InfoQ 时刻关注企业软件开发领域的变化与创新

架构师 Architect

试刊号

毛新生谈Project Zero和软件新发展

InfoQ中文站有幸与IBM中国开发中心Web 2.0首席架构师毛新生聊了聊Project Zero和软件新发展的相关话题。

Aster发布应用于数据库的MapReduce

Aster Data Systems最近发布了应用于数据库的MapReduce，这是其nCluster数据库的一个组件。

Google发布基于全新JavaScript引擎的开源浏览器

最终，Google还是进军了浏览器领域。
9月3日，Google面向全球用户发布了其浏览器——Google Chrome。

InfoQ中文站
www.infoq.com/cn

每月8日出版
InfoQ中文站制作发布

又一个开始

2007 年 3 月 28 日，InfoQ 中文站正式对外发布，迄今为止恰好一年半载。在这段时间里，我们竭尽所能将国外最新关于企业软件开发领域的资讯介绍给国内的架构师们。同时，我们还积极联络技术社区和专家，挖掘报道有价值的事件，并挑选有代表性的以英文发布在 infoq.com 网站，给国外技术社区增加一个了解中文社区的窗口。虽然，InfoQ 中文站距离优秀还有很长的路程，但我们相信，只要目标正确，就不怕路远。

《架构师》电子杂志的诞生不是拍脑袋的产物。网络在线的情况下，我们可以方便地浏览各种资讯，但如果没有网络的时候怎么办呢？这种情况想必我们很多人都遇到过。此时，存储在电脑或者手机上的电子文件就有用武之地了。我自己就曾经通过手机在外出旅行时阅读了基本时下的畅销书。《架构师》的出现，其中一个很朴素的愿望即是让我们的读者在没有网络的情况下，依然能浏览到 InfoQ 中文站的精华内容。

作为网络的一个延伸和补充，在编辑《架构师》过程中，我们将过去一个月有代表性，访问量较高的内容单独摘选出来，按“新品推荐”、“热点新闻”、“人物专访”和“推荐文章”等四个栏目划分。目前暂定每月一期，8 号出刊。所以如果你愿意，你也可以将现在的这本《架构师》看做是“InfoQ 中文站每月精选”。

作为试刊号，《架构师》定然有诸多的不足，我们期望来自读者的反馈，你的只言片语对我们都是激励，所以如果你有兴趣而且有时间，不妨发封邮件到 editors@cn.infoq.com，告诉我们你心目中的《架构师》是什么样子的，目前的这本还有哪些待改进之处。对于 InfoQ 中文站，时刻关注企业软件开发领域的变化与创新，这又是一个新的开始！

霍泰稳

目　录

[人物专访]

[推荐文章]

[新品推荐]

Bas Vodde 谈新书

“Scaling Lean and Agile"及敏捷与 CMMI 的冲突

作者 李剑

在 9 月 20 日举办的 ScrumChina Gathering Event 上，InfoQ 中文站的记者有幸邀请到 Bas Vodde 接受采访，采访的话题包括他与 Craig Larman 合作撰写的两本重量级图书，还有敏捷和 CMMI 的冲突。以下为文字稿。

InfoQ 中文站（InfoQ）：Bas，你好，请向 InfoQ 中文站的读者介绍一下自己好吗？

Bas Vodde（Bas）：大家好，我是 Bas Vodde，出生在荷兰，在几家软件开发公司中工作过，使用一些类似于敏捷的实践——当然，那个时候还没有出现敏捷软件开发这个名字，但我们的做法与之很类似。

InfoQ：可否举几个例子？

Bas：比如构建自己的持续集成服务器啊、测试驱动代码啊等等。然后 2001 年我来了北京，后来去了杭州，为诺基亚工作，几年后去了赫尔辛基，在 Nokia Networks 全公司范围内推行敏捷。

InfoQ：那你当时在公司里的职责是什么呢？敏捷教练？

Bas：唔……不，我带动着整个公司的变化，我们有教练团队，我跟他们一起工作。我参与过很多产品，小到 5 人团队，大到 400 人团队。

InfoQ :我记得你现在正在跟 Craig Larman 写两本书，都是有关“Scaling Lean and Agile”的，你们选择“Scaling”这个词，就是代表在你们在书中主要涉及的就是精益和敏捷在大型公司中的应用么？

Bas：嗯，或者说，大型产品开发。从芬兰回来以后，我又来了杭州工作，跟一个 400 人的产品团队在一起，在我的推动带领下，整个产品都换成了使用 Scrum 做开发。后来去了新加坡，办了一家公司，www.odd-e.com，主要为大型公司、大型项目提供服务，敏捷

训练、Scrum 培训、TDD 培训等等。

InfoQ：能介绍一下这两本书的内容么？

Bas：OK，第一本书的名字是 Scaling Lean & Agile Development - Thinking and Organizational Tools，这本书的主要内容是怎样思考在大型产品中使用敏捷开发和精益开发，一些有关怎样为组织考虑，怎样在大规模团队中工作的概念模型，它涵盖的范围有精益思考，系统思考等等，但也有一些具体内容，例如跨功能的自组织型团队、特性团队，还有些组织方面的东西，例如怎样构建组织结构，管理角色，职业规划等等。世界上有很多上百人的团队都在使用敏捷开发，在中国也有。

InfoQ：那你在大型团队中推行敏捷的时候遇到过哪些困难，又是怎么解决的呢？

Bas：这就是书里讲的内容了，哈哈。其实困难太多太多，最棘手的就是怎样搞定传统组织中的问题，怎样构建团队，提高开发人员的能力，怎样专注于开发而不是其他跟管理相关的东西，你也知道，尤其是在中国，很容易就光想着怎么去管理而不去想开发了。

InfoQ：那另一本书的主题呢？

Bas：Practices for Scaling Lean and Agile Development。这本书里面会讲怎么做计划、设计、测试，怎样协作，怎样多点开发，离岸开发，在一个庞大的代码库基础上怎样做持续集成，怎样处理遗留代码，怎样做产品管理等等。这本书将继续阐述具体实践的话题，里面的内容都是来自于我们在实施中发现的行之有效的方法。

InfoQ：这两本书的写作进度怎么样？

Bas：第一本书已经写完了，希望十二月可以上市，第二本差不多写完了，希望能够明年六月上市。

InfoQ：我确信这两本书会给读者带来很大帮助的。

Bas：希望如此吧~我们可是花了不少功夫来写的：）

InfoQ：我同样也希望它们可以被翻译成中文引进。顺便问一下，在实施的过程中遇到的种种困难，带给你什么样的感觉呢？

Bas：哈，感觉很爽。如果没有困难的话，这个世界可就乏味透了。你的工作就是找出阻力的来源，是什么造成了困境，你会从中成长。如果你不面对这些挑战，征服它们，最后你就会满心沮丧，无心做事&*#@^(……

InfoQ：是的，要改变的东西很多，招聘、销售、绩效考核……

Bas：第一本书一共 350 多页，我们用了 50 页来讲组织中的变化，对我们的经验做了总结，但是这个话题实在太大，完全可以新起一本书，所以我们在书里没法涵盖太多的细节，

我们也没时间，让我们期待下本书的到来吧：）

InfoQ：那么在开始在组织中推行精益和敏捷的时候，你一般都是怎么做的呢？

Bas：唔~~首先，他们既然请了我，那他们就已经下定决心要做实施了。那么接下来我做的工作就是了解情况，发起讨论，引入一些实践，讲解 Scrum，然后做回顾，对回顾中发现的问题做改进，再做回顾……

InfoQ：刚才在别的讨论组上，你还提到了 CMMI 跟 Agile 之间的冲突，能不能再多讲一下你的看法？

Bas：我认为 CMMI 一点用都没有。

InfoQ：呃……

Bas：CMMI 关注的是管理和组织，而不是开发本身。在 CMMI 的一大堆关键过程域（key process area)中，只有一个是跟开发有关系的。大多数的 CMMI 实施都会带来很大浪费。

InfoQ：不过很多人也认为，即使组织中用了 CMMI，他们照样可以使用一些敏捷实践，例如测试驱动开发，持续集成等等。

Bas：没错。我的意思是，CMMI 跟 Agile 在价值观上有冲突，而不是在实施上。我不知道 CMMI 的价值观到底是什么，但是看上去它们的价值观是过程重于人，文档重于可以运行的软件。我不是直接从实施的角度去看敏捷，而是去看敏捷的价值观和原则，但是 CMMI 的价值观和原则是什么？我不知道，因为它们从来没有被记录下来。不过我敢打赌，如果它们被记下来的话，那肯定跟敏捷是冲突的。

InfoQ：哈哈，过程重于人，文档重于可以运行的软件……

Bas：所以，即使你满足了 CMMI 5 的标准，你依然可以使用 Agile；你用了 Agile，也可以过 CMMI 5 认证，但是我还是认为，二者是冲突的。CMMI……它不会关心源代码写成了什么样子，你们团队怎么协作，你是否雇到了恰当的人……

InfoQ：好的，非常感谢你能接受我们的采访。

Bas：多谢！

原文链接：http://www.infoq.com/cn/news/2008/09/bas-interview

[新品推荐]

Aster 发布应用于数据库的 MapReduce

作者 R.J. Lorimer 译者 张龙

Aster Data Systems 最近发布了应用于数据库的 MapReduce ,这是其 nCluster 数据库的一个组件。

InfoQ 已经详细 介绍了 MapReduce，它最初是由 Google 工程师引入的一种编程模型，旨在提供一种可伸缩的方法来处理大数据集。

nCluster 是由 Aster 推出的一个高并行处理（MPP）数据库。其网站这样描述了 nCluster 的并行架构：

Aster nCluster 构建于独特、多层的 nCluster 架构之上 ,它包含三种独立的节点类 :Queens、Workers 及 Loaders。针对分析处理，该三层设计将角色完全隔离并封装起来。每层都可以独立扩展以响应负载变化——当需要时扩充容量（Workers）、加载带宽（Loaders）或者执行并发（Queens）。

Aster nCluster 提供的 MapReduce 实现利用相同的架构，为数据库中执行 MapReduce 计算留有了余地：

就像针对标准 SQL 查询的高并发执行环境一样 ,Aster nCluster 为数据库中的并行数据分析及传输实现了灵活的 MapReduce 函数。Aster nCluster 应用于数据库的 MapReduce 函数很容易编写 ,而且可以与 SQL 语句无缝集成。它们依靠 SQL 查询来操纵底层数据并提供输入。该函数可以操纵输入数据并提供输出 ,而这些输出又可以被 SQL 查询使用或者写到数据库表中。

SQL/MR 是由 Aster 引入的一个特殊的 SQL MapReduce 函数库，可在 nCluster 平台中用来调用 map-reduce 算法。Aster 支持多态函数和动态类型 ,同时 MapReduce 计算可用 Java、Python、C++等语言开发。

请访问 Aster Data Systems 站点以了解有关应用于数据库的 Map Reduce 及 nCluster 数据库的更多信息。

原文链接：http://www.infoq.com/cn/news/2008/09/aster_mapreduce

[新品推荐]

Windows HPC Server 2008 已经发布

作者 Abel Avram **译者** 黄璜

微软刚刚将其 Windows High-Performance Computing (HPC) Server 2008 交付于生产厂商。这一服务器版是 Windows Compute Cluster Server 2003 的后继者，同时代表了微软在高性能计算领域的最新解决方案。

Windows HPC Server 2008 的目标定位于需要大规模处理的业务 ,比如数据仓库或者事务处理，从桌面到拥有数千内核的集群，它都可以良好地伸缩。根据微软的说法，该系统的关键特性有：

- 提升系统管理的生产率与集群的可交互性
- 通过集成的 Visual Studio 2008 快速开发 HPC 应用
- 从工作站到集群的无缝伸缩

Windows HPC Server 2008 是 Windows Compute Cluster Server 2003 的后继者，基于 Windows Server 2008 构建。其许可证只充许集群的 HPC 运行于 HPC Server 2008 上运行。所有的计算节点必须是基于 64 位 x86 的硬件，支持最高 128G 内存，并不支持 IA-64。

Windows HPC Server 2008 可以有 180 天的评估试用期。根据发布通告，其收费标准为每一节点 475 美元。关于价格和许可证的更多信息可以在其如何购买页面找到。关于 Windows HPC Server 2008 的更多信息可于微软的 HPC 站点找到，该站点还包括了常见问题页面。同时关于超级计算机的更多信息请参阅 Top 500 Supercomputers 列表。

原文链接：http://www.infoq.com/cn/news/2008/09/Windows-HPC-Server-2008

[新品推荐]

MacRuby 0.3 发布

支持 Interface Builder，和创建 GUI 用的 HotCocoa

作者 Werner Schuster 译者 贾晓楠

现在，MacRuby 0.3 已经可以用了。

> 一个较大的变化是方法调度器，它现在完全基于 Objective-C 运行时。MacRuby 现在使用 Objective-C 运行时来实现 Ruby 的类语义，并调度纯 Ruby 方法。这是个非常重要的变化，因为这不仅简化了大量的内核实现，还让两个世界之间的界限更加清晰。
>
> [..]
>
> 在纯 Ruby 的方面，修复了很多 bug，我们现在可以运行一些 RubyGems 命令，还可以安装简单的 gem。但不要指望 MacRuby 能运行 Rails！

MacRuby 现在支持用 Cocoa 来创建 GUI—— 实际上创建 GUI 有两种方法。一种是使用和 Apple's XCode 相配套的 Interface Builder（IB）。用 IB 创建的 GUI 可以通过 action 和 outlet（GUI 元件收发消息用的）来连接到 Ruby 类。MacRuby 提供了一个用 Ruby 写的工具来创建必要的元数据，用来映射到 Ruby 代码构造器，例如从 accessor 和 method 映射到 action 和 outlet。

rb_nibtool 是用 Ruby 写的，还使用了 Ruby 1.9 的 Ripper 库。Ripper 使用 Ruby 源，让它能被 Ruby 代码访问——要么作为 Lexer 标记流，要么作为 s 表达式（相当于 ParseTree，不过 Ruby 1.9 里还没有）。rb_nibtool 根据 Ruby 源文件为.nib 文件提供类名；attr_accessor、attr_writer 和其它一些调用被解释为 outlet，一个标识符跟一个 ib_action 定义为一个 action。最终，这些收集到的信息汇入一个.nib 文件，把 GUI 定义和 Ruby 代码连接起来。

顺便提一句：使用 MacRuby 和 Interface Builder 创建 GUI 是在 OS X 上的 Ruby 的一个选

择——而在 Windows 上，Ruby In Steel IDE 能帮助 Visual Studio 创建 GUI，要通过 Ruby Connector 使用 IronRuby 或 MRI。

另一种创建 GUI 的方法是 HotCocoa，它随 MacRuby 一起提供，可以使用创建器的理念来创建 GUI，类似于一些其它的 Ruby 工具，比如 Ruby Shoes，以及其它 Ruby GUI 库。大致了解一下 HotCocoa 的实现，看看创建方法名是如何映射到 Cocoa GUI 控件上的。到现在为止，关于 HotCocoa 的文档几乎没有，因此唯一能参考的就是随 MacRuby 一起提供的 HotCocoa 源码和例程。

和其它 Ruby 工具包一样，HotCocoa 附带了一个用来建立程序框架的工具：

```
hotcocoa classlist
```

该命令用来创建一个新程序，连同必须的库和设置。

HotCocoa 程序是什么样子的呢？这里有一个简短的示例，用来列出所有加载的类及其父类的表格。这段代码建立了一个 GUI——请把代码复制到 HotCocoa 创建的程序框架中的 lib/application.rb 文件中：

```
def start
  application :name => "Classlist" do |app|
    app.delegate = self
    window :frame => [100, 100, 500, 500], :title => "Classlist" do |win|
      # Add a button to - clicking shows the data in the table
      win << button(:title => "Show classes", :bezel => :regular_square).on_action {
        klasses    = []
        ObjectSpace::each_object(Class){|x|
                klasses << {:klass => x.to_s, :ancestors => x.ancestors.join(',')}
        }
        @table.data = klasses
      }
      # create the table
    @table = table_view(
      :columns => [
        column(:id => :klass, :text => "Class"),
        column(:id => :ancestors, :text => "Ancestors")
      ]   )
      # put the table inside a scroll view
      win << scroll_view(:layout => {:expand => [:width, :height]}) do |scroll|
        scroll << @table
      end
```

```
        win.will_close { exit }
      end
    end
  end
```

Rake 文件负责程序运行的所有细节——要运行它，执行：

```
macrake
```

原文链接：http://www.infoq.com/cn/news/2008/09/macruby-03-ib-hotcocoa

[新品推荐]

NewRelic 提供免费的 Rails 监测器并增加新特性

作者 Werner Schuster 译者 张龙

NewRelic 已经更新了其 Rails 性能监测产品（此前 InfoQ 对 NewRelic 的报道从技术上谈到了该产品的工作方式）。RPM 是一个 Rails 插件，它回调 Rails 以收集性能数据，然后将数据发送到 NewRelic 服务器进行存储和评估。RPM 的特色是提供了实时仪表盘（Real-Time Dashboard）以自动查看生成的报告并对向下钻取数据。NewRelic 还对其自身的产品进行监测——RPM 产品的服务器端部分使用 Rails 编写并使用 RPM 监测。

现在 NewRelic 已经免费发布了 RPM Lite——这包括 Rails 插件、搜集数据的托管服务以及实时仪表盘。RPM Lite 可以监测的 Rails 应用的数量并没有限制。请查看 RPM Lite 的发布声明以了解更多细节信息。

RPM 的商业版本分为三个等级：铜、银、金——每个级别都增加了一些更多的特性。这些特性包括事务跟踪（允许向下钻取到事务的细节，直到 SQL）、错误跟踪（对错误分类）、自动事件检测（使用静态分析以确定传感器数据是否偏离了正常值）以及客户化报表等等（请访问 NewRelic 的站点以查看完整列表）。

当前 Rails 是唯一受支持的 web 框架——它支持 Mongrel、Thin、Litespeed、Passenger（mod_rails）以及 JRuby/Glassfish。

原文链接：http://www.infoq.com/cn/news/2008/09/rpm-lite-free-rails-monitoring

[新品推荐]

DataNucleus 访问平台 1.0 最终版发布

作者 Dionysios G. Synodinos 译者 张龙

Java 持久化平台 JPOX 的继任者 DataNucleus 发布了 DataNucleus 访问平台（DataNucleus Access Platform）1.0.0，为使用 JDO/JPA API 的 Java 应用提供了对多种数据存储的访问。

DataNucleus 访问平台 1.0.0 完全兼容于 JDO1、JDO2、JDO2.1 及 JPA1 并提供了 JDO2.2 和 JPA2 的预览特性。当前它支持对 RDBMS、db4o、LDAP、XML、Excel、NeoDatis ODB 及 JSON 的持久化。此外它还可以使用 JDO 注解/XML 或者 JPA 注解/XML 将类的配置信息持久化，不管使用哪种配置方法都提供了对 JDO 与 JPA APIs 的访问。 可以使用 JDOQL、JPQL 或者 SQL（取决于数据）来查询支持的数据存储。通过使用标准化的 APIs，在指定数据存储时，从一种数据存储到另一种数据存储的持久化的交换过程就是简单地修改一下 URL。访问平台由一系列 OSGi 兼容的 jars 所组成并且可以使用在 J2SE、J2EE 或者 OSGi 容器环境下。通过联合使用 OSGi 与 eclipse 扩展点，DataNucleus 变得可扩展并能为你自己的数据库提供支持。

