获取结果。这种做法完全使用 Java 来实现。

```
private static double calculate(String expr) throws CalculationException  {
   String className = "CalculatorMain";
   String methodName = "calculate";
   String source = "public class " + className
      + " { public static double " + methodName + "() { return " + expr + "; } }";
      //省略动态编译Java源代码的相关代码，参见上一节
   boolean result = task.call();
   if (result) {
      ClassLoader loader = Calculator.class.getClassLoader();
      try {
         Class<?> clazz = loader.loadClass(className);
         Method method = clazz.getMethod(methodName, new Class<?>[] {});
         Object value = method.invoke(null, new Object[] {});
         return (Double) value;
      } catch (Exception e) {
         throw new CalculationException("内部错误。");
      }
   } else {
      throw new CalculationException("错误的表达式。");
   }
}
```

上面的代码给出了使用动态生成的 Java 字节代码的基本模式，即通过类加载器来加载字节代码，创建 Java 类的对象的实例，再通过 Java 反射 API 来调用对象中的方法。

Java 字节代码增强

Java 字节代码增强指的是在 Java 字节代码生成之后，对其进行修改，增强其功能。这种做法相当于对应用程序的二进制文件进行修改。在很多 Java 框架中都可以 见到这种实现方式。Java 字节代码增强通常与 Java 源文件中的注解（annotation）一块使用。注解在 Java 源代码中声明了需要增强的行为及 相关的元数据，由框架在运行时刻完成对字节代码的增强。Java 字节代码增强应用的场景比较多，一般都集中在减少冗余代码和对开发人员屏蔽底层的实现细节 上。用过 JavaBeans 的人可能对其中那些必须添加的 getter/setter 方法感到很繁琐，并且难以维护。而通过字节代码增强，开发人员只需要声明 Bean 中的属性即可，getter/setter

方法可以通过修改字节代码来自动添加。用过 JPA 的人，在调试程序的时候，会发现实体类中被添加了一些额外的 域和方法。这些域和方法是在运行时刻由 JPA 的实现动态添加的。字节代码增强在面向方面编程（AOP）的一些实现中也有使用。

在讨论如何进行字节代码增强之前，首先介绍一下表示一个 Java 类或接口的字节代码的组织形式。

> 类文件 {
>
> 0xCAFEBABE，小版本号，大版本号，常量池大小，常量池数组，
>
> 访问控制标记，当前类信息，父类信息，实现的接口个数，实现的接口信息数组，域个数，域信息数组，方法个数，方法信息数组，属性个数，属性信息数组
>
> }

如上所示，一个类或接口的字节代码使用的是一种松散的组织结构，其中所包含的内容依次排列。对于可能包含多个条目的内容，如所实现的接口、域、方法和属性 等，是以数组来表示的。而在数组之前的是该数组中条目的个数。不同的内容类型，有其不同的内部结构。对于开发人员来说，直接操纵包含字节代码的字节数组的 话，开发效率比较低，而且容易出错。已经有不少的开源库可以对字节代码进行修改或是从头开始创建新的 Java 类的字节代码内容。这些类库包括 ASM、cglib、serp 和 BCEL 等。使用这些类库可以在一定程度上降低增强字节代码的复杂度。比如考虑下面一个简单的需求，在一个 Java 类的所有方法执行之前输出相应的日志。熟悉 AOP 的人都知道，可以用一个前增强（before advice）来解决这个问题。如果使用 ASM 的话，相关的代码如下：

```
ClassReader cr = new ClassReader(is);
ClassNode cn = new ClassNode();
cr.accept(cn, 0);
for (Object object : cn.methods) {
   MethodNode mn = (MethodNode) object;
   if ("<init>".equals(mn.name) || "<clinit>".equals(mn.name)) {
      continue;
   }
   InsnList insns = mn.instructions;
   InsnList il = new InsnList();
   il.add(new     FieldInsnNode(GETSTATIC,     "java/lang/System",     "out",
"Ljava/io/PrintStream;"));
   il.add(new LdcInsnNode("Enter method -> " + mn.name));
   il.add(new MethodInsnNode(INVOKEVIRTUAL, "java/io/PrintStream", "println",
"(Ljava/lang/String;)V"));
   insns.insert(il);  mn.maxStack += 3;
}
ClassWriter cw = new ClassWriter(0);
cn.accept(cw);
byte[] b = cw.toByteArray();
```

从 ClassWriter 就 可以获取到包含增强之后的字节代码的字节数组，可以把字节代码写回磁盘或是由类加载器直接使用。上述示例中，增强部分的逻辑比较简单，只是遍历 Java 类 中的所有方法并添加对 System.out.println 方法的调用。在字节代码中，Java 方法体是由一系列的指令组成的。而要做的是生成调用 System.out.println 方法的指令，并把这些指令插入到指令集合的最前面。ASM 对这些指令做了抽象，不过熟悉全部的指令比较困难。ASM 提供了一个工具类 ASMifierClassVisitor，可以打印出 Java 类的字节代码的结构信息。当需要增强某个类的时候，可以先在源代码上做出修改，再通过此工具类来比较修改前后的字节代码的差异，从而确定该如何编写增强的代码。

对类文件进行增强的时机是需要在 Java 源代码编译之后，在 JVM 执行之前。比较常见的做法有：

- 由 IDE 在完成编译操作之后执行。如 Google App Engine 的 Eclipse 插件会在编译之后运行 DataNucleus 来对实体类进行增强。
- 在构建过程中完成，比如通过 Ant 或 Maven 来执行相关的操作。
- 实现自己的 Java 类加载器。当获取到 Java 类的字节代码之后，先进行增强处理，再从修改过的字节代码中定义出 Java 类。
- 通过 JDK 5 引入的 java.lang.instrument 包来完成。

java.lang.instrument

由于存在着大量对 Java 字节代码进行修改的需求，JDK 5 引入了 java.lang.instrument 包并在 JDK 6 中 得到了进一步的增强。基本的思路是在 JVM 启动的时候添加一些代理（agent）。每个代理是一个 jar 包，其清单（manifest）文件中会指定一个 代理类。这个类会包含一个 premain 方法。JVM 在启动的时候会首先执行代理类的 premain 方法，再执行 Java 程序本身的 main 方法。在 premain 方法中就可以对程序本身的字节代码进行修改。JDK 6 中还允许在 JVM 启动之后动态添加代理。java.lang.instrument 包支持两种修改的场景，一种是重定义一个 Java 类，即完全替换一个 Java 类的字节代码；另外一种是转换已有的 Java 类，相当于前面提到的类字节代码增强。还是以前面提到的输出方法执行日志的场景为例，首先需要实现 java.lang.instrument.ClassFileTransformer 接口来完成对已有 Java 类的转换。

```
static class MethodEntryTransformer implements ClassFileTransformer {
   public byte[] transform(ClassLoader loader, String className,
     Class<?> classBeingRedefined, ?ProtectionDomain protectionDomain, byte[]
classfileBuffer)
     throws  IllegalClassFormatException {
```

```
        try {
           ClassReader cr = new ClassReader(classfileBuffer);
           ClassNode cn = new ClassNode();
           //省略使用ASM进行字节代码转换的代码
           ClassWriter cw = new ClassWriter(0);
           cn.accept(cw);
           return cw.toByteArray();
        } catch (Exception e){
           return null;
        }
    }
}
```

有了这个转换类之后，就可以在代理的 premain 方法中使用它。

```
public static void premain(String args, Instrumentation inst) {
   inst.addTransformer(new MethodEntryTransformer());
}
```

把该代理类打成一个 jar 包，并在 jar 包的清单文件中通过 Premain-Class 声明代理类的名称。运行 Java 程序的时候，添加 JVM 启动参数-javaagent:myagent.jar。这样的话，JVM 会在加载 Java 类的字节代码之前，完成相关的转换操作。

总结

操纵 Java 字节代码是一件很有趣的事情。通过它，可以很容易的对二进制分发的 Java 程序进行修改，非常适合于性能分析、调试跟踪和日志记录等任务。另外一个非常重要的作用是把开发人员从繁琐的 Java 语法中解放出来。开发人员应该只需要负责编写与业务逻辑相关的重要代码。对于那些只是因为语法要求而添加的，或是模式固定的代码，完全可以将其字节代码动态生成出来。字节代码增强和源代码生成是不同的概念。源代码生成之后，就已经成为了程序的一部分，开发人员需要去维护它：要么手工修改生成出来的源代码，要么重新生成。而字节代码的增强过程，对于开发人员是完全透明的。妥善使用 Java 字节代码的操纵技术，可以更好的解决某一类开发问题。

参考资料

- Java 字节代码格式
- Java 6.0 Compiler API
- 深入探讨 Java 类加载器

原文链接：http://www.infoq.com/cn/articles/cf-java-byte-code

相关内容：研究人员指出最近 Java 频出安全漏洞问题、JRuby 近况：1.5、AOT、Java 7

推荐文章 | Articles

半静态语言-原理和价值分析

作者 何坤

【摘要】动态类型语言在企业开发和互联网开发中应用广泛，而其弱类型的内在特点使其在这些业务复杂的应用开发中存在很多缺点：无法静态检查，程序不健壮，测试成本高；缺乏一些敏捷开发功能如 IDE 内实时验证、代码提示、代码重构等。为此，本文提出半静态语言，它的基本原理是两阶段模型，开发时运用变量类型 声明进行类型检查，运行时采用解释执行的方式。并引入“基于注释的扩展声明指令”，与现有解释器保持完全兼容。半静态语言它结合了动态语言和静态语言的优点，同时满足企业开发中的灵活性、健壮性与敏捷开发的需求。

【关键词】Semi-static Language, Dynamic Typing, Static Typing, Velocity

引言

动态类型语言在企业开发和互联网领域应用广泛，如 Ruby ,Velocity, Python 等。动态类型语言在运行时进行类型推断，以解释方式执行，修改即生效，开发灵活性高；而静态类型语言(如：Java，C/C+/C++) 在执行前做类型检查，需要编译运行，对于互联网前端开发不够灵活。

