

源自InfoQ 的富互联网应用内容精华

富互联网应用之美

Rich Internet Application



RAP
Rich Ajax Platform



HTML
5



BLAZEDS
BLAZEDS

霍泰稳 策划

葛明见 执行

InfoQ企业软件开发丛书

免费在线版本

(非印刷免费在线版)



了解本书更多信息请登录[本书的官方网站](#)

InfoQ 中文站出品



本书由 InfoQ 中文站免费发放，如果您从其他渠道获取本书，请注册 InfoQ 中文站以支持作者和出版商，并免费下载更多 InfoQ 企业软件开发系列图书。

本迷你书主页为

<http://www.infoq.com/cn/minibooks/beautiful-ria>

序一

富互联网应用，或者叫做 RIA，具备真正的革命性的在线用户体验。Macromedia 在十多年前提出了这个词条，当时最多的想法是，这是一个错字。但是在短短几年之后，RIA 在应用开发领域就证明了自己的地位和价值，迸发出迷人的光彩，并且让所有人都从中受益。

RIA 是多种技术的组合，这些技术相辅相成，以平滑透明的方式工作，创建更好的用户体验。RIA 充分利用了互联网的普及程度和应用范围。RIA 技术可以充分发挥计算机及其相关设备的能力，智能地为客户推送最合适的功能。RIA 摆脱了严重制约 Web 功能的松散耦合的页面模型，取而代之的是类似传统桌面应用的用户体验。最重要的是，RIA 提供了全方位的更好的用户体验。

RIA 已经被不计其数的后端系统和技术应用，从商业到开源，从遗留系统到尖端应用，从复杂到简单，几乎无所不包。但是 RIA 真正的应用领域是客户端，在这个领域，开发人员可以有两个选择。Flash 平台（包括浏览器中的 Flash 播放器和桌面的 AIR 客户端应用）帮助定义了 RIA 应用空间，在很多场景和开发者中保持了领先地位。最近，随着 Ajax 技术（这也暴露了之前没有充分利用 Web 浏览器的能力）的出现，并逐渐转向 HTML5，Web 浏览器自身也成为了适合 RIA 应用的开发平台。

需要着重提出的是，HTML 和 Flash 都是非常有价值的开发工具，它们的目的是相辅相成。Flash 的工作经常是 HTML 无法完成的。Flash 扩展 Web 浏览器，而不是取代它，开发者需要同时学习这两种技术，包括如何使用这两种技术解决那些单一技术无法解决的问题。

Adobe（几年前与 Macromedia 合并了）占据了 RIA 领域的主导地位。Flash 播放器仍然是下载次数最多的独立软件。现在 Flash 可以运行在所有主流平台和主要设备上，很显然，基于 Flash 进行 RIA 开发是很多开发人员的首选。有了针对设计人员的 Flash Professional、针对开发者的 Flash Builder、针对简单应用工作流的 Flash Catalyst 这一系列产品，Flash 在 RIA 领域的领头羊位置是无可争议的。同时，Adobe 全面支持和鼓励基于 HTML 的 RIA 开发，类似 Dreamweaver 功能，我们即将到来的 HTML 交互设计工具（还未命名），基于 jQuery 的工作等等。Adobe 致力于创建和提升开发工具的功能，使得开发人员和设计人员能够更有效地构建未来的富互联网应用，特别是那些目标桌面系统和相关设备。

不过，正如我们提到的，我们不是一个人在战斗！Adobe 在持续为 RIA 开发创建最好的工具的同时，我们必须从社区吸取营养，帮助我们定义和创建知识体系，从而帮助更多的开发者

和设计者走向成功。

这就是为什么我很高兴看到这个迷你书项目的原因：高度聚焦在 RIA 领域，提供开发人员需要的支持，训练他们的技能，帮助他们在激烈的竞争中取得成功。过去几年我有幸频繁地访问中国，认识了很多本地的开发者和设计者。随着每次访问，我越来越被会场中展现出来的技术、能力和激情所感动。考虑到中国有庞大的 Flash 和 Web 开发社区，我非常高兴能够看到这样一个可以满足这样重要需求的出版物出现。

Ben Forta

Adobe 全球平台技术推广总监

序二

为这本迷你书写序，似乎是我的宿命。由于工作原因把这个任务推迟了一段时间之后，后果居然是不仅写了序，还成为了另一篇序的译者。

互联网 10 年，始于 2000 年。对于互联网来说，那是个最好的年代，也是最坏的年代。互联网泡沫的破灭并没有湮灭技术与创新的光辉，反而为今后的发展打下了坚实的基础。随后互联网开始了 10 年的高速发展。时至今日，互联网像一条巨大而充满吸力的纽带，把各种 IT 服务相关的技术、应用和实现都吸引过来，形成了一个完整而庞大的互联网生态圈。在这十年中，互联网从不缺乏优秀的技术创新，其中 RIA（富互联网应用）及其相关技术便是其中重要的组成部分。

在 RIA 的江湖中，曾经有一个三足鼎立的传说。那就是 Adobe 公司的 Flex，微软公司的 Silverlight 和 Java 阵营的 JavaFX。Flash 由于 Flex SDK 的支撑，很早就从单纯的动画展示转入 RIA 领域，而且由于 Flash 的普及（Flash 播放器是下载次数最多的独立软件），Flex 目前应该是三大技术体系中市场份额最大、应用最广泛的技术；Silverlight 是微软推出的跨浏览器和跨平台的插件，能在微软的.NET 上交付炫目的多媒体体验和有丰富交互功能的 Web 应用，已经对 Flex 有了很大的冲击；JavaFX 是未被收购前的 Sun 公司在 2007 年推出的用来对抗 Flex 和 Silverlight 的桌面应用。

我个人一度对 JavaFX 的原生调用机制充满幻想，认为它与 JavaEE 的结合会为 Java 社区的 RIA 企业应用带来很大的生产力，但是非常遗憾，随着 Sun 公司的没落，JavaFX 也日渐式微。从这本迷你书也可以看出这一点，在实际应用中已经很少谈及 JavaFX 了。Silverlight 由于其技术定位的原因，应用更多的局限在 .Net 阵营。那是否意味着 Flash/Flex 技术已经一家独大了？技术领域从来不缺乏竞争者。AJAX 技术一直在与 Flex 一起支撑和丰富着 RIA 的应用，包括互联网和企业级应用。随着 jQuery、Ext 等优秀的 AJAX 框架兴起，它们已经可以完成更多的之前只能由 Flex 完成的 RIA 体验，但是似乎还不能撼动 Flex 的霸主地位。直到 HTML5 及其相关技术的出现，格局打破了，Flex 的危局到来了么？

HTML5 的出现，像一道亮丽的风景线，打破了 RIA 领域的格局。大家开始真正的认识到，Web 浏览器才是不折不扣的最大的开发平台。HTML5 同样咄咄逼人，其特征之一就是减少浏览器对外部插件的需求（比如 Flash）。当然，Flash/Flex 的特点之一也是完成 HTML 无法完成的工作，不过，现在这一点还是真的吗？HTML5 提供了更优秀的 Web 元素处理机制，提供用于绘画的 Canvas 元素，用于媒体回放的 Video 和 Audio 元素，提供对本地离线存储的

更好支持，提供针对移动设备的支持等等。HTML5 似乎无所不能，Flex 感到危机了吗？

就我个人的视角，HTML5 具备了与 Flex 竞争的能力和资格，但是现阶段 Flex 依然是 RIA 领域的老大。HTML5 的优势在于创新性的利用了浏览器自身的能力，而且有业界最富创新精神的公司苹果和谷歌的大力推进。Flex 的优势在于市场占有率，及其对游戏交互、视频和文档处理能力。浏览器的因素同样不可忽视。目前市场上占据浏览器最大份额的 IE 浏览器的所有版本都不能完全支持 HTML5 特性，除了还未发布的 IE9。而其他浏览器，Chrome、Firefox、Safari 和 Opera 正在快速抢占浏览器市场，而且这些浏览器都在积极的支持 HTML5 特性。浏览器之争会很大程度上影响 RIA 和 HTML5 的竞争走向，包括 HTML5 的普及程度。

作为从业者，无论你想挽救 Flex 的危局，还是参与 HTML5 的盛宴，无论是你想进入 RIA 领域，亦或已经置身其中，我都建议你仔细阅读这本 RIA 迷你书的每一篇文章，这些内容是 RIA 领域的专家和实际从业者的知识共享和经验总结，范围之广涉及趋势、技术、框架、语言、应用、性能等，实在是 InfoQ 为大家打造的一本居家学习之 RIA 必备读物。RIA 技术之美，读起来很美！

池建强

瑞友科技 IT 应用研究院任副院长、InfoQ 架构社区编辑



QCon全球企业开发大会（北京站）



大会分为十二个主题：

- ➔ 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
- ➔ 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

时间：2011.4.8~2011.4.10

地点：北京·京仪大酒店

网站：www.qconbeijing.com

主办方：InfoQ

电话：

13911797020

010-89880682

010-64738142

E-mail:qcon@cn.infoq.com

目录

序一.....	1
序二.....	3
虚拟座谈会：RIA 和 AJAX 技术的现状与展望.....	6
AJAX 应用开发：实践者指南	14
案例研究：ECLIPSE 富 AJAX 开发平台在 CAS SOFTWARE AG 项目中的应用.....	22
使用 FLASH BUILDER 4 BETA 进行数据为中心的开发.....	32
FLEX 技术在企业级开发中的应用.....	38
高效率的超大规模 FLEX 开发	48
SPRING BLAZEDS INTEGRATION 简介与入门	56
JAVA 程序员学习 FLEX 和 BLAZEDS 的十三个理由.....	66
微软设计产品市场总监 FOREST KEY 谈 SILVERLIGHT	72
虚拟座谈：HTML5 来了，JAVASCRIPT 框架会如何发展.....	75
RICHCLIENT/RIA 原则与实践.....	83
设计者-开发者工作流中的迭代模式	98

虚拟座谈会：RIA 和 Ajax 技术的现状与展望

作者 [Scott Delap](#) 译者 [王波](#)

InfoQ 曾经通过 Email 针对 RIA 和 Ajax 技术的现状和展望专门成立了虚拟小组。该小组召集了多位对社区有着卓越贡献的技术精英，对此进行了专门的访问。文中我们可以看到当时每位组员针对不同问题的答复，大家可以和现在 RIA 和 Ajax 的进行比较，可能会发现一些值得思考的地方。

最近，InfoQ 通过 Email 针对 RIA 和 Ajax 技术的现状和展望专门成立了虚拟小组。该小组召集了多位对社区有着卓越贡献的技术精英，他们是：

“ Mozilla 公司开发工具主管 Dion Almaer

Curl 公司的首席策略官 Jnan Dash

SFEIR Author OnGWT.com 公司的首席技术官 Didier Girard

Java Web 用户组创始人 Peter Pilgrim

微软客户端平台宣传部门主管 Tim Sneath

Adobe 公司的富互联网应用宣传人员 Ryan Stewart

随后，我们会看到每位组员针对不同问题的答复。

- 尽管 RIA 技术已经出现，但是 Web 还是由“页面”占主导而非“应用程序”。然而，去年随着网站对视频和交互浏览等嵌入了“迷你应用程序”，我们已经看到了这种现象正加速改变。而最终由 RIA 导致了这场变革。

Almaer : Web 页总会占有一席之地，但我认同我们正朝着 Web 应用程序的时代迈进。当然，Web 应用程序早已出现，但我们正经历着这场改变：

- 开发反应灵敏的 Web 应用程序技术还不成熟
- Ajax 勉强组合了一些技术和标准来给终端用户提供开发应用程序所需的简要工具
- 时至今日，Ajax 就像第一架飞机那样。它们勉强的凑合在一起并发挥几分作用，但离理想的距离还很遥远。出现问题会让人非常惊讶，但毕竟这不是喷射机的时代。简洁实用的软件就让人心满意足了

Girard：互联网聚集了大量页面。这种情况已经持续多年了。但在最近的两年，我看到了企业开发的飞速发展。现在，大部分的应用程序都是单页面应用程序，而 RIA 才是它的核心。

Dash：我认为 RIA 在消费者当中还有很大的空间，特别是从静态可刷新页面转移到动态交互应用程序的要求非常强烈。对于企业级 RIA，除了提供交互和可设定状态的交易应用程序外别无选择，这正是它们在客户端/服务器模型中过去常常用到的。然而，企业级 RIA 在美国还没起步。相反，我们在日本已经看到大量的企业在使用 Curl RIA 平台。

Sneath：回答这个问题相当困难，因为它基于“RIA”术语的业界标准假设。例如，长期以来我们身边出现了强大的互联网应用程序框架，我们推出 ASP 已经 12 年了，它提供了基于 HTML 的抽象层来跨页面支持状态属性，丰富的控件和在客户端/服务器开发之间的无缝开发。下个月 AJAX 就 10 岁了，随着 IE5 配备了第一个实现技术，这项技术后来作为某项标准而被采纳。

过去的 20 年间，最大的进步莫过于像 Silverlight 和 Flex 等框架提供的丰富图形功能，以及更加强大的编程工具和以往单单用 HTML 和 JavaScript 都难以实现的架构。我们终于可以集成媒体、控件、图像及编译代码并用互联网进行发布，同时给设计师和开发人员提供各式各样的工具。这些技术还不成熟，但是部分市场已经足够成熟来进行广泛的应用。

Pilgrim：我相信 JavaFX RIA 在去年底已经应用到桌面。我把它限定在桌面是因为我们仅得到了针对移动电话和移动设备的第一个 JavaFX Mobile 正式版。仅有少量的网站应用了该技术。例如 Adobe 开发的 Flex 应用程序 Dashboard 已经一年了。微软发布的富应用程序 Silverlight，应用到 NBC 转播奥林匹克运动会中，奥巴马总统的就职典礼也应用了该技术。除了 RIA 的新星，我想不出更好的词来形容 JavaFX 了。我知道的最大型的 RIA 应用程序是 TweetDeck，它是 Adobe RIA 应用程序。我现在每天都使用 TweetDeck，尽管在它的界面上我没有发现任何措施阻止别人建立它的 JavaFX 版本。

Stewart：我仍然不确定这项技术是否已经成功。我想在咱们扩展特定的范畴来包含像小组件、视频、更丰富的桌面应用程序和丰富的浏览器应用程序，很明显带有诸多 RIA 的元素，但是还是有很多公司并没有发现它的使用价值。然而，我确信我们已经到达了一种特殊的境地，因为设计工具逐步实现了设计师的想像。一旦这项技术成功的话，RIA 就会更容易的占领各个领域，因为设计师和开发人员可以就 RIA 的强大外观显示而更好的合作，无需牺牲开发方面的任何东西。

2. 引入 RIA 技术的同时，我们强调可移植性。然而，用户要求与文件系统、停靠和任务栏、日历和其他与 OS 进行的本地操作。你认为 RIA 平台在未来的几年会着重集成这些功能

吗，抑或继续朝着互操作的方向呢？

Stewart：这变得非常的有趣。我们仍然倾注了巨大的能量和热情来关注浏览器。在 Web 的诸多方面，我认为大家已经给桌面判了死刑。但它在很多方面都有价值，例如：访问文件系统和利用底层的操作系统等。而且我认为它会变得越来越重要。我确实觉得我们开始看到浏览器和桌面之间的联系，但是随之安全成为关注的焦点。同时，我喜欢像 AIR 或 Appcelerator 那样的技术方法让我们使用强大的 Web 开发语言来利用某些桌面功能。

Sneath：我认为用户对于最佳应用程序体验的需求是必然的，我母亲并不明白 Web 和 Windows 应用程序之间的区别，她只想完成工作，无论她是在本地进行绘图抑或是利用因特网上的资源。现在至少，你需要在这里做出选择，即唯一适合的解决方案就是本地应用程序，同样，有很多解决方案以移动方式来发布的。很难两种方案同时发布，并且每种平台目前都利用非标准和非通用的方式，这种解决方案看起来既不像本地应用程序也和其他类型的系统难以集成。

微软客户端平台的其中一项优势是具备兼容的解决方案来满足两种需求：使用 Silverlight，我们针对发布跨平台的大型解决方案提供了轻量级的运行时；同时我们有 WPF，它是 Silverlight 的父集，适合建立 Windows 应用程序来完全访问底层操作系统并可以利用本地硬件性能。因此，很自然地建立跨 Silverlight 到 WPF 的解决方案就有可能了，你可以建立自己的应用程序并扩展它的使用范围。

Dash：让我们再次来区分一下用户级 RIA 和企业级 RIA。用户对互操作性的需求比起集成需求更加强烈。无论在哪里，我们都会受益于客户 os，比如：利用视频呈现驱动，Curl 利用它们来提高性能。通过客户端集成和服务器端的互操作性实现，顺带提一句，Curl 并无任何服务器端代码。

Pilgrim：这个问题提得很好。我觉得开发 RIA 解决方案可以给 Web 浏览器用户带来更好的用户体验。Web 和桌面之间的区别已经淡化。Java 6 更新 N 系列允许用户从浏览器拖动 Java Applet 到桌面上。Adobe 有 AIR。只要我们反思这种可能性，我们就会立刻陷入思想斗争之中：谁主宰桌面呢？那些应用程序可以这样做？抑或用户或者企业会把那个应用程序植入桌面呢？

如果我们把 RIA 定位在这个水平，那么 OS GUI、隐私和安全模型的互操作性概念就会变得极为重要。还有一个项目叫做 WidgetFX，理论上允许团队内建小插件到谷歌桌面或者 Windows Live 环境。利用 Java Enterprise Web API 开发企业级应用程序就会非常有趣了，好比泛滥的 Web 框架或者可移动应用程序(JSR 186,286 和 WSRP)。

我认为用户（消费者）将影响应用程序的成功与否，例如：TweetDeck。销售捆绑的企业级 RIA 或者小插件，用 JavaFX 开发的话将需要更多的工作量。

Almaer：我们两者都需要。我们钟情于互操作性，但那会给开发人员带来麻烦（就像 Java 编程早期那样，他们链接 DLL 都不是那么容易）。这项挑战也许意味着你会说：“我觉得我不能用这项技术来完成，所以我会回到桌面 API 平台。”现在是推进 Web 和获取桌面服务 API 的时候，所以我们像本地平台那样完成所需要的功能。