DataNucleus 访问平台提供了下一代的 Java 持久化对象（Java Persistent Objects，即 JPOX）并为以下内容提供支持：

- 持久化到 LDAP 数据存储
- 使用 JDOQL 查询 LDAP 数据存储
- 使用 JPQL 查询 LDAP 数据存储
- 持久化到 Excel 文档
- 使用 JDOQL 查询 Excel 文档
- 使用 JPQL 查询 Excel 文档
- 持久化到 XML 文档
- 使用 JDOQL 查询 XML 文档
- 使用 JPQL 查询 XML 文档
- 嵌入式服务器模式的 DB4O
- 借助于 JDO 或者 JPA 使用 SQL、sql4o 查询 db4o

- 使用 JPQL 查询 db4o
- 持久化到 NeoDatis ODB
- 使用 JDOQL 查询 NeoDatis ODB 数据存储
- 使用 JPQL 查询 NeoDatis ODB 数据存储
- 持久化到 JSON
- 使用 JDOQL 查询 JSON 对象
- 使用 JPQL 查询 JSON 对象
- 为 JDO/JPA 重写了 2 级缓存
- 访问 RDBMS Schema 信息的公共 API
- JDO2.2：支持“cacheable”声明
- JDO2.2：支持“read-only”声明（以前是厂商扩展点）
- JDO2.2：支持“dynamic fetch groups”声明（以前是厂商扩展点）
- JPA2：支持@ElementCollection, @CollectionTable
- JPA2：支持 EntityManagerFactory.getCache()及一些查询方法
- 运行时增强：支持注解及特定的包
- RDBMS：支持使用 count()查询大结果集大小
- RDBMS：当遇到新的接口实现时可以动态升级 schema

DataNucleus 访问平台以 Apache 2 开源协议发布。

可以查看在线文档（或者 PDF）。

请访问 infoq.com/orm 以了解关于 ORM 的更多内容。

原文链接：http://www.infoq.com/cn/news/2008/09/datanucleus_released

[新品推荐]

Alcon 3：另一个开源的 ActionScript 调试工具

作者 Moxie Zhang 译者 张龙

Sascha Balkau 最近发布了 Alcon 3，这是一款特别针对 ActionScript 2 和 ActionScript 3 的开源调试工具。InfoQ 有幸采访了 Balkau，了解到 Alcon 3 究竟是如何辅助 ActionScript 的 RIA 开发的。

Alcon 的官方网站将 Alcon 3 描述为“面向 ActionScript 开发者的一个轻量级调试工具，提供直接且快捷的方法来调试任何 ActionScript 2 或 ActionScript 3 应用，无论这些 ActionScript 是来自于 Web 浏览器、独立的 Flash Player 还是 AIR 运行时都没有问题。”只要是支持 Adobe AIR 的平台都可以运行 Alcon，使用 Alcon 的同时还可以结合 Flex 编译器、Flash IDE 或者 MTASC。

InfoQ 的读者可能会为创建 Alcon 3 的原因感到疑惑，Balkau 这样解释：

> Alcon 的第一个版本诞生于 Flash Player 7 时代。那个版本能够提供的仅仅是一个简单的日志功能，记录输出 ActionScript 代码的踪迹（trace）。从这个角度来看，它与那个时代的其它一些日志工具其实非常类似。
>
> Alcon 2 增加了对 ActionScript 3 的支持和一些新特性，比如文件日志，可以监控日志文件，同时还增加了一个对象探测器（Object Inspector），但是这个版本中 bug 很多。
>
> 我最初编写 Alcon 的起因是因为那时没有足够的选择。你要么在 Flash IDE 中输出踪迹，要么使用现有的为数不多且功能非常基本的日志工具。但这些工具都很难用，很不合我的胃口。尤其是在使用跟踪类之前还不得不对它进行实例化，而 Alcon 的调试类是静态的，又无需实例化。因此，我非常想要一种直接且快捷的方式去使用或调试 API。

关于 Alcon 与其他 ActionScript 调试工具的比较，Balkau 说：

> Alcon 的目标是提供一种快速且便捷的方式来调试 ActionScript，避免使用过多资源。在其它一些工具比如 X-Ray 和 Flex Debugger 则可以帮助你检查应用的整个状

态，他们在各自的领域中都非常优秀。但从我自身角度来说，这些工具在大部分时间里都有些过犹不及，大多数情况下，我还是通过 trace 来调试。

Alcon 与众不同的地方在于它提供的一些的特性，这些特性是它独有的，其它类似工具都不具备。另外它还尽力实现轻量级，并且提供干净的用户接口。说 Alcon 是一个能够在多数 Flex 应用中使用的开发工具一点都不为过。

例如，Alcon 3 增加了应用监视器，这样你就没有必要把 FSPMeter 直接放在 Flash 或者 Flex 应用中。我们还重新设计了对象探测器，使它更有助于列出任何对象的属性。

当问到开发 Alcon 3 所遇到的技术难点时，Balkau 说：

实际上没有什么大的困难。我觉得目前还需要说明的也一个问题，那就是 ActionScript 的 LocalConnection 包的大小，最大不超过 40kb。这基本上意味着如果输出的数据量超出这个限度，那么，Alcon 就不再可靠。尽管 Alcon 3 尝试着在这种情景下使用本地共享对象，但结果还是没有达到最初的期望。我正在寻求解决方案，或许可以尝试通过另一个 socket 连接来解决这个问题，但希望不是很大。

对于下一代基于 ActionScript 3 的应用平台这个话题，他说：

Flash Player 10 将具备很多 Flash 开发者所期待的新特性。就像有些人热衷于游戏开发一样，我特别期盼新的实现 3D 效果、渐变和增强的声音 API 的出现。我觉得这些甚至能够为游戏编程开辟更多的可能性，尽管我必须承认自己只接触过 9 版本的部分新特性，但相对于现在的 Flash 中的 3D 来说，3D 增强肯定会突破很多限制。距完美的 3D 游戏还有很长的路要走，但我们正在一步步得接近。

我认为 ActionScript 对游戏开发感兴趣的人来说要比几年前更加引人注目。这不仅仅是因为它出色的多媒体功能，还因为 ActionScript 相对容易学习，而且在网上可以找到很多的文档和示例。

原文链接：http://www.infoq.com/cn/news/2008/09/alcon3-released

InfoQ中文站

www.infoq.com/cn

我们的**使命**：成为关注软件开发领域变化和创新的专业网站

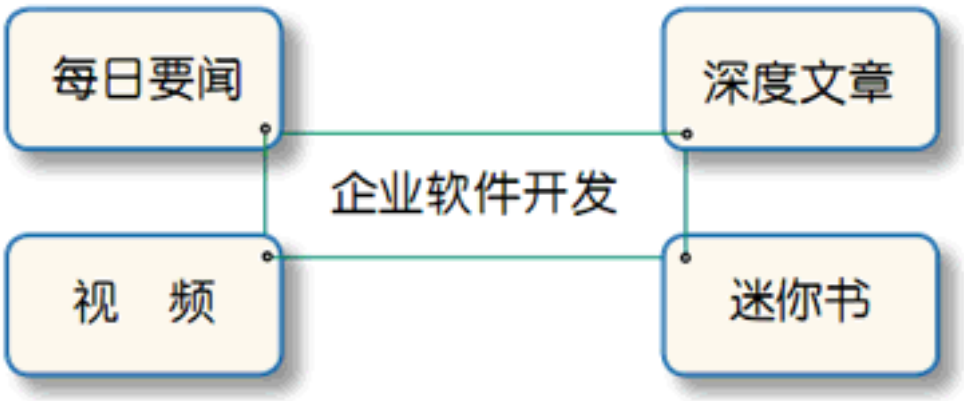
我们的**定位**：关注高级决策人员和大中型企业

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

我们的**特质**：个性化RSS定制、国际化内容同步更新

我们的**团队**：超过30位领域专家担当社区编辑

……



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

[热点新闻]

AJAX Animator 在 RIA 世界展示 AJAX 价值

作者 Moxie Zhang 译者 崔康

AJAX Animator 0.2 在八月初发布。该开源项目使用了 AJAX 技术，提供完全标准化的、在线的、协作的、基于 Web 的动画开发工具包。0.2 版显示了 AJAX 动画在 RIA 世界的良好潜质。AJAX Animator 的创建者，网名叫 Antimatter15，最近与 InfoQ 分享了他的一些见解。

对于 AJAX Animator 的开发，Antimatter15 提到：

> 本项目使用优秀的 Ext JS 2.1 框架定义 UI，这些 UI 几乎包括了一切。时间线使用原生 HTML/CSS。画图技术采用了 josep_ssv 的 OnlyPaths，也曾经使用过 Richdraw。事实上，Onlypaths 基于 RichDraw，但是它更强大。在服务器端，动画通过 Freemovie 库生成，该库使用 PHP 语言。我使用 GD2 导出 GIF 图片，其他格式在客户端导出。在实际应用的服务器上，在所有脚本都通过 YUI Compressor 压缩之后，静态内容由 Google App Engine 控制。我主要使用 Aptana(Eclipse)开发，更少的代码则在 Notepad2 里写。几乎所有图标都来自 silk icon set。

通过 AJAX Animator 创建的动画可以转换成一些通用格式，正如 Antimatter15 所提到的："通过 file 按钮，你可以使用强大的基于 JSON 的文件格式。在该按钮下面，有一个新的发布菜单，允许用户导出各种格式，例如 Adobe Flash，Processing (language)，Microsoft Silverlight (XAML)和 Animated GIF。"

关于标准化，Antimatter15 说：

> 我个人非常喜欢开放的标准，但通常它们是不可能的。如果它存在一丝机会成为一个开放的标准，我都会很高兴。同时，我也不太希望看到引入一个专制的系统（Silverlight）与开放标准（SVG）的竞争。当然，我个人观点认为 Silverlight 比动画"更好"，因为它与浏览器集成得更好，而且可以与 SVG 共享一些相似的标记代码。但是，我认为人们应该能够选择他们偏爱的格式，而且竞争是件好事。

当被问到 AJAX 如何融入富媒体 RIA 领域，Antimatter15 回答道：

我能想象到的是，一旦 Ajax 技术能力到达一定程度，人们就会最终创建出 Ajax 视频编辑工具。不过目前来说，我们对于已有的技术并不满意，因此，需要弥补它的缺点。自从 Internet Explorer 垄断之后，Web 上的创新就不再出现了，但是现在浏览器之争即将打响，包括 Google Chrome (V8)，Firefox 3.1 (TraceMonkey)，Safari (SquirrelFish)和 IE 8（其实它算不上）在内的产品给浏览器领域带来了竞争。在插件领域，也正在发生竞争，这包括曾经垄断的 Flash、新的 Silverlight、JavaFX 和 Google Gears。在这方面，用户才是真正的赢家。

原文链接：http://www.infoq.com/cn/news/2008/09/ajax-animator-in-ria

[热点新闻]

Google 发布基于全新 JavaScript 引擎的开源浏览器

作者 霍泰稳

最终，Google 还是进军了浏览器领域。9 月 3 日，Google 面向全球用户发布了其历经三年时间研发的浏览器——Google Chrome。此前，Google 一直对外否认其有开发浏览器的计划，而且就在不久前 Google 和 Mozilla 就对 Firefox 的支持签署了一份三年的合作协议。对于 Google 这一产品的推出，浏览器市场的格局将有什么样的变化，业界也是众说纷纭。有媒体表示，Chrome 的推出，受到威胁的其实不是一直视 Google 为最大竞争对手的微软所推出的 IE8，而是和 Google 一直良好合作关系的 Firefox：

> 最担心 Google Chrome 的不会是微软。尽管多年来广受诟病，但微软在浏览器市场上的份额仍然超过了 70%。与 Windows 捆绑仍然是微软的最大优势。最担心 Google Chrome 的应当是 Mozilla。
>
> Chrome 并非基于 Firefox，而是基于苹果的 WebKit 引擎。尽管 Safari 并没有获得巨大成功，但 WebKit 却向它提供了一大优势：速度。WebKit 和谷歌开发的 V8 java 脚本渲染引擎联手，理论上将使 Chrome 在速度上“百尺竿头，更进一步”。即使 Chrome 的功能达不到出色的水平，也将成为 Firefox 的强大竞争对手。

那么 Google 是如何考虑的，InfoQ 中文站编辑在今天早些时候采访了 Google 总部产品管理副总裁 Sundar Pichai，他首先介绍了 Google 推出 Chrome 浏览器的目的：

> 我们已经在这个项目上努力了三年，其中我们了解到，随着互联网的普及，越来越多的人将应用部署在网络上。而目前的浏览器因为各种原因，易用性和速度都不能满足我们的要求。Chrome 的推出就是为了解决这些不足之处，以更好地支持 Web 应用。

除此之外，对于更多人关注的相比于 IE 和 Firefox，Google Chrome 有哪些让人耳目一新的特点问题，Sundar 也从用户界面和技术内核等两个方面进行了分析：

> 开发团队一开始将主要精力放在内容而不是易用性方面，但是很快发现这违背

了 Google 一贯的风格，于是我们及时调整了方向。在使用 Chrome 时，你可以发现许多细节上的改进，比如在地址栏里面就可以针对某个网站做搜索，通过视觉和文字两种不同形式提供的标签页等，也就是说你现在可以用更少的文字和点击来完成从前相同的工作。

但是这些简洁的背后是复杂的技术内核，我可以从速度、稳定性和安全性等三个方面来解释一下。在速度方面，Chrome 选择了 WebKit 渲染引擎来处理静态页面，而用全新的 JavaScript 引擎 V8 来处理动态页面；稳定性方面，Google 采用的是一个多进程的架构，这样每个进程就可以单独完成一个任务，互不影响，避免了从前支持多标签的浏览器“一个页面崩溃全部页面遭殃”的情况；对于大家都非常关心的安全性，Google 提供了一种称为“沙盒（Sandbox）”的机制，从前黑客破坏网站时只需攻击渲染引擎，而现在他还要有能力攻破沙盒，难度提高了一倍，另外 Google 还通过自动化的测试及时发现恶意软件和应用，防止“网络钓鱼”这样的破坏性网络行为。

正如 Sundar 所言，对于 Chrome 来说支撑其速度的一个重要技术就是 Google 自己设计的全新 JavaScript 引擎——V8。从 Google 中国研发团队成员现场所做的演示可以发现，同一个 3D 动画，基于 Chrome 要比基于 Firefox 运行流畅的多。目前 Google 已经将 V8 开源，开发人员可以从 Google V8 JavaScript Engine 页面浏览详情和下载源代码。

其实 Google Chrome 也是开源的，根据 Sundar 的解释，Chrome 开发团队在设计 Chrome 时就从 Firefox、Safari 等开源浏览器产品借鉴了许多，其中所用的渲染引擎 WebKit 此前就已经被用在 Safari 上。这次将 Chrome 开源，也是 Google 对开源社区的一次回报，也是希望能推动开源社区的健康发展。

在简洁性和安全性方面，Chrome 要较 IE 和 Firefox 有更为先进的设计，不过让更多使用者感到遗憾地方在于目前 Chrome 还没有提供很好的插件机制。Sundar 对这一问题没有给予正面回答，只是告诉 InfoQ 中文站编辑，对于常用的插件，Google 会尽可能自己提供类似的产品，以满足用户的需要。考虑到直到目前为止，很多银行系统对 Firefox 都没有给予很好的支持，Chrome 在对类似应用的支持上依然还有很长的路要走。另据 Google Chrome 中国研发团队透露，未来一段时间，谷歌会在 Chrome 的本地化方面加大投入。

原文链接：http://www.infoq.com/cn/news/2008/09/google-chrome

[热点新闻]

敏捷？Scrum？吹皱一池春水，干卿何事！

作者 李剑

2008 年 9 月 20 日，ScrumChina 2008 Gathering 活动在上海壹号码头酒店顺利结束，参与者约 55 人，分别来自上海、杭州、成都、北京、香港、新加坡、美国等地。

活动归来，失望远甚于之前的期待。也许，此次活动可以作为一个侧面，反映出国内某个群体对敏捷的理解和应用现状。

会议以 Open Space 的形式进行，首先由 Bas Vodde 介绍了 Open Space 的缘起和基本概念，还有本次活动的主题——Scrum in China。接下来参会者贡献了十多个话题，随着时间的推移，也有新的话题被贴到白板上来，下面仅列举某一部分：

- 跨平台上的敏捷开发
- 怎样帮助团队成为真正的自组织型团队
- Scrum 团队与个人职业发展
- 自动化验收测试
- RobotFramework
- 在大型公司中，Scrum 团队里的领导和管理角色的转变
- Scrum 团队中开发人员与测试人员的协作
- 开发人员跟 QA 怎样协同工作
- 某团队既做新功能开发，又做 hot fix，这样的情况如何处理
- 用什么生成燃尽图？XPlanner？MS Project？
- 怎样向客户推销敏捷
- 什么样的项目适合 Scrum 开发
- TDD：怎样构建自动测试的底层架构，在没有预算或时间的情况下，怎样做 TDD 或是对测试做改进
- 怎样做大规模产品的维护工作

选了几个会场各旁听了一阵子之后，笔者不由想到了 Martin Fowler 在采访中谈到的那番话：

> 很多人都只是片面的关注具体实践，而不是它背后的哲学。如果你只是一味的采用实践，对这套体系的哲学理念置之不理，还想有多好的成效，那可能吗？
>
> ……
>
> 我觉得要学会怎么实践敏捷，最起码要花上几个月的时间。你得进入团队，用敏捷的方式工作，你需要查看所有的因素是怎么配合到一起的。这要经过几个月的练习才行。

其实，Fowler 所指出的那种倾向，在某些 Topic 名字上就已经体现出来了。

学过唯物主义认识论的人，或者说，能够有清晰缜密的思维逻辑的人应该都清楚，我们是先要认识世界，然后才能谈得上改造世界。换句话说，是先认识到问题所在，然后对症下药量体裁衣去解决问题。假如，我们能够有一个统一的认识：为客户交付高质量的软件，能够适应客户不断变化的需求，在成本和收益之间达到最佳的平衡，消除潜在的或是明显的浪费，能够让客户收获最大的 ROI（Return of Investment）；那么问题就很明显了——为了达到这样的目的，我们需要采取什么样的手段？

再或者，我们来问自己几个问题：

1. 我们做过程改进，做敏捷实施……这些事情的目的是什么？
2. 为了达到这样的目的，我们做了哪些工作？
3. 在所做的工作中，哪些事情有助于达成我们的目的，哪些事情事倍功半，哪些事情南辕北辙？
4. 第三个问题中的情况，其成因是什么？
5. 你知道问题所在了么？想到解决方案了么？

我们要做该做的事情，至于是否敏捷（且不论是否有判断敏捷与否的标准），是否用了 Scrum，“吹皱一池春水，干卿何事？”

请允许我借用 Jeff Xiong 在敏捷中国内说过的一段话作为本文的结尾：

> 我不要敏捷
>
> 我要致力于消除软件开发中的一切浪费

原文链接：http://www.infoq.com/cn/news/2008/09/scrum-gathering-finishe

[热点新闻]

何时应该打破规则？

作者 Mark Levison 译者 郑柯

作为 JUnit 测试框架的作者，Kent Beck 在《赶紧交付吧，宝贝儿》一文中提醒我们：所有的敏捷过程和实践，都是为了开发出可以交付的软件。如果有什么成为软件交付的障碍，也许你就得打破规则了。

Kent 引用了 Oakland Raider 公司总经理 Al Davis 的话“赶紧获胜啊，宝贝儿”，他还描述了自己遇到的麻烦，当时他在开发一个 Eclipse 插件，并试图先编写测试来验证之前的一个想法。