因此，许多大型互联网站选择 Freemarker, Velocity 这样的动态模板语言作为页面开发语言，在一定程度上满足了前端敏捷开发的需求。

然而，对于大型电子商务网站，不仅具有一般互联网需求频繁变更的特点，更显著特点则是业务繁多，业务模型和业务关系复杂。 因此，在此类应用开发中，Velocity 的开发也遇到了一些的问题。

前端模板开发问题

1. 降低软件质量

Velocity 是弱类型动态语言，运行时才能检查出类型错误。由于动态类型等特点，有的错误在遇到特定参数时，才能激发执行路径，软件质量不能很好的保证。

2. 测试成本高

   由于无法像静态语言一样，在运行前进行类型检查，因此软件的测试周期长，测试成本高。

3. 开发不敏捷

   缺乏一些敏捷开发功能如 IDE 内实时验证、代码提示、代码重构等。虽然能修改即生效，但对于企业级开发，效率较低。

4. 维护性差

   对于一个大型系统，在重构业务模型（Java Model）或代码时，无法知道哪些 Velocity 模板会受到影响；常常需要花费大量时间搜索相关模板，然后修改、测试。例如：笔者所在公司的一个基础产品升级，由于受影响模板众多，重构复杂，项目评估达上千人日。

这些动态语言天生的缺点在企业级和大型网站应用中非常突出，严重的影响了开发质量和开发效率。因此，在技术上亟待一种新的高质量、高效率的开发技术。

静态语言的优势

综合考虑后，我们发现动态类型语言（Dynamic Language）“解释执行方式和修改即生效”的最大优点仍是不能舍弃的。必须从问题出发，找到一条平滑的线路来解决问题。

遇到上述问题时，我们不由自主的会赞美 Java 的优点：

1. 静态语法和静态类型实时检查。

   如果赋值类型不匹配，方法不存在，参数类型错误等信息能马上在 IDE 中显示；

2. 代码提示:

   调用属性，方法时能代码提示，开发非常高效；

3. 代码热链接:

   通过变量和类名热链接到对应的 Java 类；

4. 代码重构：

修改一个 Java 类时 ,受影响的 Java 代码会被实时重新验证 ,马上会显示红色的错误； 更强大的是重构，对 Java 类，方法敏性重命名，会自动修改所有相关代码中对它的引用。

Java 等静态类型语言的这些优势就是解决问题的方向。那为什么动态语言不能做到这些呢？原因在于动态语言的根本特点是变量无类型（即弱类型特点），类型在运行时推断，这使得它无法在开发阶段进行类型检查。

那如何将动态语言和静态语言的优点结合呢？答案就是半静态语言。

半静态语言（Semi-Static Language）

定义

半静态语言，严格说应该是静态化类型的动态语言（Statically Typed Dynamic Language）。它是这样一种语言：以静态方式开发，以解释方式执行；通过变量显式声明或隐式声明，运行前可对变量类型进行推断和验证。

静态语言，动态语言和半静态语言的特点对比分析如下：

语言类型	优点	缺点	举例	适用场景
Static Language	强类型，运行前类型检查，程序健壮 对 Java 等支持反射的语言 ,可实现代码提示，重构等敏捷开发特性	需编译运行，发布慢 无法快速响应需求变化	Java C/C++	企业级后端开发 大型互联网后端开发
Dynamic Language	灵活性高，修改即生效 快速响应需求变化	弱类型，运行时类型检查，程序不健壮，测试成本高	PHP Ruby Velocity	业务简单的小型互联网前端开发
Semi-Static Laguange	开发时(Devtime)强类型，程序健壮 运行时(Runtime)弱类型，修改			业务复杂的企业级开发和大型互联网前端开发

	即生效，快速响应需求变化			

半静态语言集合了静态语言和动态语言的优点，更适合企业级和大型互联网开发，例如：电子商务，ERP，金融，保险等。

技术原理

范例

为了实现目标，需要在动态类型语言基础上，引入变量声明技术。因此本质上，半静态语言也是一种声明式语言（Declarative Language），这一点与静态类型语言一样。

以 Velocity 模板语言为例：

当前 Velocity Template 编程代码范例如下：

```
[Code 1] showBuyProducts.vm

<HTML>>
Hello $customer.Name
<table>>
#foreach( $product in $buyingProducts )
   Buy: $product.Name,  Price: $product.Price,
#end
</table>>
```

该模板执行后，HTML 页面上将用 $customer.Name 显示“客户名称”，循环显示该客户购买的每个产品的名称和价格。在 Velocity 中，运行时通过 Velocity Context 传递变量$customer 和 $buyingProducts，而开发时这两个变量是未定型的（Untyped，或者说都是 Object 类型）。

为了实现静态化开发，引入变量声明，在模板顶部对变量$customer，$buyingProducts 进行显式类型声明。变量声明指令为“##$”。

格式为：

```
##$ <Type> <var1[,var2[,[…]]]>
```

带有变量声明的半静态模板代码为：

```
[Code 2] showBuyProducts_static.vm

##$ com.abc.crm.Customer customer
##$ com.abc.saling.Product product
##$ List<Product> buyingProducts

<HTML>
Hello $customer.Name
<table>
```

```
#foreach( $product in $buyingProducts )
   Buy: $product.Name,  Price: $product.Price,
#end
</table>
```

上述代码中 ,指定了变量 customer 的类型为 com.abc.crm.Customer ,变量 buyingProducts 的类型为 Product 泛型集合。由于 "##"是 Velocity 的注释指令，因此 "##$" 在 Velocity Engine 解析（Parse）和渲染（Render）时不会与现有语法冲突，Velocity 引擎能正常执行，从而保证了兼容性。

动态语言一阶段模型

在动态类型语言中，只有一个运行时（Run Time）阶段，运行阶段由解释器（Intepreter）来对源代码进行解析（Parsing）、执行（Evaluation）产生执行结果。过程如下：

Dynamic Typed Language Execution

由于动态语言无类型的特点，在解析步骤中产生的抽象语法树（Abstract Syntax Tree，AST）所有变量被存储为统一的类型，例如 JavaScript，Velocity 中变量都作为 Object 类型。在执行步骤，一般由类型推断系统（Type Inference System）负责根据变量的实际值动态判断变量的类型，并判断函数、方法或属性调用是否正确，由解释器进行执行或计算，从而产生结果。

半静态语言两阶段模型

而半静态语言，分开发时（Develop Time）和运行时（Run Time）两个阶段，两个阶段互不干扰。

1. 开发时阶段。

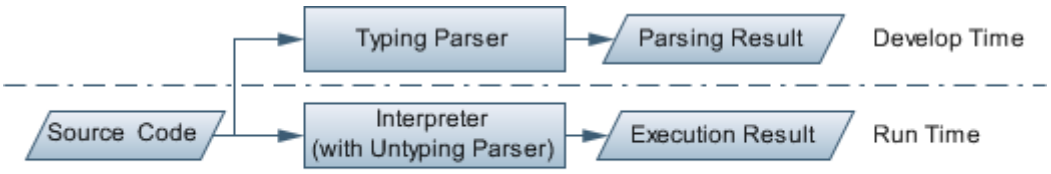
   开发时进行类型检查。一个“编译器”，更严格说是类型化解析器（Typing Parser）负责对源代码进行解析和类型检查，然后输出检查结果。“变量声明”是类型检查的必要条件。检查结果包含类型检查失败的错误信息和警告信息，类似于 Java 编译时的错误信息。

   与静态类型语言不同，此编译器不输出机器代码或字节码，只输出类型检查错误信息。

2. 运行时阶段。

   此阶段中，源代码仍由解释器以解释方式执行，同动态语言的解释执行过程。

半静态语言的两阶段模型如下图所示：



Two Phases Model of Semi-static Language

需要指出的是，运行时阶段仍采用无类型解析器（Untyping Parser），是一个类型推断系统。而开发时采用的是一个新的类型化解析器(Typing Parser)，是一个类型检查系统(Type Checking System)。

开发流程

半静态语言的开发流程涉及 5 个步骤：

1. 编码
2. 编译(类型检查).

   半静态语言的编译与静态类型语言很不相同，它的编译只进行类型检查，不产生机器码或字节码。因此，半静态语言的编译可以称为“检查”（Checking)。

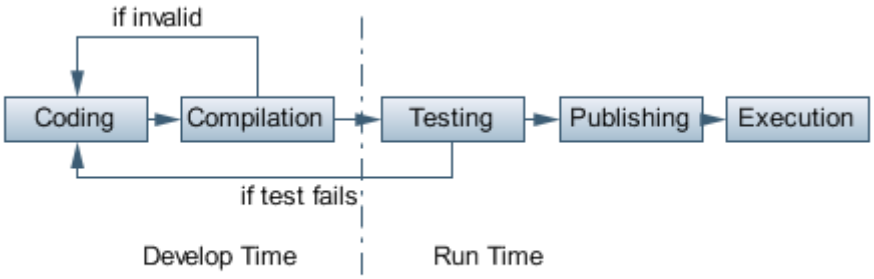
   在这个步骤中，如果代码存在类型错误（Error），编译失败，那么你必须退回到步骤 1）修改代码 bug，直到代码编译正确。
3. 编译过程还可以产生警告（Warning），程序员可以有选择的忽略。 测试