我们披上了一层安全的面纱。“我们不能那么做，因为我们是在 Web 环境中”。但是这非常的虚伪。要让用户获取相关功能，我们之前都是在本地平台上编写应用程序，并要求我们的用户下载和安装它。基于这一点，用户不可能完全安全。他们运行我们的可执行文件，这些文件可以完全控制他们的系统并为所欲为。我们要 针对 Web 应用程序建立安全的不会对用户进行干扰的可信的模型，我之前已经讨论过。（<http://almaer.com/blog/application-trust-models-expanding-web-applications-out-of-the-sandbox>）。

3. 当前视频应用程序是 RIA 最主要的类型之一。你认为在未来的 12-18 个月里面，RIA 技术会被那一种类型的应用程序所采用？

Dash：Curl 针对基于 Web 应用程序所要求的高可扩展性、可靠性、安全性、性能和可预测性。在过去的 15 年，从客户端-服务器应用程序的转变，大大地降低了 TCO（总体成本）。坦白的说，视频在这些应用程序中并没有很高的优先级。那就是 Curl 为 400 家大企业客户用作关键业务应用程序。而并非单个客户发布视频。

Pilgrim：对于其他媒体，很明显在音频方面。2008 年，JavaFX 取得了巨大的成功。这是第一次，开发人员可以编写简单的视频播放器并把它作为 Applet 发布。

对于 JavaFX 的其他应用就是移动应用程序。我看到有游戏和移动应用程序。实质上，是因为 JavaFX Mobile 是基于 JavaME 栈之上。

Sneath：视频文件仍然非常庞大。今天，大部分的视频内容的发布还远远没有到达“高清”的要求。视频还有许多革新的空间，不管是在前端抑或是后端，我们都很清楚看到它在未来的几个月内都给 Silverlight 带来的巨大的机会。

除此以外，我们认为我们还将在来年看到 AJAX 更加依赖于客户端框架。我常常惊叹于众多高手能够提取 HTML 和 JavaScript 代码，但创建像 Gmail、Outlook Web Access 或者 Facebook 这样的高端体验所需要的技巧，却只有少部分的开发者掌握，即使有像 jQuery 和 ASP.NET AJAX 这样的高级框架。像 Silverlight 这样的框架通过抽取掉不必要的复杂性使得创建富英特

网应用程序的过程变得更加大众化，所以开发人员可以集中在体验本身。

Almaer：我看到了人们对于转用这项技术在 Web 上能完成那些功能的总体趋势有所改变。曾经有那么一段时间，人们会认为创建基于 Email 客户端的 Web 应用程序的人是疯了。现在却有很多人用 Gmail/Yahoo!Mail 等等。还有一些人进行地图搜索。我们开始建立 Bespin 的原因是为了弄清楚我们可以多深入地建立富 Web 代码编辑器，而不是华丽的代码。我预期这个趋势还会继续，并在你知道以前，我们会看到像视频编辑器和 Photoshop 等实实在在出现在 Web 上。看一下 iWork.com 站点就会给你一点启发。

Girard：在我的领域里，RIA 主要采用合作开发。我看到越来越多的公司会因他们的团体需求而且要求更加复杂的应用程序。

Stewart：视频仍然是关键。但我认为更加重要的一件事是即时协作应用程序。Web 开始进入即时状态，我们常常网聊的同时，却不能做其他的事情。随着人们上网的时间越来越长，像 Facebook 那样的网站会成长地越快，对于点之间和朋友之间的即时协助存在着巨大要求。我认为 RIA 特别能满足该需求，因为它结合了视频、音频和丰富的界面。Adobe 和 Microsoft 都有一些强大的终端解决方案，使得 RIA 前端能够满足这项需求。

4. 相对于其它的框架和语言（如：Ajax、GWT、Curl、Flex、Silverlight、JavaFx 等），它的最大优势在哪里呢？

Sneath：我只能选择其中一项吗？

Silverlight 其中一项核心竞争力就是在如此小的程序包内却实现了如此强大的功能。如果你正寻找媒体播放器，Silverlight 实现了平滑的流媒体、数字内容保护和真正的高清支持。如果你正寻找一个平台来建立下一代业务应用程序，Silverlight 提供多种控件、数据绑定、Web 服务支持，全部都配备了强大的.NET Framework 和 Visual Studio 工具。如果你想创建让人叫绝的用户体验，Silverlight 提供强大的图形和媒体集成，配备了新技术 DeepZoom，以及可集成到 Adobe Illustrator 和 Expression 设计套件工具的基于 XML 的标记语言。全部加起来还不到 4M，支持 VB、C#、JavaScript、Ruby 和 Python，并在 PC 机和 Mac 上免费可用。

Dash：Curl 的最大优势是开发者的生产力（一种涵盖了一系列的文本、图形、网格以及面向对象的类型和类）运行时的可扩展性优势、大容量的数据处理、把客户端-服务器代码快速编译成机器码和高级别的安全性。这是所有大公司对于关键业务应用程序的基本需求。

Pilgrim：JavaFX 最有用的是和 Java 兼容。所以你可以利用源代码的知识库，例如：Apache 软件基金会、Codehaus 和 SpringSource。

第二个有用之处是用于开发的标准场景图像 API、内建到语言和平台中的动画绑定和触发器。

Almaer：我的观点是开源 Web、Ajax 等。好处是在这些技术前面不会标有公司名。既然已经有 MICROSOFT Silverlight 和 ADOBE Flex。我们就需要 Ajax——开源的 Web 平台。这意味着我们有机会让开发人员控制平台，而不是今天和某个公司密切合作，但明天呢？然而，我们不能仅以贩卖“开放相关的内容”生存，我们需要在平台级别上竞争，同时我们庆幸有通用的技术，尽管别人已经完成了这些工作。

Girard：对于生态系统来说，GWT 更紧密的技术。

- 许多开发者懂得 java 语言，他们都是忘我的 RIA 开发者，对于 java 开发人员转到 GWT 只需一天时间
- 浏览器就是 GWT 应用程序的运行时。它是广泛使用：GWT 应用程序可以在所有浏览器上运行，甚至在 iPhone 或 Android

Stewart：我认为 Flash 的渗透性是一个大课题，因为进入的门槛非常低，但有很多好处。第一，Flex 是目前最成熟的技术。我们正使用该框架的第四个版本，Flex 来自于企业背景所以该框架非常牢固并被广泛使用。我认为 Adobe 的设计社区也带来了好处。设计是 RIA 的核心之一，Adobe 做得比任何公司都好，我们正给我们的开发者社区注入这种精神。我也觉得 ActionScript 非常强大。从第一个版本至今经历了漫长的过程，它是真正的 OOP 语言，对于 JavaScript 开发人员来说很容易掌握，对于 C# 或 java 开发者也一样。

5. 相对于其它的框架和语言，它的最大不足在哪里呢？

Almaer：其他平台都有归属。它们有统一的文档。你知道去哪里查找。使用 Ajax，存在的普遍问题是它难以查找到相关的资料。作为一个庞大的社区，我们在 2009 年会做相应的工作。

Pilgrim：第一个致命的缺点是它缺乏基于组件的复杂场景图形库，这并不使用 Swing。例如，没有纯粹的场景图 SplitPane 组件、无 TabbedPane 或者 JTable 的替代物。你不得不在编程的时候自己动手。我已经编写完 SplitPane.fx 和 BorderLayoutPanel.fx 组件。然而，方便可用，附在 JFXtras 项目中。除此以外，我不确定 Sun 公司还做了什么。在高级组件方面，FX 远远落后于 Silverlight 和 Flex。

第二个致命的弱点是 JavaFX 媒体目前没有录制功能。换句话说，使用的客户端没有媒体编码功能。目前，你编写 JavaFX 版的 Sims-On- Stage (Electronic Arts 收购的流行 Web 卡拉OK 网站) 非常困难。必须加强 Java Media Component API。如果你准备进行流媒体录制的话，可以使用 JavaSound API 并做大量的技术性高级编程。在这种情况下，FX 落后于 Adobe。Java

和 JMC 云可能超过 Adobe，如果它们开始视频的子像素呈现并提供混响音效，例如，录制和处理功能。

JavaFX 是目前最经济的应用程序。你可以使用相对简单的 GUI 根据市场数据给用户实现流业务信息。建立带表格数据的高性能 JavaFX 应用程序也许还要等上一段时间。

Dash : Curl 最大的不足之处是大家对它的认识相对比较模糊。我们大部分的客户在发现 Curl 可以解决高性能和安全需要之前，尝试使用 Ajax 和 Flex 都失败了。同时，视频呈现不是我们的强项，之前也不是我们的目标之一。

Sneath : 我们从竞争对手那里听到的异议是 Silverlight 并没有在客户机上大量的安装。事实上，我们很高兴 Silverlight 在市场上的引入力度。Silverlight 2 已经发布 5 个月了，在这段时间，我们看到数以千万计的电脑进行安装，机器上 Silverlight 的安装率是 FireFox 和 Google Chrome 的总和。很自然地，我们不会满足于取得的成绩，我们会继续对该平台进行投入，但随着我们的展示网站像 Netflix、March Madness、Sky UK 和 AOL，我们有信心安装率会继续提高。

Stewart : 其中一个感觉是，Flash 仍有很多观念上的问题需要解决，因为人们并不把它视为实现应用程序的方式而是骚扰广告媒体。庆幸的是，视频某种程度上改变的这种观念，但周围仍然有很多“骚扰 Flash”。另外一个弱点是我认为 Flash 需要适应 Web 生态系统。很多东西像 SEO、链接、分析技术都建立在 HTML 模型之上，所以大块的 Web 如何运作以及赚钱实际上依赖于 HTML 和 JavaScript 的。我们只是看到 Flash 在某些方面表现更好，但是最终它总让人觉得有点不同，这正是我要研究的缺陷。

Girard : 浏览器的弱点就是 GWT 弱点。最重要的是关于 2D 图像。如果浏览器更好的支持画布和 SVG 的话就好了。

6. 大多数 RIA 语言并不同时用于开发客户端和服务器。主要的后台程序用 PHP、Java 和.NET 等实现。如何看待这种[多语言编程模型](#)对 RIA 的影响呢？

Dash : 我喜欢“精通多种语言”这个词。坦白说，这会让人混淆。我观察到业界正分化为双语言方案（例如：C# 和 XAML、ActionScript 和 MXML、JavaFX 和 Java）。有些人会说 2 种比 4 种好而 4 种又比 6 种好。但是我们使用 Curl 的人相信一种语言包含演示层和逻辑层。因此，MIT 的研究人员设计一种统一的语言来解决整个方案。这就导致了可怕的“程序员经济”，一些我们在 RIA 业界没有注意到的东西。我们的客户经验极大地证实了这项优点。Curl 是相当好的建立富客户端应用程序的多范例语言。我们希望你能够掌握这项语言。

Stewart : 我希望它很快会得到采纳。学习新语言是令人畏惧还是有趣取决于你如何看待它，

但文中的观点，针对应用程序结合不同的语言变得非常重要。最后，我认为让程序员自由使用最适合他们的语言组合才是最有生产力的。如果每个人都同意那种说法，我们认为每个 RIA 供应商都会看到高采用率和广泛应用，这也会让我们得到鼓舞。

Almaer：我喜欢 RESTful 的世界。我认为它有很好的解耦功能。然而，我们正探索统一的前端和后端观念，在我们当中使用 JavaScript。使用富服务器端 JavaScript 平台，移植 JavaScript 到浏览器，在接下的日子中，我们会得到 Smalltalk 支持的有趣解决方案吗？我们正要探索。

Girard：对于 GWT 情况却并非这样。你可以对客户端和服务器端使用相同的语言。这实在是太好了。

Sneath：这当然是 Silverlight 的独特之处。如果你是.NET 开发人员，你肯定知道 Silverlight，相同的编程模型，它只是整个.NET Framework 的一个子集。实际上，你可以利用在基于 ASP.NET 服务器上运行的相同源码，并在基于 Silverlight 的客户端应用程序中使用它，而无须任何修改。由于.NET 的普及性，你不仅仅可以分享代码还可以跨越多种不同的设备类型来分享技巧：从 Xbox->移动设备 ->Web 浏览器->Windows 客户端应用程序->云数据中心服务器。

Pilgrim：对于 JavaFX 开发人员，我认为你需要用 Java 编程解决难题。实质上，如果你使用 EJB、JMS 或者使用客户端服务器协议，像无线发送 XML。我看不到 JavaFX 对服务器端的用处。给我一个用例好吗？

也许某人可以编写动态生成 JavaFX 代码的 Grails 应用程序 编译它然后回发引用动态生成 JAR 文件的 JNLP 文件。是的，你可以用技术实现，但是使用它的商业价值在哪里呢？换句话说，当我们遇到所谓的信贷紧缩的时候，它浪费时间和金钱。相反，使用这些语言来解决业务问题，才是真正的改革方式。基于这一点，我认为 JavaFX、Flex 和 Silverlight 都是可行的解决方案和平台。

原文链接：<http://www.infoq.com/cn/articles/ria-panel>

相关内容：

- [Grails Ajax富客户端插件大比拼](#)
- [创建并扩展Apache Wicket Web应用](#)
- [Joe Walker谈DWR](#)
- [利用GWT开发高性能Ajax应用、JavaScript多线程编程简介](#)

Ajax 应用开发：实践者指南

作者 成富

Ajax 技术的出发点在于改变传统 Web 应用使用时的“操作-等待页面加载-操作”的用户交互模式。这种交互模式会打断用户正常的使用流程，降低用户的工作效率。Ajax 技术的交互模式是“操作-操作-操作”。用户并不需要显式地等待页面重新加载完成，而是可以不断地与页面进行交互。页面上的某个局部会动态刷新来给用户提供反馈。整个交互过程更加平滑和顺畅。这是 Ajax 技术得以流行的一个重要原因。本文简要介绍了 Ajax 应用开发的各个方面以及相关的最佳实践，但对一些细节内容没有展开讨论。

目前的 Web 应用开发基本上都是围绕富互联网应用（[Rich Internet Application](#)，RIA）展开。RIA 的实现技术有很多种：[Ajax](#)、[Flash](#)、[JavaFX](#) 和 [Silverlight](#) 等。Ajax 技术的优点在于它是构建在开放标准之上，不存在厂商锁定的问题；同时也不需要额外的浏览器插件支持。Ajax 应用对搜索引擎也比较友好。对开发者来说，Ajax 所需技术的学习曲线也较平滑，容易上手。本文简要介绍了 Ajax 应用开发的各个方面以及相关的最佳实践，但对一些细节内容没有展开讨论。

Ajax 简介

Ajax 技术的出发点在于改变传统 Web 应用使用时的“操作-等待页面加载-操作”的用户交互模式。这种交互模式会打断用户正常的使用流程，降低用户的工作效率。Ajax 技术的交互模式是“操作-操作-操作”。用户并不需要显式地等待页面重新加载完成，而是可以不断地与页面进行交互。页面上的某个局部会动态刷新来 给用户提供反馈。整个交互过程更加平滑和顺畅。这是 Ajax 技术得以流行的一个重要原因。

Ajax 构建于一系列标准技术之上，包括 [HTML](#)、[JavaScript](#) 和 [CSS](#) 等。在这些技术中，HTML 是作为应用的骨架而存在的，展示给用户最基本的内容。CSS 则为 HTML 表示的内容提供易于用户阅读的样式。JavaScript 则为应用添加丰富的交互行为，为用户提供良好的使用体验。

Ajax 应用开发：实践者指南 Ajax 技术的出现使得应用中一部分的逻辑从服务器端迁移到了浏览器端。浏览器的作用从简单的渲染页面和表单处理，上升到了处理一部分业务逻辑。

一般来说有两种类型的 Ajax 应用风格，一种是仅少量使用 Ajax 技术来适当增强用户体验

(Ajax Lite) , 另外一种则是大量使用 Ajax 技术来达到与桌面应用相似的用户体验 (Ajax Deluxe) , 提供诸如鼠标右键、拖拽和级联菜单等。开发人员应该根据应用的特征选用合适的风格。

浏览器兼容性

开发 Ajax 应用的时候要面对的一个重要问题就是浏览器兼容性。虽然 Ajax 技术基于 HTML 、 JavaScript 和 CSS 等标准技术 , 但是不同的浏览器厂商对于这些标准的实现程度有着很大的差别。同一浏览器的不同版本之间也会有一些不同。新版本可能会修复旧版本上的问题 , 也可能会带来新的问题。不过总体的趋势是浏览器的实现越来越向标准靠拢。

解决浏览器兼容性的第一步是确定应用要支持的浏览器种类和版本。这个决策取决于应用的目标用户和特定的应用需求。对于一般的通用 Ajax 应用来说 , 可以按照浏览器的市场份额和支持某种浏览器所需的代价来确定。雅虎提出的分级式浏览器支持 ([Graded Browser Support](#)) 是一个很好的参照 , 从其中给出的 A 级浏览器开始是一个不错的选择。从特定的应用需求来说 , 某些使用了 ActiveX 控件的 Ajax 应用就只能在 IE 上运行 ; 而开发针对 iPhone 的应用则只需要考虑移动版 WebKit 就可以了。

就 Ajax 应用的三个组成部分来说 , HTML 的兼容性问题比较少 , 毕竟目前主流的 HTML 4.01 规范已经有 10 年的历史了 ; 在 JavaScript 方面 , JavaScript 语言核心部分基本上没什么问题 , 而文档对象模型 (DOM) 和浏览器 对象模型 (BOM) 部分的兼容性问题相对较多 , 这主要是因为浏览器长生对规范的支持程度不同以及各自添加了私有实现。使用一个流行的 JavaScript 库就可以解决这些问题 ; CSS 方面的兼容性目前是问题最多的 , 而且没有比较好 的库的支持。在下面会着重介绍 CSS 的兼容性问题。

富含语义的 HTML

HTML 语言本身上手比较简单 , 只是一些元素的集合 , 只需要了解这些元素及其属性的含义即可。这些元素既有与文档结构相关和富含语义的元素 , 也有与页面的展示相关的元素。一个好的实践是只使用与文档结构相关和富含语义的元素。从 HTML 语言规范的历史也可以看到这个趋势。 HTML 语言规范的历史比较长。在 [HTML 最初的草案](#) 和 [HTML 2.0](#) 中 , HTML 只包含描述文档结构的元素。在 [HTML 3.2](#) 中 , 很多与展示相关的元素被引入进来。[HTML 4.01](#) 规范试图解决这个问题 , 许多与展示相关的元素被标记为废弃的 , 不推荐使用。[HTML 5](#) 更进一步 , 引入了更多的富含语义的元素 , 同时移除了一些在 HTML 4.01 中被废弃的元素。应用这种实践进一步划分清楚了 HTML 和 CSS 在 Ajax 应用中的职责。