> 几周的时间里，我总共花了 6 到 8 个小时来编写第一个测试并使之运行。为 Eclipse 插件写测试可不容易，所以遇到问题也很正常。我就是反复不断地琢磨这个问题，想让我的第一个测试顺畅运行。

8 个小时的工作之后，他还是没有得到一个有用的测试，而且也没能成功测试最早的想法。几天后，他在 Eclipse 中尝试了另外一种不用做测试的方法。他投入了三个小时的工作，发现原先的想法并不怎么样，这让他觉得很不开心。

Piergiuliano Bossi 认为 Kent 传递了错误的信息。他觉得 Kent 只是做了一个探索性的开发（spike），对 Eclipse 的插件架构体系进行了了解，并且试图找出如何在那个环境中编写测试。Piergiuliano 认为不通过 TDD 的方式来搞清楚 api 的用途是很正常的，等到对 api 有了足够的理解之后再重新编写代码，也许效果更好。他想知道是不是有可以暂时不用考虑规则的情况，以及这样做会带来什么后果：

> 咱们考虑这样一种状况：如果可以暂时不遵守某些实践，就能马上得到一些短期的交付。在这种状况下，团队会很容易欠下一些很严重的技术债务，甚至有可能大大影响系统的质量。
>
> ……
>
> 还有其他的状况：系统已经糟糕到一定地步，不允许再发生任何技术债务，软

> 件再有任何差池，都将会导致财务上的损失（甚至更差）。经验告诉我，这些损失是很容易发生的，一旦出现，结果很可能更具破坏性。光说“赶紧交付吧，宝贝儿”会造成灾难性后果，而且是完全不负责任的做法。

他担心 Kent 的文章会“鼓励粗鄙的、不计后果式的编程文化”。

五年前，Bob 大叔写过同样的话题：

> 傻瓜才会盲目遵从规则。我们的脑子足可以分辨规则何时有用，何时不行。我们有责任不断判断规则是否有用。

但是他又从另一面做了解释：

> 我们的职业自豪感才是解决问题的根本。这种自豪感既冰冷残酷，又炽热夺目。它不会让我们因为恐惧而将规则摒弃一边，如果这么做，也是因为职业自豪感让我们发现：某些规则会让我们交付垃圾软件。

所以，Piergiuliano 和 Bob 大叔一定认为：要将“质量第一”牢记心中。

原文链接：http://www.infoq.com/cn/news/2008/09/break_rules

[热点新闻]

排序应该在数据库还是在应用程序中进行？

作者 冯大辉

在网站开发中，究竟是在数据库（DB）中排序好，还是在应用程序中排序更优，这一直是个很有趣的话题。DBANotes.net 博主，在数据库方面比较有研究的冯大辉就这一问题日前和读者明灵（Dragon）做了探讨，本文是关于该问题的总结。

问：请列出在 PHP 中执行排序要优于在 MySQL 中排序的原因？

答：通常来说，执行效率需要考虑 CPU、内存和硬盘等的负载情况，假定 MySQL 服务器和 PHP 的服务器都已经按照最适合的方式来配置，那么系统的可伸缩性（Scalability）和用户感知性能（User- perceived Performance）是我们追求的主要目标。在实际运行中，MySQL 中数据往往以 HASHtables、BTREE 等方式存贮于内存，操作速度很快；同时 INDEX 已经进行了一些预排序；很多应用中，MySQL 排序是首选。而在应用层（PHP）中排序，也必然在内存中进行，与 MySQL 相比具有如下优势：

1. 考虑整个网站的可伸缩性和整体性能，在应用层（PHP）中排序明显会降低数据库的负载，从而提升整个网站的扩展能力。而数据库的排序，实际上成本是非常高的，消耗内存、CPU，如果并发的排序很多，DB 很容易到瓶颈。

2. 如果在应用层(PHP)和 MySQL 之间还存在数据中间层，合理利用的话，PHP 会有更好的收益。

3. PHP 在内存中的数据结构专门针对具体应用来设计，比数据库更为简洁、高效；

4. PHP 不用考虑数据灾难恢复问题，可以减少这部分的操作损耗；

5. PHP 不存在表的锁定问题；

6. MySQL 中排序，请求和结果返回还需要通过网络连接来进行，而 PHP 中排序之后就可以直接返回了，减少了网络 IO。

至于执行速度，差异应该不会很大，除非应用设计有问题，造成大量不必要的网络 IO。另外，应用层要注意 PHP 的 Cache 设置，如果超出会报告内部错误；此时要根据应用做好

评估，或者调整 Cache。具体选择，将取决于具体的应用。

问：请提供一些必须在 MySQL 中排序的实例？

答：在 PHP 中执行排序更优的情况举例如下：

1. 数据源不在 MySQL 中，存在硬盘、内存或者来自网络的请求等；

2. 数据存在 MySQL 中，量不大，而且没有相应的索引，此时把数据取出来用 PHP 排序更快；

3. 数据源来自于多个 MySQL 服务器，此时从多个 MySQL 中取出数据，然后在 PHP 中排序更快；

4. 除了 MySQL 之外，存在其他数据源，比如硬盘、内存或者来自网络的请求等，此时不适合把这些数据存入 MySQL 后再排序。

必须在 MySQL 中排序的实例如下：

1. MySQL 中已经存在这个排序的索引；

2. MySQL 中数据量较大，而结果集需要其中很小的一个子集，比如 1000000 行数据，取 TOP10；

3. 对于一次排序、多次调用的情况，比如统计聚合的情形，可以提供给不同的服务使用，那么在 MySQL 中排序是首选的。另外，对于数据深度挖掘，通常做法是在应用层做完排序等复杂操作，把结果存入 MySQL 即可，便于多次使用。

4. 不论数据源来自哪里，当数据量大到一定的规模后，由于占用内存/Cache 的关系，不再适合 PHP 中排序了；此时把数据复制、导入或者存在 MySQL，并用 INDEX 优化，是优于 PHP 的。不过，用 Java，甚至 C++来处理这类操作会更好。

从网站整体考虑，就必须加入人力和成本的考虑。假如网站规模和负载较小，而人力有限（人数和能力都可能有限），此时在应用层（PHP）做排序要做不少开发和调试工作，耗费时间，得不偿失；不如在 DB 中处理，简单快速。对于大规模的网站，电力、服务器的费用很高，在系统架构上精打细算，可以节约大量的费用，是公司持续发展之必要；此时如果能在应用层(PHP)进行排序并满足业务需求，尽量在应用层进行。

原文链接：http://www.infoq.com/cn/news/2008/09/sort-in-database-applications

[热点新闻]

关于复杂事件处理和事件驱动架构的争论

作者 Steven Robbins 译者 霍泰稳

复杂事件处理（Complex Event Processing ，CEP）系统和事件驱动架构（Event Driven Architecture，EDA）都被认为会在目前和未来的精致繁杂的系统设计中扮演重要角色。但是它们的角色是什么？会对业界产生什么样的影响？最近社区又开始了关于这些问题的争论。David Luckham 和 Roy Schulte 还编撰了一个用于 CEP 和 EDA 的术语概览和词汇表。

抛开实现细节，Luckham 和 Schulte 对每个术语做了定义：

> 首先“事件”这个最普遍的术语，是有问题的。基本上它包含两个截然不同的含义：（1）一个发生的活动；（2）计算机系统里面代表某个活动的事物。按理说，应该引入两个不同的术语，比如“事件（event）”和“事件对象（event object）”。但是，事实是在每个稍微长些的讨论中，你都会发现这样做太晦涩难懂了，它们【注：event 和 event object 这两个分开的术语】的区别要不就是被误用，要不就是被忘记甚至忽略了。举个例子，如果要使用两个术语，那么很有可能导致你在说“事件处理（event processing）”时，其实意思是指“事件对象处理（event object process）”。所以说，最好的办法是复用“事件（event）”这个单词，通过每个词的上下文来理解它所要表达的意思。

Luckham 和 Shulte 将“复杂事件”定义成“一个对多个其它子事件的抽象的事件。”在提到穆迪投资者服务系统中的问题导致不正确的评级时，Joe Mckendrick 谈论到了复杂事件的话题。Mckendrick 说“也就是说，目前即使没有上亿美元，也有数百万美元的投资决定是由此类系统产生的错误数据造成的。” Mckendrick 的立场是，复杂的识别和感应系统也许仍然需要人类的参与，以阻止问题或者错误的发生。

Mckendrick 提到 K. Mani Chandy 博士，加州理工学院的一个正在做识别和感应研究的计算机科学教授，他曾经表示在基于复杂事件做决策时，要保证这个过程中有人的参与。Chandy

说在有些情况下，比如战术军事上的某个涉及到使用武器的操作，“它会一直有个对此事最终行为负责的人参与其中。”

Chandy 和 Micahel Olson 谈到为何 事件处理与 ‘识别和感应’ 应用（PDF）也许将在业务活动监测和业务仪表盘领域普遍存在。Chandy 和 Olson 对 Web 识别和感应应用有非常深入的研究，这些应用从 Web 数据源提取事件和数据：

> Web 数据源可以是活跃的或者休眠的。客户端可以通过请求-应答协议轮询服务器，以获得信息。而信息也可以通过 RSS 或者 ATOM 流，或者其他的数据协议，推送给客户端。休眠的数据源也可以有个活跃的接口，方法是让代理定期轮询它，并在接下来的轮询中传输更改的信息。

但是 CEP 真的需要一个完全不同的架构类型吗？

Brenda Michelson 就事件处理写了一篇文章——事件驱动架构概览。他定义了 EDA 中的 5 类组件：

- 事件元数据：事件元数据包括事件说明和事件处理规则；
- 事件处理：事件处理的核心是引擎和事件发生数据；
- 事件工具：事件开发工具用于定义事件说明和事件规则，以及管理订阅等。事件管理工具提供事件处理基础架构的管理和监测，事件流的监测以及显示事件生成和处理状态等；
- 企业集成：一个企业集成中枢在事件驱动架构中扮演着重要的角色。需要集成的一些服务包括：事件预处理（过滤、路由和转变等）、事件通道传输、服务调用、业务流程调用、发布和订阅，以及企业信息访问等；
- 源和目的：创建事件和/或执行一个事件驱动动作的企业资源（应用、服务、业务流程、数据存储、人员和自动代理等）。

Michelson 还谈到了 EDA 和 SOA 之间的关系：

> 我相信 SOA 和 EDA 是平等和互补的。所以，我不认同那些努力传播 SOA 的同学们所说的“EDA 只是 SOA 的一个子集”的论断。一个更广泛的事件驱动架构概念，不仅是超越事件驱动 SOA 的，还应该包括实时信息流和分析，以及复杂事件处理。

Ivar Jacobson 博士在 EDA 方面有自己独到的见解。Jacobson 提出的问题是：我们需要事件驱动架构吗？在回答他自己的问题时，Jacobson 说，“当 EDA 认为事件是系统中最重要的

组成时，你最好注意那些组件或者服务，以及组件之间的‘通道’”。事件可以被生产、传递和消费，甚至在系统中被传播。这种类型系统的一个最大好处就是：

> 最有意思的组件是那些服务。你同时有了面向服务的架构（SOA），甚至更多。

不论哪一种情况，EDA 和 SOA 都不会彼此不相容或者排斥。它们都能被用来处理复杂事件处理系统，并为你的企业提供自动的或者有效的产出。

原文链接：http://www.infoq.com/cn/news/2008/09/cep-and-eda

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

[人物专访]

毛新生谈 Project Zero 和软件新发展

InfoQ 中文站(下文简称 InfoQ)有幸与 IBM 中国开发中心 Web 2.0 首席架构师毛新生(下文简称 Mao)聊了聊 Project Zero 和软件新发展的相关话题，其中包括 Project Zero 的组织形式、支持的语言、以及未来发展方向等等。

毛新生，现任 IBM 中国开发中心 Web 2.0 首席架构师。此前他曾任 IBM 软件集团企业解决方案部大中华区和北亚地区首席架构师与 IBM SOA 中国设计中心技术主管 ,在企业级软件方面拥有广泛、扎实、深厚的理论功底和丰富的设计与项目实施经验。

InfoQ：先给我们 InfoQ 中文站的读者来介绍一下您自己和您现在做的事情吧？

Mao：好，谢谢。我在 CSDL（IBM 中国研发中心）主要是负责 Web 2.0 的开发工作和研究工作。我们大家都知道说 SOA 之后，Web 2.0 是另外一个很热的事情。在 IBM 来讲也非常关注，我们怎么样把 Web 2.0 在企业里面用起来。我们从 2006 年就开始了，在此之前我是主要是主管 SOA 设计中心，因为 Web2.0 很重要，所以这是切换到 Web 2.0 这边。

InfoQ：给我们简单介绍一下你心目中的 Project Zero 这个项目吧？

Mao：Project Zero 是 IBM Web 2.0 的一个项目，这个项目是非常好玩的。它实际上是它首先采用全新的方式来开发这个软件，所谓的 Open Commercial Software。Open Commercial Software 就是在社区里面其实还有些争议 ,大家有非常赞同的 ,也有觉得这个跟 Open Source 的精神有所偏离。但是从 IBM 的角度来讲呢，它是个非常可贵的探索，它代表了这样一种理念。就是说，我们今天怎么样能够让软件的开发既可以有非常好的奉献给社区，同时可以又让公司有利润、有钱可以赚，只有这个样子，我们才会是一个比较可以持续的一件事情，这是很大的一个点。

另外呢，从 Project Zero 角度讲，它也是一个关于应用的架构，以及应用的形态的一个新的探索。为什么这么讲呢？整个 Project Zero 它就是一个 Incubation（孵化项目），是 IBM

自己通过我们的投资来做的一个尝试。这个尝试后面一个很重要的理念是说，应用走到今天，我们实际上从 Web 2.0 的 Community 看起来，这个软件的形态在发生变化。这个形态上的变化就是牵扯到说我们怎么样去定义应用，应用的整个 Style 究竟是什么样的？是不是我们这个现有的这个企业里面这些应用，社区里的这些应用就是这个样子的，实际上我们都是有好多变化的。今天大概还没有办法做一个结论，什么样是好的，什么样是不好的，未来究竟展示是什么样子的，但是毫无疑义，我们看到 Project Zero 在摸索这个非常重要的点，一个就是怎么样采用动态的脚本语言，让这个程序员的工作变得更加的快速和有效，让程序员的学习曲线变得更加的低一些。

第二个是我们怎么样采用 REST 这种架构的风格，来组织你的应用和组织你的应用群落。REST 这个事情我们不需要多讲，大概技术人员对它们都有比较多的、比较好的一个理解。作为一种新的软件的架构，它使得这个应用程序本身非常的轻量级，非常的 Scalable，而且很 Open。更重要的是说，这样若干个特点将会使得应用程序可以自然而然的构造出来那种开放的、容易整合的，运行起来对资源消耗又比较小的这样一些应用，这是非常了不起的。

另外还有一些，站在这个 Web 2.0 角度来讲，你看到所带给项目的丰富用户体验，丰富的用户体验以 Ajax 技术为基础，它其实意味着 Web 应用的一个巨大的变化。过去这个 Web 应用，大家都了解，浏览器像是一个哑终端，它用 HTML 和 CSS 来描述这个 Presentation 层。但是它本质上并不跟 3270 的 Domain Terminal（域终端）有太大的区别，它还是从属于服务器。将每一次用户的交互都送回服务器去处理。有了 Ajax 为代表的这样一个技术之后呢，浏览器开始摆脱了这个从属的位置，逐渐的变得独立。它从一个独立的可以去部署应用，去运行应用，这样的一个环境和平台，从而导致说浏览器其实可以独立的跟各种各样的这个来自网络中心的服务器去打交道，它使得 Web 应用的结构本身突然之间发生了一个巨大的变化。这个变化它当然是伴随着丰富体用户验来的，可是它对 Web 应用从安全，从计算的分割，从逻辑的分割，从结构上的适配，这些若干个方面都存在着巨大的问题，这个是 Project Zero 所尝试的另外一个方面。

那么综合起来是说，发现 Project Zero 不仅仅是在技术上来探索这些东西，它同时也在探索，我从一个端到端的角度去看，我们怎么样子将这些技术结合起来使用，使得一个应用从设计到开发，到运行，到部署，到管理，到运营，这若干个层面都是非常的轻量级的。换言之，是省钱的，这后面我们会谈到，你后面有个问题谈到说，这个我们将会做些什么，进一步的谈为什么 Project Zero 做的那些事情，使得我刚才讲得这个端到端的整个过程当中都变得非常轻量级、很省钱。

InfoQ：那我们有一个比较感兴趣的问题：就是说为什么要把这个项目命名为 Zero 呢？它有没有一个特别的含义？

Mao：是的，之所以讲 Zero，就是跟我前面谈的这个精神是互信配合的，我们希望是 Zero 的 Complexity，意思就是说它是高度简洁的，人们前去学习它，使用它，整个端到端的过程当中都来得比较容易。可能今天我们还有些距离，但是你知道事情的进展和发展是需要花一些时间的。第二是说，我们希望它这个 Cost 是能够比过去要低很多，你部署这个应用、运行这个应用所需要的 Infrastructure（基础设施），要比过去更轻量级一些，那么这个 Cost 会降下来。我们用 Zero 来形容这件事情，使得用户非常直观的而且具有冲击感的一个感觉，总体上来讲是这样的。

InfoQ：那为什么会选择 PHP 和 Groovy 来支持 Zero 呢？

Mao：这是一个好问题，只是这个取决于说，在社区里面，哪些语言比较流行，那我们知道 PHP 在这个社区里目前来看的话，从动态语言来讲，它应该是最流行的，目前的程序员的量也是比较大的。作为 Project Zero 来讲，我们刚才谈到了动态的脚本语言作为它的一个重要的成分或者应用，我们不支持 PHP 是说不过去的 透过 PHP，这样就使得说这个社区语言，以千万计的 PHP 程序员可以利用 Zero 这个平台来开发轻量级的应用，这是非常有吸引力，同时对这个社区来说是件非常好的事情。关于 Groovy，Groovy 实际上在过去来讲，它发展很快，虽然今天我还没有 PHP 那么多用户，但是在 Java 的社区里边，Groovy 作为一个极其重要的脚本语言，我想对任何一个 Java 程序员来讲，这都是件很好的事情。你可以想象一下，PHP 的程序员通常难得是 Java 程序员，而 Java 程序员通常用 PHP 来讲，在我的观察里边也比较少，所以对于 Java 社区里的这些工程师们来讲，Groovy 看起来是一个很自然的选择，Groovy 通常带来了很好的脚本语言特性，同时呢，还有它又天生的继承了 Java 的特性，对于 Java 程序员来讲，这是非常难得的一个好的语言。这是我们为什么选择支持 Groovy 的一个原因。

InfoQ：那有计划支持其它的语言吗？比如说 Ruby，还有像 Python 这样的动态的脚本语言？

Mao：目前从官方的计划来讲，还没有。我不敢保证我们将来有没有，但是有一点我特别希望能够给你讲一下，就是说对脚本语言的支持，以及其它各种各样新的特性的支持，在 Project Zero 里边来讲，其实都是比较容易的。因为本身，这就是我们前面讲的这个，Community Driven 这样的一个开放模式。社区可以在这个基础上，做它们自己的事情，比如说我们现在有一些合作伙伴，有一个 CRM 的这个系统的开发商，我不方便讲它的名字，但

是它们在利用我们这个平台来做它们的事情，它们在增加、扩展这样的一个平台，这些东西将会成为 Project Zero 社区的一部分，换言之，别的人也将会受惠于这个厂家所做的一些事情。