   QA 执行功能测试，集成测试和系统测试。

   如果测试失败，必须退回到步骤 1）。
4. 发布

   将代码发布到生产环境
5. 执行

   最终用户访问用半静态语言开发的应用功能。



Development Process of Semi-static language: 5 steps in 2 phases

从上面的开发流程可见， 开发时阶段覆盖了步骤 1）、2），运行时阶段覆盖了步骤 3)、4)、5）。

为了保证只有编译合法的半静态语言程序在生产环境运行，需要有以下两条约束规则来保证：

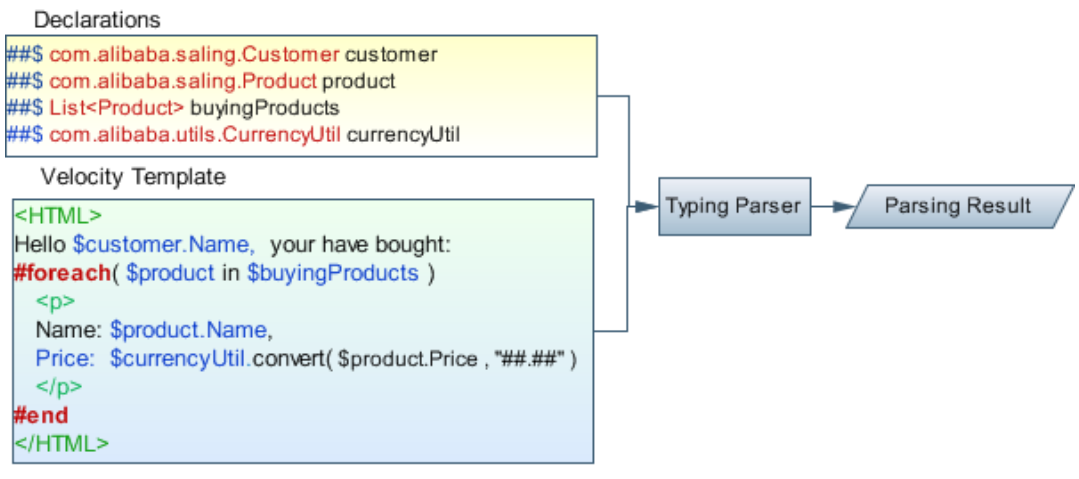
1. 代码编译合法后，才能提交到测试阶段；
2. 测试正确的代码才能发布上线。

由于半静态语言仍用解析器运行，理论上代码仍具有修改即生效的特点。但从软件质量保证角度，这个缺点应该规避。因此上线后的代码不允许未经编译、测试的随意修改。

类型检查系统和原理

半静态语言的类型检查系统中的核心组件编译器 Compiler（或称为 Checker），它本质上是一个类型化解析器。编译时，该系统采用类型检查算法（Type Checking Algorithm）；而在运行时阶段，仍由解释器执行代码，采用类型推断算法（Type Inference Algorithm)。

半静态语言的类型检查基本原理是，根据变量声明对源码进行解析、类型检查和语义检查，输出检查结果。这个系统中类型检查系统的基本原理如下图所示：



Mechanism of Typing Checking System of Semi-static Language

我们使用一个命令行工具 vmcheck 来编译半静态语言代码。格式为：

```
Format: vmcheck templateFile
```

以前面的声明式 Velocity 源码为例，类型检查系统包含以下几个基本规则和检查点：

1. 变量是否声明；

   如果变量 $customer 未声明，编译错误如下：

   ```
   Error: line:2,column:7,variable $customer not declared !
   ```

2. JavaBean 的属性和方法是否存在、

   如果 com.alibaba.saling.Customer 类没有属性 'Name' ， 编译错误如下：

   ```
   Error: line:2, column:7, property 'Name' not found for $customer.
   ```

   如果 com.alibaba.utils.CurrencyUtil 类没有方法 'convert' ， 编译错误如下：

   ```
   Error: line:6, column:22, method 'convert' not found for $currencyUtil.
   ```

3. 方法调用的参数匹配;

   1) 如果这样调用 'convert' 方法：

      ```
      $currencyUtil.convert()
      ```

      则产生如下编译错误信息：

      ```
      Error: line:6, column:22, insufficient parameters for method call
      'convert' .
      ```

   2) 如果这样调用 'convert' 方法

      ```
      $currencyUtil.convert( $customer , "##.##" )
      ```

      则产生编译错误信息：

      ```
      Error: line:6, column:22, parameter type mismatched of $customer for
      method call 'convert' , Double is required.
      ```

4. 特定语句的类型匹配，如条件，循环语句：

   如果有下面的复制语句调用

   ```
   #set( $customer.Name = $product.Price)
   ```

   则产生编译错误信息：

   ```
   Error: line:11, column:5, type mismatched of assignment statement.
   ```

   'if', 'foreach' 等语句使用的类型匹配规则类似。这与 Java 等强类型语言一样。

   1) 集合泛型的类型匹配

      对于 Java 语言，JDK5+支持泛型特性。因此，类型检查也需支持泛型。对于以下代

码

```
##$ List buyingProducts
$buyingProducts.add( $customer)
```

编译错误如下：

```
Error: line:12, column:5, parameter type mismatched of $buyingProducts
for method call 'add' , 'com.alibaba.saling.Product' is required.
As for the previous Velocity code snippet [Code 1], after executing
'vcheck' command on console,
```

变量声明

变量声明就是对变量的类型进行声明。变量声明根据放置的地点分为两种，显示声明（Explicit Declaration）和隐式声明（Implicit Declaration）。

1. 显式声明

   显式声明采用特殊指令（Directive）或语句（Statement），在源码中对变量进行类型声明。

   显式声明通常的格式为：

```
<Declaration Directive>  <Type>  <varList>
```

为了保持与运行时解释器的兼容性，我们引入一种“基于注释的扩展声明指令”技术。以 Velocity 模板语言（VTL）为例，在 Velocity 注释指令“##”上扩展“##$”指令用于变量声明。如下例所示：

```
[Code 3] showBuyProducts_static.vm

     ##$ com.abc.crm.Customer customer
     ##$ List buyingProducts
     ##$ String flag, sss, abc
```

对于其他动态类型语言，同样使用“基于注释的扩展声明指令”来实现兼容性的半静态语言。

Language	Comment Instruction	S2L Declaration Instruction
Velocity	##	##$
Javascript	//	//$
Ruby	#	#$
Python	#	#$

2. 隐式声明

隐式声明不用在源码中编写声明语句，而从配置文件或其他地方分析变量声明。例如，使用 Velocity 进行 Web App 开发时，如果需要直接频繁操作 request,response,session 等 Servlet 容器对象，编译器可以将它们作为内置变量，使用隐式声明。如下表所示：

Built-in variable	Type
request	HttpServletRequest
response	HttpServletResponse
session	HttpSession
application	ServletContext

以下代码使用隐式声明变量 request, session,

```
<html>
     <body>
     Hello, $request.getParameter("username") ! <p/>
     Your logged in at $session.getAttribute("loginTime") last time.
     </body>
     </html>
```

这段代码看起来，对现有 Velocity 语法没有任何扩展. 但实际上，在编译时，编译器使用内置变量对源码进行类型检查。

如果编写了一段错误的调用，例如：

```
$session.getParameter("loginTime")
```

则编译器输出一条“方法不存在的”错误信息：

```
Error: line:12, column:5, method 'getParameter' not found for $session!.
```

语法约束

半静态语言基于某种动态类型语言进行实现，但它在语法语义上更接近与静态类型语言。在这两个端点，存在一些矛盾的地方，比如：变量动态定型，ducking type 等。因此，半静态语言需要有语法约束：

1. 变量先声明，后使用
2. 变量在作用域 scope 内置能声明为一个类型；
3. 禁止 Ducking type 也就是说， 动态语言的无继承多态特性不允许使用，因为这与静态类型系统是冲突的。

如果违反这几个规则，编译器会产生相应的编译错误。以 ducking type 为例（Ruby 支持，而 Velocity 等 Java 系列脚本不支持），如果尝试访问一个不存在的方法，则会产生下面的错误。

```
Error: line:12, column:5, method 'quack' not found for $dog.
```

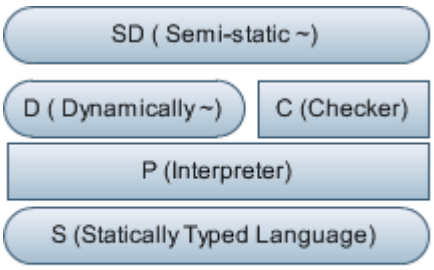
而在 Ruby 中，只要 dog 存在 quack 方法，代码运行是正确的。

半静态语言组成模型和实现方式

半静态语言本质上是动态语言思想和静态语言思想的结合的产物。一种基本的半静态语言实现，核心功能是在运行前进行类型检查和语义检查。其组件集合 SS 包括：

1. 一种静态类型语言 S，S 以编译方式运行；
2. 一种以 S 语言为基础的动态类型语言 D。D 以解释方式由 P 执行，解释器 P 由 S 编写；
3. 在语言 D 的语法集合上扩展变量声明语法，新语法集合名为 SD ；
4. 用语言 S 对解释器 P 进行扩展，实现 SD 的类型编译器 C；
5. 开发时，遵循 SD 语法集合的代码由 C 进行类型检查；
6. 运行时，遵循 SD 语法集合的代码由 P 进行解释执行。

因此，新的半静态语言 SS 是基本组成是：新语法集合 SD 和类型编译器 C.