编写 HTML 文档的时候首先需要选用合适的[文档类型声明 \(DTD \)](#)。目前来说最合适的是[HTML 4.01 过渡型](#)，即<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">。在编写 HTML 文档的时候，需要使用合适的元素。HTML 规范 中的一些元素，如、、<abbr>、<blockquote>、<cite> 和<code> 等，对开发人员来说比较陌生。但是这些元素富含语义，有适合它们使用的场景。如果使用的是<div> 和等通用元素，需要使用富含语义的 class 属性来增加语义，说明元素的作用。在 HTML 文档编写完成之后，最好使用 W3C 提供 的 HTML [文档验证器](#)来验证文档。

CSS

CSS 的语法非常简单，包含的元素也非常少，其中最主要的是样式规则集。样式规则集是一系列样式声明规则的集合。每个样式规则由选择器和声明两部分组成。选择器用来选择文档中的元素。这些元素将被应用上与选择器对应的样式声明。CSS 不同版本规范所支持的选择器类型不同，尽量使用常用并且简单的选择器以获得更好的浏览器兼容性，如 ID 选择器、class 选择器和元素选择器。

使用 CSS 的时候会遇到的一个很大的问题是浏览器兼容性。经常会遇到的情况是某种样式在 A 和 C 浏览器上应用正常，而在 B 浏览器上则使用不正常。等到把 B 浏览器调好了之后，却发现 C 浏览器上显示出现错误。解决这个问题的基本原则是要首先确定几个基准的浏览器和开发基本的布局样式表。基准浏览器一般是所要支持的浏览器中对 CSS 规范支持较好的浏览器。基本的布局样式表保证在基准浏览器上应用的页面布局是正确的。目前的浏览器在 CSS 页面布局这一块的兼容性最差，尤其在盒模型（box model）、浮动定位等方面。而在显示样式，如字体大小和颜色等方面，则基本上没有什么问题。

接下来要做的是让基本的布局样式表在除基准浏览器外的其它浏览器上正确工作。这个时候就需要对某种版本的浏览器来应用特殊的样式，从而修正样式上的不一致。一种做法是利用一些招数（hack）。招数是利用浏览器本身对 CSS 规范支持的不完善或是实现上的 bug 来识别浏览器并应用样式。另外一种做法是通过 JavaScript 来检测当前浏览器并应用样式。招数可能会随着浏览器的版本更新而变得不可用，因此尽量少使用。

在一般 Ajax 应用，最常被应用招数的是 IE 6。因为 IE 6 对 CSS 规范支持不完善，而且存在比较多的 bug，但是 IE 6 的用户目前还是数量众多，还是有支持的必要。对 IE 浏览器应用特殊样式的更好做法是使用 IE 独有的[条件注释](#)。

当应用所包含的 CSS 文件比较多的时候，开发和维护这些 CSS 文件就成为一件比较困难的事情。一个解决办法是把面向对象的思想引入到 CSS 的编写过程中。两种重要的原则是组件化

和单一职责。组件化的做法是开发出针对页面上某类元素的样式组件。这些样式组件可以在不同的页面中任意组合使用。单一职责指的是把表示结构和外观的样式分开。与结构相关的样式包括大小和位置，外观的样式包括字体大小、颜色和背景图片等。

DOM 查询与操作

DOM 操作是 Ajax 应用中页面动态和局部刷新的实现基础。DOM 定义了文档的逻辑结构，以及对文档进行访问和操作的方式。通过 DOM，开发人员可以在文档中自由导航，也可以添加、更新和删除其中的元素和内容。通过 DOM 规范提供的 API 就可以完成对文档的查询与操作。不过 DOM 的原生 API 使用起来比较繁琐，最好使用 JavaScript 库来完成查询和操作。

通过 DOM 操作对当前页面进行修改一般都是通过响应用户的事件而发生的。这些 DOM 操作中一部分是纯浏览器端实现的，另外一部分则需要服务器端的支持。服务器端可以选择返回数据或 HTML 片段。返回数据的好处是传输的数据量小，易于与第三方应用集成。不足之处在于浏览器端需要额外的操作来完成展示。浏览器端可以使用 DOM 操作或是模板技术来生成 HTML 片段。服务器端也可以通过 JSP 和 Apache Velocity 等模板技术来生成 HTML 片段，并直接返回给浏览器。浏览器只需要直接使用即可。这种做法的好处是浏览器端实现简单。不足之处在于与展示相关的逻辑同时存在于服务器端和浏览器端，不容易维护。

有一些比较好的实践可以提高 DOM 操作的性能。首先是使用文档片段(document fragment)。当需要插入大量节点的时候，首先把这些节点添加到一个文档片段中，再把此文档片段添加到文档上。这样可以减少页面的重新排列。其次是使用 innerHTML 来更新文档内容，速度比使用 DOM API 要快。最后是通过 cloneNode() 来创建多个结构相同的元素。

事件处理

Ajax 应用与用户的交互是通过响应用户事件的方式来完成的。浏览器负责捕获用户的行为并产生各种不同的事件，应用处理这些事件。浏览器中可以产生的事件种类比较多。事件产生之后，会按照一定的过程在当前文档树中传播。事件所产生的节点称为目标节点。完整的事件传播流程是从文档的根节点开始向下传播到目标节点（捕获阶段），然后再往上传播回根节点（冒泡阶段）。当事件传播到某个节点上的时候，就会触发此节点上绑定的处理方法。（IE 只支持冒泡阶段。）需要注意的是事件处理方法中 this 所指向的对象的值，有可能是当前节点或是 window 对象。通过 JavaScript 库提供的支持来绑定事件处理方法，可以避免这些不一致。

在绑定事件处理方法的时候，可以利用事件的传播机制来减少事件监听器的数量。如当需要

为一系列元素添加鼠标点击的事件时，可以把该事件添加到其父节点上。在完成对事件的处理之后，可以终止事件的传播，还可以阻止浏览器的默认行为。

选用合适的 JavaScript 框架

目前存在非常多的 JavaScript 框架，有开源的也有商业的。比较流行的有 [jQuery](#)、[Dojo](#)、[YUI](#)、[ExtJs](#)、[MooTools](#) 和 [Prototype](#) 等。选用流行框架的好处是有比较大的社区支持，遇到问题的时候容易获得帮助。流行框架的文档和示例也比较丰富。使用不同的框架会给应用带来不同的实现风格。jQuery 的使用者对方法的级联情有独钟，Dojo 的爱好者则倾向于把页面上的不同部分划分成 dijit 来实现。

选用什么样的框架的因素很多，技术的和非技术的都有。一般来说，轻量级的框架，如 jQuery 和 Prototype，上手比较容易，但是可复用的组件较少；而比较庞大的框架，如 Dojo 和 ExtJs，则学习曲线较陡，但是可复用的组件非常多，适合快速开发复杂的 Ajax 应用。

构建过程

Ajax 应用也需要一个完整的构建过程。构建过程的主要目的是提高 Ajax 应用的质量和性能。这个构建过程可以包含的步骤有：

1. JavaScript 代码的潜在错误和代码风格检查。通过集成 [JSLint](#) 可以找到代码中潜在的问题
2. JavaScript 文件的合并、缩减和混淆。通过合并可以把多个 JavaScript 文件合成一个，减少页面加载时的 HTTP 请求个数；通过缩减可以去掉 JavaScript 代码中多余的空白字符和注释等，从而减少文件大小，降低下载时间；通过混淆则是可以替换有意义的变量名称，从而进一步减少文件大小，同时在一定程度上保护代码免被反向工程。可以执行这些操作的工具有很多，Apache Ant 就可以完成合并，[JSMin](#) 和 [YUI Compressor](#) 可以完成文件的缩减，[Dojo Shrinksafe](#) 可以进行混淆。
3. CSS 文件的合并和缩减。与 JavaScript 类似，CSS 文件也可以执行同样的合并和缩减操作，从而减少 HTTP 请求数目和文件大小。YUI Compressor 可以完成 CSS 的缩减。
4. 图片文件的压缩。通过对图片文件进行格式转换和压缩，可以在不损失质量的前提下，减少图片文件的大小。

测试

Ajax 应用的测试包含服务器端和浏览器端两部分。对于服务器端来说，测试的技术和工具

都已经比较成熟。只需要根据服务器端采用的技术来进行选择即可。一个比较重要的原则是服务器端和浏览器端尽量实行松散耦合，以方便测试。从这个角度出发，服务器端返回数据，而不是 HTML 片段是更好的做法。可以通过工具来测试服务器端返回的结果是否正确。

浏览器端的测试目前情况不是非常理想。已经有一些单元测试的框架，如 [QUnit](#), Dojo [D.O.H](#) 等，也存在一些集成测试的工具，如 [DOH robot](#) 和 [Selenium](#) 等。就单元测试来说，目前对仅用 JavaScript 实现的纯逻辑代码较容易实现，而对于包含了与页面上节点交互的代码则较难实现。不管是单元测试还是集成测试，目前自动化程度都不是很高。

为了便于测试，Ajax 应用中各部分之间的耦合应尽可能的小。事件处理方法的方法体应尽可能的简单。

调试

Ajax 应用的调试一直是一个比较麻烦的问题，其主要原因是不同浏览器之间存在着各种各样的兼容性问题，同一浏览器的不同版本之间也会存在很多不同。为了在所支持的浏览器上达到一致的效果，开发人员往往费劲了周折。目前的情况要好了不少，不同的浏览器都有了自己比较好用的调试工具，如 Firefox 上的 Firebug，IE 上的 developer toolbar 等。当出现问题的时候，可以通过这些工具来直接修改页面上的 DOM 结构和 CSS 样式来进行试验。找到正确的解法之后再用代码来实现。很多工具都支持直接在控制台输入 JavaScript 语句来执行，通过这种方式可以快速的查看程序中变量的值以及调用 JavaScript 方法来改变应用的内部状态，从而发现问题的原因。

内存泄露

Web 应用内存泄露的问题一直存在，Ajax 应用的出现把这个问题进一步暴露出来。目前很多的 Ajax 应用都是单页面应用（[Single-page Application](#)，SPA）。用户通常会在单个页面上使用比较长的时间而不关闭浏览器。在用户操作过程中产生的一些小的内存泄露会累积起来，导致浏览器占用内存不断增加，应用运行起来越来越缓慢。

面对内存泄露问题，一般来说需要注意下面几点：

- 熟悉常见的内存泄露模式。最典型的是由于错误使用闭包造成的包含 DOM 节点的循环引用。打断循环引用就可以解决此问题。
- 很大一部分内存泄露与 DOM 节点相关。尽量不要为 DOM 节点对象添加额外的属性，尤其是 JavaScript 方法。

3. 当内存泄露发生的时候，使用 [Drip](#) 等工具来找到发生泄露的节点并修正。

安全

Ajax 的出现并没有解决存在的一些安全问题，同时也带来了一些新的安全隐患。传统 Web 应用中存在的跨站点脚本攻击(XSS)、SQL 注入和跨站点请求伪造(CSRF)等安全问题在 Ajax 应用中仍然需要解决。对于 XSS 来说，一般的解决办法是不信任用户的任何输入。输出的时候对所有的东西进行转义(escape)。只对那些明确知道是安全的(白名单)的东西恢复转义(unescape)。对于 CSRF 的解决办法是对所有的请求添加一个验证令牌，用来确保请求是来自于自己的站点。

Ajax 带来的新的安全隐患主要与 JSON 有关。一部分 Ajax 应用的服务器端暴露 JSON 格式的数据。JSONP 允许通过<script>标签来获取数据，而不受浏览器同源策略(Same -origin Policy)的影响。不过 JSONP 可能造成数据被恶意的第三方窃取。攻击者还可能通过重定义 JavaScript 对象方法(如 Array)的方式来窃取数据。

性能

Ajax 应用的性能是一个非常重要的方面，应该在应用开始开发的第一天就把性能这个因素考虑进来，并贯穿整个开发过程。如果在开发后期才考虑性能的话，就可能与陷入一个两难的境地。一方面应用的性能达不到用户的要求，造成用户的抱怨和流失；另一个方面为了提升性能就需要对应用已有的架构做出非常大甚至是颠覆性的调整。

Ajax 应用性能的决定因素在前端。简单来说有下面几条基本的原则：

1. 减少与服务器端交互的次数与数据大小。这点主要是减少浏览器端发出的 HTTP 请求的数目和降低服务器端返回的数据内容的大小。前面提到的 JavaScript 和 CSS 文件的合并和缩减都是服务于这个目的。
2. 页面的渐进式增强。在 Ajax 应用中，HTML 文档所包含的内容对用户是最重要的，而 CSS 则帮助用户方便的查看 HTML 文档。因此这两者是要被优先加载的。JavaScript 文件可以稍后加载或延迟加载。因此，在 HTML 文档中，对 CSS 文件的引用要放在文档的上面，即<head>元素中；而 JavaScript 文件的一般作为<body>元素的最后一个子节点出现。部分的 JavaScript 文件可以等到页面完全加载成功之后再延迟加载。

Google 的 Steve Souders 在前端性能这一领域做了很多开创性的工作。他写的两本书《[高性能网站](#)》和《[更快速网站](#)》都是非常好的总结性材料，值得深入研读。

个人简介

成富，目前任职于 IBM 中国开发中心，参与 IBM 产品的开发工作。对前端开发和 Dojo 框架有着比较丰富的经验。对新兴的 Web 2.0 技术也有比较浓厚的兴趣。他的个人网站是 <http://www.cheng-fu.com>。

参考资料

- [Ajax 应用风格](#)
- [Browser Compatibility Tables](#)
- [HTML 4.01 Specification](#)
- [富含语义的 HTML](#)
- [ECMAScript Language Specification 第五版](#)
- [面向对象的 CSS](#)
- [精通 CSS--高级 Web 标准解决方案](#)
- [Pro CSS and HTML Design Patterns](#)
- [Memory leak patterns in JavaScript](#)
- [Understanding and Solving Internet Explorer Leak Patterns](#)
- [XSS Cheat Sheet](#)
- [Ajax Security](#)
- [Unit testing Web 2.0 applications using the Dojo Objective Harness](#)
- [High Performance Web Sites](#)

原文链接：<http://www.infoq.com/cn/articles/ajax-guide>

相关内容：

- [FireAtlas : ASP.NET AJAX 查看器](#)
- [如何选择最合适的 Ajax 框架？](#)
- [应用 JSF、Ajax 和 Seam 开发 Portlets \(1/3 \)](#)
- [使用 JSF、Ajax 和 Seam 开发 Portlets \(2/3 \)](#)
- [使用 JSF、Ajax 和 Seam 开发 Portlets \(3/3 \)](#)

案例研究：Eclipse 富 Ajax 开发平台在 CAS Software AG 项目中的应用

作者 [Craig Wickesser](#) 译者 [连小剑](#)

本文案例研究的重点将会放在 Eclipse RAP (富 Ajax 平台) 以及它是如何应用到 CAS PIA 架构中的，同时也会涉及到 RAP 的一些有趣的应用、CAS 在使用过程中的一些经验教训以及未来他们的产品的发展方向。

简介

CAS Software AG 是来自德国 Karlsruhe 的一家软件公司，创建于 1986 年。该公司专注于 CRM (客户关系管理) 领域，尤其是在 SME (中小企业) 方面。近年来 CAS Software 的软件产品在 CRM 领域获得了非常好的声誉，包括他们在特定领域诸如面向教育、自动代理权和基于会员的组织或者协会等的 CRM 产品。

近来 CAS Software 正在开发被称为 CAS PIA(个人信息助理)的一个产品，这个产品会以 SaaS (软件即服务) 的方式发布，这也是他们首个以这种方式发布的产品。该产品使用了许多技术，我们会在以后作详细介绍。这篇案例研究的重点将会放在 Eclipse RAP (富 Ajax 平台) 以及它是如何应用到 CAS PIA 架构中的，同时也会涉及到 RAP 的一些有趣的应用、CAS 在使用过程中的一些经验教训以及未来他们的产品的发展方向。

范围

CAS Software AG 为自由职业者以及中小企业开发 CRM (客户关系管理) 软件。对 CAS 来说，他们把软件的受众定位在那些不使用特定 CRM 应用、而是更倾向于用工具和应用的组合来跟踪他们的客户、市场等的用户。CAS 认为对那些可集中而易于访问的客户信息管理软件，市场将会有很大的需求。在特定行业内，已经有基于桌面的应用，然而这些应用仅仅提供局限于 CRM 专有的功能。通过使用诸如 Eclipse RAP 和 Eclipse Equinox 之类的技术，CAS PIA 可以开发出不仅提供“标准”的 CRM 功能，而且还可以有管理市场战略、电子邮件整合以及个性化信件等功能的应用，而所有这些都被放入一个干净而模块化的软件包之内。

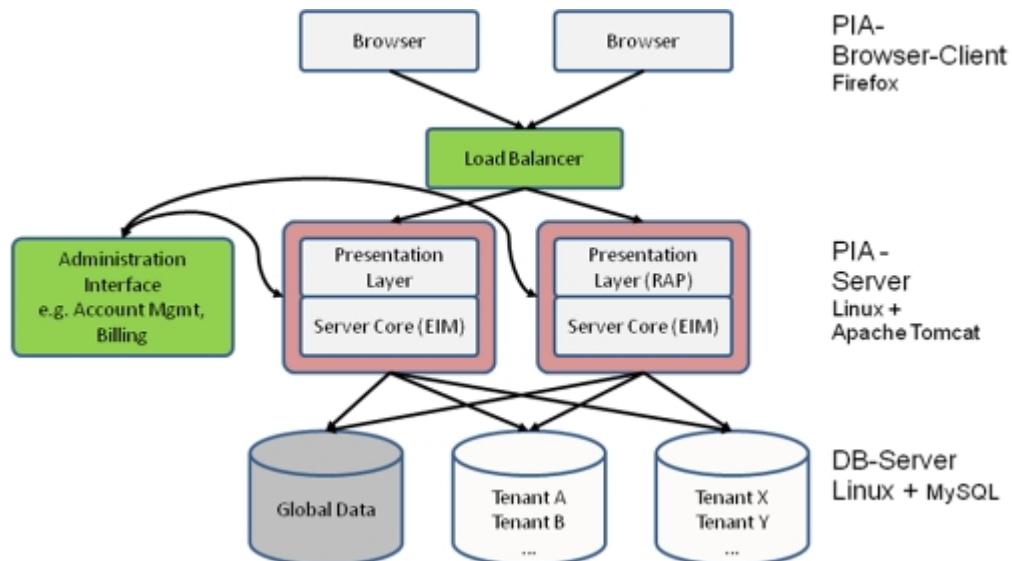
通过在用户界面开发中使用 Eclipse RAP，CAS PIA 提供了一个使用户感觉非常友好的桌面风