InfoQ：我们也知道，在这次 IMPACT 2008 上，IBM 宣布了一款名为 WebSphere sMash 的新产品，它是 Zero 的商业版，那么这个商业版和现在我们所谈的 Zero 有什么区别和联系吗？

Mao：非常好的问题，Project Zero 其中我谈到是 IBM 的一个 Incubation 项目，然后它采用社区去开发，同时我也提到说它是 Open Commercial Software，所以 Project Zero 代表的是社区，代表的是大家一起来贡献和开发，大家一起来做，那 WebSphere sMash 它实际上是 Project Zero 给这些 Technology，尤其是说，当 IBM 自己开发出来的一些技术的商业化版本。这个商业化版本就是它代表了 IBM 会对这个商业化版本，进行商业化的运作，进行商业化软件的渠道，技术的支持，以及培训等等。那我们知道，这个 Project Zero，它就像像其它的社区一样，使大家自己帮助自己，你可以去 Download，你可以去学习，你可以去贡献，但是在那个地方大家基本上是以社区的方式来交互，这是很不一样的。

InfoQ：那么在 sMash 这个产品里面，它有一个基于浏览器的 IDE，那么是什么原因让你们去决定要做一个基于浏览器的 IDE，和我们传统的 IDE 相比有什么优势吗？

Mao：这是个非常有趣的话题，至于浏览器这件事情呢，我们花了很长时间才做了决定，事实上我这个小组负责的是这个 WebSphere sMash 或者 Project Zero 里面 Simple Flow 的一些 Toolkit。我们做的一部分也是 IDE 的一部分。在两年前，我们就已经做了浏览器的版本，看起来，这个并不比 YahooPipe 或者说互联网上的一些很酷很炫的 Rich Application 有什么差别，都在同一个量级。我们在浏览器上做了很长时间，让我们发现说，原来当我们谈论轻量级的时候，我们还习惯用一个 Eclipse 去做这件事情，也许重了一些。你去看一看，这个社区里的很多应用，实际上越来越多的人在尝试让人们用浏览器去做一些东西。比如说这个我们看到的 Google Application Engine。你可以下载一些文本编辑器来做，但是同时你也可以在浏览器中用简单文本编辑器去做一些事情。不过我们做的这件事情会比简单文本器功能要复杂的多，要高级的多，比如语法的着色，还有怎么对应用进行管理，以及这个应用的启动、停止等等这些功能，所以这些功能都将会跟我们未来这个 WebSphere sMash 的发展方向相配合，比如说我们会提供这个 Hosting，即像 Software as a service 的方式。

那么在这样的一个情况之下的话，基于浏览器就使得这个程序员可以去做开发，去做应用的部署与管理，这个当然是非常有趣的，也是比较合适的，同时它也是整个 Webshpere

sMash 说，我们希望它和我们这个端到端的轻量级的理念是相吻合的。我想关键的一个技术基础来支持这个决定是因为，还是我们前面谈到 Web2.0 的进展，使得浏览器本身已经成为一个独立的、应用的、运行的、部署的地方，所以它提供了一个独立的基于浏览器 IDE 的技术基础。

InfoQ：我们知道 Zero 对 Dojo 这个框架进行了一个支持，那么我们想问的是，它是否还支持其它流行的 JS UI 库，比如说 ExtJS 这样的？

Mao :Dojo 是 IBM 自身在社区里面推动的贡献 ,Ajax 的一个 Library ,所以 Dojo 作为 Zero 的一个缺省的支持这并不奇怪。那同时我想提的是说，据目前我提到的整个 Project Zero，整个 WebSphere sMashp 它是基于社区的开发方法 ,所以人们希望把 ExtJS ,或者 Prototype、jQuery 等等这些不同的 AJAX Libraries 结合进来是非常容易的。因为整体的架构是非常 Module 的 ,所以当你用其它的 Library 去开发你的应用 ,你可以非常容易的去利用 WebSphere sMash 在 Server 上提供的，因为 sMash 提供出来的这些接口都是 RESTful Style，并不限制你用任何的 Ajax Library。

InfoQ：随着 Ruby on Rails 的成功，现在市场上出现了很多像 Rails 风格的 Web 开发框架，比如说 Groovy 领域就是 GRails，那么和 Zero 和 GRails 相比有什么优势吗？

Mao：这是一个非常好的问题，其实对我来说也是个非常难回答的问题，我非常希望告诉你说，一二三，它有哪些好；一二三，它有哪些不好。那么在我的理解里面，实际上我觉得它们是这个有着相同的目标，就是我们看得到 Rails 作为这个就是“不要去重复你自己做过的事情”，这样一个哲学理念，同时它也采用了这个像 Convention over Configuration（惯例优于配置)这样的一些各种各样最佳实践。那这些东西你会发现说 ,在 Project Zero 里面，在 WebSphere sMash 里面也有很好的体现 ,尤其是像 Convention over Configuration 等等都是非常清晰的在 WebSphere sMash 里面被使用。那整个 Grails 在 Groovy 上来支持 Rails 这样一个方法去组织和架构你的 Web 应用，由于 Ruby on Rails 的成功，所以其实 Grails 也变得相当的流行。事实上我们看到说，在 Rails 里面你定义个 Model，然后它会帮你处理关系数据库的连接，然后提供相应的这个模块让你去使用。那么在 WebSphere sMash 里你可以看到有类似功能的 Zero Resource Model，你也能简单地定义你的 Model，和数据库打交道，做 Persistence 等也可以做得到。从时间上来讲，开发的速度上来讲，开发一个 Rails 的应用，一个简单的应用，也许是十几分钟的事情，那么在 Zero 里面，我也可以用一两分钟给你开发出，所以我不觉得它们在这个简单性、在开发 Web 应用轻量级这个上面，彼此有太大的差别，我认为它们是不分伯仲的，第二是是说哲学，以及各种各样的在社区里面经过这么多

年发展起来的最佳实践，我认为也是类似的，不管是对动态语言的采用，还是说对 Rest 的支持，你也知道 Rails 对 REST 的支持也还不错，看到 REST for Rails Application，我觉得这都是非常聪明的选择。然后对这个 Ajax 的一个自然的支持 ,比如数据 ,尤其是对 Database Driven（数据库驱动）应用的支持都很好。我觉得这些方面，这二者并没有太大的差别，某些方面我认为 Project Zero 会好些，某些方面 Rails 更好一些。

那从另外一个角度来讲 ,我觉得 Project Zero 会比它考虑的更多。因为你会发现说 ,GRails 更多的是应用的框架，它本身并不去考虑端到端的故事，从设计到开发，然后怎么去部署，怎么去运行，怎么去运营。这一系列的事情，Rails 更多的侧重于说，我有一个应用，那 MVC 的方式去组织它，然后你定义一个 Model，我给你一些必要的 Utility，使得这件事情必要的 Convention ,使得这件事情变得比较容易。当然你如果需要去做进一步定制的时候 ,其实 Rails 会给你一些压力，可是在，它在 Project Zero 里面，首先来讲的话，它着眼点并不是一个某一种类型结构的应用，它着眼的是一个非常通用的平台，在这个平台上你可以做各种各样的应用，换言之说，今天我可以在 Project Zero 重新构造出一个 Rails 的框架来，然后你可以在 Project Zero 上以 Rails 的方式去做这件事情，但是你只是 Project Zero 去支持的，一种情况，它可以支持各种各样的情况。比如说我现在纯粹是 Ajax 加 REST style 的应用，我们可以支持的很好。而在这样一个基础上，你可以演变出各种各样的模式出来，换言之说，我们实际上需问程序员一个问题，是不是 Rails 就是唯一的 Web 应用结构上的方式？我怀疑，很多的程序员都可能会给某一个答案，说，“No，that’s not true.”所以，更重要的情形是说，Project Zero 我觉得说 ,相对来讲 ,很大的一个区别。打个比方说 ,我们讲得具体一点 ,当 Project Zero 考虑 hosting environment 的时候，这是一个非常有趣的话题，我们怎么去支持 Multi-Tenancy（多重租赁），我们怎么样去支持大型系统的可扩展性，我们怎么样去很好的支持安全性，等等 ,这些若干个问题出现的时候 ,这都不是亲爱的 Rails 想去解决的问题 ,但是它是 Project Zero 需要去面对和解决的问题，所以我想那些方面都是很重要的一个差别 。

InfoQ：OK，那我们还回到 Zero 项目本身来去谈一下，我们也知道 Zero，它采用的是非常开放，但它不是开源的方式来组织的。为什么会采取这种形式？

Mao：前面我已经很诚实的提到这个问题，我希望有一个非常好的，每一个人都喜欢的答案，很可惜没有。实际上，这个里面，一直有一个悖论，就是说我跟中国很多开源，搞开源的一些公司的朋友聊过，他们也都非常的痛苦，就是说今天，我怎么样既通过开源很好的回馈和贡献给社区，同时我又可以生存下去，其实它一直是一个非常艰难的话题，我想我非常诚实。这个话题不光是在开放的社区里面，在努力和奉献的这些可贵的朋友们的话题，同

时它也是 IBM 这样一个，IT 行业里面最大的公司一直在思考的话题。IBM 在历史上做了无数的开源方面的贡献，从著名的 Eclipse 到 WebSphere Application Server Community Edition，到数据库，到 Linux 系统的大量的支持，IBM 在 Open Source 方面的贡献几乎涵盖了所有的范围，而且我怀疑它可能是最大的 Open Source 这样一个捐献者。事实上 IBM 并没有停止它的脚步，IBM 一直还在努力的在开源这个方面，一如既往的在做贡献。比如我们刚才提到了 Dojo，IBM 其实是 Dojo 最大的一个贡献者，同时 IBM 也是围绕着 Ajax 发起的交互性和规范化方面运动的首倡者—— OpenAjax(http://www.openajax.org/)。我不想举更多的例子，我只是想讲有一件事情我们需要严肃的面对，当我们既需要透过 Open Source 的方式来集合社区的力量，使得我们可以做出非常好的东西，同时使得人们拥有这样一个自由，去得到这些知识，去传播和使用这些知识。但是它可能并不意味着全是免费，事实上 Open Source 是 free software，并不是 money free，指的是 freedom，你有这样一个自由，追根溯源，其实我们都需要去问这样一个事情，什么样的一个模式可以使得软件这个行业，青春永驻，每一个人都可以很好的贡献，同时又可以生存和发展下去，我想这是在整个这个生态系统里面，每一个程序员都在思考的问题。我不认为今天有谁会给我一个非常完美的答案，但是有一点是非常清楚的，那就是我们需要发挥我们的倡导力寻找到一个合理合适的方式，我们不知道现在做得方式是不是最好的方式，或者是不是未来一定这是一个答案，它是一个探索，是一个尝试，就是为什么我们一直在讲 Project Zero 是一个孵化项目，这个孵化项目不仅仅是技术，所以请大家不要只是看到它是 Dynamic scripting，在讲 REST style Architecture，在讲 Feed，在讲 Ajax。同时它也是在探索软件业新的形态，怎么样去支持 Software Service，怎么样去端到端的轻量级，更重要的它也在探索非技术的方面，那就是软件形态和它的商业模式与开发方式究竟将走向何方。我想这是 IBM 作为一个 IT 行业的 Leader 应该去思考的事情，同时我也非常希望我们的朋友们、来自社区的这些个同行们，如果大家有很好的想法，我非常希望有机会跟大家一起去交流，我们完全有这种灵活性，来调整我们的做法，事实上今天的这样一个做法，它在努力的尝试给予这样的灵活性，我们一方面通过 Project Zero 来给社区足够的开放性，可以去 contribute，可以去 download 我们的开发计划和我们的 source code，你可以自由的尝试，但是我们与此同时也提供正规的商业版本，使得我们可以按照常规的商业方式来提供支持、教育和各种各样的后续性的开发。

InfoQ：那么我们知道目前在 Zero 这个项目上面有两个 PHP 项目已经能运行，比如说 Sugar CRM 和 PHPBB。那么，我们想问的是，是否所有的 PHP 应用，目前都能够部署在 Zero 上面，如果说还不能的话，什么原因？

Mao：我是多么的能够希望告诉你这个答案是 Yes，但是不是的，它是 No。这个原因很简单，目前我们 Project Zero 支持是在语言这个级别上对 PHP 进行支持，换言之就是说，你可以，它认识 PHP 语言，但是我们知道 PHP 不仅仅是一门语言，它还有非常非常多的，我们称之为 Library，非常非常多的这些 Library，作为整个 PHP runtime 的一部分，它们实际上是社区里面无数的这些可爱的朋友们贡献出来，分享出来的。那些东西呢，也是 PHP 这个语言非常有趣的、有价值的部分，现在我们 IBM 支撑的力量，就是我们也没有，到现在为止还没有这个能力和精力去做一些 PHP Library 的支持，你知道很多 PHP Library 都是用 C 写的一些扩展的话，事实上 Project Zero 它是以 Java 为基础的这样一个运行时间基础，所以做这样的一个移植呢，它需要花费很大的力气的，那么我们特别欢迎社区里面的朋友，大家为第三方软件开发的厂家来所用到的各种各样 PHP 的 Library 逐步的做一些必要的移植，使得它们可以在 Project Zero 的 Java 环境里面，去做到同样的效果。我想这件事情，应该会在 Project 社区里面慢慢的发生，比如说 SugarCRM，PHPBB，实际上就是第三方的软件厂家它们很认可 Project Zero 这样一种哲学、设计理念，它们愿意把它们所用到的东西，这样换言之就变成是说 SugarCRM 做完之后，它的好多东西让社区受益，我们希望看到更多的社区人员加入到这个行业里面来。

InfoQ：那么接下来，我们就畅想一下未来，Zero 它的下一步计划是什么？

Mao：Zero 下一步有非常多的内容，我现在讲几个比较主要的、重要的内容，第一个部分就是关于我们前面谈到的怎么样子让它们运行、运营基础轻量化，这个部分其实我们有一个名字叫做 New reality runtime，也不知道怎么翻译比较好，它谈的内容是什么呢，它谈的是说，我们可不可以让 Java 的虚机非常快速的重启，可以在一台机器上启动很多个实例，然后呢，这样第一是说，我们希望这个 Java 的虚机呢，运行一个应用就好了，不要运行多个，换言之，我一个虚机运行一个应用，这个变化是很大的，你想过没有，J2EE 是一个 Container 形式的，所以一个 Java 虚机运行多个应用，对不对？那运行多个应用的话，你会发现对程序员来讲是个巨大的挑战，因为它要有并发的问题，它要有这个怎么样去做区隔，做这个 isolation，对内存的访问、对资源的访问、对各种各样东西的访问，这会带来很多的麻烦，然后其中有一个人要是坏了的话，按照俗话所说一只老鼠坏了一锅汤，他坏了，他把内存渗漏了，或者是他怎么样了，然后整个 J2EE 的 stack 就 crash 了。事实上有些东西我们可以从 PHP 上学习一下，你看 PHP 要支持一个应用，人，噌，就起来了，很快，毫秒级别就起来了，运行完了，人家就没了，它不需要 container。所以对我们讲，这里面有一个观念叫做 application as the server，也就是说每一个 application 是一个独立的应用，它有自己完整的

stack，这个应用坏了不影响另外一个，可是这件事情对现有的 Java 虚机，提出一个巨大的挑战，我们过去的 Java 虚机，一台机器上启动两个 Java 虚机就了不得了，在上面 run 很多很多小应用，大家来分享这个机器，现在突然想说，噢，在一台机器我要 run 上百个应用，然后每个应用都是独立的，这个是新话题，这是 IBM 在 Java 虚机部分希望做的一些优化和改善，从而使得程序员可以拥有一个比过去简化的，多的多的运行时环境，与此同时，这样的一个好处是带来说 ,程序员可以非常清楚的看到和用到了什么 ,而不是像 container 里面，基本上你只是看到这个 container 本身的这个接口层层面的东西，也看不到 container 之外的一些东西 ,那个对你是一个黑箱子 ,我们大家都有 J2EE 的一个应用 ,出了一些问题的时候，我们去调试痛苦的经历，那是不容易的一件事情，因为它意味着你需要对整个 J2EE 的所有东西都有非常深厚的理解，而且很不幸，那个部分的东西实在是太多了。所以我们讲这样的话，站在我们角度来讲就是说，怎么样就使得程序员看一个相对比较透明的环境，我用到哪些东西 ,这个东西是怎么运作的 ,我非常清楚 ,由我控制 ,这个实际上也是带来这些 simplicity 的一个重要基础。换之言说 JVM 拥有若干个目标 ,希望使得整体的 Project Zero ,在轻量级，在编程的简单性这些方面能够提供足够的支持，那个是我们所谈论的 New Reality Runtime。与此同时，我们前面谈到过说，Project Zero 将会支持 Hosting，将还会更加自然而然的支持 software as a service 的软件模式 ,今天就我们的中间件来讲的话 ,要去支持这个 Multi-Tenancy 还是需要做进一步的工作。

那么，如何使得能够建立一个高度轻量级，高度可扩展，高度 Scalable 这样的 runtime 基础 ,它可以对 software as a service 这样的一个 hosting 的环境以及以 hosting 方式来开发做一个很好的支持，这是我们下一步要继续做的。这件事情其实同时也牵涉到我们怎么样，前面提到，一个 new reality runtime 怎么做 clustering，一台机器进去了，它自己就开始工作，然后怎么样可以开始自己更多的访问的人。所以这些都是 Project Zero 下一步要去做的。另外 IDE，这个 IDE 将越来越多的更加绚丽的功能，比如 Web IDE 支持你开发更多的 widget，这个 widget 自然而然的部署在其它的环境和平台之上，这个是我们期望去做的。还有就是说我们的 zero resource model，比如说在 persistence 等等方面的支持。所以总的来讲，Zero 下一步将会进一步推进简洁的 Scalable 轻量级的，这样一个越来越多的 Web2.0 风格、技术和最佳实践融合进来的这样一个方向去发展。

观看采访视频 ,请移步至 :http://www.infoq.com/cn/interviews/maoxinsheng-project-zero

[推荐文章]

跌跌撞撞的持续集成之路

作者 仝键

推荐理由：作者在自己的亲身经历中，摸索出了一条属于自己的持续集成之路，最重要的不是结果，而是作者探索而最终做出决策的心路历程。

> “天下事头绪纠缠，兴一利必也生一弊。”

一句话，道破了改进难点所在。最近在项目中围绕持续集成做改进的时候，对这一点感受颇深。跌跌撞撞的一路走来。我们的持续集成的过程已经变得有些“个性化”，反过头来看我们一路的变化，非常有意思。

从项目的技术架构说起，我们的项目是采用的 J2EE+Flex 的方式进行开发的。在我进入项目组的时候，一个比较健壮的持续集成环境已经搭好了。工程分为两个，一个是 Java 后端的工程，一个是 Flex 前端的。我们的持续集成服务器是 CC。整个开发工作是围绕着持续集成展开的。一周为一个迭代。

那个时候，我们采用的是比较标准的方式：

> 后台采取 TDD 的方式开发。
>
> 每次提交代码之前更新所有代码，然后运行所有测试用例，全部为绿色的时候才提交。
>
> 前台 Flex 比较麻烦，所以采取了用功能测试覆盖单元测试的方式。用基于 Ruby 的 FunFx 写单元测试。工作方式与后台差不多，每次前台功能测试全部通过了才提交。
>
> 持续集成的流程是每隔 5 分钟检测一边代码库，有更新就 build。
>
> build 的流程是先编译后台，跑单元测试，单元测试通过了，再编译 Flex，将 swf 和 html 以及后台的文件打成 war 包，部署到 tomcat 上去，跑功能测试。

build 成功之后发布到特定的目录，形成发布列表。有 war 包供人下载。

那个时候，build 一次大概是 15 分钟，因为 Check In Gate 环节是按照标准流程走的，所以 build 出错的几率也小。CC 大多数时候是绿的。哪怕偶尔出问题红了，也很快被修正了。