Semi-static Language Composite

举例：

Java 是一种静态类型语言，运行前进行编译和类型检查；

Velocity 是一种基于 Java 的动态模板语言，通过 Velocity Engine 以解释方式运行；

基于 Velocity 实现半静态语言的方式为：为 Velocity 基本语法增加变量声明指令（语句），基于 Velocity 解释器 实现类型编译器 ，负责在开发时对模板进行类型检查。

实践中，Java 体系的动态类型语言一般与 Java 语言天生的结合使用，应用广泛。以它们为基础，很容易通过扩展方式实现类型编译器，进而实现半静态语言。 例如 Freemarker，Groovy，JRuby，Bean Shell 等。其他动态类型语言也可以基于此原理设计半静态语言，如：Python，Ruby。

IDE 敏捷开发（Agile Development in IDE）

对于 Velocity，Freemarker 这类动态类型语言，它们基于 Java 等强类型语言，在模板内能直接操作传入的 Java 对象。由于 Java 等语言有反射（Reflection）机制。因而，除了静态类型检查的基本功能，可以在类型检查和反射技术基础上，实现一系列 IDE 敏捷开发功能。包括：

1. 代码提示：编辑器内的 Java 对象的属性，方法代码提示；
2. 参数提示：编辑器内的 Java 对象的方法参数提示；
3. 全量构建和增量构建：Java 类修改对相关 Velocity 模板的增量检查；
4. 代码重构 :修改 Java 属性或方法名称，自动批量修改相关模板中所有对应类型的 JavaBean 属性或方法名称。

其中 3），4） 功能对于大型系统的维护和重构价值尤为明显。以上这些敏捷开发功能可独立实现或结合集成开发环境（IDE）如 Eclipse 插件来实现。

结论

通过以上分析可见，半静态化语言结合了静态语言和动态语言的优点，能很好的解决动态语言编程的开发质量和开发效率问题。半静态化语言保留了动态语言的灵活性优点，同时达到了静态语言在开发时强类型检查优势，能有效提升程序健壮性，减低测试复杂性和测试成本。通过与 IDE 结合，实现代码提示，代码重构等敏捷开发功能，有效提升动态语言的开发效率。在企业级应用和互联网应用开发中有着良好的应用价值。

参考资料

[1] Gordon Plotkin. *A Semantics for Static Type Inference.* (1992). http://homepages.inf.ed.ac.uk/gdp/publications/Stat_Type_Inf.pdf

[2] Velocity Template Language Reference. *Apache Software Foundation*. (2010)

[3] Michael Furr. *Static Type Inference for Ruby*. (2009)

http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.144.5525&rep=rep1&type=pdf

[4] http://en.wikipedia.org/wiki/Duck_typing

关于作者

何坤，Raymond He，阿里巴巴技术部中文站架构师。多年 JavaEE 领域开发经验，喜欢钻研崇尚创新。专注敏捷 Web 框架设计，敏捷 Java 开发技术和分布式 Java 系统。2009-2010 作为 Webx2.5 项目架构师，推进中文站 Web 框架敏捷化升级、开发模式敏捷化等工作，通过一系列技术创新，将中文站开发效率每人每天平均节省 30-90 分钟。目前是阿里巴巴中文站开发效率领域负责人，正在研究半静态语言及其在大型网站的应用。

原文链接：http://www.infoq.com/cn/articles/hk-semi-static-language

相关内容：

- 原生 IDE 新军：JRuby 阵营的 RedCar，JavaScript 阵营的 Cloud9
- 什么是 IronRuby？开发者如何在 Rails 中使用它？
- 引入 JavaScript 虚拟机的语言：CoffeeScript 1.0、StratifiedJS、利用 C/C++的 Emscripten 和 Python、配置管理的五项最佳实践
- 重构 TekPub——从 ASP.NET MVC 框架迁移到 Ruby on Rails

访谈与书摘：George Fairbanks 与《恰如其分的软件架构》

作者 Srini Penchikala 译者 马国耀

由 George Fairbanks 编著的《恰如其分的软件架构》一书致力于通过风险驱动的方法进行软件架构开发。

George 从多个视角阐述了架构建模过程，比如用例模型、概念模型、域模型、设计模型和代码模型。此外，他还探讨了多种架构风格，如大泥球模式（Big ball of mud）、管道过滤器模式（Pipe-and-filter）、Map-Reduce 模式，讨论了架构模式与架构风格之间的差异。这次采访还谈到了演化设计、架构重构以及如何对架构模型进行分析、测试和验证相关的话题。

十多年来，George 曾在 Kinetium、Valtech、和 Platinum Technology 等公司从事软件架构和面向对象的设计方面的教学工作。2008 年春天，他担任卡内基梅隆大学研究生软件架构课程的合作讲师；George 一直是软件体系结构工作会议 WISCA（Working International Conference on Software Architecture）、国际软件维护大会 ICSM（International Conference on Software Maintenance）以及欧洲软件架构会议的委员会成员之一；此外，他还是 IEEE 软件工程汇刊和 IEEE 软件的仲裁委员。

InfoQ 就此书采访了 George，探讨了他的写作动机、他的论文项目——软件框架领域的设计片段，以及其他话题。此外，InfoQ 还为读者争取到了本书的书摘（第一章，简介；第三章：风险驱动的模型；第七章，软件架构的概念模型）

InfoQ：本书背后的主要写作动机是什么？

George Fairbanks：有两个方面。其一，转变架构就是摆在架子上的文档的错误观念。面向对象的设计和软件架构都是解决软件设计问题的方法——二者都是工程方法，是有助于开发者提高效率的工程技能。软件架构更多地关心设计问题并解决它们，而非编写的文档。

其二是想纠正一个误解——敏捷与架构水火不容。像许多常年混迹于电信行业中的开发人

员一样，我所学习到的是，若实现一个电话交换，则一定要在 40 毫秒内发出拨号音，这是一个难以实现的需求。后来，我在金融界工作，又学习到建设一个缺乏安全的银行系统是行不通的。若你需要 50 毫秒才能发出拨号音，或者给坏蛋留下盗窃的机会，那么，任何其他特性都不再重要了。

并非每个项目都有一些让你寝食难安的失败风险。但是，当你的确碰到此类问题时——我的网站能否支持扩展到 100M 的点击率？我的投票系统安全吗？ ——你需要工程技术帮助你解决问题。往往会误认为这需要好几个月的时间才能解决，而实际上一个合适的架构设计可能仅仅需要几个小时或几天的时间。

InfoQ：你正在开展的一个论文项目叫做“软件框架领域的设计片段”。可否和我们谈一谈该项目，以及它对写作本书的影响吗？

George：有这么一个强有力的观点：框架是解决软件问题最佳方法。人们曾经抱怨在软件很难继续改善的同时硬件却正迎头赶上，可现在他们抱怨的是操作系统中有着太多的软件。通过框架，解决一个复杂的问题往往仅需几十行代码。可是，正如所有人都经历过的，问题是怎么找到这几十行代码，而这项工作却需要一整天的时间。

设计片段是一些模式，它们展示了客户代码如何插进框架之中。我开发了一个工具，它可用来浏览模式，将你的代码与它们绑定起来，用它来静态地检查你对模式的使用。这是一项在卡内基梅隆大学进行的论文研究项目，该项目是 David Garlan 和 Bill Scherlis 建议的，并与 Jonathan Aldrich 与 Ralph Johnson 组成委员会成员。

在今年的 OOPSLA/SPLASH 大会上，我碰 到了 Ralph，他再次跟我提起他的建议。他回想起在自己曾努力普及重构和重构浏览器（refactoring browser，这是 Bill Opdyke 等人的论文）。他告诉我，问题不在于别人拿走了他的想法，而是人们忽略了这些想法！重构被 IDE 普遍接受花了 10 年的时间。我希望你这样的人 能够继续探究设计片段，这样也许能使好想法不至于淹没在历史的大潮中。

InfoQ：在书中，您谈到将风险驱动的模型应用到敏捷流程。那么在实际工作中如何将架构相关的任务及工作引入敏捷软件开发过程之中，您有何建议？

George：书中描述的风险驱动的模型非常简单。你需将风险（风险是那些你担心可能会导致项目失败的事情，譬如容纳一百万并发用户或解决集成问题）按优先级排序，选择并应用架构方法来减低风险，然后判断这些风险是否已有应对。这个过程通常需要几个小时或几天的时间，而且很容易融入 2 周一次的功能交付的迭代周期。

但是，接受该想法的同时也意味着要抛弃一些别的想法。首先，架构意味着需要事先设计好

一切。可是，在这里你只需探究并设计具有最大风险的部分。有时你需要对一些风险深入分析从而减少担忧，有时则不需要。

应抛弃的第二个想法是完备的架构描述。有时你的确需要一个完整的蓝图，而那通常是应为你正在开展一个巨大的项目，你需要和许多合作团队就设计进行探讨。那些有过这类经历的人应该已经知道这一点，因为你知道，最大的问题是组织结构（如何让 1000 个开发人员同心协力）问题，而非技术问题。而对其他项目而言，最好对一部分进行建模，比如只为服务器进行吞吐率模型，因为这是你最担心的风险。

通过抛弃 Big Design Up Front（译注：编码之前按照既定程序进行大量费时费力的设计工作）和完满的文档，就能方便地将架构融入那些有益的敏捷思想之中，譬如迭代开发、支持重构的测试套件和结对编程。

InfoQ：敏捷软件开发框架通过每日迭代总结、风险储备等方式已包含风险管理，该方法与您书中所介绍的风险驱动模型在管理软件架构方面有何差别？

George：乍一看，本书在风险管理问题上的建议可能与敏捷方法的建议有所不同，但是实际上它们探讨的是不同类型的风险。敏捷在降低项目管理风险上做的非常好，这些风险如构建了错误的程序包或交付延期。但对于工程风险，它大多依赖于开发者通过自己的技能解决。所以，如果你的项目需要成功地解决 UI、数据库或架构风险，开发者必须要掌握相应的技能。极限编程方法中没有关于该使用点对点架构或 3 层架构的建议。它为你扫除了许多管理上的风险，让开发人员可以专心地发挥他们疯狂的工程技能。

本书描述了一组架构技术，这些技术可使你成为更好的开发者。但是，如果你试图将它们全部应用起来，那就又回到了 Big Design Up Front，这不是个好主意。所以，本书推荐风险驱动模型，它指导你应用最有价值的架构技术，然后回到交付系统的工作之上。

敏捷，尤其是短迭代，可帮助你尽早发现问题。它可以使你的客户更早告诉你，你设计的 UI 不能用或者（间接地表明）数据库太慢。它所带来的益处同样适合于架构——能更早地发现 EJB 或 Hadoop 是否适用。