格的界面体验。用户界面根据人类工程学 (ergonomics) 来开发，使用了人们熟悉的窗体部件和行为，比如拖放等等用户对 web 应用程序所期望的很多东西。另外，在线功能使所有同一公司内的同事可以通过该应用来管理委派、任务和文档，也可以进行其他的重要工作流程比如地址转换和路径计划。在下一章中，我们将深入到 CAS PIA 的开发架构中来看看开源软件在当中如何起到至关重要的作用。

解决方案概述

CRM 解决方案已经出现好几年了，桌面的和在线的都有，在最近一段时间里功能变得非常强大。用户在使用它们时的期望远远超出了基本功能，诸如报表、安全、亲和的外观和体验以及反应迅速的用户界面。基于许多用户对 CRM 应用的期望和需求，CAS 选用开源软件作为他们整个解决方案的基础。

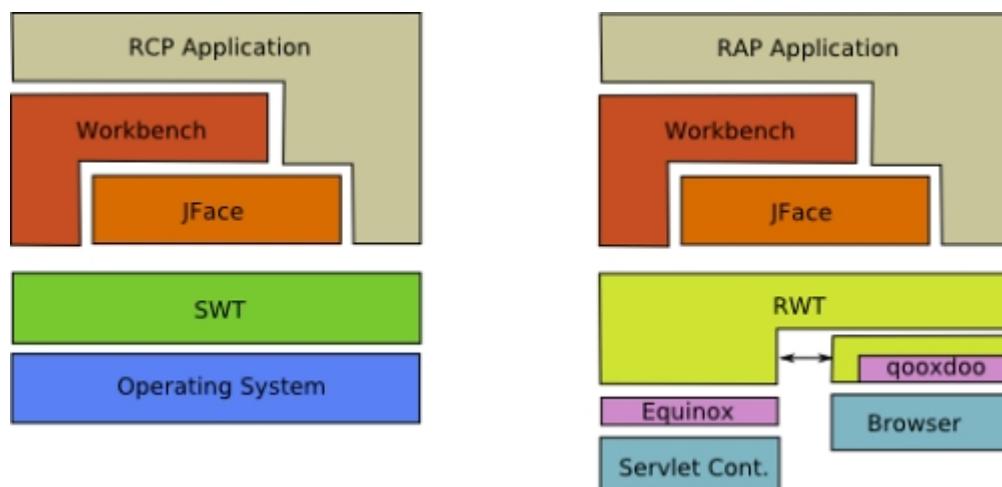
从设计角度来讲，CAS PIA 是一款易于扩展的 web 应用程序，它可以利用多个服务进程，而这些服务进程可以分担访问压力和提供冗余。每一个服务进程都运行在作为应用程序服务器的 Apache Tomcat 之上，包含了基于 RAP 的应用层和一个服务核心。应用层负责展示用户界面和处理用户请求，而服务核心则提供商业逻辑和数据库访问。



CAS 选用基于 web 的应用程序，从而使得用户可以不必担心安装、硬件成本和配置以及数据的安全。而且，基于 web 的应用可以使用户在任何地点来访问 CAS PIA，而不必在每台计算机上安装重客户端。作为 web 应用前端的表示层使用了 Eclipse RAP 来开发。RAP 项目主页把它描述为一个为开发人员提供了下述功能的框架：

“通过使用 Eclipse 开发模型、基于 Eclipse 工作台扩展点的插件以及用 SWT API (以及 JFace) 开发的组件工具箱来构建基于 Ajax 技术的富 web 应用 RAP 非常像 Eclipse RCP , 但是它不是在桌面计算机上启动 , 而是运行在一台服务器上 , 可以被标准浏览器访问。这主要是因为它提供了一套特殊实现的 SWT (一个 SWT API 的子集) 。
<http://www.eclipse.org/rap/about.php> ”

下面是 Eclipse RCP 和 Eclipse RAP 的一个简单架构对比图。



CAS PIA 用 Eclipse RAP 在表示层中构建用户界面有以下原因 :

- 用户感观 - Eclipse RAP 可以构建出非常符合人体工程学而且可切换主题的富用户界面 , 已经非常类似于胖客户端的感受。
- 开发效率 - AJAX 和 JavaScript 被包装成对开发者透明的组件 , 从而使得程序员可以用他们熟悉的 Java 类库和 IDE 来开发。
- 易于扩展 - 尽管 RAP 使程序员可以不直接使用 JavaScript 、 HTML 和 CSS 来开发 , 但是它也提供了足够的可扩展性 , 使定制的组件和风格可以毫无问题的加入到应用当中。
- 工程质量 - Eclipse 和它的产品族拥有最好的软件设计和体验 , RAP 也不例外。
- 单一代码库 - RAP 能够被编译为 AJAX 或者 RCP 应用程序。

表示层也包含了 OSGi 运行时环境 , 这为它在别的 CAS 产品中的使用提供了良好的模块性和复用性。 CAS 选择 [Eclipse Equinox](#) 项目作为他们实现 OSGi 的工具 , 定义如下 :

“...一个 OSGi R4 核心框架规范的实现，即一组实现了若干可选的 OSGi 服务和其它架构的软件包，可以运行在基于 OSGi 的系统上。

总的来说，Equinox 项目的目标是成为一流的 OSGi 社区和使 Eclipse 成为界面组件视觉化的开发工具。

通过利用 Equinox 提供的分离机制，CAS 已实现了自己的核心模块，这些模块包含了许多软件包，这些包可以作为通用组件应用到不同的应用程序中。每个模块都提供了一些扩展点，通过这些点，根据所开发的不同应用的需求，可以实现不同的特定的行为。例如，用户管理组件可以用在许多应用程序当中，而联系人管理模块就比较特殊，只会被用到 CRM 相关的应用中。OSGi 提供的这种扩展性使模块很容易被扩展，比如在构建和部署阶段。

CAS PIA 的另一部分是商业逻辑和典型的服务器端相关功能，即服务核心或者 EIM（企业信息管理）。设计和开发 EIM 是把它作为 CAS 整个产品线的核心。服务核心提供了通过 Sun JAX-WS、RMI 和 REST 服务来远程访问的功能，核心同样也利用 Spring 框架设计成组件化的风格，都是可以被扩展的。

持久层用了 MySQL 数据库，同时也包含了 CAS 特别开发的定制组件。定制组件包含了一个可扩展的数据模型、一套定制的查询语言（CAS-SQL）和一个权限管理组件。该权限管理组件利用 ACEGI 框架来进行用户鉴权，也对数据库层的每个对象都提供了 ACL（访问控制列表）。这套权限管理系统和 Oracle 的 OLS 比较类似，而该系统还支持 MySQL 之外的其它数据库，从而使 CAS 可以在别的产品上使用它。

RAP Eclipse RAP 的单元测试

单元测试在任何软件开发中都是非常重要的一个环节，即使是在软件的客户端也不例外。很多时候，应用程序的界面开发人员发现很难对代码做单元测试。通常，这是因为表示层和应用逻辑紧密耦合从而使得单元测试代码的开发非常复杂和难于维护。CAS 的程序员们设法把尽量多的逻辑都放在服务器端，从而使单元测试（代码）非常健壮。然而，不是所有的东西都可以放到服务器端，而这也是为什么好的 UI 设计成为一个很重要因素的原因。

通过用通用设计模式比如 MVC、表示层模型（Presentation Model）、模型视图代理（Model View Presenter）等等来实现用户界面，将视图从逻辑当中解耦出来，从而使单元测试变得相对容易。即便进行了良好的设计，也还是有很多的问题需要面对，尤其是 RAP 用户界面的测试。首先，RAP UI 组件包含了一个 Java 层和一个 JavaScript 层，这意味着有两个部分的代码需要测试。[Qooxdoo](#)，即 RAP 使用的 Ajax 应用程序框架，提供了类似于 JUnit 和 JUnit 的单元

测试工具。CAS 利用这些工具来对组件的 JavaScript 层进行测试，用 JUnit 来测试 Java 层。下面是 CAS 提供的一个单元测试代码，展示了对定制组件 JavaScript 层的测试。

```
/**  
 * Memory leak test.  
 * * Creates and disposes an objects, and checks if there are some leaking instances.  
 *  
 * @type member  
 * @return {void}  
 */  
  
testMemoryLeak : function() {  
  
    var ms1 = de.tests.MemoryLeakUtil.getMemorySnapshot();  
    // create  
    var dc = new de.cas.qx.ui.widget.calendar.datechooser.DateChooser();  
  
    qx.ui.core.Widget.flushGlobalQueues();  
  
    // dispose  
    dc.dispose();  
    var ms2 = de.tests.MemoryLeakUtil.getMemorySnapshot();  
  
    var msg = de.tests.MemoryLeakUtil.checkMemoryLeak(ms1, ms2);  
    this.assertEquals("", msg, "There are some leaking objects!");  
},
```

这个特定测试试图找到由 DateChooser 组件引起的潜在的内存泄露(DateChoose 是 CAS 开发的定制组件中的一个)。MemoryLeakUtil 类是一个 CAS 创建的定制工具类，它使用了 Qooxdoo 提供的一些功能，比如列出内存中的所有对象。通过 Qooxdoo 提供的功能，他们可以轻易的对内存泄露问题来做测试，而这类问题是在 JavaScript 组件开发中很常见的。在测试 RAP 用户界面时需要面对的另外一个问题是处理 UI 的异步和动态的状态。有一些工具可以用来记录用户界面并且可以把这个过程存储起来，从而可以反复运行。这种类型的测试有助于检查

UI 的行为和交互，仿佛是有用户在真正使用它，但是它们也有局限性。在 CAS 对 web 应用程序测试工具做的大致评估中，他们还没有发现一款工具可以处理异步和非基于页面的用户界面，如遇到基于 Ajax 的应用程序，它的内容是动态装载的而非改变整个页面。

单元测试中有时也会遇到的一个问题是服务器和/或数据层进行测试。通常单元测试代码直接与服务器、数据库等直接通信来完成它们的测试。这种类型的测试有其缺陷，CAS 的开发者们也遇到过，单元测试由于需要和其他层的通信而变得很慢。一个通常的解决办法是用假对象，在测试中用假对象来代替“真对象”。在 java 中有很多假对象的框架，包括 [Mockito](#)、[EasyMock](#) 和 [JMock](#)，它们用来简化假对象的创建过程。对 JavaScript 来说，也有这样的假对象框架比如 [JSMock](#) 和 [Mock4JS](#)。

之前您看到了用于测试 JavaScript 层的单元测试代码，之后您将会看到一段测试 Java 层的例子。对 Eclipse 1.1 来说，这个框架基本上包含了相当于 JUnit 的测试的功能，不同的是它可以使需要 OSGi 环境的测试正常运行。如果您需要执行期间更新 UI 的单元测试，您可以非常简单地从 `org.eclipse.rap.junit.RAPTestCase` 扩展。但是，如果对单元测试来说不需要更新用户界面，那么相应地，你可以扩展 JUnit 的 `org.junit.TestCase` 类。下面是一个关于包含了用户界面交互的 RAP 的测试用例：

```
public class RapJUnitTest extends RAPTestCase {  
  
    public void testOpenView() {  
        try {  
            IWorkbenchPage page = getPage();  
            page.showView( "org.eclipse.rap.demo.DemoTreeViewPartI" );  
        } catch( PartInitException e ) {  
            e.printStackTrace();  
        }  
        assertEquals( 1, getPage().getViewReferences().length );  
  
        getPage().hideView( getPage().getViewReferences()[ 0 ] );  
        assertEquals( 0, getPage().getViewReferences().length );  
    }  
  
    private IWorkbenchPage getPage() {
```

```

IWorkbench workbench = PlatformUI.getWorkbench();
IWorkbenchWindow window = workbench.getActiveWorkbenchWindow();
return window.getActivePage();
}
}
    
```

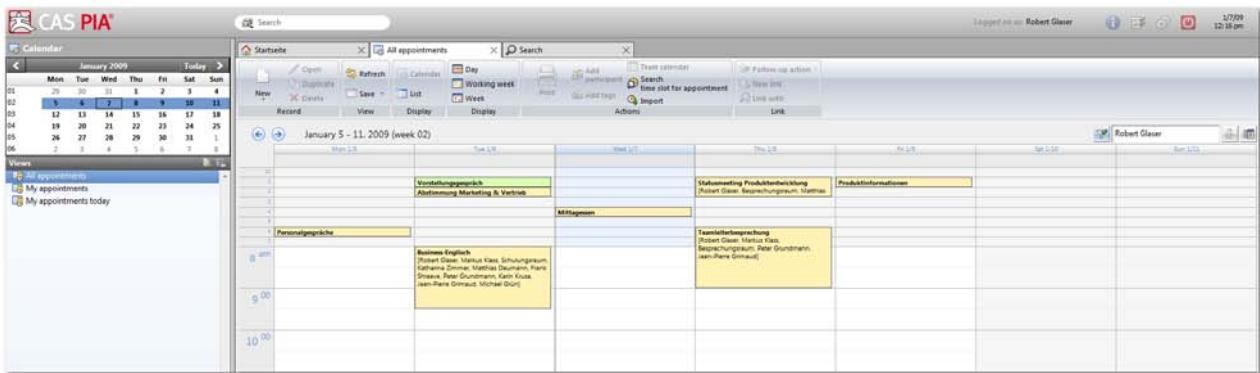
这个例子展示了如何测试一个实际的 UI 组件来验证可视视图的数量。而这也正展示了如何正确地测试基于 RAP 应用程序测试的一个步骤，然而，它并没有提供自动化用户驱动交互。通过模拟一个用户点击按钮或者在输入框中键入值这样的测试 UI 的能力将会是 CAS 在未来所研究的目标。

定制用户界面组件

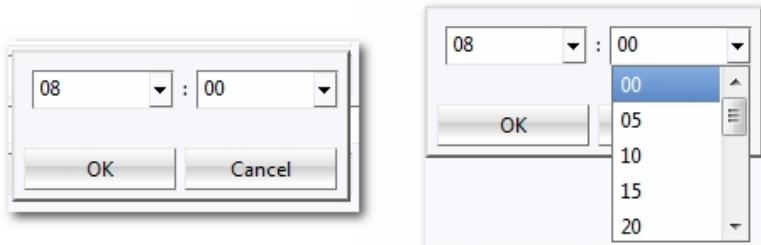
除此而外，RAP 还提供了还提供了用 SWT 构建的一个超大的组件子集，称为 RAP 控件工具箱，或者称为 RWT，这套工具箱可以满足许多应用程序的需要。然而，当有“标准”以外的需求时，Eclipse RAP 也支持开发和使用定制组件。开发定制组件的第一步是确定这是一种什么类型的控件，在 RAP 中，有两种控件，一种是“复合”式 ("compound") 另外一种是“自构建”式("owner drawn")。复合控件是把已有的 RAP 组件组合到一起来提供一种新的 UI 功能。而自构建式组件则源于 JavaScript，通常需要许多重型开发，有时则需要第三方库。RAP 开发向导提供了一个一步步教你如何创建“自构建”定制控件的教程，教程中用截屏和例子代码阐述了整个过程。对于定制组件的开发流程，教程中列出了四个主要的步骤：

- 为组件创建一个运行在服务器上的 Java 实现
- 为组件创建一个运行在浏览器上的 JavaScript 实现
- 用 Java 创建一个适配器，这个适配器把 JavaScript 组件和 Java 组件连接起来
- 通过在 org.eclipse.rap.ui.resources 扩展点上增加插件的方式注册该 JavaScript 文件

CAS 的开发人员必须开发一些自定义控件包括一个日历、日期选框、工具条和可折叠的导航控件，看起来像 Microsoft Outlook。这个日历和工具条就是用 JavaScript 加上 CSS 和 HTML 开发的“自构建”组件的例子，转化而成为 CAS PIA 所用的 RAP 组件。Qooxdoo 提供了很多功能来开发控件，这些功能可以大大的降低开发的难度。下图是展示了一组“自构建”组件的截屏，特别是一个日历和一个工具条。



可以看到日历组件提供了很多功能比如左上角的一个迷你日历，中间的一个比较详细的日历，可以加入任务，而且可以定制视图来显示（例如，“所有的任务”）。这个特殊部件由大约 20000 行代码构成，实现它需要大量时间和精力。上图中所示的另一个“自构建”组件是一个工具条，它提供的功能类似于很多应用例如 Microsoft Office 和 Microsoft Outlook 的工具条。组合组件的例子是一个时间选择器，我们可以在 CAS PIA 中看到它，如下图所示。



这个复合组件由一组控件包括一个对话框，若干按钮以及可选框构成，他们一同构建了这个时间选择器。在 CAS PIA 的拷屏图中另外值得注意的一点是应用的整体样式或者说主题。RAP 通过使用层叠式样式表 (CSS) 提供了主题功能，同时使应用可以接受加在 org.eclipse.rap.ui.themes 扩展点和 plugin.xml file 扩展点上的扩展。

开发定制的 RAP 组件时，在设计和开发阶段必须对下面几点加以考虑。首先，开发人员必须熟悉 HTML、JavaScript、CSS 和 Qooxdoo。我提到这个是因为 RAP 的一个优越之处就是开发人员可以用 Java 来编程而可以避开 JavaScript，但是在开发定制组件时却不是这样。其次，开发人员必须设法使控件具有跨浏览器特性。写过 RAP 核心组件的开发人员竭尽全力来确保控件的浏览器兼容性，而在未来版本的 Qooxdoo 中有望在这方面得以增强从而使开发人员可以免于考虑浏览器兼容性问题。最后，另外一个需要注意的问题是在 RAP 中的 Qooxdoo 和你从网站上下载的不是同一个版本。极端情况下，对 RAP 的开发人员来说，这意味着一些特定功能和类可能不能使用，尽管它们出现在 Qooxdoo 的 API 中。

开发中遇到的问题

对开发人员来说任何新技术都一定要有一个学习曲线，发展过程中也会有很多问题。在 CAS PIA 的开发过程中开发团队陷入了一些与性能和部署相关的问题当中。

他们马上遇到的第一个问题就是客户端和服务器端都出现较低的性能和高的资源开销。CAS 利用可以复用 GUI 控件而不是反复创建它们的对象池和缓存来提升一些性能。尽管 CAS 所做的努力有所帮助，但是对于完全解决他们在 Internet Explorer 上的性能问题还远远不够，这也使得 CAS PIA 不能支持 IE。然而 CAS 对短期内 Qooxdoo 框架的性能提升和 Internet Explorer8 的即将发布对 CAS PIA 表现出可接受的性能和对 IE 的支持相当有信心。另一方面，在别的浏览器上特别是 Firefox，都已经在近期取得性能和资源开销问题方面的提升。