随着项目的开发，代码规模越来越庞大。功能测试越来越慢，比起自动执行脚本那种速度，开发人员更乐意手动点两下，加之上面对工作进度的压力。更改了一些工作方式：

后台的工作方式不变。

前台，将功能测试脚本的工作交给几个测试人员编写。几个测试人员也坐在附近，基本可以看作团队成员（到后来编制也是我们团队的成员了）。 开发人员人肉测试一下保证没有问题了再提交。

测试人员写脚本的流程是拿到上一次 build 成功的 war 包，在本地写脚本，本地测通过了再提交。

持续集成服务器上功能测试不通过的时候，由测试人员提交 BUG，开发人员修改。

持续集成的流程依旧。

这样，从后来收集的数据看，工作效率是提升了，因为参照以前的统计，开发人员的工作一下减轻了 1/3。以进度来衡量的速度自然很轻易就可以让上级满意了。

新的解决方案总会产生新的问题，测试人员在测试方面的专业性，使得他们写出来的脚本测的更细。功能测试的时间占耗，增长的更快了，短短半个月，就增长到了 1 个小时。每当出现问题，作出反应之后，要在 1 个多小时以后才能知道结果。而且，持续集成方面并没有做到，一旦出错，谁也不能提交代码这么严格。模块化的设计所带来的假想的安全感和进度的压力，使得开发人员修正问题的激励不高。于是修正问题的速度不如产生问题的速度快。持续集成服务器在那两个个礼拜里只有两头是绿的，周一早晨和周五下午。

最早，我们发觉，由开发人员重构造成的脚本失败占大多数，而测试人员每次拿到的上一个版本是没有错误的。所以会出现自动化脚本本地跑得过，服务器上跑不过的情况发生。于是我们修改了发布的逻辑，在后台单元测试通过、flash 编译完成的情况下打的那个 war 包，复制一份，放到某特定目录指定为 current build。供测试人员写测脚本使用。

过程改进之后，测试人员可以快速的修正脚本了，虽然对于开发人员重构造成测试人员工作的返工无疑是一种浪费，但是毕竟自动化的测试省了回归测试的不少时间，还是可以接受。

脚本的修正速度解决之后，工作似乎有了些起色，但很快，问题的本质就暴露了出来--build 的时间太长了，修得速度还是跟不上问题产生的速度。尤其是中间缺少当 build 失败时强制阻止代码提交的环节。这之后依然是周一和周五两头绿，中间都是红的。于是，我们觉得问题还是出在 build 速度上。我们人工的将功能测试脚本分到四个 suite 里去，然后以多线程的方式运行。速度被提高了 4 倍。于是又消停了两天。

好景不长，多线程的测试似乎不太稳定。很多本地可以跑通的测试用例，到了服务器上就失败。险些一个礼拜都没有 build 出一个版本。最后不得不改回单线程。这时，build 一次已经占到了 100 分钟。第一期的产品 Backlog 还没有完成 1/3。

持续集成走到这里已经进入一个困境，有必要做一些更深一步的改进。经过多次讨论，归纳出了几套方案：

分冒烟测试和 all test 两套测试用例集是我们当中呼声最高的一种方案，当我的代码提交之后在跑完所有单元测试和基本的冒烟测试之后就发布 beta 版，由测试人员接到 beta 版，进行更细致的自动化测试并带一些人肉测试。但是反对的声音认为，不跑完全部的测试用例就失去了持续集成的意义。而且会更降低开发人员修正 Bug 的积极性。于是作为修正，支持的声音则提出，在 Check-In Gate 处把关，恢复每个人提交代码之前跑测试用例的实践。可这明显会给开发人员带来更大的工作负担，估计以此时的进度压力，开发人员的安全感肯定会大幅下降。很可能会推行不下去。

另一个方案是从细节处调优，把 WEB 应用部署到另外一台机器上去，或许就会稳定一些了。但是反对的声音认为，以测试用例的这个增长速度，他早晚会不稳定的，而且可能撑不过两周。作为修正，想考虑分布式，但是我们所有人的知识储备中，并没有一个人清楚 CC 有没有分布式能力。所以想的是购买 Cruise，但是价格的障碍就摆在眼前了，在项目前景还不是很明朗的情况下，估计很难申请到资金，但也不是不可能，只要我们敢于冒这个风险。

第三，便是更为高级的分支式开发，将版本库划分一下分支，以分支来搭配持续集成，以分支合并来触发自动构建。这样做，开发的过程就更加有板有眼，粒度可以划分的更细。可是分支的划分，一时想不清楚。但是假设想清楚了，似乎这也使得我们的工作流程更加复杂了，做如此之大的改变，风险有多大？效果有多大？

成本有多大？到底是值不值得？一时也想不清楚。

方案有了，听起来都很有道理，也都有问题。该如何做出决策，是个问题。现在大家众说纷纭，都有理，就变得难以抉择。而且到现在了是不能随便尝试的，这种尝试也是一种风险，一旦出问题造成的成本上升都会加大我们身上的压力。

迷茫之下到 AgileChina 上跟大家讨论了一下。非常高兴的收集到了几方面的建议：

Jeff Xiong 觉得可以将测试分级，并将 build 分为两个环节，一个跑基本的用例，一个跑全部的用例。这跟我们的第一套方案的思路吻合。但是这种行为是不是失去了持续集成的意义呢？他也不是很确定，他说：

> 我也不确定……不过，不全面的持续集成至少比不能用的持续集成要好。哪怕一个 quick build（只？）能抓到 80%（70%？60%？50%？）的 defect，如果它只要很少的时间就能跑一遍，似乎值得这样做。
>
> 不过同时就需要(可能是专门的)人来关注 slow build 的健康状况 ,不然 broken functional tests 可能被忽视并累积。

因为是在论坛上，互相之间的交流容易造成理解上的偏差，我便阐述了一下我的理解：

> 嗯……也就是说分一个 fast build 和一个 slow build 然后有专人关注 slow build。
>
> 那我的理解，这个 fast build 应该是反映了后台的健康和前台与后台的基本集成的健康。主要用来完成保障集成的角色。
>
> 而 slow build 则是反映了从用户接口来看的软件的健康状况，定期回归，防止发生过的错误再次发生。
>
> 这样逻辑上看着很清晰，但是两者之间的同步……会不会有什么问题呢？

Jeff Xiong 认可了我的理解，并提出了更进一步的解释和建议：

> slow build 实际上是运行完整的回归测试套件。当然理想的情况是 slow build 基本上不出错，因为*逻辑*用单元测试都覆盖到了，功能测试只是在描述*表现形式*。那么因为 slow build 基本上不出错，就没有价值每次都去运行它，让它在后台慢慢的跑着，过一段时间（半天或者两小时）去关注它一下，没问题就好，偶尔出了问题就马上解决并且加上对应的单元测试。这样你既节约了时间又不会严重降低对质量的保障力度。
>
> 实际上的情况可能比较难这样理想，但是和所有好的环境一样，这个环境不是

> 说一下子规划好就万事大吉的。你可能大概的分一分，然后不断的维护，在两组 build 之间交换测试案例，一些覆盖到大量功能的、经常出错的案例也许要换到 fast build 的冒烟套件里面，一些看起来永远不会出错的案例也许可以换到 slow build 去。一直琢磨这个事，它才会变得越来越好。
>
> （题外话：最近我觉得稍微大一点的项目应该有比较专注的 build master，developers 往往并不是特别认真的考虑 build 和 CI 的持续改进。）

而胡凯则提出了一些基于 CC 采用分布式的解决思路：

> CruiseControl 是支持一个叫做 Distribute 的 Contrib，它的 Idea 是：因为 CruisControl 支持叫做 Composite 的 build 方式，那么你可以起多个 build server 一起来 build 同一个项目，当且仅当所有的子 build 通过，整个 build 才算成功。
>
> 对于每个 build server，你是在单线程测试，对于整个项目，你却是并发测试，因为有多个 server 同时在跑测试。

但是他也说到 CC 毕竟是开源的东西，配置起来十分麻烦，于是最后也提出了采用 Cruise 的方案^_^:

> 或者你也可以尝试一下 ThoughtWorks 新发布的 Cruise，基本的理念很相似，都是把测试分布到不同的机器上执行。
>
> 在试用期内可以同时跑 6 个 Agent.你可以在每个 Agent 上执行不同的 Target 比如
>
> Agent1 : ant ft.suite.1
> Agent2 : ant ft.suite.2
> Agent3 : ant ft.suite.3
>
> 你可以拿来玩儿下，根 Open Source 的那个比起来，应该容易设置很多。
>
> 缺点是：
>
> 你必须使用 hg 或者 svn 作为 SCM
>
> 试用期过后，你只有 2 个免费的 Agent

另外，糖醋鼻子还提到了分支式开发，大家围绕这个还展开了激烈的讨论，不过我考虑到分支式开发对我们的利可能要远小于弊，最终还是放弃了这个方案。

在吸取了两方面的建议之后，经过一番思考，决定开始两方面的准备，一方面，对测试

用例的分级方面做了一些工作，重构了一部分用例的结构。另一方面我去调研分布式构建的实现手法。CC 的配置果然非常麻烦，调研期间发现了有 RemoteAnt 这个东西，试了一下基本满足我们的需求，考虑到一个 Agent 不用做的太过重型，于是就采用了这个方法。在分布式的技术调研已经完成的情况下，测试分级要不要做成了一个问题，但考虑到目前需求只作了 1/3，如果这个都抗不住要分级的话，后面的工作就没法做了，所以分级的事情，虽然做了，但是也暂缓实行。

现在实践已经采用，会不会产生新的问题呢？前文说过“兴一利必也生一弊”，这个事情是肯定的。那么问题就来了，既然弊端肯定会滋生，我们怎么知道作出的决策是正确的呢？

其实，倒回去看这一路走来的过程，除了一个可以运行的过程以外，还有一个很重要的收获，那就是：如何进行决策。而所谓决策，并不是在黑白分明的事情之间做出选择，而是在都有理的事情中做出选择。就像我们都知道做软件设计的时候，设计是没有好坏之分的，只有适不适应你的具体情况之别。所以当我们面临几套解决方案的时候，就好象面对几套设计方案一样，真的是很难选择。如何做？各自就有各自的思路了。像我们就吸收了敏捷开发的思想中所强调得不做过度设计。选择立杆见影的改进去做。同时，像前文所说，抱着拥抱变化的态度，相信兴一利必生一弊并不是坏事。相反，他可以从一定程度上，带领我们找到真正的问题。

作者简介：仝键，网名咖啡屋的鼠标，06 年大学毕业，普通程序员，专注于 Java、Flex 方面的开发、Agile 等软件开发方法论的学习。爱好参加社区活动。

原文地址：http://www.infoq.com/cn/articles/road-of-ic

[推荐文章]

构建的可伸缩性和达到的性能：一个虚拟座谈会

作者 James Cox 译者 宋玮

推荐理由：延续 InfoQ 传统，请重量级的人物谈重量级的问题——请 Twitter、eBay、Betfair 和 FiveRuns 的人来谈可伸缩性和性能。"性能是服务于一个单独请求时的资源使用问题。可伸缩性是当要服务更多（或更大）请求时资源消费量如何随之增长的问题。""语言是不能伸缩的。框架是不能伸缩的。而架构是可以伸缩的。"

实现伸缩性和性能调优的经验所保有的价值容易被低估。两者都是“晚些时候”或“我们真正流行起来时”所面临的难题。早期创业公司确实不应该立即花这份钱，而大公司通常不能做出足够快速的反应以实现所需的改变。再加上这需要一个多学科的团队，它立刻就变成了一个需要解决的政治上和工程上的难题。

但是它从来不会从我们的视线中消失——在过去的少数几个 Qcon 会议上，“Architectures You've Always Wondered About（你始终感到惊讶的架构）”专题被淹没了，正因为如此，我们渴望学习“大人物”们如何去做这些事情的技巧和诀窍。

在这次 InfoQ.com 的虚拟座谈会上，我们从几个最大的公司和项目中邀请了几个在实现可伸缩性和性能方面的专家，让我们进入他们取得了成果而我们却依然梦想的神秘世界吧。

座谈会的参与者包括有 Blaine Cook，Twitter 的前任架构师，他领导开发者进行了 Starling 消息队列 ruby gem 的开发。Blaine 在“如何让 Ruby 和 Rails 伸缩自如”的会谈中发出了积极的声音，在这一领域有非常多的经验。

他是被 Randy Shoup 加入的，Randy Shoup 是 eBay 市场架构小组的一个分布式架构师。从 2004 年起，他一直是 eBay 搜索基础架构的主要架构师。而 Matt Youill，则是 Betfair 的首席技术专家。在超过 6 年的时间里，他引领了许多 Betfair 的技术架构。最近他又与 Betfair

的高级技术小组转向一个更长期的高级项目——包括了 Flywheel 技术及其他技术的策略、研究和特殊项目。

为了让与会者来源更多样，我们请专业开发 Rails 性能工具的 FiveRuns 也参加进来。他们派了整个工程团队用了一个下午来商讨其答案——简直可以另外开一个座谈会了。

问题 1：许多人把性能和技术混为一谈。你们怎么回应这种误解呢？

Randy：我同意独立于任何特定语言或框架来讨论伸缩性问题确实很有价值——不管实现策略如何 ,模式都是一样的。可是 ,首先让我们确保自己已经把性能和可伸缩性区分开了。性能是服务于一个单独请求时的资源使用问题。可伸缩性是当要服务更多（或更大）请求时资源消费量如何随之增长的问题。

它们是相关的，但是不是同一件事情:-)。幸运的是，许多改善其中一个问题的方法通常也会改善另一个问题。但是，速度非常快的系统可能并不是最具伸缩性的系统，反之亦然。

FiveRuns：我们关心性能、可伸缩性、可用性的对比。我们把性能定义为一个操作（或一些操作）如何快速地完成，比如，响应时间、每秒事件处理的数量等等；而可伸缩性——就是应用怎样很好地按比例扩大规模以应对更大的使用需求（比如，用户数、请求速度、数据容量）。

问题 2：当你碰到性能瓶颈时，你如何开始调查是什么导致了这一问题？

Matt：每个问题都是不同的，但是我想这通常是一个搜集观测资料（即，与许多人进行交谈）并缩小问题范围的过程——通过检查什么仍在继续执行，你可以分离出不再执行的东西。

有两个重要的事情要谨记在心。确保它确实是问题的瓶颈所在。例如，忧虑处理器使用效率低下的问题与一个受 IO 束缚的应用可能没什么关系。从产品环境去进行观测非常重要，不要在阶段环境或开发环境亦或是其他环境上进行观测。试图在“假”的环境里再现问题并收集信息将只能产生“假”的结果。

Blaine：直觉（快速但不可靠，而且容易产生错误）和监测（需要预先的工作，但是这能让你随心所欲地分析问题）。人们经常谈论测试驱动开发，但是可伸缩性需要测量驱动（metric-driven）开发。

像 JMeter 和 Tsung 这样的工具 ,可以让你模拟负载 ,但是接受任何事情都要有所保留。如果你有很长的时间（到底多长时间取决于你的流程和构建测试的人）去构建非常周到的测试，你可以获得更好的一手资料。但是，不可能做到和真正的流量一样逼真。

Randy：通常我们会从注意到问题的地方开始，然后向前追溯。在应用栈的所有层次所

进行的监测/勘测得越多，就越容易这样做。它还可以帮助有各种学科专长团队里的人们来检查这个问题。

问题 3：当处理 Web 应用的问题时，你发现什么工具最有用？

FiveRuns：在浏览器层，firebug 是我们所选的工具，因为它提供了对用户执行单页面的感觉的一些深入观察。我们还喜欢 ab & httperf，它给我们提供了一些自动化的负载测试，因此对比于我们从 firebug 中获得页面级的计时，使用这个工具可以获得会话级的计时。

Matt：我们混合使用成品工具和定制工具。成品工具是非常好的，但是成品意味着它们是通用的。它们没有你应用程序中非常重要的特定标准的知识。例如，在 Betfair 的 Web 应用上下文里，这可能与请求频度和含有相应业务事件（就像足球比赛中的射门、进球）的特殊页面类型相关。通用工具没有“足球比赛”选项。

Blaine：Ganglia 是非常优秀的。同时 Nagios 或 Zabbix（举个例子）将告诉你何时资料遭到破坏，使用少量加工你就能够让 ganglia 给你提供任何东西。对于 MySQL，Innotop + slow query log 帮了大忙。使用 mysql 微妙级日志补丁来察看所有查询中哪些查询的时间在 1 毫秒到 1 秒之间。

Randy：我非常同意使用日志——我们可以支配的最有用的工具是我们的监测框架，我们的应用服务器使用它来记录我们所做的各个操作的调用/事务发生时间——整个 URL 的执行情况、嵌套的数据库访问、嵌套的服务调用等等。这就让我们能够远程诊断大多数产品问题，而不需要在任何地方都配备调试器。我们还保留了这些日志的可查询历史，我们可以用之比较以前和当前的性能，以及随时间推移的趋势。

问题 4：当你有一个堆栈（或核心）问题需要诊断时，什么工具可以提供帮助？

Randy：GDB 和 DTrace 是用于 C++的基础架构。core 或 pstack 是个颇有价值的工具。对于 Java，我们使用了 tracing 和各种 Java 调试器，它们组成了 eBay 基础架构的主要部分。可是，调试器不应当是你用的第一个工具；它应该是你用的最后一个工具。产品代码需要用足够多的仪器来远程诊断问题。

Matt：我们有许多魔法！我们使用各种工具来重现问题并调试它们（包括栈的问题）——Visual Studio、 Eclipse、 WinDbg、 cdb、 Purify、 Fortify、 dtrace 以及许多定制的东西，为我们的架构所构建的东西。

问题 5 :你认为像 Yahoo!性能前十名等等这样的文章有用吗？领域或应用的可伸缩是个问题吗？

Blaine：这两者的答案都是肯定。从某点上讲，伸缩性已经从领域问题（即，如果你不

使用内存缓存或者一个等价的分布式哈希表和基于内存的缓存）转移了，而你仍然处于“领域”范围。一些应用充分地理解了这一点。当今静态内容的可伸缩性已不那么重要了，那只是花钱的问题并需要公司有好的社会组织的问题。

FiveRuns：我们认为像 Yahoo!性能文章这样的资源是有用的。我们认为把应用架构与问题领域进行匹配是非常关键的——实际上，我们对问题领域的所有假设都定义了软件系统质量方面的约束。

换句话说，我们都期望问题领域可以定义针对性能、伸缩性、可用性等等的预期。当然，需要在这些（或其它）质量之间进行权衡，比如，功能性和上市时间，它们将能显著地改变问题领域。

Matt：当然，所有有用的素材都在这些文章中了。虽说这样，完成一个高质量（性能）系统是相对的——我从未看到关于如何维护品质一个好的建议——这是治理部分的内容。看上去 IT 系统似乎注定一开始很鲜亮、然后衰退、最终消亡，接着再全部重新构建一遍。连确保它们至少偶尔有一个漂亮的外观都很难。