InfoQ：您在本书的第 **14** 章（架构风格）谈到了许多架构风格。在新兴的云计算世界里，应用程序（代码和数据）在外部系统中托管，对于这种架构转型，哪些架构风格和模式更适合呢？

George：有关架构风格，需要记住的最重要的一点是，它们就像“立刻能穿上身的胸衣”，能够帮助你的系统实现某方面的质量。譬如，管道过滤器网络可支持重配置、点对点系统在增加和删除节点时能保持运转、Map-reduce 可以跨多个廉价硬件进行并行计算。

书中有一个架构风格（或模式）目录，而且他们都能应用于托管计算环境，如云计算。租借硬件而非所有硬件可能会改变某些设计决定，但是对此进行报道仍为时尚早。

InfoQ：您认为特定域语言（**DSL**，**Domain Specific Languages**）可在软件架构的建模和执行中扮演什么角色？

George：学术界在定制架构语言领域的研究已有近 15 年了。这些语言，如 ACME、C2、AADL 和 SysML，能够帮助你描述诸如组件、连接器和风格（它们不能直接通过 Java 表述）等架构概念。尽管我曾看过有些人使用 SysML 进行代码生成，但是它们大多用于建模。

然而，正如你说的，大多数人都已将 DSL 用于执行，尽管我们也许根本就没有想过这个问题。当你为 Struts 或 EJB 写配置文件，你就是在为系统定义拓扑。配置文件实际上是框架作者设计的 DSL 文件，开发人员用它来配置运行时系统。许多系统都遵循这种模式，在这些系统中，你编写一些模块/组件，然后通过一个外部配置文件定义他们之间的连接。

InfoQ：书中描述的模型如何用于管理企业内的安全体系架构（**Security Architecture**）的程序。

George：我不准备直接回答这个问题——Srini，你是安全体系结构方面的专家！那么，我要强调的是本书透露的一个信息：你的模型必须要根据问题进行裁剪。如果你建立一个模型来指定列车时刻表，你会使用当初用来计算火车折旧的模型吗？应该不会，对于软件也一样。如果你建立的模型对于预测用户请求的延时非常有效，它不一定能帮你发现安全漏洞。

工程师们长期以来一直在建立了很多模型。根据定义模型不包含所有细节。所以，为了在建模中浪费时间，首先应该明确你的模型要解决的问题，然后再建立一个模型来回答这个问题。

总的来说，软件开发者都是非常聪明的家伙。但是，交给我们的问题总是太大，所以单凭一个开发者的智慧是无法解决的。安全就是一个很好的例子： 我不相信有这样的开发者，他能从 1 千万行代码中发现所有的安全问题。可是，该系统的模型也许有所帮助。例如，系统的运行时视图就能方便地展现哪些服务可在防火墙外访问；或者，你也可以创建一个显示请求通路的定制视图，这样你就可以得知那些通道不对输入进行检测。模型是通过忽略细节、简化问题（将问题简化到能够装入一个人的大脑中并进行推理的粒度）的方式工作的。

InfoQ：对于本书以及软件架构（**Software Architecture**）的当前状态你还有什么想法和建议？

George：我很怀念 20 世纪 90 年代，不仅仅是因为当时的音乐很好听，另一原因是人们对软件设计的激情。当时的大会挤满了学习对象、设计模式和设计技术的开发者们。（还记得吗？）书店里摆满了关于最新软件设计思想的书籍。但是这些先进理念却受到官僚的企业软件开发流程的限制。最后，敏捷运动兴起啦！

今天，令人振奋的是敏捷开发。开发者们都致力于消除软件开发流程中的低效之处。他们转向短迭代，回归测试套件和结对编程。谢天谢地！他们似乎已经从之前的十年中学到了软件设计的经验教训，尽管，让我伤心的是，我的确碰到过一些并不懂得其设计模式的人。

笼统地说，每过十年软件就会增长 10 倍。所以，当我们关注解决流程问题时，我们的软件经历了一个数量级的增长。一点儿也不开玩笑，.NET 的文档比 《大英百科全书》还要厚。糟糕的设计带来的痛苦也成倍地放大了，所以，解决设计糟糕的 10M 行代码的系统中的问题比解决 1M 行代码的系统中的问题的复杂程度要超过 10 倍。而且你知道吗，明年软件又会变得更大!

我认为，此时我们应该将注意力转回设计。软件框架就像巨兽一样统治着地球，开发者们为生存挣扎着。因特网一级的系统及其非常吸引人的工程方法 （这些方法曾经只是少数几个 Bay Area 公司的突发奇想）已经成为家常便饭。我期待阅读有关这些新设计的书籍。

InfoQ：谢谢你参与这次 **Q&A** 形式的采访。最后一个问题，你最喜爱的技术和非技术书籍是什么？

George：在这里我就不再点名那些家喻户晓的经典书籍了，我要推荐的两本书是对我有很大影响但并非广泛传阅的书籍。其一是 Kent Beck 编著的《Smalltalk Best Practice Patterns》。 他坐下来，反思他如何编写那些类、方法、和变量，好像在向我们展现那些描述其思维过程的一组模式一样，所以，读者在一定程度上能够走进 Kent 编程过程中 的思维。我的一个朋友开玩笑地说，Smalltalk 是导致这本书的销量低的原因，而且他准备将它重写成“Java Best Practice Patterns”，因为这些概念仍然适用。

第二本书是《Catalysis》， 由 Desmond D'Souza 与 Alan Wills 合著。这本书的宏大和深度使得令人害怕，而且它所使用的精细的规范也吓走了许多读者。我现在写的不是那本书中的那些前置条件和后置条件，但是我 发现，“思考”那些契约促使我编写更好的代码。那本书在软件设计问题上提出了很有用的建议，包括从方法到框架，乃至组件。

原文链接：http://www.infoq.com/cn/articles/fairbanks-jesa

相关内容：

- 演进架构中的领域驱动设计、SOA 治理成熟度--一名架构师的观点
- 如何进行平台型网站架构设计？、SQL Azure 架构--有竞争力的区别
- 将架构作为语言：一个故事

Dojo：不容忽视的 RIA 框架

作者 王沛

受邀写这样一个专题，非常乐意。这也让我看到了 **infoQ** 是一个真正以技术为导向的社区，不追求最热的话题，只看技术的价值，很欣赏。

2005 年 5 月，Ajax 概念被第一次提出。而在此一年之前，Dojo 框架已经写下了第一行代码。作为 Ajax 之前的“Ajax”框架，Dojo 官网至今 一直用着朴素的名字来定义自己：javascript toolkit。看上去仅仅是个工具集，而事实上它却有框架的力量，推进着大型 Web2.0 应用的开发。这也是 Dojo 一直给人的印象，低调、沉稳，却很强大。如果你仅仅想让 WordPress 页面的下拉菜单效果更加酷，我不反对用 JQuery；但如果你需要以 Web2.0 技术为基础去架构一整个应用，那 么 Dojo 一定是你的最佳选择。

为什么选择 Dojo？

看完前一段，一定会有人说，Dojo 给人的印象没有强大，只有庞大，难用以及糟糕的性能。这完全可以理解。Dojo 文档的缺乏，社区的不活跃，相对严格的设计模式，让很多初学者难以上手，于是转投用起来更加友好的其它框架。这是一个偏见导致的恶性循环，造成用 Dojo 的人始终不多。而消除偏见，正是这个专栏的目的。

如果我们相信大公司 CTO 的眼光，不妨先看看 Dojo 的这些用户：

这只是 Dojo 用户的一小部分，实际用 Dojo 的公司更多，甚至中国的腾讯，百度。如果觉得这些公司的技术离自己很远，那么 Java 的 Struct 2 框架内置 Ajax 模块用的是 Dojo，也许更能说明 Dojo 的价值。

为什么选择 Dojo？这个问题其实很大，也许当你看完整个专题系列，心中自有选择。但在这里，我还是想先简单列出一些 Dojo 有吸引力的特性：

特性	描述
优秀的面向对象体系	类定义，多继承，基类方法调用，在 Dojo 中都设计的自然而合理。良好的 OO 体系，是大型应用的基础。
命名空间	和 Java 一样的命名空间管理，对应文件夹层次结构，让我们能有效管理大量源代码以及他们的依赖关系。
Dijit 界面控件系统	Dijit 是 Dojo 的最大特色，它很容易创建和使用，方便了界面模块化开发且易于维护 ;基于内容构建的天然特性更方便创建 SEO 优化的 Ajax 页面。
严格的设计模式	很多人说 Dojo 难用，严格的设计模式是一个重要原因。很多东西也许第一眼看上去不是很直观，但一但掌握，用这种风格写出的代码会非常容易扩展和维护。
灵活的单元测试框架	DOH 是 Dojo 自带的测试框架，除了标准的函数级别单元测试，你还能用它自动控制鼠标移动和点击来进行自动的界面测试。而且 DOH 同样可以用于非 Dojo 框架的其它 Ajax 测试。
强大的构建工具	Dojo 的 Build 工具其实有很多限制，需要遵循一定的编码规范，这为丰富的功能提供了前提。它能自动处理文件依赖关系，自动分离 i18n 文件，自动嵌入模板文件，让交付的系统运行的更快。