另外一个令人头痛的问题是用一种连续累计的构建过程开发和部署整个应用程序的 RAP 组件。由于 CAS 选择将 Eclipse Equinox 部署到 Tomcat 中而不是将 web 服务器嵌入到 Equinox，所以这是唯一的问题。CAS 使用了 Eclipse 中的 Releng- Tools，这个工具可以支持夜间自动构造，然而它们在使用 Ant 的过程中发现文档太少而且有很多奇怪的问题（比如，动态生成构建脚本）。最终确保夜间构建的正常运行花费了大量的监控和测试。

与此同时 CAS 不得不解决将 Equinox 部署到 Tomcat 中的问题。对这个配置问题提出的解决方案是，生成一个单独的 WAR 文件，这个文件中包含了所有的商业组件、运行时环境、Equinox 和 RAP。但是，CAS PIA 也需要 EIM 这么一个非 OSGi 组件和 RAP 部分一起集成到应用程序当中，从而使得两个部件之间可以不通过 web 服务和 RMI 直接通信。解决这个问题需要分两步，首先必须把 EIM 服务器组件放到 WAR 文件的“lib”目录中。第二步涉及到对 web.xml 中 [servlet bridge](#) 的特殊配置。CAS 用到了 servlet 桥中的“[extendedFrameworkExports](#)”参数，这个参数能使 EIM 和 RAP 组件根据需要集成在一起。

经验教训

CAS Software 积极致力于开发和部署 Eclipse RAP 应用程序 即便在开始的时候碰到很多问题，他们仍然非常乐于使用 Eclipse RAP 来开发产品。他们发现用 Eclipse RAP 后开发人员的效率得到显著提高，这主要是因为 Java 程序员已经习惯于 Eclipse 的集成开发环境、调试工具和组件模型。

一旦 CAS 克服了以上提到过的问题而走过了陡峭的学习曲线，每个人都会乐于使用 RAP 来做开发。CAS 希望 RAP 在未来产品中增加的唯一功能是从服务器端来触发客户端动作的能力。目前 CAS PIA 使用自主开发的一套方案，其实实际上称不上真正的解决方案，但 CAS 对

RAP 的未来充满信心。

未来发展方向

CAS Software AG 目前在中小企业 CRM 市场上居于领导地位，他们计划到 2010 年时把这种领导地位拓展到整个欧洲。CAS PIA 在 2009 年第一季度将会面试并且推广。最终目标是成为 SaaS CRM 产品欧洲地区供应商的前 20 位。

CAS 会继续在使用和支持 Eclipse RAP 上发挥他们重要的作用，他们还将于 2009 年 2 月在匈牙利的赛格德大学开办有关 RAP 的课程。

参考链接

- [CAS PIA](#)
- [Eclipse RAP Book](#) (2008 年 12 月)
- <http://rapblog.innoopract.com/2007/12/rap-deployment-part-2-deploying-your.html>
- http://www.eclipse.org/equinox/server/http_in_container.php
- GUI 测试工具
 - [QF-Test](#)
 - [Squish for Java](#)
 - [QA Wizard Pro](#)

原文链接：<http://www.infoq.com/cn/articles/eclipse-rap-casestudy>

相关内容：

- [Ajax、Comet、HTML 5 Web Sockets 技术比较分析](#)
- [Eclipse RAP 1.0 给 Ajax 带来了 RCP 和 OSGi](#)
- [Grails Ajax 富客户端插件大比拼](#)
- [选择你所需要的 Ajax 框架](#)

使用 Flash Builder 4 beta 进行数据为中心的开发

作者 [Tim Buntel](#) 译者 [曹如进](#)

Adobe Flash Builder 4 的优势在于它的跨平台和跨浏览器特性，它允许程序在所有的操作系统和所有的浏览器上以同样的方式运行。它为开发者们提供了更多的机会来创建以数据为中心的富互联网应用。这篇文章用创建一个简单数据管理应用的主要步骤向大家表现了一个使用 Flash Builder 4 beta 进行以数据为中心的开发。

Adobe Flash Builder 4 beta 为 Flex 开发者们，不管是新手还是老手，提供了更多的机会来创建以数据为中心的富互联网应用。这样一个由专业工具，一个开源框架以及无处不在的客户端所组成的 Flash 平台，让你能够发布出令人瞠目的表现内容和应用程序。

尽管如此，大多数的应用还依赖于平台之外的服务。也许你的应用程序为企业数据库中存储的信息提供了报表以及数据可视化的功能，抑或你的富电子商务应用程序需要与现有的订单管理系统，或者第三方的支付服务进行集成。那么你只需要连接到相应的服务器或者服务，这种应用特性就可以使得上述一切像发送电子邮件般的简单，例如通过使用云托管服务和第三方的 API 查询数据库等等。

在以往的版本中，开发者必须学会各种技巧手工编写连接服务器和服务的代码。例如，需要知道连接一个 SOAP 服务使用的 MXML 标签不同于连接 ColdFusion 组件或者 PHP 类。此外，你通常还得编写一些对于 web 开发者来说很少见，很困惑的代码，如事件监听和故障处理。

在 Flash Builder 4 beta 中，Adobe 改变了这一切，它采用了一种全新的方式创建以数据为中心的应用程序。新环境下的 Flex 开发者可以快速的连接到数据和服务，并将它们绑定到富 UI 控件上。这些创建面向数据的高级应用的新方法使得经验丰富的开发者们受益匪浅。

使用 Flash Builder 4 beta 进行以数据为中心的开发主要包括三个步骤：

- 定义一个数据/服务模型
- 将服务操作绑定到 Flex 组件上
- 实现高级的数据处理，例如分页和数据管理

在这篇文章中，你将会经历创建一个简单数据管理应用的主要步骤。在这个场景中，你有一张 Oracle 数据库中的表，并想要创建一个 Flex 应用来允许用户查看它的数据以及新增，更

新和删除记录。

要求

为了更好的理解这篇文章，你需要使用下面的软件和文件：

Flash Builder 4 beta

- [下载 \(InfoQ 中文站独家高速提供 \)](#)
- [了解更多](#)

基础知识

之前有过 Flex Builder 的使用经验会很有帮助，但这不是必要的。你需要熟悉一种服务端技术例如 ColdFusion，Java 或者 PHP。

第一步：创建一个服务

由于在 Adobe Flash Player 中运行的应用程序不能直接与 Oracle 数据库交互，因此你需要利用一个服务来完成该任务：它可以接受来自 Flex 发来的请求并传递到数据库中；还可以将数据库中的数据用一种可理解的格式发回给 Flex。这样的远程服务有着相当多的实现方式，在 Flash Builder 4 beta 中已经内嵌支持了使用 ColdFusion，PHP 和 Java 创建服务，而其他类型的服务可以像 SOAP web 服务或者 HTTP 服务一样使用。使用 ColdFusion 是个理想的选择，因为它可以和任何后端数据库交互，且它语法的简单性使得你仅仅需要少量的 标签即可完成创建一个数据接入服务。加之 ColdFusion 支持一个高性能的名为 AMF 的协议与 Flex 应用程序进行数据交换。另外 PHP 和 Java 也同样支持 AMF，因此你大可以随心所欲的使用自己最熟悉和认为最高效的服务端技术。

使用 ColdFusion，你需要为 Flex 应用程序执行的每一个数据操作创建一个 ColdFusion 组件 (CFC) 函数，如：获取一张表的所有记录，向表中增加一条新的记录，以及删除一条记录等等。CFC 中的函数可以返回弱类型和强类型的数据（例如，如果你正在采用一种更加面向对象的方式进行开发的话，可以使用 getAllRecords 函数返回一个 ColdFusion 的查询对象或者一个对象数组）；Flex 两种类型的数据都能处理。最后一定要确保 cffunction 标签的 access 属性标记为 remote 后再测试组件。好了，到此为止你已经可以在 Flex 应用程序中使用这个服务了。

第二步：在 Flash Builder 中创建模型

在 Flash Builder 4 beta 中，新的数据/服务面板位于中心位置，主要是用来管理和交互你的应用程序中使用到的所有服务器和服务。它采用一个树状视图来表示所有服务中可用的数据和操作。视图中呈现的数据和服务可以来源于不同的地方。例如，其中一个可能是 ColdFusion 组件或者 PHP 类，另外一个可能是云托管的第三方 RESTful 服务。尽管如此，你不必担心它们在服务端如何实现，因为现在将结果绑定到 UI 组件、编写代码来调用操作都能统一到一个方法中。

为了让服务得以使用，Flash Builder 4 beta 会自动检查内部服务并创建树状视图。在 Flash Builder 4 beta 中选择数据->连接到 ColdFusion（或者你的服务使用到的技术）。对 ColdFusion 而言，你只需要简单地提供一个想使用的服务名称（例如，EmployeeSvc），并将它定向到文件系统中的 CFC 即可。这一步骤会依服务使用技术的不同而略有变化（例如，你也许会为 web 服务指定 WSDL），但是结果一定是一样的：Flash Builder 4 beta 通过在内部检查服务来发现返回的操作和数据类型，继而在数据/服务面板上创建服务的树状视图。

如果有必要的话，你还可以继续向服务树状视图中加入其它服务，或者也可以马上就在应用程序中使用已有的服务。如果服务是弱数据类型，那么需要一个额外的步骤。因为一个弱类型的服务仅返回数据，而没有关于数据所代表含义的信息。

比方说，你的 CFC 函数返回了一个 ColdFusion 查询对象，而 Flash Builder 4 beta 看到的只是一堆记录，它并不知道这些记录代表的是产品集合还是员工集合或是销售订单的集合；这仅仅是一堆数据而已。为了关联操作结果的数据类型，Flash Builder 4 beta 允许你手工配置操作返回的数据类型。当然，如果你在服务端使用强类型的数据类型，这步是可以略去的。

想要设置弱类型服务的返回类型，你可以右键点击数据/服务面板（例如，getAllItems 操作），然后选择配置返回类型。向导会帮助你建立服务端弱类型数据与 Flex 应用程序中的强类型的映射关系。它通过给出一个真实的操作样例数据来让你决定选择什么样的类型。过程中你需要为操作返回的自定义类型指定一个名称，例如可以把返回的每一条记录称为 Employee 或者 SalesOrder，还可以指定数据类型中的字段和格式——如将 name 的类型设置为 string，员工的 id 设置为数字（见图 1）。

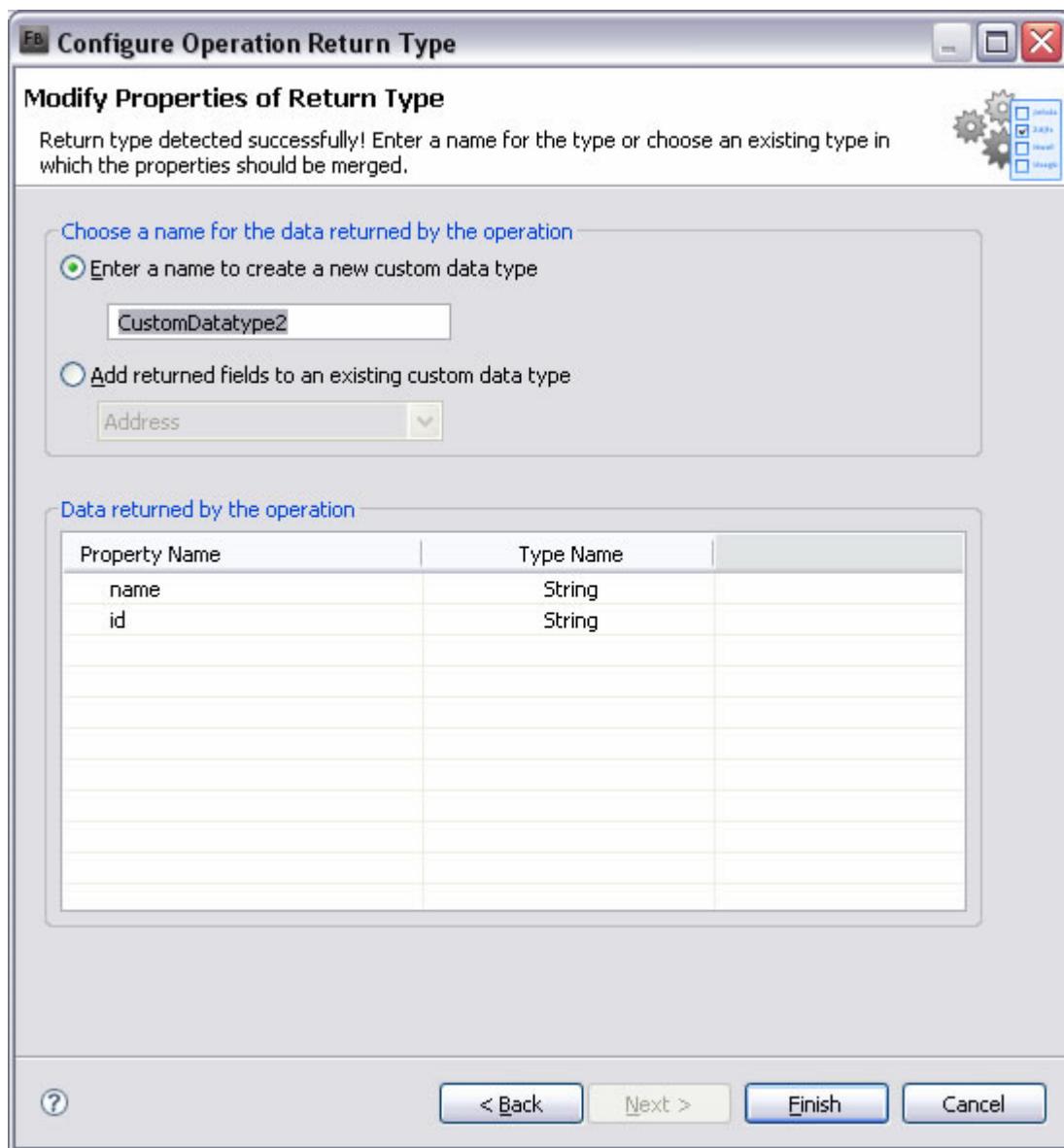


图 1. 配置操作返回类型

第三步：将服务连接到 UI 控件

既然你已经定义好了服务中所有的操作以及返回的数据类型，那么现在需要做的就是在应用程序中的某个地方显示那些操作的结果。Flex 框架中包含了大量的控件用以数据绑定，包括数据网格 (data grids)，列表控件 (list boxes)，表单域 (form fields) 等等。这些组件可以显示数据并允许用户与你的服务进行交互。

一开始就在设计视图中对 UI 进行布局，以及绑定操作到组件上会很简单。只要切换编辑器从源代码视图到设计视图，你就可以从组件面板中拖动组件到应用程序的画布 (canvas) 上并进行精确定位。

选择 DataGrid 组件（在组件面板里数据控件组的下面），将它拖放到页面中。你会发现它没有绑定到任何数据；如果运行程序，会发现它仅仅是一个三列的空网格。为了能够让网格显示从你的服务操作中获取的数据，你只需要简单地将数据/服务面板中的操作拖拽到网格上即可。结束之后你会发现，网格将会显示从操作返回的列。这时，保存项目，运行，就得到了一个正在使用你的 ColdFusion 服务填充网格的应用程序。这一切都无需编写任何代码，无需事先任何事件监听器，无需知道服务端是 ColdFusion 还是 Java 或者 SOAP。你还可以用很多其他方法来快速创建基于数据类型和服务的应用程序 UI。如可以从一个数据类型生成表单并且创建主从表，可以将一个服务拖拽到按钮组件中，然后每当用户点击这个按钮，就会触发操作的执行（例如，调用保存操作），还可以将操作拖拽到图表控件上等等。

数据和服务特性并不是仅仅在设计视图中有用。通过使用服务模型生成的子类，你可以获得关于所有操作和数据类型，甚至值对象的自定义行为的代码提示。

高级数据特性

Flash Builder 4 beta 以数据为中心的新特色功能，可以极大地提高你在创建以数据为中心的应用程序时的生产力。虽然在 Flex Builder 3 中也可以创建同样的应用，但是要花费更多的精力。这种新的高级数据特性，已经超越了生产力；它们能够让你实现在以前看来极度困难或是不可能的功能。比如，客户端数据管理特性可以让你将客户端的常见数据服务操作（选择，创建，更新和删除记录）与服务端相应的数据操作进行映射。这将使得你能够批量处理操作，而撤销功能可以使用户重做一些改变等等。另外一个强大的特性是支持自动分页。如果你要显示大量的记录，那么在应用程序一次性读取和加载它们的时候，会有性能问题。而分页会自动地每次按需取出一小部分的记录；你需要做的只是提供一个能够接受某行开始以及所需读取的记录数为参数的服务，而 Flash Builder 4 beta 负责实现客户端的所有逻辑。

下一步怎么做

不管你是一名经验丰富的 Flex 开发者还是刚刚接触这个技术的新手，Flash Builder 4 beta 都能够让你充分利用已有的服务端数据和服务逻辑知识，轻松的创建富应用开发体验。[下载好软件](#)后，今天就可以开始让你的用户看到数据新的呈现方式。同时也别忘了看看 [Adobe 实验室的视频和教程](#)哦。

关于作者

Tim Buntel 是 Flash Builder（以前叫做 Flex Builder）的高级产品经理。在 2007 年加入 Flex 小

组之前，他曾担任多年的 Adobe ColdFusion 高级产品经理。

原文链接：<http://www.infoq.com/cn/articles/flex-datacentric-development>

相关内容：

- [用Flash Builder 4 beta建立连接BlazeDS远程端的Flex应用程序](#)
- [Flex 4 的十大变化](#)
- [Flash Builder 4 beta中五个重要的新特性](#)
- [富Office客户端应用](#)
- [用Flex Builder构建即时聊天应用](#)

Flex 技术在企业级开发中的应用

作者 [池建强](#)

2009 年，对于技术开发者来说，是一个互联网技术风起云涌的年代，互联网技术纽带般的把各种 IT 服务相关的技术应用和实现，交织到一起，形成一个庞大的互联网生态圈。作者根据自己十几年的技术从业经历从三个方面简单介绍了企业应用系统的互联网化趋势，并着重介绍了 RIA 技术如何在互联网和企业应用中发挥出它的作用。