问题 6：实现可伸缩性和性能调优通常被看作是一种“救火”措施；目标是是立刻修正问题。你们在一个成熟的代码库上是如何跟踪性能衰退的？

Randy：在 eBay，所有项目都经过了严格的负载和性能测试阶段。在将项目发布到网站之前，我们使用这一阶段来检测并解决潜在的性能问题。

Matt：我们经过的是一个不同的过程——我认为这些都是在应用程序运行的时候才能检测到的。要确保一个应用被有效地划分并部署到真实环境的服务器上。如果一切正常，再把它部署到另一个服务器上，然后再是另一个，等等。确保在早期初次公开展示过程中，你投入了有效的监测和管理基础架构以捕获性能问题。

不要试图在部署之前就捕获性能问题。你不可能重建真实环境中的条件，因此你不可能得到真实可靠的测量结果。

Blaine：是的，监测。非常仔细的监测。我提到的是监测吗？监视迟缓现象。当你看到一个衰退时，回去检查一下你的改变记录。应该很容易定位，因为你通常所做的是增量部署，不是吗？

FiveRuns：我们得补充一下，适当地建立一个基线是很重要的：一个已知的好的性能表现。并且试图梳理出任何交织于最终用户功能瑕疵的性能问题——有时这些瑕疵看起来只和功能有关，但实际上也包含了性能问题。

问题 7：那么捕获问题的真正重要的衡量标准是什么？

Randy：衡量标准就是那些影响你业务（例如，页面负担、用户响应时间）的衡量标准，以及使你受限（例如，内存、CPU、网络带宽等）的衡量标准。正如你可能想象的，eBay 基础架构的一些部分是受 CPU 限制的，另一些是受内存限制的，还有一些则是受 IO 限制的。当然，墨菲法则（一种幽默的规则，它认为任何可能出错的事终将出错）确保了你没有严密跟踪的衡量标准就是那个对你不利的标准！

Blaine：取决于你的应用。任何事都有响应时间。找到你认为花了最长时间的地方，对它们进行计时，如果结果随你所测量的比特位数的多少而非常不同，找到它们。页面装载时间在高于 250ms 时都是很重要的问题。在这个响应时间之内，用户（通常）都看不出差别，而任何优化（例如，针对 API）都是性能和成本的优化。

了解集群的当前状态；如果是超出了集群的处理能力，那就去增加处理能力，句号。没什么好说的，去买更多的机器就是了。不要再阅读本文了，去买更多的机器吧。如果处理能力是一个影响因素，而且你的应用并没有大的问题（无明显性能问题，你已经调优了 mysql 等等），去买机器吧。你可以让它运行得更快并节约了后面的花费，但同时，购买机器意味着你将减轻很多压力并给自己一个喘息机会以对应用正确实现可伸缩性。

Matt：确保它与你的业务相关，确保你所采取的测量是有意义的。除非你知道当时正在执行什么业务功能，否则一个 CPU 测量是无意义的。

FiveRuns：我们全体都同意最终的衡量标准是捕获了客户期望响应的那一个。通过响应，我们真正把整个系统中的其它元素汇集到了一起。例如，gmail 比其它邮件服务递送邮件更加迅速；因为它的“响应”（例如，从邮件被发送到在 gmail 中看到邮件之间所花的时间）满足了用户的预期，其它技术上的标准是无关紧要的。

问题 8：性能对应用程序的总体“健康”能起什么作用？

Matt：性能是一个重要的部分，但仅仅是部分。虽然旧了一点，但 ISO 9126 依然是建立有关测量系统的衡量标准的很好入手点。

“健康”是个很有意思的概念。在医学里，身体里每个细胞的性能不能够被测量，但是大量细胞汇集在一起就有了“生命信号”，如脉搏、温度、血压和呼吸都是可以测量的。大系统应该以类似的方法进行测量——单个服务器的内存使用与整个系统在线用户数相比起来没什么用处。每个大系统需要一些关键的生命信号。

Randy：性能绝对是基础。差的性能直接反映了底层基础架构的花销，以及用户的体验。这两者都是底线。一个性能低于预期的应用不是“完成”了的应用，一个缺乏必须功能的应用同样是没有完成的。

FiveRuns：通常，我们都同意把性能不佳看做是一个大 bug，而当作一个大 bug，我们还得关注存在大 bug 的地方，这可能是另外一个大 bug。因此，我们认为健康是依赖于用户/所有者体验的一种东西。

可是，也有反例。例如，针对很小一部分受众的一个意在提供一些新颖功能的 Web 应用，其可伸缩性问题不被计入到应用的“健康”范畴。如果是一个工业级应用，比如 gmail，它一向有性能（或伸缩性或可用性）问题，那么其可伸缩性就给应用“健康”的带来了消极影响。

Blaine：有时一个应用就是慢，因为它要处理很复杂的事情，加快速度是不可能的。这变成了一个交互和负载容量问题。有时应用慢是因为代码太烂，当负载增加时所有事情都慢下来了（尤其与触发锁的代码有关，例如，sql 锁或 nfs 所或任何种类的分布式锁）。

问题 9：真正的性能和感知的性能之间的存在概念差异。对修复来说哪个更重要？

FiveRuns：感知的性能是最终用户感觉得到的，因此最应该修正。可是我认为感知的性能是很难测量的——即，它是依赖于上下文的，作为观察者，我们不能总是访问同一个上下文环境，而且我们不可能控制组成上下文的所有东西。

Blaine：掩饰。感知的性能更重要。使用后台队列卸载工作，显示用户所期望的东西并且断定是怎么泄露到应用的其它方面的。使用 Javascript 轮询廉价的请求而不是集中到一个单一的昂贵请求。在请求开始的时候将开销大的事情拆成并行的，在请求结束的时候再把结果收集起来。

Matt：我不同意——真实更重要。与其假装成某种情况，不如真的变成那种情况。

Randy：这两者都重要:-)。感知的性能影响了用户体验，而所谓的真实性能会影响成本。这最终就是一个业务决策：此刻最大的痛苦是什么，因此要看哪个相对更优先。

问题 10：几乎每个星期，总有些人发布一个新的 Web 或应用服务器或数据库/ORM 层软件。怎么看待这种情况？这是好事情——还是让 Job 调优更加困难了？

Blaine：我使用 apache。它有其缺点，但是工作良好。我使用 MySQL，它也有其缺点，但是工作良好。没有应用可以解决你所有的问题；它们有些是让事情更容易理解或修正，有些的确更好，但是有些使得调试和修正更加困难。

至于数据库，拥有 DHT 并不能使你获得索引（index）。你需要一个分布式索引，只是你还不知道。BigTable 和相关开源项目都不是分布式的索引。MapReduce 不是分布式的索引。查明你的分布式索引是怎样的，自己构建它吧。

Matt：这很好，但是可以更好。大多数同类产品都是大同小异的。能看到持续创新挺好，

但是如果能看到每一类工具的数量少一些而完全不同类型工具的数量和差异多一些，那就更好了。

FiveRuns：作为这种产品的制造者，我们对这个问题特别感兴趣！我们认为它是好事——它鼓励创新并给我们带来新的思路。但是我们专注于使用最好的工具组合——新的不见得就是好的。

Randy：有选择是好事情。这取决于你怎么去选择。追赶潮流此刻不再比顽固坚持不能工作的东西更有生产效率。我要说的是在一个像 eBay 这样 24x7 都要运转的站点，我们在使用经证明是好的技术时需要非常小心。这是我们对公司的责任。在我们考虑引入任何新东西的时候，在我们考虑将其应用到站点之前都要经过非常广泛的评估。

问题 11：那么在这一领域还有哪些内容经常被曲解？

Matt：嗯，我觉得让人有点失望的是，当一个好思想被商业化或者更通用化时，这个思想的质量就变差了。比如数据存储——很难建立一个既快速又可存储大量数据的软件。很明显，为了商业化（即，有大量潜在客户）两者都可实现当然最好了，但是这就意味着最终是一个妥协的产物，既不快也不大——不管厂商怎么宣传！

不是说这就是普遍现象或者总会是这样。

Blaine：你只能通过使用软件实现伸缩性。“语言是不能伸缩的。框架是不能伸缩的。而架构是可以伸缩的。”

伸缩性是一个社会问题。协调你的业务、操作和工程师们是极其关键的，如果你在这方面不成功，你实现伸缩性的努力也会失败。如果你的公司不理解需要多方协调以培育成功，那么你也会失败。YouTube 正是通过正确的处理这一社会问题实现了伸缩性。他们购买服务器、有一个好的投资计划、有足够的带宽。他们选择一件事情，把它做好，以组织的方式决策伸缩性，在天才工程师们的努力下，伸缩性如魔术般被实现了。

你可能会失败。快速吸取教训，然后成功。没问题。不要构建开箱即可伸缩的服务，因为你或许会发现你的交互设计是错误的，你必须回过头重新来一遍。在这期间，那些没有构建伸缩性的人可能正确设计了交互，因而足足领先了你 6 个月的时间。

FiveRuns：性能改进/伸缩性提升是一件“容易”的事情——修正性能或伸缩性问题不总是像外界看得那么简单——很容易因此受到指责且很难真正解决“所有”给定的约束。以我们的经验，修正性能和伸缩性问题需要对技术和问题领域两方面都有深入了解和经验。

Randy：经常被曲解的是性能和可伸缩性这两者间的区别 ;-)

原文链接：http://www.infoq.com/cn/articles/scalability-panel

[推荐文章]

使用 ClickOnce 细分发布版本

作者 David Cooksey 译者 朱永光

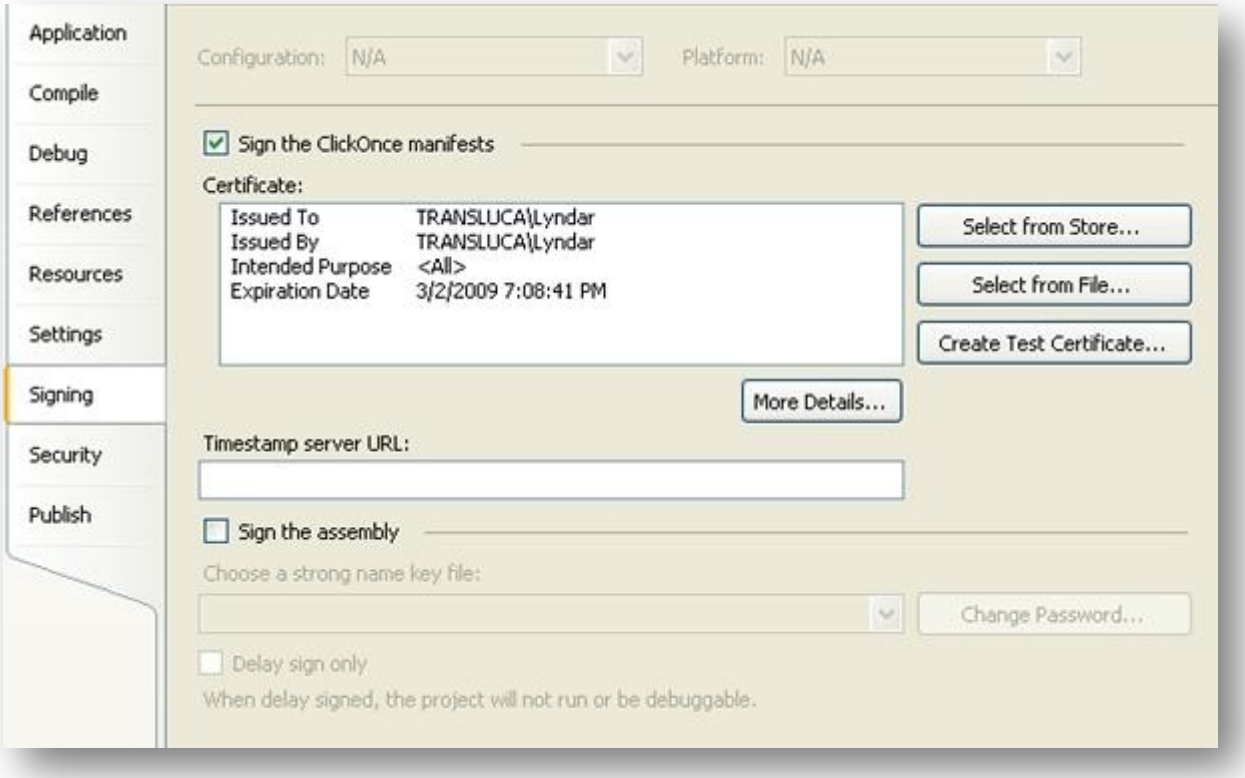
推荐理由：ClickOnce 让 WinForms 应用程序的部署轻而易举。David Cooksey 演示了如何在 ASP.NET 中编写一个 HttpHandler 来实现对 ClickOnce 部署的版本细分。

ClickOnce 是微软在.NET 2.0 框架中发布的一项技术，允许大家在 Visual Studio 中方便地部署和更新.NET 的 Windows 应用程序。部署功能是通过把应用程序文件复制到一个文件夹中、FTP 目录中或者某个 Web 位置 上，并同时附加一个清单文件来实现的。清单文件是一个具有.application 扩展名的 XML 文件，它包含了所有程序文件的一个列表，以及类似发布者 标识这样的安全相关信息。ClickOnce 应用程序每发布一个新的版本，一个新的子目录就被创建，并把更新的文件放在这个目录中。程序的所有部署版本都 是相互独立存在的，这意味着我们只需要识别用户，并把用户导向应用程序适合的版本，从而控制它们需要接受那个版本的程序。

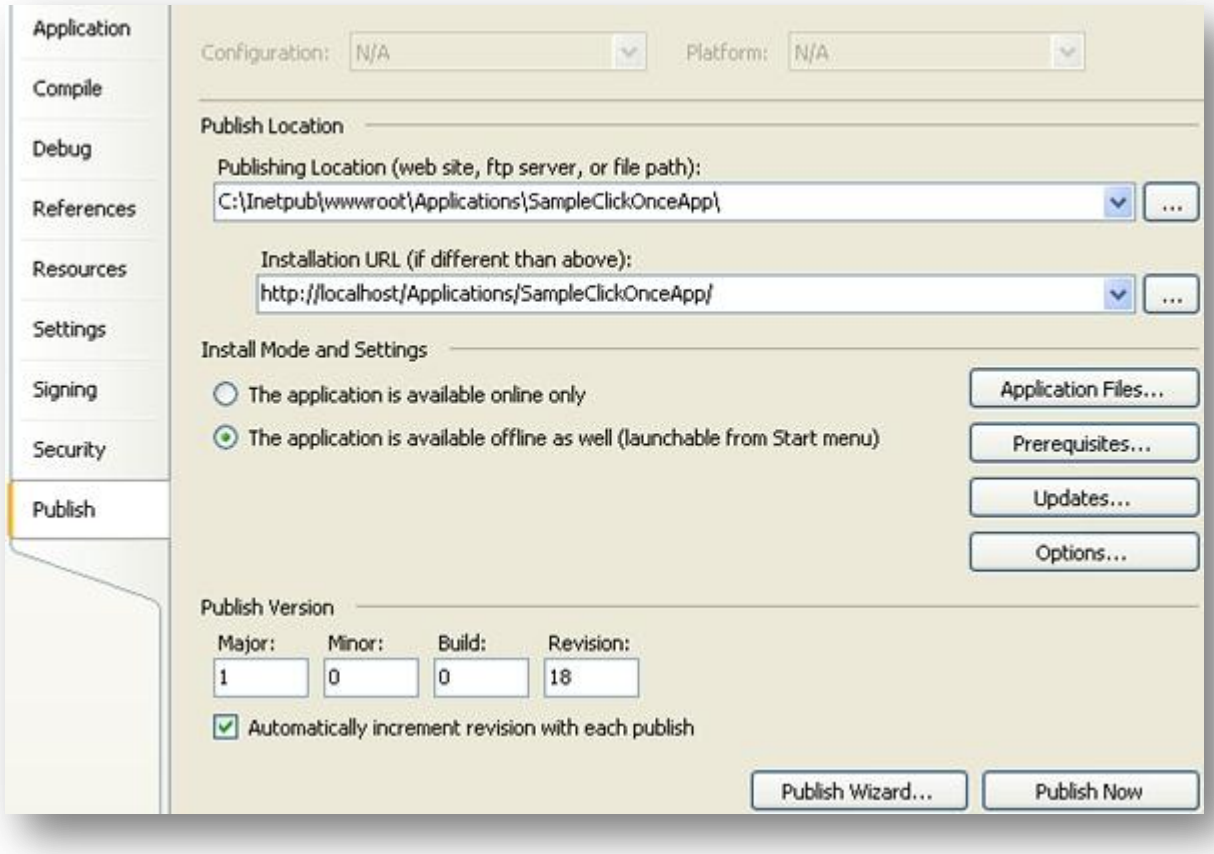
首先，让我们来创建一个简单的 Windows 应用程序，并通过 ClickOnce 来部署它。创建一个新的 Windows 应用程序项目，放置一个标签控件到窗体上，并在 New()过程中编写一些代码，以便让我们能看到我们拥有的是哪个版本。

```
label1.Text = "Version: " & _
System.Reflection.Assembly.GetExecutingAssembly.GetName.Version.ToString
```

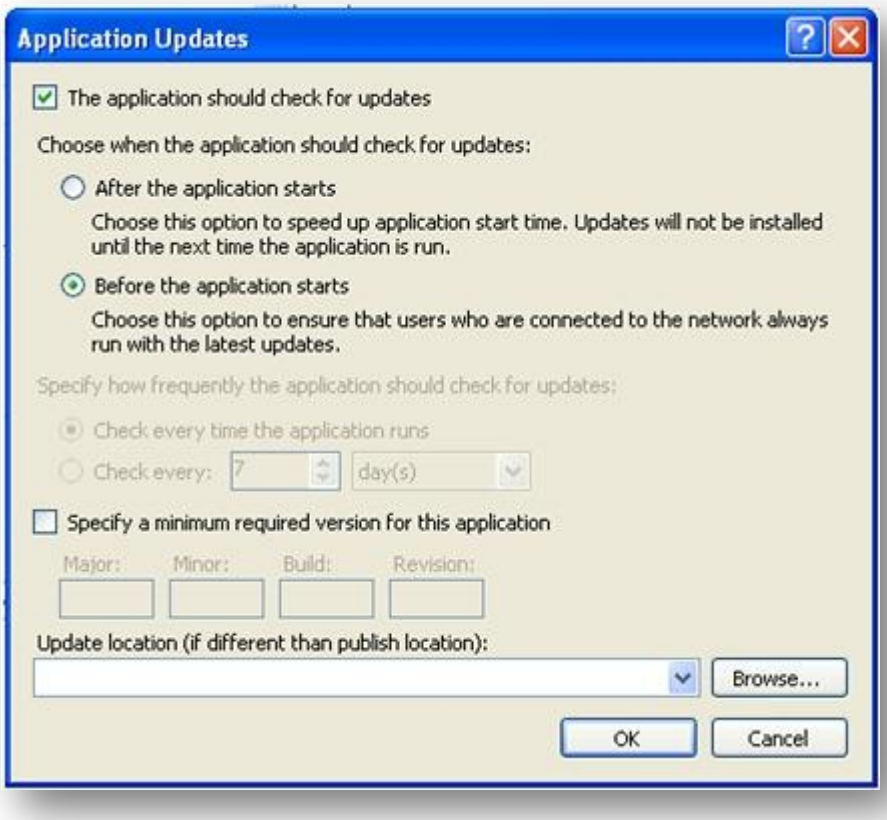
现在，打开项目属性界面并浏览签名(Signing)标签页，勾选“Sign the ClickOnce manifest”复选框，并通过点击“Create Test Certificate”按钮及输入密码来创建一个测试证书。