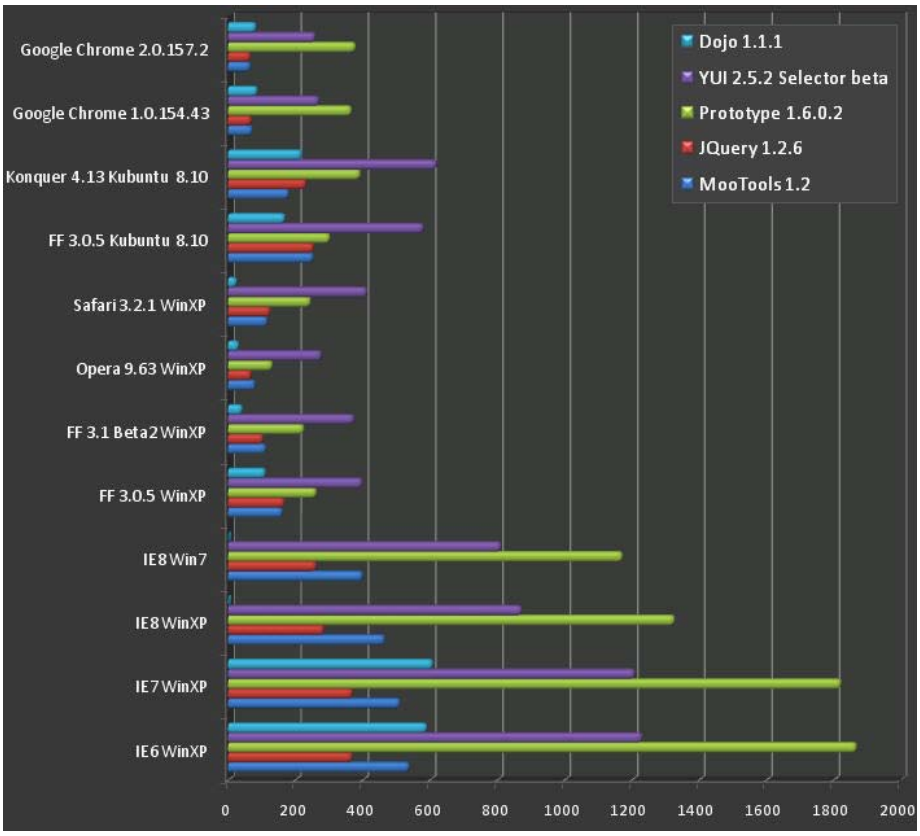
性能其实很优秀

性能很难对比，即使相同的一个 Calendar 控件，功能都会有差别。这里从 2 个点来看性能问题：DOM 节点的查询速度，以及 Dijit 展现的速度。前者大家的功能完全相同，可以有个比较；后者则专属 Dojo，看看绝对的性能。

（**1**）节点查询速度

JQuery 最早引入基于 CSS 选择器进行 DOM 节点查找的机制，也是其最大特征。如今成为了各个框架都具备的功能。这应该算是最能体现 Javascript 算法设计的一点。下图来自于：http://blog.creonfx.com/javascript/mootools-vs-jquery-vs-prototype-vs-yui-vs-dojo-comparison-revised

是各个框架查询速度的对比，时间越少表示性能越好。虽然用的版本比较老，但在查询的实现上各个框架都没有太大的变化。



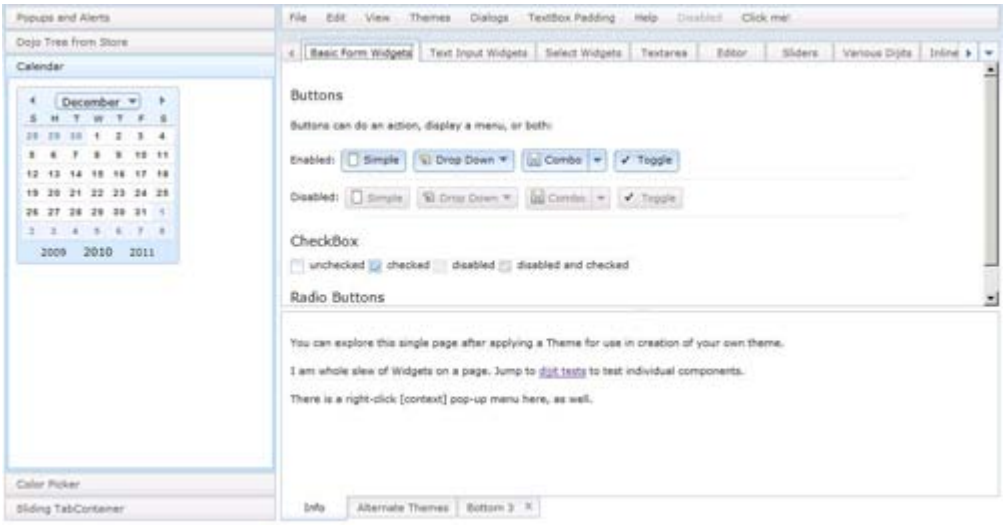
一图胜千言，Dojo 的性能优势一目了然。如果你想亲自做这个测试，可以访问如下地址。

http://mootools.net/slickspeed/

（2）**Dijit** 展现速度

Dijit 是 Dojo 的界面展现体系，性能好坏直接决定着页面的响应速度。在声明方式下，Dojo 需要遍历页面 Html 找到所有的 Dijit 进而创建并展现它们。这个过程比你的想像要快的多，Dojo 的 ThemeTester 是一个很好的例子：

它的访问地址是：
http://archive.dojotoolkit.org/nightly/dojotoolkit/dijit/themes/themeTester.html



此页面包含了 399 个 Dijit ,全部通过声明方式创建。在 FireFox3.6 下总共的展现时间是 2000ms 左右，平均一个 Dijit 用时 5ms。页面中包括了多个 Tree，多个 TabContainer，多个 Menu 以及富文本编辑器这样的复杂 Dijit。因此，如果你发现你的界面远没有这么复杂，却展现的很慢，通常需要检查自定义的 Dijit 设计的是否合理。

综上所述，Dojo 本身，包括自带的 Dijit，拥有着相当不俗的性能。至于如何写出高性能 Dijit，正如如何写出高性能的 JQuery 插件，需要的是经验和积累。

性能是软件开发的一个永恒话题，无处不在。没有一个统一的解决模式，桌面程序也好，Web 程序也好，要提高性能，合理的设计、高效的算法永远是解决问题的王道。而 Dojo 在这一点上，绝不会是你的绊脚石。

开始使用 Dojo

作为专栏的开篇，这里并不会详细介绍很多具体的技术细节。而是通过几个重要 Dojo 相关网站的介绍来了解 Dojo，如下表所示：

网站	介绍
Dojo 官网	本质上官网总是没有太多实际性的作用，作为一个产品门户，起到总体介绍的作用，内容相对很少，而且个人觉得 Dojo 官网的设计实在差强人意，很多过期内容，而且导航非常不便。
Dojo 下载地址	在这里可以下载到所有的 Dojo 稳定版本，每个版本都有几个不同的包可供下载，以 dojo1.5 为例：

	1. **dojo-release-1.5.0.zip** Dojo 标准包，已经压缩，可供生产环境直接使用。 2. **dojo-release-1.5.0-src.zip** Dojo 源代码，包含测试，build 工具，开发环境使用，方便调试。另外这个包下面包含了所有的 test case，通过它们能快速掌握各个控件的用法。 3. **dojo-release-1.5.0-docs.zip** Dojo 离线文档，内容与 http://api.dojotoolkit.org 基本相同，但没有在线的使用方便，一般直接看在线文档即可。 4. **dojo-release-1.5.0-demos.zip** Dojo 功能演示，其实这是非常直观的一块，可以看到很多很酷的用法，每个 Demo 都有具体的代码示例。下载解压后放到 dojo 同级目录即可使用。 5. **dojo-release-1.5.0-shrinksafe.zip** Dojo 的 build 工具，通常专用于 Dojo，已经包含在 src 包里，如果要用于其它的 Javascript 程序打包，可单独下载。
在线 API 手册	经过改版，最近刚刚推出。相对于旧版本已经方便很多。可以查到所有类的属性，方法，事件的介绍和使用示例。
Dojo Campus	在这里能找到几乎所有的类或方法的使用 guide，相当于 Dojo 的 Wiki，内容不断更新。通常在 google 中搜索 Dojo API，来自 Dojo Campus 的是最有价值的。
Sitepen	这是来自 Dojo 创建者的博客，里面会有 Dojo 相关的技术文章，也会有业界的一些观点文章。
Dojo 中文博客	近期比较活跃的 Dojo 中文博客，包括原创内容以及国外 Dojo 技术文章翻译。翻译部分基本会同步于 Sitepen 的文章。

捷径：从 **Dojo Test Case** 学习 **Dojo** 控件用法

除了上述网站之外，有一个快捷学习使用 Dojo 的方法，就是看 Dojo Src 包下面的 test case。在源代码包下面（例如 dojo-release-1.5.0-src.zip），每个命名空间下都会有一个 tests 目录，这里面包含了每个控件或者工具类的各种用法的实例，通过它们能够快速上手它们的用法。

例如：http://yourhost/dojoroot/dijit/tests/test_Dialog.html，这个页面就展示了 Dialog 的各类用法。任何一个 Test Case 都是一个完整的 Dojo 环境，你完全可以模仿它写出自己的第一个 Dojo 程序。

从对象、事件、闭包看 dojo 对 javascript 的扩展

和任何一个 Ajax 库一样，Dojo 对 Javascript 语言本身也做了扩展，例如用 dojo.forEach 来方便的遍历一个集合。因此如果熟悉了一个框架，上手 Dojo 基本上只需要熟悉不同的 API 命名方法 ,dojo 的这些功能基本都在 dojo 命名空间下 ,通过 dojo.doSomething()的形式来调用，或者 dojo.string.doSomething()这样的形式。

在这个功能上我们从 3 个方面的例子来看 dojo。

（1）面向对象支持

开始写代码之前，我们先为源代码位置指定一个命名空间，用如下代码：

```
dojo.registerModulePath('com.infoq', '../infoq');
```

第一个参数表示你的根命名空间，第二个是相对于 dojo.js 的目录。这样整个目录结构如下：

```
/script
/dijit
/dojo
/dojox
/infoq
```