从我个人的从业经历来看，在长达十几年的软件研发过程中，无论是研发的产品或实施的项目，大部分是在为企业客户提供服务。当然，期间我还从事过两年左右的互联网应用的开发。早期的互联网应用开发和企业级应用开发的区别还是很明显的，无论是技术、架构、业务和用户体验，都有很大的不同。举个简单的例子，比如开发语言，最早在 2000 年左右，大家都用 Perl 和 Asp 做网站，后续陆续开始使用 PHP、Ruby 和 Python 这样的动态语言来构建丰富多彩的互联网应用，当然这其中也少不了 Flex 技术。而为企业客户构建的应用，则更多地倾向于静态语言，比如 Java 和 C# 等。当然随着技术的发展，这两者之间的交集越来越多，大家会越来越多的发现，很多大规模的互联网站点是基于 Java 或 C# 构建的，也有一些企业应用开始使用动态语言。这一点也很明显地展示了企业级应用与互联网的融合。

什么是企业级应用？

说了这么多，需要为企业级应用系统做一个定位。事实上这个概念在业界并不是十分清晰，没有一个明确的定义，什么是企业级，为什么叫企业级呢？有的观点是从系统规模上划分，有的是从团队规模上划分，有的是从开发周期上划分。我个人对企业级应用系统的定义比较简单，主要是用来区分互联网应用和个人软件。什么是互联网应用呢，四大门户（如新浪、网易等）百度和淘宝、各种 SNS 网站、博客系统和微博系统等等；个人软件呢，就是指安装在个人 PC 上的客户端软件，例如编辑器、绘图软件、开发工具等。这两种类型的应用和软件受众都是普通大众，而企业级应用系统的受众是企业客户，是为企业服务的，企业级应用系统的使用者是企业内外部客户以及与企业业务关联的人员。

2009 年，在技术层面可以说是一个风起云涌的年份，互联网像一条巨大而充满吸力的纽带，把各种 IT 服务相关的技术、应用和实现都吸引过来，形成了一个完整而庞大的互联网生态

圈。那身处其中的我们认识到了什么呢？随着我们持续的通过技术、平台、产品和项目为企业客户提供服务，我们发现企业应用不再局限在 Intranet 内部，企业应用系统的互联网化趋势越来越明显，主要体现在以下三个方面：

1. **Intranet 到 Internet 的转变**：企业应用系统由局域网转到互联网，企业应用开始要求多浏览器支持，国际化的支持，全球业务的互联互通。同时企业应用不再满足简单的表单和表格界面，富互联网应用（RIA）的需求应运而生，企业客户越来越倡导用户体验，RIA 也是我们后续要重点讨论的话题。
2. **企业应用的内容转变**：除了企业的核心业务系统，这样一些需求渐渐浮出水面：交互性门户系统、电子商务平台、企业级 2.0（博客、Wiki、RSS 和微博等）、企业级 SNS（社区平台）和无线企业应用等。
3. **需求的转变**：除了功能需求，客户对于安全、性能、大容量和大并发等特性愈发关注，在可预见的未来，企业应用一定是构建在互联网而非局域网，可能是在云端，也可能在其他的新技术上实现

作为现阶段的 IT 服务提供商，必须从技术层面和业务层面去适应和支持这样的趋势变化，否则我们会变得步履艰难。

好了，谈了这么多，主要讲了一个趋势的变化。下面我们来看一下在互联网和企业应用中都能发挥巨大作用的 RIA 技术。

RIA 简介和选择 Flex 的原因

RIA 技术的全称是富互联网应用（Rich Internet Application），RIA 首先应该是一个网络应用程序，其次它还要具有桌面应用程序的特征和功能。可以这样理解，如果你的桌面程序能在网络上（目前主要是基于浏览器）运行，并且能保持其原来的功能和特征，那么我们就可以称它们为 RIA 应用（富互联网客户端应用）。

目前 RIA 的主流技术主要包括 Adobe 公司的 Flex、微软公司的 Silverlight 和 Java 阵营的 JavaFX。Flash 由于 Flex SDK 的支撑，很早就从单纯的动画展示转入 RIA 领域，而且由于 Flash 的普及，Flex 目前应该是三大技术体系中市场份额最大、应用最广泛的技术；Silverlight 是微软推出的跨浏览器和跨平台的插件，能在微软的.NET 上交付炫目的多媒体体验和有丰富交互功能的 Web 应用，已经对 Flex 有了很大的冲击；JavaFX 是未被收购前的 Sun 公司在 2007 年推出的用来对抗 Flex 和 Silverlight 的桌面应用，但由于起步较晚，目前应用并不广泛，但其 Java 的原生性和开源性质对 Java 社区的开发人员还是有很大的吸引力。

基于以上三种技术，我们最终选了 Flex 做企业级的富客户端应用开发，虽然苹果公司的 CEO 乔布斯老师已经开始公开表示不在苹果的移动设备上支持 Flash，尽管 HTML5 和 CSS3 来势汹汹，但是在企业应用开发这样一个不是非常激进的领域，考虑到 Flash 广泛的群众基础，最终我们还是选择了 Flex。

事实上在互联网应用中，RIA 技术早已散发出夺目的光辉和迷人的魅力，无论是电子商务中的产品展示，还是 SNS 网站上的交友游戏，亦或是游戏和教育领域里的交互性设计，已经为广大互联网用户带来了无以伦比的客户体验。那么在企业应用系统中，企业客户还在满足于呆板的树形结构、简单的表格和文字性质的描述吗？就我们的经验来说，2008 年开始，企业客户就开始向我们提出这样的需求了，例如操作复杂的表单、图形化内容展示、动态报表绘图、图形化流程配置、流媒体视频播放和文档播放等，这一切都是在浏览器上进行的。对于大部分这样的需求，我们都是笨手笨脚的使用了 Javascript、Extjs、Jquery 和 Activex 等前端技术勉强实现了，对于不能实现的需求，我们只能腼腆的告诉客户，这些功能我们还实现不了，或者说浏览器不应该有这样的操作等等，当然这种话事实上也很难说服我们自己。

直到我们决定采用 Flex 技术来实现富客户端操作之后，我们才发现很多问题在 Flex 面前迎刃而解了。在 Full-Stack 系统中，如果 Ajax 技术和 Flex 技术配合形成前端组件体系，将大大提高开发效率、系统性能和改进客户体验。

Flex 是 Adobe 公司开发的可以输出成基于 Flash Player 来运行的互联网应用程序。Flex 基于标准的语言，与各种可扩展用户界面及数据访问组件结合起来，使开发人员能够快速构建具有丰富数据演示、强大客户端逻辑和集成多媒体的应用程序。Flex 目前最新版本是 4，一个 Flex 应用程序应该有两种语言代码完成，那就是 ActionScript 和 MXML。ActionScript 是一种面向对象的脚本语言，MXML 则是一种标记语言，非常类似于大家所熟悉的超文本标记语言(HTML)，扩展标记语言(XML)。简单来说 MXML 用来描述界面，ActionScript 用来处理业务逻辑。

以下是 Flex 的一些基本特点，也是我们采用 Flex 的重要原因之一：

1. 可可视化开发，通过拖拽方式开发界面
2. 对于有 XML 和脚本开发经验的人员，很容易上手
3. 可实现表现层与后台的真正分离
4. 丰富的媒体支持和动画效果，良好的用户体验
5. 支持多种通讯方式和数据格式

6. 同时支持客户端和浏览器模式
7. 跨平台，支持各种操作系统和浏览器

基于 Flex 构建企业级应用开发平台

与普通开发者使用 Flex 技术不同的是，我们采用了一种组件化的方式引入 Flex，这是因为我们对这部分技术的引入并不是从零开始，为了应对企业级应用开发的需求，我们很早就构建了一个企业级应用开发平台——GAP (Global Application Platform) 平台，这是一个 Full-Stack 的应用开发平台，除了底层框架、组织权限、工作流引擎、数据字典等等，还包括界面框架、通用 Web 控件，Ajax 控件等，Flex 的引入是对现在平台的补充和完善。

基于以上考虑，我们对 Flex 的应用分为三个阶段。

- 首先，进行 Flex 与 GAP 平台的整合，包括组件化集成、前后台通信机制的设计，在这一阶段我们主要采用了 ant 和 xdoclet 技术进行组件的打包、资源文件的合并，采用 Spring BlazeDS Integration 技术与 GAP 平台框架进行交互访问，通信方式采用了 Flex 提供的 RemoteObject。（ Spring BlazeDS Integration 是 Adobe 与 Spring 共同联合开发一个开源项目，其目标是开发者可以利用 Spring 开发模型通过 Flex、BlazeDS、Spring 以及 Java 技术创建 RIA ）。
- 其次，构建 Flex 图形组件框架，对 Flex 提供的控件进行封装、扩展，形成针对企业应用的个性化 RIA 控件库。在这一阶段我们主要是基于 Adobe 的开源项目 Cairngorm 进行构建的。Cairngorm 是一个基于 Flex 技术的微内核的 MVC 框架，设计简洁而易于扩展，非常适合构建自己的 RIA 控件库。
- 第三，使用 Flex 技术解决企业应用中的实际问题。

下面我们主要从实际应用的角度来看一下 Flex 在企业级系统开发中能够做什么。

一、企业组织结构的图形化展示

凡是为企业开发过系统的人都知道，企业的组织结构管理和权限管理几乎是每个项目或产品不可或缺的基础组件之一。从功能角度分析，GAP 平台的组织权限系统已经非常完善了，无论是多关系的组织结构、细粒度的权限控制，都可以非常好地满足客户对于组织管理和安全的需求。直到有一天一个客户提出，能不能把那棵呆板的组织机构树变成组织结构图，如果能支持图形化操作就更好了。听到项目组给我们反馈的这个需求，第一个反应就是拒绝，因

为实现起来太麻烦了。

原有的组织结构树如下图所示：



The screenshot displays two windows of an organization structure management system.

Left Window (Organization Structure Management):

- Group Company
 - Ruiwei Technology
 - Research Department
 - tina
 - Sales Department
 - + Sales Department
 - Human Resources Department
 - tina
 - Administrative Department
 - + Administrative Department
 - Quality Department
 - + Quality Department
 - Testing Department
 - + Testing Department
 - Customer Service Department
 - + Customer Service Department
 - Logistics Department
 - + Logistics Department
 - Technical Support Department
 - + Technical Support Department
 - Implementation Department
 - + Implementation Department

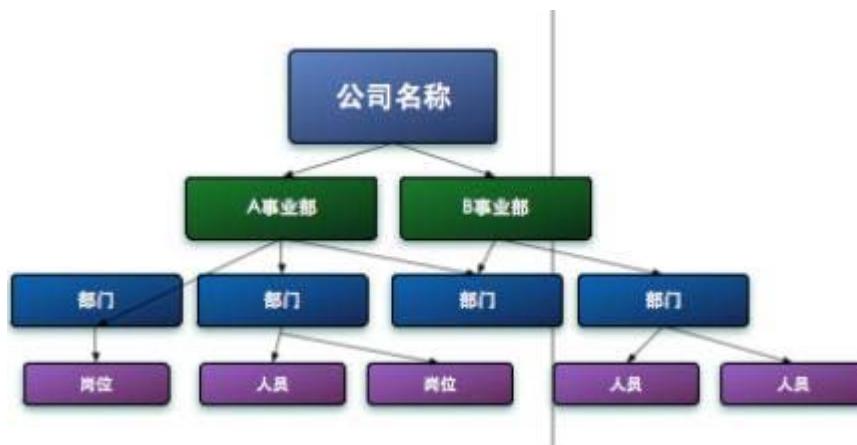
Right Window (Detailed View):

Buttons: 新增 (Add), 编辑 (Edit), 删除 (Delete), 排序 (Sort), 调级 (Promote/Demote)

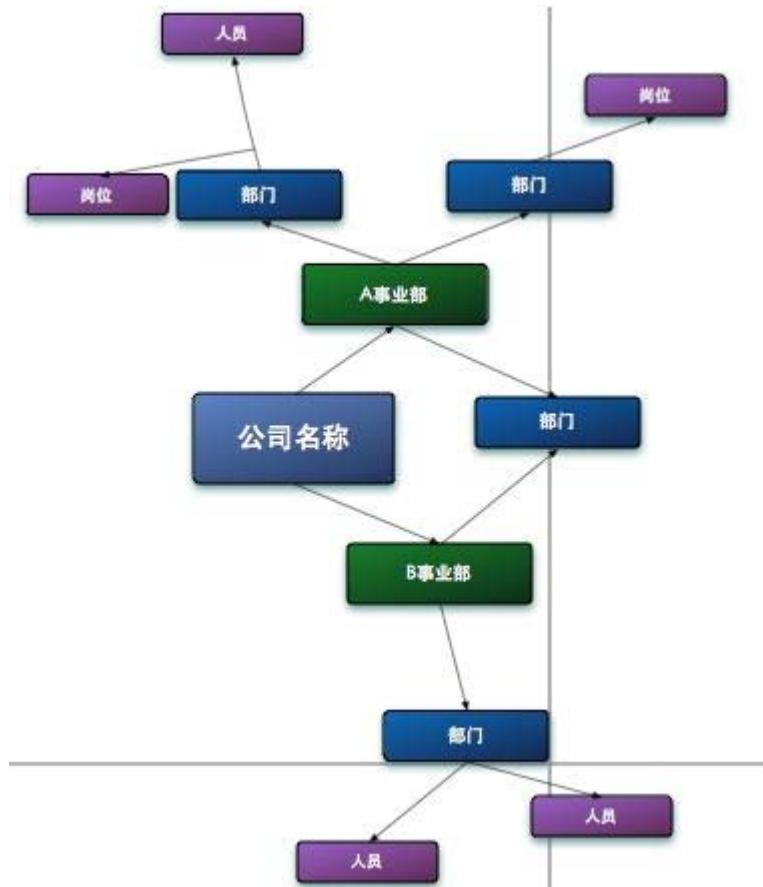
Form Fields:

- 部门名称: 人事部
- 编 号: 1079100700000000023
- 部门标识:
- 部门类型:
- 部门级别:
- 负责人: 花椒
- 备注:
- 创建时间: 2009-08-26 10:47:59
- 修改时间:

客户需要的展示方式可能是：



也可能是：



同时企业客户还希望能够进行图形化操作管理，这种功能如果通过 Javascript 来实现无疑是困难的，而 Flex 技术在处理类似的功能时则具备先天的优势。首先 Flex 是一种可以运行在网络上的客户端技术，它提供了一套成熟的图形化控件和类库，可以很容易的实现图形和布局控制。同时，Flex 可以通过多种通讯方式（HttpService、RemoteObject、WebService）与 Server 端的服务进行数据交互，使得图形化操作变得非常简单，例如把人员拖到另一部门，双击显示该机构的详细信息等。最终我们也是通过 Flex 技术实现了客户的需求。

二、表单操作

Flex 同样可以构造出复杂的表单功能，操作便捷，响应迅速，适应企业不同场景的需求。例如这样一个基于 Flex 技术的表格，看似简单，实际上是包含了排序、过滤、表头拖拽、表头固定、合并等功能，类似的功能如果用 Ajax 的方式来实现代码量会很大，但是在 Flex 中，这些特性基本上是原生的，或经过简单开发即可实现，代码量非常小，而且性能远远超过普通列表控件和 Ajax 列表控件。经测试，在同一场景下，通过 Flex 列表控件加载 1000 条数据，平均响应时间是 0.1 秒，Ajax 控件 0.5 秒，普通刷新页面的方式最慢，从发出请求到返回并显示数据，大概需要 1 秒钟。

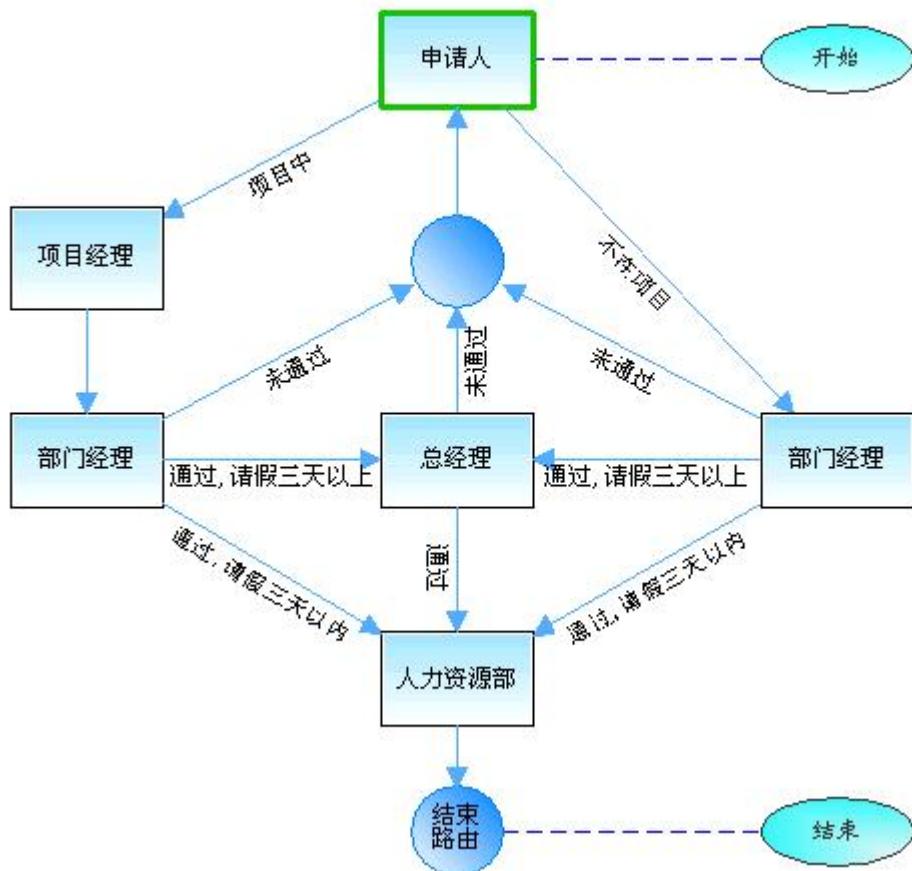
请输入要查询的内容				查询	重置
序	支持事由	支持日期 ▼	详细内容	支持人	状态
1	Flex使用指导	2010-06-04	提供Flex开发IDE以及使用文档，并对开发人员进行使用指导。对其提出的问题进行解答。	臧	已解决
2	培训组织权限和excel导出	2010-06-02	安排培训场所以及培训项目，对开发人员培训组织权限系统和讲解excel导出功能。	臧	已解决
3	给加贸版ERP系统生成代码	2010-06-01	项目中文名称：加贸版ERP系统 项目英文名称：JM_ERP	马	已解决
4	flash播放器使用支持	2010-05-31	需求：Flash播放器支持定制窗口尺寸 解决：通过flex应用与js函数交互的方式实现窗口尺寸的定.....	臧	已解决
5	组织权限API使用支持以及机构树	2010-05-31	需求：根据机构树获得机构信息 解决：说明使用的API接口，并提供参考实现的jsp代码	臧	已解决
6	Flash播放器支持	2010-05-28	需求：在页面中使用flash播放器，按指定尺寸播放视频文件 解决：使用一个开源的flash播放器的.....	臧	已解决
7	中电财项目	2010-05-28	项目中的业务需求转化为流程设计的问题	马	已解决
8	陕西联通项目问题	2010-05-28	陕西联通项目问题：业务表单中的数据部分提交，部分审批是否可以实现？表单设	马	已解决
9	组织权限admin使用支持,按钮授权	2010-05-28	需求：组织权限的管理员admin拥有最大权限，如何管理	臧	已解决