现在，我们已经拥有证书了，因而可以设置 ClickOnce 部署了。选择发布（Publish）标签页，输入发布位置即安装 URL（你也可以使用底 部的发布向导（Publish Wizard）来直接输入这些）。发布位置是将要存放部署文件的物理位置。安装 URL 是用户用于下载和安装应用程序的 URL。我们打算让这个应用程序能够 脱机运行，所以选择在“Install Mode and Settings”之下的第二个单选框。为了实现我们控制版本的功能，我们需要把程序文件部署到一个 Web 应用程序的文件夹树中，在选择部署位置时务必注 意这一点。



注意，更新选项要通过点击“Updates...”按钮来显示。



至此，我们已经具有了一个可以进行部署的 ClickOnce 应用程序了。一旦部署并安装好，利用显示出来的更新选项设置，应用程序将会在每次启动的时候检查更新。

如果有新版本存在，那么用户将被提示进行更新。

此时，我们已经部署好一个 ClickOnce 应用程序了，接下来可以准备实现版本控制。进行版本细化控制的原理是为主文件.application 设置一个基于 Web 的转向功能.首先 ,我们要把.application 扩展名添加到 ISS 的允许文件列表中。这可以通过 IIS 里的 Web 站点属性，设置 界面中的 Home Directory 标签页上的 Application Configuration 配置节来实现。

一旦完成这步工作 ,就可以创建一个新的 WebApplication 并添加一个 Handler 类到其中。

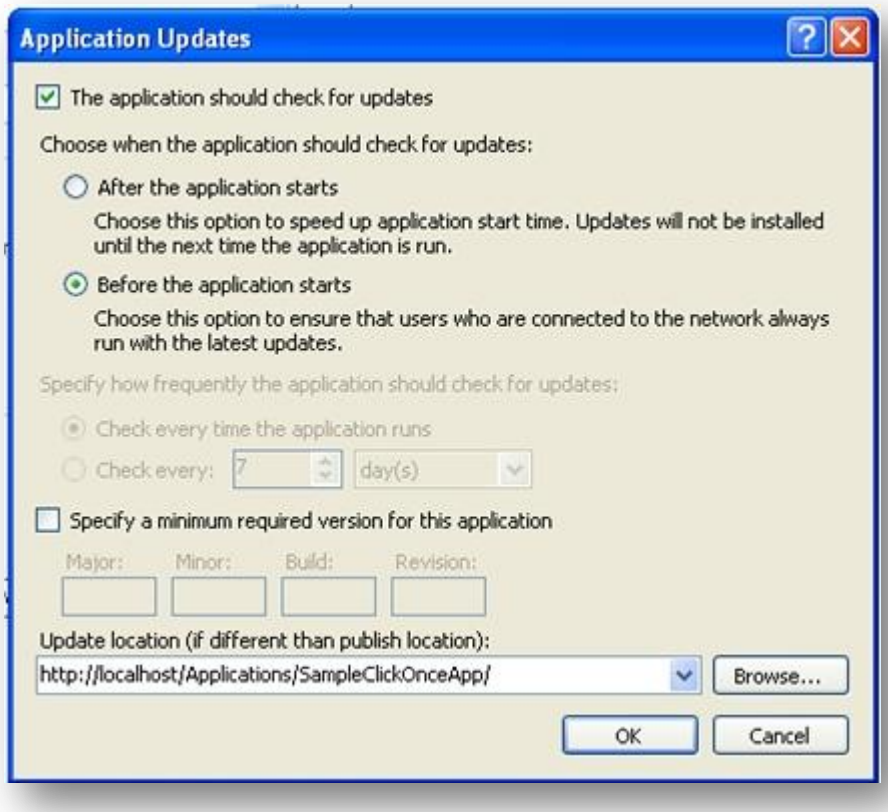
```
Imports System.Web
Imports System.Web.Services

Public Class MyHandler
    Implements System.Web.IHttpHandler

    Sub ProcessRequest(ByVal context As HttpContext) Implements IHttpHandler.ProcessRequest

        context.Response.ContentType = "text/plain"
        context.Response.Write("Hello World!")

    End Sub

    ReadOnly Property IsReusable() As Boolean Implements IHttpHandler.IsReusable
        Get
            Return False
        End Get
    End Property

End Class
```

接着，在应用程序的 web.config 文件中注册这个 Handler。Handler 类型的格式是 [Namespace].[ClassName], [Assembly]

```
        <httpHandlers>
        <add verb="GET" path="*.application"
type="WebApplication1.MyHandler,WebApplication1"/>
```

```
    httpHandlers>
  system.web>
configuration>
```

现在，返回到我们的示例 ClickOnce 应用程序，重新打开 Publish 标签页下的 Updates 对话框。输入指向这个 Web 应用程序的更新位置。



最后，编辑我们之前创建的 HttpHandler，让其针对请求返回适合的版本。这里，我们返回一个硬编码的版本，不过一个真实的实现应该处理验证等功能，以提供更好的灵活性。

```vb
Imports System.Web
Imports System.Web.Services

Public Class MyHandler
  Implements System.Web.IHttpHandler

  Sub ProcessRequest(ByVal context As HttpContext) Implements IHttpHandler.ProcessRequest

    context.Response.AddHeader("Content-Type", "application/x-ms-application")

    Dim filePath As String = context.Request.FilePath
    filePath = context.Request.MapPath("/Applications/SampleClickOnceApp/SampleClickOnceApp_1_0_0_16.application")
    context.Response.TransmitFile(filePath)
  End Sub

  ReadOnly Property IsReusable() As Boolean Implements IHttpHandler.IsReusable
    Get
      Return False
    End Get
  End Property

End Class
```

标头可以在 IIS 中进行配置或如这里所示的在代码中设置。

一些注意事项

在为 ClickOnce 签名使用临时证书时，一旦证书过期，就需要使用一个新的来代替，那么，应用程序的当前安装就不能再升级了，所有用户都必须重新安装应用程序。这是一个已知的缺陷。幸运的是，已经有一种方式能够无限地延长证书的过期时间。访问这个地址来获取详细信息：http://support.microsoft.com/Default.aspx?kbid=925521。

使用更新位置来同步部署应用程序清单的位置非常重要。IIS 不能处理未在列表中的文件，所以如果更新路径无效的话你的处理器将永远不会触发。

如果使用 Visual Studio 2008，部署文件夹略有不同。每个版本的.application 文件都放在他们特定的子目录中，而非在主文件夹中。如果你有一个在 VS 2005 下设计的版本方案，并在 VS 2008 中来部署应用程序的话，这就会引起问题。

原文链接：http://www.infoq.com/cn/articles/ClickOnce-Versioning

[推荐文章]

Ruby/Rails: 不一样的'Web'应用

作者 郑功梓

推荐理由 :本文以一个实际应用的例子为引子 ,探讨 Ruby/Rails 在非传统 web 系统中应用 , 以及研究如何定制以 Rails 为基础的领域特定的 MVC 框架。

从一个有趣的项目说起

近些年来，在 Web 开发领域发生了许多有趣的变化，涌现出大量的框架，使得快速开发 Web 应用程序变成现实。其中 Ruby on Rails 尤其引人注目。Ruby 的强大语法和表达能力配合 Rails 的思想，不仅创造了 Web 开发效率提升数倍的奇迹，也为非应用 Web 技术解决非传统 Web 领域的问题提供了一个很好的契机。

现在我就来介绍一个刚刚结束的有趣的实际应用项目（暂且称之为 W 项目）。W 项目是一套用于农场的自动化系统。该系统中包含了众多工业设备，如用于工人们佩 戴的移动 W 设备，RFID 数据收集设备（分散安装在农场各处），闸门控制设备，地磅电子称，显示屏.... 还有许多名目繁多的设备。这些设备有的通过电缆，有的通过 ZB（ZigBee）无线网络将数据汇集到一台工控服务器上。这台服务器负责收集和处理所有设备传回来的数据，并发送指令给相关设备，例如控制闸门。

系统中大部分设备都是可以直接采购到的产品，但还是有几个设备是需要定制开发，其中最重要的定制设备 ,就是 W 设备。W 设备可以看作是一个工人佩戴在手臂上的移动 PDA。由于工作环境恶劣，W 设备与普通的 PDA 有很大差别，它只有很小的显示屏，数个按钮，内部配有 RFID 识读器，并且通过 ZB 无线网络与服务器通讯，外壳坚固并可防水。

由于与客户沟通顺畅，该项目的开发工作进行得非常顺利。系统的软件部分集中在服务器，服务器软件的主要任务是：

- 从众多设备采集数据，处理数据，控制设备。
- 提供数据浏览，统计报表等功能。

服务器运行 Debian，用 Ruby On Rails 来完成第(2)项任务，用 Ruby 配上小部分 C 扩展来完成第(1)项任务。

W 设备上的嵌入式软件很简单，基本上就是采集数据，并通过 ZB 网络将数据发送到服务器，显示一些简单的信息等。

系统很快投入试点使用，运行良好。

变化/挑战

由于试点应用很成功，客户很快就有了把这套系统作为产品推广的想法，因此二期工程随即上马。

二期工程的性质发生了变化，由仅供自家用的“专用系统”要发展为一个“通用的产品”。由此带来的一个显著变化是客户为 W 设备赋予了更多更重要的角色。现在，W 设备上要完成许多比以前复杂得很多的功能，并且要求日后能够方便地对 W 设备的功能进行定制（二次开发），以快速适应不同农场的需要。然而 W 设备是一个资源非常有限的嵌入式系统，仅有数十 KB 的内存，各种外设的驱动加上 ZB 通讯协议栈已经消耗掉了 80%的资源，按照原有架构去实现大量新功能已经不可行了，然而维持现有硬件设备不增加硬件成本是项目的一个关键目标…又要马儿跑又要马儿不吃草？

我们来看看面临的挑战：

- 增加很多复杂的操作界面(超出了 W 设备的现有资源能力)
- 功能变化快
- 需要日后可定制功能(二次开发)
- 维持低成本(意味着维持现有硬件架构不变)

解决这些问题最有效的方法就是把计算转移到服务器。所幸的是 W 设备和服务器是通过 ZB 无线网络实时连接的，这就为透过网络转移计算创造了条件。

解决方案

最经典的基于服务器端的解决方案莫过于传统的“UNIX 终端”模式，而且 ZB 无线网络窄带宽低延时特性似乎与终端模式可以很好地匹配。

但是要想直接应用“UNIX 终端”模式也有一些问题：

- 服务器端编程比较复杂（例如基于 cursor/ncursor 库编程），日后要进行二次开发效率低而培训成本会很高。

- 在 W 设备上实现标准 Terminal Emulator 比较困难，有可能要为 W 设备增加内存而修改硬件设计。

提到终端，我不禁想起 DHH 在 RailsConf 2007 上的那个 keynote，他把浏览器和 IBM 3270 做了有趣的对比。是的，那是个绝妙的对比，它给我留下的印象远大于对 Cargo Cult 调侃。浏览器和终端本质上要解决的是同一个问题。不同的是，由于 web 技术取得长足的发展，在服务器端开发 Web 程序比开发终端模式的应用程序要方便快捷得多。Rails 正是其中一个可以支持快速开发的绝佳例子。对这个项目而言，更重要的是服务器已经在跑 Rails 了，继续用 Rails 作为基础架构将会节约很多资源。

那么客户端呢？与实现“Terminal Emulator”相比，实现“Web Browser”难度更大。但是，仔细地去考察浏览器，复杂的原因在于 HTML 多媒体渲染以及客户端脚本。文本 Web 浏览器是可以相当小巧的。W 设备并不需要华丽的界面。而且由于 HTTP / HTML 效率太低并不合适 ZB 网络的低带宽，定制传输协议和标记语言势在必行。新的自定义的标记语言相当简洁（我们称之为 MML：Micro Markup Language），在 W 设备中实现对应的“浏览器”变得异常轻松，最终只用了大约 2k 行左右的 C 代码。

再回到服务器端，如何使 Rails 适应这些定制的协议和标记语言？

从外部来看，Rails 提供了以下主要服务：

- MVC 编程架构
- 透过 ActiveRecord 与数据库打交道
- 为 HTML 渲染提供服务
- 其他如测试，数据迁移，插件...等等

其中份量最重的 ActiveRecord 部分与 web 其实完全无关，可以“拿来就用”。很多便利工具，例如测试，数据迁移，插件机制等等，其实与 web 也无多大关系。

既然 Rails 的 MVC 中，M 与 web 无关，C 部分主要留给业务逻辑，而 V 部分对于非 web 领域价值不大，倘若要基于 rails 再建一个领域特定的 MVC，工作量也就是集中在 V 部分了。而这个 V 部分，既然是领域相关，则无论采用什么方案都是一个不可避免的工作。由于定制的 MML 是一个非常简单的标记语言，因此 V 部分也相当简单。当我们把 Rails 的思想与习惯用法再应用到这个新的 MVC 上时，就得到了一个基于 Rails 并且与 Rails 神似的框架。我们可以称这种框架为“领域特定的框架”（Domain Specific Framework，DSF）。

归纳起来，解决方案就是：客户端使用定制的 MML 展现数据，服务器端为基于 Rails 的 DSF，这样可以满足客户的所有要求，并且具有良好的可扩展性。

实现基于 Rails 的 DSF

要基于 Rails 来实现这样一个 DSF 以适应非传统 Web 应用，有两个途径：一是对 Rails 的各个部分进行修改，例如修改 router 以适应新的协议，修改 render 适应 MML 等等；二是利用 Rails 的现有设施，按照 Rails 的思路重新构造某些部件。

第一种方法实施起来并不容易。虽然使用 Rails 容易，而 Rails 本身是比较复杂的，修改起来并不容易，另外还享受不到未来 Rails 升级的好处。而第二种方法除了能够享受未来 Rails 升级的好处外，还可以与现有的 Rails 程序和平共处，并且无缝连接，因此选择后者来实现。要实现 full stack 的 MVC framework，除了直接利用 Rails 的 ActiveRecord 之外，还需要实现：Parser，Router，Server，Controller 和 MML render。

Parser: 负责解析自定义的通讯协议，获取从 W 设备传来的请求，并解析参数等。

Router: 扫描并装载 Controllers，缓存 Controller 对象，根据 Parser 的结果取得相应 Controller 的对象。Router 还负责监控 Controller 是否已被修改和重新装载 Controller，这样就可以和 Rails 一样，在开发的时候可以立即看到修改的结果而不用重启服务器。

Server: 总控制，从串口中读取 ZB 网络数据，调用 Parser 解析,把结果传给 Router，从 Router 中获取 Controler 对象并根据 Parser 结果调用 Controller＃Action，以及缓存 Controller 的输出等等。

Controller: 用户应用的业务逻辑都放在 Controller 的子类里实现。在 Controller 里直接调用 rails 的 Models 访问数据库。

MML Render: 使用 erb 作为模板文件，再加上一些辅助的功能，例如 render link，menu 之类的，和自定义的 MML 特性密切相关。MML Render 作为 module 最终 mixin 到 Controller 里面。

从上面各个模块的功能描述可以看出，这个 DSF 模型几乎和 rails 是一样的，但由于是面向自定制的协议和 MML，其复杂度和实现难度大大低于 Rails，再加上 Ruby 语言的强大表达能力，最终这个 DSF 仅用了不到千行代码！

再来看看在这个 DSF 下开发应用程序会是怎样的，以一个最简单的"Hello World"为例：

contollers/main.rb：

Ruby 代码：

```
class MainController < WSController
  controller_map_to :m
  action_maps :index => "i"
```

```
        def index
      @text = "Hello world"
    end
end
```

views/main/main_index.erb：

Ruby 代码：

```
<%= "<p3.20crs2e3.5> #{@text}" %>
```

在 MainController 中的 controller_map_to 起的作用是把自己（main）映射到一个较短的名字"m"，而 action_maps 则对 actions 取短的别名（本例中，index 的别名是 i），这样做的目的是为了减少请求串的长度。（ZB 网络的带宽非常有限，节约每一个字节意义都很大）。

再看看 view，其中“<P3.20CrS2E3.5>”表示在第 3 行 20 列处(P3.20)，以红色(Cr)2 号字(S2)显示 @text，经过 3.5 妙后清除屏幕(E3.5)。当然，如果想更方便地描述这些 MML 属性，则可以定义一系列 helper 函数，或者干脆定义一套 DSL。

这个简单的例子没有演示 Models，因为它可以直接使用 Rails 的 Models，因此使用起来和在 Rails 中没有丝毫差别。例如可以使用 has_many，belongs_to 等等。

通过这个例子，我们可以体会到这个 DSF 与 Rails 有多么“神似”，熟悉 Rails 的人通过几分钟简单的了解就可以立即在这个新的 DSF 下开发应用。二次开发的问题迎刃而解。

结论

当我们把 MVC Web 框架的概念推广到 web 之外，那么这个 V 就可以是任意的领域特定的数据展示格式。它既可以是基于文本的，也可以是基于二进制的；既可以是自定义的，也可以去兼容已有的格式。通过定制的 DSF 来解决问题不是没有意义的“重复造轮子”，而是解决领域特定问题的一个有效手段。而基于 Rails 来实现 DSF 更是有着非常显著的优势，整个复杂的 Model（ActiveRecord）可以不经任何修改而直接利用，所需要的代价仅仅是在 DSF 里面增加一行代码：

```
require File.dirname(__FILE__) + '/../config/boot'
```

Rails 中的众多优秀工具例如数据迁移，测试，调试，插件等等，大部分也可以直接使用，可以说基于 Rails 来实现 DSF 只需要付出 5%的努力，就可以达到 100%的效果。

当遇到需求变化时，运用恰当的技术手段有时候可以柳暗花明，特别是跨领域交叉应用，往往能收到意想不到的效果。web 技术的蓬勃发展带来了异彩纷呈的诸多框架技术，开发工具，以及丰富的人才储备，这些资源对于非 Web 领域也有巨大的吸引力。本文所举的这个

例子就是巧妙地通过基于 Rails 的 DSF 来解决实际问题。实际上这个例子还有一个精彩的插曲值得一提，那就是客户希望可以脱离 W 设备和 ZB 网络来开发应用程序，简单地说，就是希望有一个 W 设备的硬件模拟器。在传统解决方案里面，硬件模拟器是一项非常复杂的工作，但在这里，由于整个解决方案采用的是 Web 技术，因此实际上模拟器的核心就是一个简单的 MML 到 HTML 的转换程序加上少许 JavaScript 而已，浏览器就摇身变成了一个硬件模拟器。突破传统思维的束缚，就容易找到金矿。

我不是一个 Web 程序员，也从未开发过用户超过十个人的传统 Web 程序，但这并不意味着 Web 技术对我无用。正相反，Web 技术经常被应用到我所从事的嵌入式系统领域。基于 Rails 的 DSF 解决方案为加速 Web 技术在其它领域的应用开启了一道光明之门。

作者简介：郑功梓，资深潜入式系统工程师，技术涉猎广泛，前新大陆自动识别技术有限公司总工程师，现旅居新西兰。

原文链接：http://www.infoq.com/cn/articles/ruby-rails-diff-app

[推荐文章]