你可以注册多个命名空间到多个文件夹，方便组织源代码文件。这个结构和 java 非常类似，甚至比 java 少了 com/infoq 这样的多余目录。下面的代码就定义了一个 Dojo 类，这个类文件位于 infoq/demo/Class1.js。代码中示例了继承，基类方法调用，构造函数的使用。

```
dojo.provide('infoq.demo.Class1'); //dojo特有，表明这个文件所属命名空间
dojo.require('infoq.BaseClass'); //引入基类定义
dojo.declare('infoq.demo.Class1', [infoq.BaseClass], { //类定义
  constructor: function(){
    //构造函数代码逻辑
  }
  ,doSomething: function(){
    //调用基类infoq.BaseClass的doSomething方法
    this.inherited(arguments);
  }
});
```

示例中演示了 Dojo 的类定义，继承，基类方法调用的方法。可见其过程与传统面向对象的

编程非常类似。Dojo 中通常一个类就是一个文件，并且文件路径和 dojo.provide 声明的类路径严格对应。

（2）事件支持

Dojo 事件系统最大的特点是统一了 DOM 节点的原生事件和自定义事件，例如：

```
var handler = dojo.connect(obj, 'onLoad', callback);
```

这个 onLoad 可以是 DOM 节点的原生事件，也可以是 obj 对象的一个纯 javascript 方法，调用 onLoad 的参数也会传个 callback 方法。dojo.connect 函数返回一个 handler，可以用 dojo.disconnect(handler)的方式取消一个事件的绑定。

（3）闭包及 **dojo.hitch**

闭包是 javascript 的重要特色，它让函数回调的定义变的非常便捷。当年看到 Prototype 提供的 Function.bind 方法惊为天人，dojo 中也提供了同样功能的函数：dojo.hitch。假设某对象的定义如下：

```
var obj = {
   doSomething: function(arg1, arg2, arg3){
      //code
   }
};
```

那么要把 obj.doSomething 进行一个闭包封装可以使用：

```
var callback = dojo.hitch(obj, 'doSomething', 'arg1Value');
```

那么执行：

```
callback('arg2Value', 'arg3Value');
```

完全等价于：

```
obj.doSomething('arg1Value', 'arg2Value', 'arg3Value');
```

可以看到 dojo.hitch 甚至可以连接 2 个部分的参数传递，这个功能非常有用。

忘掉$(id)，看 Dijit 中如何使用 DOM 节点

$(id)曾经是 Prototype 框架的特色，也是大家最为津津乐道的常用函数。后来各个框架均提供并加以扩展（JQuery 中用于基于 CSS 选择器的节点选择）。dojo 中也有一样的函数：dojo.byId(id)。但与其它框架对于$方法的不可或缺相比，dojo.byId 使用非常少，因为它意味着对 DOM 节点的紧密耦合。Dojo 中严格的模块化设计思想极力避免紧密耦合的出现。

开发中用到 DOM 节点时，通常和 UI 相关，而 Dojo 中所有对 UI 的操作和渲染都抽象成 Dijit 的形式，对于复杂的 UI 结构，可以嵌套使用 Dijit。创建 Dijit 有 2 种方式，一种基于已有 DOM 节点，另一种基于模板。

（1）使用已有 **DOM** 节点创建 **Dijit**

首先自己定义一个 Dijit：

```
dojo.declare('infoq.demo.SomeDijit', [dijit._Widget], {
   //dijit code
});
```

然后在页面中以声明方式创建实例：

```
<div dojoType="infoq.demo.SomeDijit" sampleProp="sample"></div>
```

那么在 Dijit 中的代码就可以用 this.domNode 来引用到这个节点。而不再需要通过 dojo.byId 的方式来获取 DOM 节点的引用。

（2）使用模板创建 **Dijit**

顾名思义，模板定义了一个 Dijit 的骨架，假设有如下 Dijit 定义：

```
dojo.declare('infoq.demo.TemplatedDijit', [dijit._Widget, dijit._Templated],
{
   templateString: dojo.cache('infoq.demo', 'templates/templated_dijit.html')
   intro: 'demo template introduction'
   //dojo.cache用于直接获得指定文件内容，会在build时自动嵌入代码
   //templateString定义了一个dijit的模板内容
   //other code
});
```

这就创建了一个基于模板的 Dijit，模板文件位于 script/infoq/demo/templates/templated_dijit.html（前面我们已经注册 script/infoq 对应于 infoq 命名空间）。假设模板文件内容如下：

```
<div>
   <img src='xx.png' dojoAttachPoint='imgAvatar'/>
   <p dojoAttachPoint='introductionNode'>${intro}</p>
</div>
```

那么在页面上的 Dijit 就会以这个模板为骨架。在 dijit 代码中，可以使用 this.imgAvatar 来使用 img 节点。这里的 dojoAttachPoint 就是模板 dijit 中的特有属性，用于获取模板中包含的 DOM 节点或者 Dijit 的引用。例子里模板文件中的${intro}则会直接被替换为实例对象的 intro 属性。

测试及 Build 工具

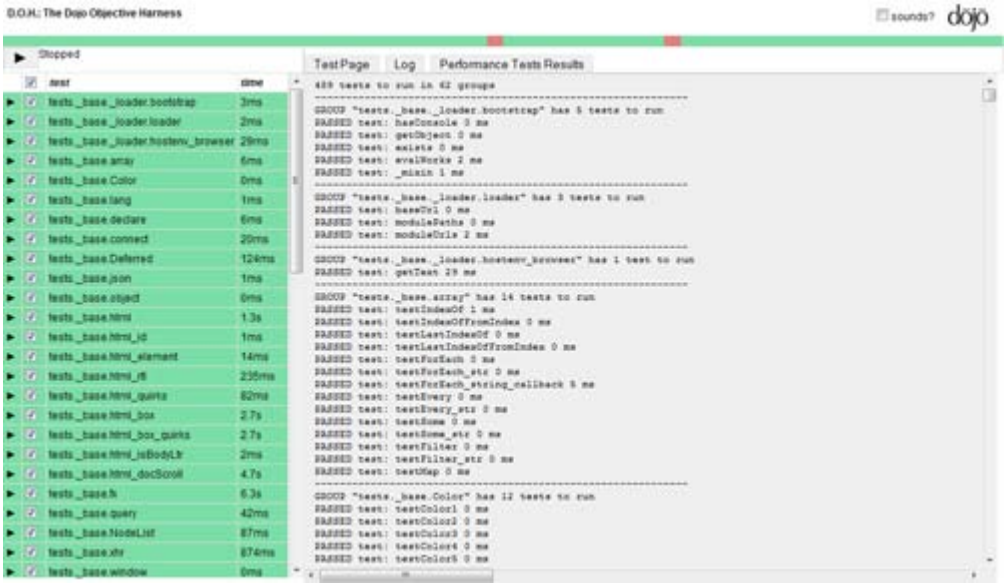
测试及 Build 是完整开发的重要组成部分，如果大家感兴趣，在后续系列中将详细介绍这两块内容，这里作一个简单的介绍：

（1）测试工具：**D.O.H**

DOH 全称是 Dojo Objective Harness，是一个基于 Dojo 的单元测试工具，具有如下特点：

- 既能够测试基于 Dojo 框架的 Ajax 应用，也能够测试基于非 Dojo 框架的 Ajax 应用（老版本只支持 Dojo 框架的）。最新版本现在是 0.9（独立于 Dojo 版本）。而且是 08 年就发布的，可见其还是相当稳定和够用的。
- 能够将多个单元测试用例（多个 javascript，html）文件，在同一个界面上集中展示测试结果。
- 既可以在浏览器中运行测试，也可以用命令行的方式来运行测试，命令行方式仅限于纯 javascript 代码的测试。
- 提供基于浏览器的 Robot 插件，用于模拟鼠标键盘操作，方便界面测试。

大家可以通过 http://yourhost/dojoroot/util/doh/runner.htm 来运行 Dojo 自身的单元测试，体验其强大的功能。下面是其运行结果截图：



（2）**Build** 工具：**Shrinksafe**

ShrinkSafe 同样独立于 Dojo，但针对 Dojo 特别优化，用于打包和压缩 Ajax 应用。其主要提供了如下功能：

- 自动处理源文件的依赖关系，打包多个文件到一个文件
- 压缩 javascript 代码，去除注释，无用空白等
- 处理及打包 i18n 文件
- 自动嵌入 dojo.cache 引用的文件（如 dijit 的模板文件）

通常经过 build 过的代码，体积会减少 30%左右，并且因为将多个文件打包成一个，大大加快了浏览器加载速度，让交付的应用程序运行的更快。

这就是 Dojo

看完上文，相信大家对 Dojo 有了一个大概或者更深入的了解。一直以为 Dojo 是一款强大的工具，熟练掌握能让你的开发事半功倍。而 Dojo 的难用让它的用户相对较少，尤其是在国内。本文的目的正是希望通过对 Dojo 的系统性介绍，消除大家对 Dojo 的偏见，或者帮助那些想用 Dojo 的人用好 Dojo。

如果大家有任何问题或建议，或者希望看到哪些具体的 Dojo 功能介绍，欢迎留下评论，会在后面的文章中针对热点问题进行详细介绍。

原文链接：http://www.infoq.com/cn/articles/fairbanks-jesa

相关内容：

- Dojo 框架：误解与现实
- 使用 Dijit 实现界面组件化开发
- Dojo 联合创建者 Dylan Schiemann 就 AJAX、Comet、Bayeux、RIAs 和 Dojo Toolbox 的问答
- Nexaweb 向 Dojo JavaScript 工具集贡献代码
- 如何选择最合适的 Ajax 框架？

时刻关注企业软件开发领域的变化与创新

InfoQ中文站
www.infoq.com/cn

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

Google 和 Mozilla 相继推出浏览器应用商店

作者 崔康

在 Google 最近推出 Chrome Web Store 商店之后，Mozilla 也表示将在明年初发布浏览器 Web 应用商店，其所倡导的“开放 Web 应用生态系统”预计将会给 Web 社区带来更多发展机会。