三、流程设计器

2004 年我们开始研发工作流平台，其核心功能是工作流引擎和流程设计器。为了开发出 Web-Based（基于 Web）的流程设计器，我们投入了极大的人力物力，最终采用 ActiveX 控件实现了复杂的流程设计、流程监控等功能。到目前为止基于浏览器的流程设计和监控仍然是我们的功能特色之一。但是随着技术的发展，基于 Activex 控件的流程设计器越来越显示出局限性，例如不支持多浏览器，不支持国际化，在各种 Windows 和 IE 版本中的自动安装经常会出现问题，最重要的是扩展起来比较复杂。

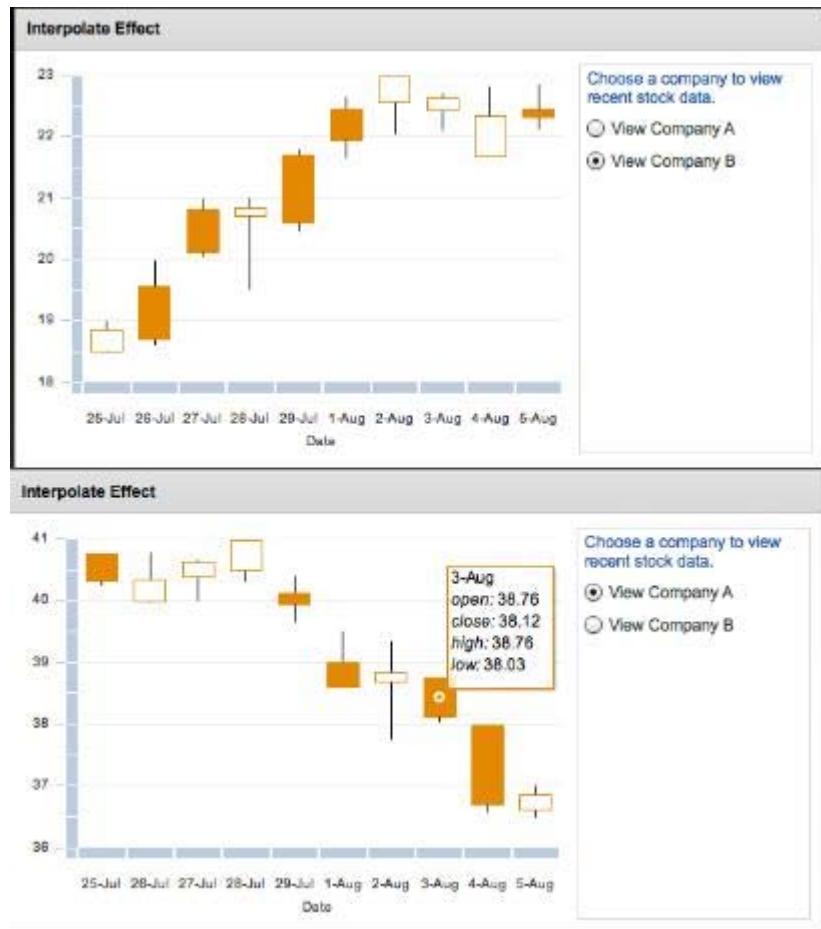
下图就是基于 ActiveX 技术的流程设计器。看上去很美，但的确存在着问题。事实上我们正在积极准备基于 Flex 技术对流程设计器进行改造，改造完成后，上面提到的问题即可迎刃而解。技术的进步带来应用的改进，所以我一直强调，技术创新才是软件企业的原动力。

边框颜色图例： [初始化] [任务已分配] [执行中] [挂起] [已中止] [执行完毕] [已完成]



四、动态图表

通过图表描述业务数据，加强数据的展现能力。每个图表都可以支持参数的动态变化，响应点击事件，实现动态效果，而这些几乎不需要额外编写代码。传统的报表工具或制图工具，例如 BIRT 或 JFreeChart 等，处理报表图片时都是通过流的方式输入静态图片，一旦生成，就是静态页面，用户无法与之交互。而使用 Flex 进行图表的开发，则可以轻易突破这一障碍，Flex 提供了大量内置的图表控件，来进行图表展示，在浏览器上的表现方式为 Flash，可以在生成图表之后，继续实现前后台数据的交互和展示。例如使用 CandlestickChart 控件来实现蜡烛图的动态展示。从一下两张图可以很清楚的看出，通过选择不同的选项，可以显示不同的趋势变化，点击图元还可以显示该图元的相关数据。这些复杂的数据操作基本上是由 CandlestickChart 控件完成的，开发者只需要组装数据即可。事实上要实现这个功能样例，只需要 100 行左右的代码。



五、知识管理

企业信息化 10 年，积累了大量的数据、文件、视频需要进行展示，通过 Flex 可以构建通用的播放器，利用流媒体技术、全文检索技术实现企业内部的知识管理。类似的技术和功能在互联网上已经不是什么新鲜事了，例如 slideshare、youtube，国内的豆丁、优酷等，这些网站要么是基于文档的管理，要么是基于视频的处理，但它们的共同点是核心技术都采用了 Flex。对于互联网领域，领域的细分无疑是非常明智的，但是在企业级应用领域，企业客户更关注的是应用的整合，我们拟基于 Flex 技术开发通用的前台播放器，可以同事播放视频和文档，后台采用 Flash Midea Server 和文件服务器，结合内容管理、全文检索和标签云技术，实现企业信息数据的全流程管理。

六、动态商品展示

基于 Flex 技术的动态商品展示已经在很多中大型电子商务公司应用，通过 Flex 可以实现很多眩目的效果来进行商品展示，同时可以动态设置商品的属性，让客户更好的了解商品细节，增加客户粘度。这样的技术在企业级应用中同样适用，主要业务领域应该是企业的电子商务

平台。

七、全键盘操作

某些特殊领域的客户，比如要快速录入大量数据，就会对全键盘操作有需求，要求在浏览器做的应用要能像 Excel 一样全键盘操作，除了支持 Tab 加 Enter 键之外，还需要能够支持四个方向键的操作支持，就像是 Excel 一样，当单元格中的文字处于全选状态的时候可以通过方向键进行导航。这样的功能用普通的 Javascript 实现一个是复杂，而是会降低网页性能。但用 Flex 来实现这种特殊场景就完全没由这些问题，本质上来说 Flex 还是客户端技术，可以很好的提供键盘支持。

结束语

企业客户越来越认识到 RIA 技术对于企业应用的重要性，而 Flex 就是实现 RIA 重要的选择之一。但是，无论技术也好，创新也好，有用的才是好的，不能因为技术而使用技术，因为创新而创新。就 Flex 而言，从根本上说它还是一个客户端程序，所以一定会比普通的 Web 页面更多的占用更多的客户端资源，所以我不建议大家在构建企业应用时大量采用 Flex 技术，一定要用其所长。Flex 在互联网领域早已大放异彩，那么它是否能成功应用于企业应用的开发呢？不必拭目以待，我想这个答案无疑是肯定的。

关于作者

池建强，12 年软件从业经验，先后在洪恩软件和用友集团任职。目前在用友集团瑞友科技 IT 应用研究院任副院长，负责公司基础应用平台的研发和整个公司的技术管理。主要关注领域：企业应用软件平台研发、领域驱动设计，OSGi，动态语言应用、云计算、移动互联等相关技术。

原文链接：<http://www.infoq.com/cn/articles/flex-application>

相关内容：

- [使用Grails和Flex开发JEE应用](#)
- [InfoQ案例研究：纳斯达克市场回放、使用Flex构建以数据中心的企业应用](#)
- [Flex SDK与Flash Builder 4：深度对话Andrew Shorten](#)
- [Blaze Data Services还是LiveCycle Data Services？](#)

高效率的超大规模 Flex 开发

作者 [Ryan Knight](#) 译者 [曹如进](#)

文章着重介绍了 Adobe Flex 开发与传统的 Web 开发的不同之处，作者告诉大家如何正确理解和利用这些不同，来为我们创建实用性更强，浏览和更新速度更快的网站。

Adobe Flex 开发与传统的 Web 开发有很多不同之处。正确的理解和利用这些不同之处，可以帮助我们创建更丰富的用户体验，也可以反过来增加网站的可用性以及浏览和更新速度。Flex 还提供了大量的组件和技术库来让 Web 开发更加轻松——它提供的强大工具要远远超过传统 Web 2.0 中的异步 JavaScript 和 XML(Ajax)应用。此外，Adobe 公司最近发布的 Adobe Flash Builder 4 beta (以前叫 [Adobe Flex Builder](#)) 中提供了许多新的和改进的工具，他们可以用来开发大规模的 Flex 应用程序。Flash Builder 4 着重于提高开发人员的生产力以及设计人员和开发人员的工作流。

Flex 开发中的关键要素之一是使用模块和运行时共享库 (Runtime Shared Libraries , RSLs)，这些模块和库可以方便系统不同部分的并行开发以及客户端上高效的内存管理等等。另外一个关键要素是使用 Sprint BlazeDS 集成，它大大简化了后台不同技术服务器端的开发和集成，如使用 Java 消息服务(JMS)和 Spring 安全进行的通信。

以下是 Flex 开发中的一些关键要素：

- 利用 Flex 的不同之处。了解为什么 Flex 开发与传统 Web 开发有所不同，并利用那些不同之处让应用程序从中获益。
- Flex 模块。了解 Flex 模块以及怎样模块化 Flex 开发和部署。使用 Flex 模块来解耦项目中的不同层次。
- 了解反模式和模式。预先了解正确的应用程序反模式，这样你可以按照正确的架构模式实现优秀的设计。
- 使用 Spring BlazeDS 集成项目简化开发。使用 Spring BlazeDS 集成项目简化 Java 服务器集成。
- 有效地设计。与设计团队紧密合作，充分利用 Flex 和 Adobe Flash 的优势。
- 测试。预先计划测试。

是什么让 Web 开发与 Flex 有所不同？

与传统 Web 应用程序不同，Flex 应用程序开发更像传统的桌面应用开发。例如，整个应用程序会被下载到客户端——这是一个庞大的下载，而且取决于应用程序如何组织。当然，有许多方法可以优化这个过程，例如使用组件库或将工程分解成各个模块。

这套方法与传统的超文本标记语言（Hypertext Markup Language，HTML）编写的应用程序不一样——即使它是用 Ajax 开发的。一个 HTML 的 Web 站点通常可以划分成不同的页面，以使得下载量非常小。并且即使使用 Ajax，下载量还是相对较小，因为大部分 Ajax 库本身就相对较小。使用这种方法可以让单个页面的下载量比整个应用程序更小。

Flex 与传统 Web 应用程序开发的另外一个不同之处在于，状态主要在客户端上维护。这超出了在 Web 应用程序上使用 Ajax 控件，因为后者一般只在 Web 浏览器上维护少量的客户端状态。而现在，应用程序中的所有状态都在局部变量中进行维护。

如果你还不太熟悉 Flex 的话，另外一个不同之处可能对于你来说是一个挑战。Flex 使用了一个高度的事件驱动编程风格，主要原因是 Adobe Flash Player 是单线程的，因此任何需要长时间运行的调用都需要注册一个回调函数。单线程对于图形化用户界面（GUI）开发来说是有利的，因为它不存在死锁或者一个线程窃取所有 CPU 周期，从而保持了 GUI 的响应。单线程还使得开发更加简单，因为你不需要处理多线程编程。（还记得在 Swing 中处理线程的噩梦吗？）

Flex 中用以处理事件的主要机制是函数指针，或者作为函数参数进行传递的闭包。这些指针可以是用户为了某个异步事件结束时发生的事件或者提醒而注册的。（函数指针对于 Java 程序员来说很有挑战，因为核心 Java 语言还不支持闭包。）

Flex 模块至关重要

Flex 提供了一些选项可以将应用程序划分成模块，包括创建库以及主应用程序的子模块。通过将项目设计成层次化的形式，开发和编译都可以在单个的分支中进行。

模块化 Flex 项目的主要好处在于，这么做可以优化客户端上下载和启动的时间。应用程序的一小部分可以预先下载到客户端，而剩下的部分可以在后端按需加载。此外，模块还可以被加载和卸载，以减少客户端上的内存占用量。

模块化 Flex 项目的另一个主要好处是，它可以让开发过程更容易模块化。它允许开发人员

在不同的模块和库上进行工作以避免开发团队的死锁发生。

然而构建大型 Flex 应用程序的一个挑战之处在于，编译过程所花的时间要比大部分程序员熟悉的 Java 编译时间要慢很多。重新编译一个大型 Flex 应用程序可以花上好几分钟。想要优化这个过程，你可以创建分离的库并且只编译你当前工作的应用程序部分。

一种方法是采用 SWC 文件形式的库。SWC 是 Flex 中使用的主要存档格式，它有点类似于 Java 中的 Java 存档 (JAR) 文件。你可以使用 SWC 档案来完成一些工作：

- 组件库。组件库是一个 SWC 文件，它包含 Flex 应用程序中使用的类和其他资源。
- 主题。主题定义了应用程序的观感。它包含主题所需的资源（例如图片文件和字体），以及定义这些资源如何被使用在应用程序中的层叠样式表(Cascading Style Sheet ,CSS)。
- 资源包。这些都是本地化的属性文件和 Adobe ActionScript 类集。

有两种方法可以在项目中添加 SWC 文件：要么静态地连接它们，这种情况下它们会编译进你的项目；要么把它们当做运行时共享库 (RSL , Runtime Shared Library) 来使用。使用多个 RSL 的好处在于它们可以被分开下载和缓存到客户端上。此外，如果客户端有多个模块的话，它们还可以共享同样的 RSL。

你还可以将主应用程序进行模块化。一种方式是采用 Flex 模块；另一种是使用子应用程序。子应用程序的好处在于它们独立于主应用程序进行开发和测试。

注意当编译模块和库时，Flex 并不像 Java 一样层次地遍历整个树。相反它只编译链接到主应用程序中的部分。这种方式会使得模块和库的开发非常微妙，因为你可能注意不到某个文件被链入了。当进行库开发的时候，有必要定义哪些文件会被包含。

库具有相对直接的可被利用的优势。本质上你是向 Flex 模块管理器传递了想要加载模块的 URL。

```

protected function getModuleFromServer(url:String, onSuccess:Function) :void
{
    var module:IModuleInfo = ModuleManager.getModule(url);
    module.addEventListener(ModuleEvent.READY, loadModuleSuccessHandler);
    module.load();
}

public function loadModuleSuccessHandler(moduleEvent:ModuleEvent) :void {
    var module:IModuleInfo = moduleEvent.module;
    .... handle initializing the module
}

```

像这样进行模块加载的有趣之处在于你可以指定另一个 URL 来加载这个模块——例如，在某个故障情况下，客户端丢失了与主服务器的连接。这时客户端可以继续从另外一个 URL 中加载模块，以维护重要的业务功能。

让应用程序轻松地开发和测试

通过适当地激活 View 栈 (Stack)，你可以用单个 URL 轻松地访问应用程序的任何点——这一功能拥有超越单纯可用性上的优势。除此之外它还使得应用程序上的工作变得很轻松。一个常见的错误发生在你改动部分应用程序，而这个应用程序需要在好几个屏幕和样式中进行导航。如果你不得不为每一处改动而手工进行导航时，这个过程将非常耗时。取而代之，我们可以使用一个固定的 URL 来访问应用程序的不同部分，这将大大减轻开发中的痛苦并且加速整个过程。相同的模式同样可以让应用程序变得很容易测试，因为应用程序单个部分可以通过 URL 来访问和测试。

在 Flex 中，有两种方法可以将视图栈构建到应用程序中。一种方法是使用状态 (states)，而另一种方法则是使用一个 ViewStack 组件 (ViewStack.mxml)。视图栈只是用来显示对象的栈：这些对象可以是一个包装简单的表格或是一个复杂组件的画布 (canvas)。使用视图栈的好处在于你可以通过在代码中设置当前的子组件来轻松的操作它们。例如 myViewStack.selectedChild=accountInfo 将会设置视图栈去显示 accountInfo 视图。这个功能可以让你在测试和开发中轻松地操作视图栈。

客户端架构：反模式

在软件编程中，会有一个通用的范式来定义软件开发中的反模式和模式。模式是解决常见问题的重复解决方案；反模式则是不熟悉语言的程序员最会发生的常见错误。定义好 Flex 中的模式和反模式后，我将会展示一些新手 Flex 程序员常犯的错误，然后再展示一些 Flex 编程开发的最佳实践。

这些实践可以帮助项目在增长为大规模的过程中，避免常见的错误。第一个通用的反模式是将所有的 MXML 视图放在根包中。这么做好处不是很明显，因为你可以将 MXML 文件放入到包中，并且通过它们的名字空间（由于实际的包的名字可能没有在文件中指定）进行引用。其实 MXML 文件是根据其所在的目录结构来放入包当中的。因此将 MXML 文件放入到根包中等价于不将它们放入任何包中，这样做会使一个大型项目变得一团糟。

举一个例子，[Flex 商店示例](#)不仅将主应用程序文件放入到根包中，还将主页，商品页和支持

页 (HomeView.mxml, ProductsView.mxml, SupportView..mxml) 都放入根包中。这个过程和把所有根目录中的视图包放入一个 Java 包中比较类似。