Spring 2.5 :Spring MVC 中的新特性

作者 Rossen Stoyanchev 译者 张凯峰

推荐理由 :Spring 2.5 推出了可用于 Spring 管理对象的自动发现、依赖注入、生命周期方法、Web 层配置和测试的全套注解。

Spring 框架从创建伊始就致力于为复杂问题提供强大的、非侵入性的解决方案。Spring 2.0 当中为缩减 XML 配置文件数量引入定制命名空间功能，从此它便深深植根于核心 Spring 框架（aop、context、jee、jms、 lang、tx 和 util 命名空间）、Spring Portfolio 项目（例如 Spring Security）和非 Spring 项目中（例如 CXF）。

Spring 2.5 推出了一整套注解，作为基于 XML 的配置的替换方案。注解可用于 Spring 管理对象的自动发现、依赖注入、生命周期方法、Web 层配置和单元/集成测试。

探索 Spring 2.5 中引入的注解技术系列文章由三部分组成，本文是其中的第二篇，它主要讲述了 Web 层中的注解支持。最后一篇文章将着重介绍可用于集成和测试的其它特性。

这个系列文章的第一部分论述了 Java 注解（annotation）是如何代替 XML 来配置 Spring 管理对象和依赖注入的。我们再用一个例子回顾一下：

```
@Controller
 public class ClinicController {
      private final Clinic clinic;
      @Autowired
      public ClinicController(Clinic clinic) {
           this.clinic = clinic;
      }
      ...
```

@Controller 表明 ClinicController 是 Web 层组件，@Autowired 请求一个被依赖注入的 Clinic 实例。这个例子只需要少量的 XML 语句就能使容器识别两个注解，并限定组件的扫描

范围：

```
<context:component-scan
base-package="org.springframework.samples.petclinic"/>
```

这对 Web 层可谓是个福音，因为在这层 Spring 的 XML 配置文件已日益臃肿，甚至可能还不如层下的配置来得有用。控制器掌握着许多属性，例如视图名称、表单对象名称和验证器类型，这些多是关乎配置的，甚少关于依赖注入的。通过 bean 定义继承，或者避免配置变化不是很频繁的属性，也可以有效的管理类似的配置。不过以我的经验，很多开发人员都不会这样做，结果就是 XML 文件总比实际需要的要庞大。不过 @Controller 和@Autowired 对 Web 层的配置会产生积极的作用。

在系列文章的第二部分我们将继续讨论这个问题，并浏览 Spring 2.5 在 Web 层的注解技术。这些注解被非正式的称为@MVC，它涉及到了 Spring MVC 和 Spring Porlet MVC，实际上本文讨论的大部分功能都可以应用在这两个框架上。

从 Controller 到@Controller

与第一部分讨论的注解相比，@MVC 已不只是作为配置的一种替换方案这样简单了，考虑下面这个著名的 Spring MVC 控制器签名：

```
public interface Controller {
    ModelAndView handleRequest(HttpServletRequest request,
HttpServletResponse
response) throws Exception;
 }
```

所有的 Spring MVC 控制器要么直接实现 Controller 接口，要么就得扩展类似 AbstractController、 SimpleFormController、 MultiActionController 或 AbstractWizardFormController 这样的基类实现。正是 Controller 接口允许 Spring MVC 的 DispatcherServlet 把所有上述对象都看作是“处理器（handlers）”，并在一个名为 SimpleControllerHandlerAdapter 的适配器的帮助下调用它们。

@MVC 从三个重要的方面改变了这个程序设计模型：

1. 不需要任何接口或者基类。
2. 允许有任意数量的请求处理方法。
3. 在方法签名上具有高度的灵活性。

考虑到以上三个要点，就可以说很公平的说@MVC 不仅仅是个替换方案了，它将会是 Spring MVC 的控制器技术演变过程中下一个重要步骤。

DispatcherServlet 在名为 AnnotationMethodHandlerAdapter 的适配器帮助下调用被注解的控制器。正是这个适配器做了大量工作支持我们此后将会讨论的注解，同时也是它有效的取代了对于控制器基类的需求。

@RequestMapping 简介

我们还是从一个类似于传统的 Spring MVC Controller 控制器开始：

```
@Controller
 public class AccountsController {

     private AccountRepository accountRepository;

     @Autowired
     public AccountsController(AccountRepository accountRepository) {
         this.accountRepository = accountRepository;
     }

     @RequestMapping("/accounts/show")
     public ModelAndView show(HttpServletRequest request,
                              HttpServletResponse response) throws
Exception {
         String number = ServletRequestUtils.getStringParameter(request,
"number");
         ModelAndView mav = new
ModelAndView("/WEB-INF/views/accounts/show.jsp");
         mav.addObject("account", accountRepository.findAccount(number));
         return mav;
     }
 }
```

此处与以往的不同在于，这个控制器并没有扩展 Controller 接口，并且它用 @RequestMapping 注解指明 show()是映射到 URI 路径 “/accounts/show” 的请求处理方法。除此以外，其余代码都是一个典型的 Spring MVC 控制器应有的内容。

在将上述的方法完全转化到@MVC 后，我们会再回过头来看@RequestMapping，但是在此之前还有一点需要提请注意，上面的请求映射 URI 也可匹配带有任意扩展名的 URI 路径，例如：

```
/accounts/show.htm
 /accounts/show.xls
 /accounts/show.pdf
```

```
...
```

灵活的请求处理方法签名

我们曾经承诺过要提供灵活的方法签名，现在来看一下成果。输入的参数中移除了响应对象，增加了一个代表模型的 Map；返回的不再是 ModelAndView，而是一个字符串，指明呈现响应时要用的视图名字：

```
@RequestMapping("/accounts/show")
 public String show(HttpServletRequest request, Map<String, Object> model)
 throws Exception {
     String number = ServletRequestUtils.getStringParameter(request, "number");
     model.put("account", accountRepository.findAccount(number));
     return "/WEB-INF/views/accounts/show.jsp";
 }
```

Map 输入参数是一个“隐式的”模型，对于我们来说在调用方法前创建它很方便，其中添加的键—值对数据便于在视图中解析应用。本例视图为 show.jsp 页面。

@MVC 可以接受多种类型的输入参数，例如 HttpServletRequest/HttpServletResponse、HttpSession、Locale、InputStream、 OutputStream、File[]等等，它们的顺序不受任何限制；同样它也允许多种返回类型，例如 ModelAndView、Map、 String，或者什么都不返回。你可以查看@RequestMapping 的 JavaDoc 以了解它支持的所有输入和返回参数类型。

有种令人感兴趣的情形是当方法没有指定视图时（例如返回类型为 void）会有什么事情发生，按照惯例 DispatcherServlet 要再使用请求 URI 的路径信息，不过要移去前面的斜杠和扩展名。让我们把返回类型改为 void：

```
@RequestMapping("/accounts/show")
 public void show(HttpServletRequest request, Map<String, Object> model)
throws Exception {
     String number = ServletRequestUtils.getStringParameter(request, "number");
     model.put("account", accountRepository.findAccount(number));
 }
```

对于给定的请求处理方法和“/accounts/show”的请求映射，我们可以期望 DispatcherServlet 能够获得“accounts/show”的默认视图名称，当它与如下适当的视图解析器结合共同作用时，会产生与前面指明返回视图名同样的结果：

```
<bean
class="org.springframework.web.servlet.view.InternalResourceViewResolver">
     <property name="prefix" value="/WEB-INF/views/" />
     <property name="suffix" value=".jsp" />
```

```
    </bean>
```

强烈推荐视图名称依赖惯例的方式,因为这样可以从控制器代码中消除硬编码的视图名称。如果你想定制 DispatcherServlet 获取默认视图名的方式，就在 servlet 上下文环境中配置一个你自己的 RequestToViewNameTranslator 实现，并为其 bean id 赋名为“viewNameTranslator”。

用@RequestParam 提取和解析参数

@MVC 另外一个特性是其提取和解析请求参数的能力。让我们继续重构上面的方法，并在其中添加@RequestParam 注解：

```
    @RequestMapping("/accounts/show")
     public void show(@RequestParam("number") String number, Map<String, Object>
model) {
         model.put("account", accountRepository.findAccount(number));
     }
```

这里@RequestParam 注解可以用来提取名为“number”的 String 类型的参数，并将之作为输入参数传入。 @RequestParam 支持类型转换，还有必需和可选参数。类型转换目前支持所有的基本 Java 类型，你可通过定制的 PropertyEditors 来扩展它的范围。下面是一些例子，其中包括了必需和可选参数：

```
    @RequestParam(value="number", required=false) String number
     @RequestParam("id") Long id
     @RequestParam("balance") double balance
     @RequestParam double amount
```

注意，最后一个例子没有提供清晰的参数名。当且仅当代码带调试符号编译时，结果会提取名为“amount ”的参数，否则，将抛出 IllegalStateException 异常，因为当前的信息不足以从请求中提取参数。由于这个原因，在编码时最好显式的指定参数名。

继续@RequestMapping 的讨论

把@RequestMapping 放在类级别上是合法的，这可令它与方法级别上的@RequestMapping 注解协同工作，取得缩小选择范围的效果，下面是一些例子。

类级别：

```
    RequestMapping("/accounts/*")
```

方法级别：

```
    @RequestMapping(value="delete", method=RequestMethod.POST)
    @RequestMapping(value="index", method=RequestMethod.GET,
params="type=checking")
```

```
@RequestMapping
```

第一个方法级的请求映射和类级别的映射结合，当 HTTP 方法是 POST 时与路径“/accounts/delete”匹配；第二个添加了一个要求，就是名为“type”的请求参数和其值“checking”都需要在请求中出现；第三个根本就没有指定路径，这个方法匹配所有的 HTTP 方法，如果有必要的话可以用它的方法名。下面改写我们的方法，使它可以依靠方法名进行匹配，程序如下：

```
@Controller
 @RequestMapping("/accounts/*")
 public class AccountsController {

     @RequestMapping(method=RequestMethod.GET)
     public void show(@RequestParam("number") String number, Map<String,
Object> model)
    {
         model.put("account", accountRepository.findAccount(number));
     }
     ...
```

方法匹配的请求是“/accounts/show”，依据的是类级别的@RequestMapping 指定的匹配路径“/accounts/*”和方法名“show”。

消除类级别的请求映射

Web 层注解频遭诟病是有事实依据的，那就是嵌入源代码的 URI 路径。这个问题很好矫正，URI 路径和控制器类之间的匹配关系用 XML 配置文件去管理，只在方法级的映射中使用@RequestMapping 注解。

我们将配置一个 ControllerClassNameHandlerMapping，它使用依赖控制器类名字的惯例，将 URI 映射到控制器：

```
<bean
class="org.springframework.web.servlet.mvc.support.ControllerClassNameHandlerMap
ping"/>
```

现在“/accounts/*”这样的请求都被匹配到 AccountsController 上，它与方法级别上的@RequestMapping 注解协作的很好，只要添加上方法名就能够完成上述映射。此外，既然我们的方法并不会返回视图名称，我们现在就可以依据惯例匹配类名、方法名、URI 路径和视图名。

当@Controller 被完全转换为@MVC 后，程序的写法如下：

```
@Controller
 public class AccountsController {

      private AccountRepository accountRepository;

      @Autowired
     public AccountsController(AccountRepository accountRepository) {
         this.accountRepository = accountRepository;
     }

      @RequestMapping(method=RequestMethod.GET)
     public void show(@RequestParam("number") String number, Map<String,
Object> model)
    {
         model.put("account", accountRepository.findAccount(number));
     }
     ...
```

对应的 XML 配置文件如下：

```
<context:component-scan base-package="com.abc.accounts"/>

  <bean
class="org.springframework.web.servlet.mvc.support.ControllerClassNameHandlerMap
ping"/>

  <bean
class="org.springframework.web.servlet.view.InternalResourceViewResolver">
     <property name="prefix" value="/WEB-INF/views/" />
     <property name="suffix" value=".jsp" />
 </bean>
```

你可以看出这是一个最精减的 XML。程序里注解中没有嵌入 URI 路径，也没有显式指定视图名，请求处理方法也只有很简单的一行，方法签名与我们的需求精准匹配，其它的请求处理方法也很容易添加。不需要基类，也不需要 XML（至少也是没有直接配置控制器），我们就能获得上述所有优势。

也许接下来你就可以看到，这种程序设计模型是多么有效了。

@MVC 表单处理

一个典型的表单处理场景包括：获得可编辑对象，在编辑模式下显示它持有的数据、允

许用户提交并最终进行验证和保存变化数据。Spring MVC 提供下列几个特性辅助进行上述所有活动：数据绑定机制，完全用从请求参数中获得的数据填充一个对象；支持错误处理和验证；JSP 表单标记库；基类控制器。使用@MVC，除了由于@ModelAttribute、@InitBinder 和@SessionAttributes 这些注解的存在而不再需要基类控制器外，其它一切都不需要改变。

@ModelAttribute 注解

看一下这些请求处理方法签名：

```
@RequestMapping(method=RequestMethod.GET)
 public Account setupForm() {
      ...
 }

  @RequestMapping(method=RequestMethod.POST)
 public void onSubmit(Account account) {
      ...
 }
```

它们是非常有效的请求处理方法签名。第一个方法处理初始的 HTTP GET 请求，准备被编辑的数据，返回一个 Account 对象供 Spring MVC 表单标签使用。第二个方法在用户提交更改时处理随后的 HTTP POST 请求，并接收一个 Account 对象作为输入参数，它是 Spring MVC 的数据绑定机制用请求中的参数自动填充的。这是一个非常简单的程序模型。

Account 对象中含有要被编辑的数据。在 Spring MVC 的术语当中，Account 被称作是表单模型对象。这个对象必须通过某个名称让表单标签（还有数据绑定机制）知道它的存在。下面是从 JSP 页面中截取的部分代码，引用了一个名为“account”的表单模型对象：

```
<form:form modelAttribute="account" method="post">

      Account Number: <form:input path="number"/><form:errors
path="number"/>
      ...
 </form>
```

即使我们没有在任何地方指定“account”的名称，这段 JSP 程序也会和上面所讲的方法签名协作的很好。这是因为@MVC 用返回对象的类型名称作为默认值，因此一个 Account 类型的对象默认的就对应一个名为“account”的表单模型对象。如果默认的不合适，我们就可以用 @ModelAttribute 来改变它的名称，如下所示：

```
@RequestMapping(method=RequestMethod.GET)
 public @ModelAttribute("account") SpecialAccount setupForm() {
```

```
        ...
    }
      @RequestMapping(method=RequestMethod.POST)
    public void update(@ModelAttribute("account") SpecialAccount account) {
        ...
    }
```

@ModelAttribute 同样也可放在方法级的位置上，取得的效果稍有不同：

```
    @ModelAttribute
     public Account setupModelAttribute() {
        ...
     }
```

此处 setupModelAttribute()不是一个请求处理方法，而是任何请求处理方法被调用之前，用来准备表单项模型对象的一个方法。对那些熟悉 Spring MVC 的老用户来说，这和 SimpleFormController 的 formBackingObject()方法是非常相似的。

最初的 GET 方法中我们得到一次表单模型对象，在随后的 POST 方法中当我们依靠数据绑定机制用用户所做的改变覆盖已有的 Account 对象时，我们会第二次得到它，在这种表单处理场景中把@ModelAttribute 放在方法上是很有用的。当然，作为一种两次获得对象的替换方案，我们也可以在两次请求过程中将它保存进 HTTP 的会话（session），这就是我们下面将要分析的情况。

用@SessionAttributes 存储属性

@SessionAttributes 注解可以用来指定请求过程中要放进 session 中的表单模型对象的名称或类型，下面是一些例子：

```
    @Controller
     @SessionAttributes("account")
     public class AccountFormController {
        ...
     }


      @Controller
     @SessionAttributes(types = Account.class)
     public class AccountFormController {
        ...
     }
```

根据上面的注解，AccountFormController 会在初始的 GET 方法和随后的 POST 方法之间，把名为 “account”的表单模型对象（或者象第二个例子中的那样，把所有 Account 类型的

表单模型对象）存入 HTTP 会话（session）中。不过，当有改变连续发生的时候，就应当把属性对象从会话中移除了。我们可以借助 SessionStatus 实例来做这件事，如果把它添加进 onSubmit 的方法签名中，@MVC 会完成这个任务：

```
@RequestMapping(method=RequestMethod.POST)
 public void onSubmit(Account account, SessionStatus sessionStatus) {
     ...
     sessionStatus.setComplete(); // Clears @SessionAttributes
 }
```

定制数据绑定

有时数据绑定需要定制，例如我们也许需要指定必需填写的域，或者需要为日期、货币金额等类似事情注册定制的 PropertyEditors。用@MVC 实现这些功能是非常容易的：

```
@InitBinder
 public void initDataBinder(WebDataBinder binder) {
     binder.setRequiredFields(new String[] {"number", "name"});
 }
```

@InitBinder 注解的方法可以访问@MVC 用来绑定请求参数的 DataBinder 实例，它允许我们为每个控制器定制必须项。

数据绑定结果和验证

数据绑定也许会导致类似于类型转换或域缺失的错误。不管发生什么错误，我们都希望能返回到编辑的表单，让用户自行更正。要想实现这个目的，我们可直接在方法签名的表单模型对象后面追加一个 BindingResult 对象，例程如下：

```
@RequestMapping(method=RequestMethod.POST)
 public ModelAndView onSubmit(Account account, BindingResult bindingResult) {
     if (bindingResult.hasErrors()) {
         ModelAndView mav = new ModelAndView();
         mav.getModel().putAll(bindingResult.getModel());
         return mav;
     }
     // Save the changes and redirect to the next view...
 }
```

发生错误时我们返回到出现问题的视图，并把从 BindingResult 得到的属性增加到模型上，这样特定域的错误就能够反馈给用户。要注意的是，我们并没有指定一个显式的视图名，而是允许 DispatcherServlet 依靠与入口 URI 路径信息匹配的默认视图名。

调用 Validator 对象并把 BindingResult 传给它，仅这一行代码就可实现验证操作。这允

许我们在一个地方收集绑定和验证错误：

```
@RequestMapping(method=RequestMethod.POST)
 public ModelAndView onSubmit(Account account, BindingResult bindingResult) {
     accountValidator.validate(account, bindingResult);
     if (bindingResult.hasErrors()) {
         ModelAndView mav = new ModelAndView();
         mav.getModel().putAll(bindingResult.getModel());
         return mav;
     }
     // Save the changes and redirect to the next view...
 }
```

现在是时候结束我们的 Spring 2.5 Web 层注解（非正式称法为@MVC）之旅了。

总结

Web 层的注解已经证明是相当有用的，不仅是因为它能够大大减少 XML 配置文件的数量，而且还在于它能成就一个可自由访问 Spring MVC 控制器技术的精致、灵活和简洁的程序设计模型。我们强烈推荐使用“惯例优先原则（convention-over-configuration）”特性，以及以处理器映射为中心的策略给控制器派发请求，避免在源码中嵌入 URI 路径或是定义显式的视图名引用。

最后是本文没有讨论，但值得关注的一些非常重要的 Spring MVC 扩展。最新发布的 Spring Web Flow 版本 2 添加了一些特性，例如基于 JSF 视图的 Spring MVC、Spring 的 JavaScript 库，还有支持更先进编辑场景的高级状态和导航管理。

原文链接：http://www.infoq.com/cn/articles/spring-2.5-ii-spring-mvc

架构师 试刊号

每月 8 日出版

总编辑：霍泰稳

总编助理：刘申

编辑 :宋玮 朱永光 李剑 胡键 郭晓刚 李明

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

13810038718 010-84725788

《架构师》月刊由 InfoQ 中文站出品（ www.infoq.com/cn ），Innobook 制作并发布。更多精彩电子书，请访问 Innobook。

InfoQ中文站 www.infoq.com/cn

innobook 创造 · 共享 · 传播