原文链接：http://www.infoq.com/cn/news/2010/12/google-mozilla-appstore

VS 2010 急需的服务包已经快发布了

作者 Jonathan Allen 译者 朱永光

Visual Studio 2010 的第一个服务包接近完成。据 Brian Harry 所说，.NET 4 SP1、VS 2010 SP1 和 TFS 2010 SP1 的测试版一共解决了 800 到 1000 个错误。完整的错误修正列表还不能查到，不过他已经提供了一个针对 Team Foundation Server 的 80 个最重要错误的修正列表。

原文链接：http://www.infoq.com/cn/news/2010/12/VS2010-Beta1

OS 发布：Pyxis 2 Beta 2

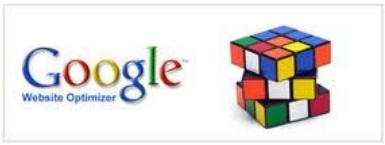
作者 James Vastbinder 译者 张龙

近日，Thomas Holtq 发布了面向.NET Micro Framework 设备的 Pyxis 2.0 操作系统的 Beta 2 版。Pyxis 操作系统一开始本打算替代声名狼藉的 Chumby，但团队现在瞄准了来自于 GHI 的 FEZ Cobra 和 Chipworks X。其想法是构建这样一个设备：其功能与能力都要超越原来的 Chumby。

原文链接：http://www.infoq.com/cn/news/2010/12/pyxis2beta2

使用 Google Website Optimizer 优化页面

作者 Abel Avram 译者 丁雪丰



Google 为 Google Apps 用户提供了一个免费的工具——Website Optimizer，它允许用户测试并度量不同版本 Web 页面的成功访问情况。

原文链接：http://www.infoq.com/cn/news/2010/12/Google-Website-Optimizer

Windows Phone 7 开发人员向导已经发布

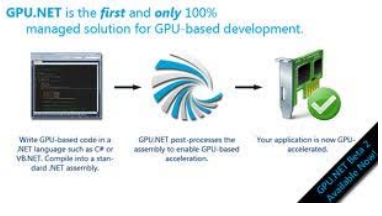
作者 James Vastbinder 译者 朱永光

上个周末，微软的模式与实践团队在 MSDN 上发布了 Windows Phone 7 开发人员向导的 最终版本。微软团队在 CodePlex 上建立了一个开放社区来协作编写这个向导，基于这种模式，在过去 1 年中根据大家的反馈对这个向导进行完善，并且已经 被下载超过 5000 次年。它最终的目的是要创建一个 ,连接到 Windows Azure 的后端的 Windows Phone 7 客户端示例应用程序。

原文链接：http://www.infoq.com/cn/news/2010/12/wp7-dev-guide

使用 GPU.NET 针对 GPU 编程

作者 Abel Avram 译者 侯伯薇



GPU.NET 是为.NET 开发者提供的、整合在 Visual Studio 2010 中的托管解决方案，它的目标是为 GPU 创建带有增强计算功能的应用程序。

原文链接：http://www.infoq.com/cn/news/2010/12/GPU.NET

MySQL 5.5 全面上市增强 Web 应用和性能

作者 霍泰稳

在本周举行的甲骨文（Oracle）全球大会上，Oracle 外宣布，MySQL 5.5 全面上市。在新版的 MySQL 产品中明显增强了 Web 应用，性能也得到显著提升。Oracle 的这一举措也证实了此前在收购 Sun 时，他们对开源社区的承诺。

原文链接：http://www.infoq.com/cn/news/2010/12/mysql5-new-improvements

Open Group 发布新技术标准--SOA 本体

作者 Boris Lublinsky 译者 马国耀

为了通过 SOA 架起业务与 IT 之间沟通的桥梁，Open Group 发布了一项新技术标准——SOA 本体——旨在更好地定义 SOA 的相关概念、术语与语义。

原文链接：http://www.infoq.com/cn/news/2010/12/SOAOntology

Verve 已经发布--一种类型安全的操作系统

作者 James Vastbinder 译者 侯伯薇

最近微软研究院发布了一篇通告，声明 Verve 的发布， 它是来自于 Singularity 项目的一种操作系统，基本前提是要使用类型化汇编语言（Typed Assembly Language，TAL）和 Hoare 逻辑达到高级的保密性（security）和安全性（safety）。Verve 操作系统由内核 （nucleus）、核心（kernel）和一个或多个应用程序组成。

原文链接：http://www.infoq.com/cn/news/2010/12/verve-msft

推荐编辑 | 翻译团队编辑 侯伯薇

又是年终岁尾，每当这个时候，铺开纸写东西的时候，就会习惯性地在最上面写上四个大字——年终总结。在这里，我也不例外，就来做一次在 InfoQ 的年终总结。

不知不觉之间，加入到 InfoQ 中文站已经有一年多的时间了，在过去的 2010 年中，有很多的经历，也有很多感想。还记得《艺术人生》会在这个时候都让大家来点亮属于自己的一盏灯，我也想在《InfoQ 人生》中为自己的 2010 年点上一盏，至于上面的字，我希望是——坚持。

的确，就是坚持。

生活之中没有什么会总是一帆风顺的，即便是在 InfoQ 也是一样。由于各种外在因素的影响，我曾经犹豫过，也彷徨过，也因此而懈怠过，但不管怎样，还是努力坚持了下来，每个月都或多或少地为 InfoQ 中文站贡献自己微薄的力量，也为广大的读者朋友起到一点点儿的用处。也正是在这个坚持的过程中，越来越坚定了要和 InfoQ 中文站一起成长的信念。

之所以坚持，有很多原因，最主要的当然是，InfoQ 真的就像是一个大家庭，里面有很多志同道合的兄弟姐妹。在出现问题、遇到困难的时候，大家不会袖手旁观，只要讲述出来，总是会有人热心地提供无私的帮助。而且，InfoQ 中文站一直在发展，让我可以看到希望，看到更多崭新的东西不断出现，给人以惊喜，我的坚持终究会有一些结果，这也是主要的驱动力所在。

正因为坚持，我在 InfoQ 中文站的这一年中得到了很多，且不说参加了 Qcon 北京大会，结识了好多业界的思想领袖，并且交到了很多心仪已久的好友；也不说翻译了不少新闻和文章，让它们能够通过我以中文的形式和大家见面，而在这个过程中，我学到了很多日常工作中接触不到的思想和知识；更重要的是，这段时间里，我的视野变得更加开阔，对自己更加自信，更喜欢与大家一起交流提高，也更准确地定位了自己的人生目标，那就是——做个快乐的程序员。

2011 年即将来临，我希望可以继续守住自己的信念，继续坚持，并且更加努力，把更多、更好的东西通过 InfoQ 中文站奉献给国内的广大和我一样的程序员们。也希望能够结识更多的朋友，大家一起相互鼓励，相互交流，共同进步，共同提高。也希望 InfoQ 中文站能够在新的一年中有更大的创新和发展，能够为更多人带去更好、更新的思想和理念。

最后还想说一句，我以自己为 InfoQ 中文站的一员而感到自豪，如果你也想体会这种感觉的话，就不要犹豫，加入我们吧！

侯伯薇

2010 年 12 月 28 日 于大连

封面植物

珙桐

珙桐，有“植物活化石”之称，是国家 8 种一级重点保护植物中的珍品，为我国独有的珍稀名贵观赏植物，为世界著名的珍贵观赏树，常植于池畔、溪旁及疗养所、宾馆、展览馆附近，并有和平的象征意义，属于被子植物。

生长分布：在我国，珙桐分布很广。正如其名字一样，“珙桐之乡”的珙县王家镇分布着全国数量众多的珙桐。

珙桐喜欢生长在海拔 700～1600 米的深山云雾中，要求较大的空气湿度。生长在海拔 1800-2200 米的山地林中，多生于空气阴湿处，喜中性或微酸性腐殖质深厚的土壤，在干燥多风、日光直射之处生长不良，不耐瘠薄，不耐干旱。幼苗生长缓慢，喜阴湿，成年树趋于喜光。

植物形态：珙桐为为落叶乔木，树皮呈不规则薄片脱落。单叶互生，在短枝上簇生，叶纸质，宽卵形或近心形，先端渐尖，基部心形，边缘粗锯齿，叶柄长 4-5 厘米，花杂性，由多数雄花和一朵两性花组成顶生头状花序。花序下有 2 片白色大苞片，纸质，椭圆状卵形，长 8-15 厘米，中部以下有锯齿，核果紫绿色，花期 4 月，果熟期 10 月。

珙桐的花紫红色，由多数雄花与一朵两性花组成顶生的头状花序，宛如一个长着“眼睛”和“嘴巴”的鸽子脑袋，花序基部两片大而洁白的苞片，则像是白鸽的一对翅膀。4～5 月间，当珙桐花开时，张张白色的苞片在绿叶中浮动，犹如千万只白鸽栖息在树梢枝 头，振翅欲飞。非常美观，因此英语称为“鸽子树”。

用途：材质沉重，是建筑的上等用材，可制作家具和作雕刻材料。


1kg.org
多背一公斤
爱自然 | 更爱孩子

架构师 1月刊

每月 **8** 日出版

本期主编：张龙

总编辑：霍泰稳

编辑 :李明 胡键 宋玮 郑柯 朱永光 池建强

读者反馈：editors@cn.infoq.com

投稿：editors@cn.infoq.com

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

商务合作：sales@cn.infoq.com 13911020445

本期主编：张龙，**InfoQ** 中文站翻译团队主编

张龙，同济大学软件工程硕士，曾就职于理光软件研究所。主要从事文档工作流和协同知识解决方案的研发工作。热衷于 Java 轻量级框架的研究，对敏捷方法很感兴趣。目前对 Ruby 及 Flex 产生浓厚兴趣。曾有若干年的 J2EE 培训讲师经历。联系方式为：zhanglong318[at]gmail.com

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
Enterprise Software Development Community

商务合作：sales@cn.infoq.com

读者反馈/投稿：editors@cn.infoq.com