开发人员在创建第一个应用程序时常常会碰到一个问题：那就是如何用视图来捆绑住应用程序模型。一个通用的反模式是在类之间传递模型或者在调用视图前设置模型。反模式的一个例子——也来自于 Flex 商店——即在主应用程序中将目录传递进 ProductsView。

```
<ProductsView id="pView" label="Products" catalog="{catalog}"
```

这段代码在小的例子中可以正常工作，但是随着应用程序逐渐变大，它会带来一些问题，如层之间的紧耦合，以及很难开发和测试。另外一个问题是当同样的数据需要在应用程序不同部分显示时，这些数据需要分开地传递给每一个组件。首先，这看起来似乎不是一个问题，但是试想下如果尝试模拟每一个传递和使用数据的实例，结果会怎样。

客户端架构：推荐模式

多年来众多的 Flex 架构和框架已经被开发出来。最有前景的一种方法使用了一个像 Swiz 一样的轻量级依赖注入框架，并用它来划分项目结构。你可以在 Swiz 中定义类似于 Spring 中的应用程序 bean，并将它们注入到应用程序中的不同部分。一个典型的应用程序框架会包含一个中心类，这个类会定义服务怎样组合在一起——例如，ClientBeans.mxml 类为应用程序声明了通用的模型和控制器：

```
<model:StoreModel id="storeModel" />
```

然后 View 类连接上控制器：

```
[Autowire(bean="storeModel")]
[Bindable]
public var storeModel: StoreModel;
```

对于控制器和其他通用组件，你也可以这么做。用这种方式组织项目结构有一些优势。一个优势在于它可以让模拟服务轻松地植入控制器，以促进开发和测试。另外一个优势在于它能够除去和客户端架构不同层之间的耦合性。

一个精心设计的主视图状态有助于实现前面提到的可用性和可测试性目标。视图栈的思想是让应用程序中的任何视图可以通过单个 URL 进行访问。除此之外，视图栈的参数需用通过通用控制器传入，这个通用控制器能够让你在应用程序中各个地方使用通用的 URL 和必要的参数来访问期望的视图。

一个 Flex 架构示例：重构 View Store

作为架构的一个示例，让我们先看看怎样将

http://www.adobe.com/devnet/flex/samples/flex_store/ 上的 Flex Store 的例子进行重构。

首先，将主源代码移动到子目录中。在这个例子中，我将所有的资源，产品视图，以及样例目录移动到了 flexstore/src/flex 中并将其作为源代码主目录。我还把所有的 MXML 和 CSS 文件移动到了这个目录中。

接下去，将三个主视图类移动到视图文件夹中。在 src/flex 目录下创建一个 com/anvilflex/view 目录。将 HomeView.mxml , ProductsView.mxml 以及 SupportView.mxml 文件放入到这个目录中。为此，你必须对 flexstore.mxml 做出如下改动：

- 在<mx:Application>标签中加入 xmlns:view="com.anvilflex.view.*" 。
- 将视图变为使用<view>标签（因此<HomeView>将变成<view:HomeView ... />）。

将 Product.as 类移动到分开的域文件夹中——com/anvilflex/view。为此，你必须修复 Product 类的 import 语句，并删除 Product 单词的最后一个字母。接下去，按住 Ctrl-空格键来使用自动完成功能，它同样也可以为 Product 类加入正确的 import 语句。

最后，引入 ProductControl 来管理产品页面和结算间的过渡。

使用 Spring BlazeDS 集成简化 Java 开发

Spring BlazeDS 集成项目大大简化了与客户端交互的 Java 服务端代码开发。在最新发布的项目中，团队成员可以实现核心的配置文件，如 web.xml 以及 Flex 服务配置文件。接下去，团队人员需要简单地了解一下如何在他们的类中增添适当的标注来将它们作为服务展现给 BlazeDS。举个简单的例子，你可以将 Product 服务类展现为一个远程服务：

```

@Service("productService")
@RemotingDestination(channels={"my-amf"})
public class ProductDAO {

    @Autowired
    public ProductDAO() {
        //... initialize the ProductDao class
    }

    @RemotingInclude
}

```

```
public Product findProductName(int id) {  
    Product product = database.findProduct(id);  
    return product.name;  
}  
}
```

在这个例子中，我将 ProductDAO 类声明为了远程服务，它可以通过使用 productService 进行调用。单个方法可以使用 @RemotingInclude 将其展现为远程方法。其实这些方法都会作为远程服务展示给 BlazeDS，其中 Flex 可以通过远程对象调用远程服务。

与图形设计团队一起工作

刚开始的 Flex 项目中一个常见的问题是，确保图形设计团队了解 Flex 中可以做什么。Flex 为很多有趣的设计提供了可能性：图形设计团队需要了解 Flex 中可以做什么，以及 Flex 怎样来表示一个基础设计转变。

通常情况下，碰到的问题是，图形设计团队用一些像 Adobe Photoshop 的工具创建了一个模拟。尽管设计看起来非常好，但是它与工作中的应用程序还相距甚远。整个设计将不得不削减并且颜色和字体都得与应用程序的 CSS 相匹配。根据设计的复杂度，自定义的 Flex 组件也需要被创建。这些组件可能是需要自定义皮肤的按钮，也可能是定义了一系列新功能集合的组件。困难之处在于设计和实现之间变得极其耗时，设计人员需要搞清楚什么是可能的并从设计中实现这些变化。每一次迭代都会变得更加复杂。

幸运地是，Adobe 已经引入了一个新的工具来改进这个过程，这个工具就是 Flash Catalyst。它可以让类似 Photoshop 这样工具产生的艺术品能够轻松地转变为可用的原型。它还可以创建一个原型的 GUI 并最终应用到应用程序中。通过使用此工具，设计人员可以更快地遍历他们的设计，从而使该组织能更有效地调整设计，以满足其要求。

单元及功能测试

Flex 开发中有两种主要的方法来测试一个应用程序。第一个是使用 funit 或者 fluint 的功能测试。这个方法可以进行服务以及控制器的测试。第二个方法是功能测试，它可以测试与 GUI 间的实际交互。功能测试可以使用像 Mercury 一样的商业工具，或者像 FlexMoney 一样的开源工具。

两种类型测试的关键是要在松耦合层创建可以独立测试的代码。例如可以通过将默认捆绑改变为使用模拟服务，来轻松地修改服务。

总结

使用 Spring BlazeDS 集成的 Flex 为大规模企业应用程序提供了一系列卓越的技术。开发人员的挑战在于设计模块化，易调试，可扩展的架构。这里的可扩展，我指的不仅仅是软件，还包括开发过程。解决这些挑战可以帮助项目成功，并为组织带来一个更高的投资回报率。

原文链接：<http://www.infoq.com/cn/articles/flex-development-ryan-knight>

相关内容：

- [从标签时代到富客户端：从Web 1.0 到Flex](#)
- [FlexMonkey将单元测试引入Flex用户界面开发](#)
- [Java程序员ActionScript 3 入门](#)
- [Flex与JSON及XML的互操作](#)
- [构建Flex应用的 10 大误区](#)

Spring BlazeDS Integration 简介与入门

作者 [Ryan Knight](#) 译者 [张龙](#)

Spring BlazeDS Integration 是 2008 年年底 Adobe 与 Spring 共同宣布联合开发一个新项目。其目标就是让开发者可以利用 Spring 开发模型通过 Adobe Flex、BlazeDS、Spring 以及 Java 技术创建 RIA。其优势在于将 Spring 的易用性与 Flex、BlazeDS 以及 Java 整合起来以共同创建应用。本文作者根据这些改变介绍了本项目与以往传统方式的优越之处。

去年底 Adobe 与 Spring 共同宣布将联合开发一个新项目 : Spring BlazeDS Integration。其目标是：开发者可以利用 Spring 开发模型通过 Adobe Flex、BlazeDS、Spring 以及 Java 技术创建 RIA。这样我们就可以通过 BlazeDS 公开 Spring 管理的服务而无需额外的配置文件。其优势在于将 Spring 的易用性与 Flex、BlazeDS 以及 Java 整合起来以共同创建应用。

我将在本文中介绍 Spring BlazeDS Integration 项目对传统开发方式有哪些改观，同时展示一些相关示例。首先，我们一起来看看它是如何改变应用的集成方式以及如何对现有的 Spring 项目进行转换使之可以利用新的集成。最后我将对该项目的其他特性以及优势进行适当的介绍。

本文所用的示例应用是个简单的苏打 (soda) 服务，它提供了基本的账户信息。其所用的数据模型是 SodaAccount，代表了客户端账户信息。

以 Spring 的方式开发 RIA

Spring 的横空出世完全颠覆了传统 Java 服务端的开发方式。它鼓励通过依赖注入的方式来装配 POJO，这极大地简化了应用的开发与测试。

Spring 的核心配置是通过 Java bean 实现的。借助于 bean，任何 Java 类都能被公开成为服务。比如说，下面的配置片段就将 Soda 服务声明为一个 Spring bean：

```
<!-- Implementation of soda bean-->
<bean id="sodaBean" class="com.gorillalogic.sodaBank.SodaService"
      init-method="initSodaAccounts">
    <property name="numAccounts" value="1000"/>
</bean>
```

为了将这些 bean 公开成为 Flex 客户端所用的远程服务，Integration 项目采用了 Spring Web

MVC。Spring Web MVC 将 DispatcherServlet 作为一个中央分发器，用以处理任何类型的 HTTP 请求或是基于 HTTP 的远程服务。我们可以通过相同的 JavaBean 配置方式来配置该 DispatcherServlet 以将请求转发给相应的处理器进行后续处理。

之前，BlazeDS 项目会通过 MessageBrokerServlet 将请求路由给相应的 BlazeDS Message Broker。现在借助于 Spring BlazeDS，Spring Web MVC DispatcherServlet 已经替代了 MessageBrokerServlet，接下来就需要配置 DispatcherServlet 以将请求 转发给 MessageBrokerHandlerAdapter。该适配器本身是个 Spring 工厂 bean，它会在 Spring Web 应用上下文中创建一个局部 BlazeDS Message Broker 实例，然后将 Spring bean 公开成为远程服务，之后 Flex 客户端就能够直接调用该服务了。

这种配置 BlazeDS Message Broker 的方式可以与 Spring 项目结合的更加紧密，同时还减少了将 Spring bean 公开成远程服务所需的配置量。比如说之前，我们需要在 messaging.xml 中声明一个单独的条目来公开 Java 服务，但现在可以轻松地在声明 Spring bean 的那个配置文件中公开远程 bean。

Spring BlazeDS Integration 也使用了一些标准的 BlazeDS XML 配置文件来配置消息基础设施。这包括通道定义等一些内容。

该项目的下一版本将要增加与 Spring Security 的集成。最初的实现会通过一个 pointcut advisor 来保护 BlazeDS 端点。Pointcut advisor 是 Spring AOP 支持的一部分。

建立全新的 Spring BlazeDS Integration 项目——服务器端

无论是建立全新的项目还是为现有的项目增加支持，步骤都是大同小异的。第一步需要将所需的 jar 文件增加到程序库目录中。可以通过 [Spring Source 站点](#) 下载，也可以使用示例项目中的程序库。

对于这个示例来说，我们打算将一个简单的 Soda Service 项目修改为 Spring BlazeDS 项目。首先要修改 web.xml 文件。将该文件中所有对 BlazeDS MessageBrokerServlet 的引用都删掉，然后加上对 Spring DispatcherServlet 的引用：

```
<servlet>
  <servlet-name>Spring MVC Dispatcher Servlet</servlet-name>
  <servlet-class>org.springframework.web.servlet.DispatcherServlet</servlet-class>
  <init-param>
    <param-name>contextConfigLocation</param-name>
    <param-value>/WEB-INF/config/web-application-config.xml</param-value>
  </init-param>
```

```

<load-on-startup>1</load-on-startup>
</servlet>

<servlet-mapping>
    <servlet-name>Spring MVC Dispatcher Servlet</servlet-name>
</servlet-mapping>

```

以上配置通过标准的 Servlet 映射模式将所有的请求路由给 DispatcherServlet , 同时还将上下文配置信息指定为 web-application-config.xml。

标准的 BlazeDS 文件位于 WEB-INF/flex 中 , 其主文件为 services-config.xml , 其中定义了通道、日志及其他系统配置。该文件的一个变化就是标准 AMF 通道的 URL 变成通过 DispatcherServlet 来路由请求 :

```

<channel-definition id="my-amf" class="mx.messaging.channels.AMFChannel">
    <endpoint url="http://{server.name}:{server.port}/{context.root}/
        gorilla/messagebroker/amf"
        class="flex.messaging.endpoints.AMFEndpoint"/>

```

web-application-config.xml 是主配置文件。事实上 , 一旦配置好了其他文件 , 那么在大多数情况下只需要修改该文件就行了。在 web-application-config.xml 文件中声明了 MessageBrokerHandlerAdapter , 这样就会将 HTTP 消息路由给 Spring 管理的 Message Broker。

```
<bean class="org.springframework.flex.messaging.servlet.MessageBrokerHandlerAdapter"/>
```

文件中声明的框架的另一部分是 MessageBrokerFactoryBean 。它对我们想处理的 URL 进行了映射 , 也就是发往 Dispatch Servlet 的所有请求 :

```

<bean class="org.springframework.web.servlet.handler.SimpleUrlHandlerMapping">
    <property name="mappings">
        <value>
            /*=mySpringManagedMessageBroker
        </value>
    </property>
</bean>

<bean id="mySpringManagedMessageBroker"
    class="org.springframework.flex.messaging.MessageBrokerFactoryBean"/>

```

由于 DispatcherServlet 监听着/gorilla 路径 , 因此该配置会将发送给/gorilla URL 的所有请求都传给 Spring 管理的 MessageBroker 。然后由于 Message Broker 使用了 WEB-INF/flex/services-config.xml 外的通道配置 , 这样发送给/gorilla /messagebroker 的消息就会被路由给声明为服务的那些 bean 了。

这样 , Message Broker 就与 bean 发生了联系 , 下一步配置就是声明应用中的 Spring bean 并

将其公开成为远程服务。在该示例中，我们将 SodaService 声明为 sodaBean：

```
<bean id="sodaBean" class="com.gorillalogic.sodaBank.SodaService"
    init-method="initSodaAccounts">
    <property name="numAccounts" value="1000"/>
</bean>
```

接下来，为 BlazeDS remoting 公开 sodaBean：

```
<bean id="sodaService"
    class="org.springframework.flex.messaging.remoting.FlexRemotingServiceExporter">
    <property name="messageBroker" ref="mySpringManagedMessageBroker"/>
    <property name="service" ref="sodaBean"/>
</bean>
```

上面的配置引用了之前的 Message Broker bean 来公开 sodaBean。默认情况下，类中所有方法都会被公开成为远程服务。FlexRemotingServiceExporter 拥有众多选 项来对服务进行配置。比如说，我们可以通过 includeMethods 和 excludeMethods 来选择公开或是不公开哪些方法。

建立全新的 Spring BlazeDS Integration 项目——客户端

用于连接服务器的客户端代码使用了标准的 Flex RemoteObjects。对于该示例应用，我们声明了如下的 RemoteObject：

```
<mx:RemoteObject id="remoteObject"
    destination="sodaService"
    result="resultHandler(event);"
    fault="faultHandler(event);"
    channelSet="{sodaChannels}"/>

<mx:ChannelSet id="sodaChannels">
    <mx:AMFChannel uri="/gorilla/messagebroker/amf"/>
</mx:ChannelSet>
```

凭借该 remote 对象，Flex 客户端可以调用远程的 Java 服务器。有两种手段能让客户端知道该向哪个通道发起调用。其一是针对服务端配置文件 services-config.xml 来编译客户端，通常这不是一个好办法，因为它将客户端与服务器端紧密耦合在了一起。其二是通过一个通道集将通道配 置在客户端上。

对 RemoteObject 的调用与本地对象调用大同小异，区别在于返回结果的过程是异步的。基于这个原因声明了一个 resultHandler 以在服务端的结果返回时进行回调。在本示例中，对服务器端的调用形式如下：

```
remoteObject.getSodaModel();
```

返回的结果是个 ResultEvent , 然后将其转换为 sodaModel :

```
sodaModel = event.result as SodaModel;
```

保护远程服务——服务器端

为了保护与服务器端的通信 ,Spring BlazeDS Integration 项目使用了一个客户化的认证和授权过程。该过程将 Spring Security 与 BlazeDS 安全过程集成起来了 (注意 , 在本文撰写之际 , 该处所使用的代码仅仅存在于 SVN 上。同时我将示例所用代码的快照放到了 jar 文件中)。

在服务器端进行安全配置的第一步就是定义安全上下文。这需要为用户定义用户名、密码以及相关角色等信息。在该简单示例中 , 我们仅仅将用户信息定义在文件中。真实的企业项目很可能会将这些信息放到数据库中或是使用单点登录。

我们通过一个单独的配置文件 (security-context.xml) 来声明系统中的用户。需要将该文件加到 DispatcherServlet 上下文配置中以便服务器启动时就能对其进行加载。如下配置片段展示了如何在 web.xml 文件中配置该文件 :

```
<servlet>
  <servlet-name>Spring MVC Dispatcher Servlet</servlet-name>
  <servlet-class>org.springframework.web.servlet.DispatcherServlet</
    servlet-class>
  <init-param>
    <param-name>contextConfigLocation</param-name>
    <param-value>/WEB-INF/config/security-context.xml
      /WEB-INF/config/web-application-config.xml
    </param-value>
  </init-param>
  <load-on-startup>1</load-on-startup>
</servlet>
```

接下来 , 在 security-context.xml 文件中声明系统用户 :

```
<authentication-provider>
  <user-service>
    <user name="ryan" password="monkey"
      authorities="ROLE_USER, ROLE_ADMIN" />
    <user name="alex" password="chimp" authorities="ROLE_USER" />
  </user-service>
</authentication-provider>
```

第二步就是通过客户化的安全配置文件来配置 Message Broker。为了将额外的服务或是安全

配置到 Spring 管理的 Message Broker 上 , 我们需要对 MessageBrokerFactoryBean 增加一些额外的配置处理器。该处理器实现了两个方法 : processBeforeStartup 及 processAfterStartup , 它们可以在 Message Broker 的启动前后为其设定相关的配置。

要想配置 Message Broker 的安全信息 , 我们需要设定两个处理器。其一是登录命令 , 它提供了认证与授权 ; 其二是一个安全配置处理器 , 它保护个别的通道与 URL 。

LoginCommand 是 BlazeDS 中的一个接口名 , 它用于定义客户化的认证与授权过程。接下来 , SpringSecurityLoginCommand bean 就将 Spring Security 与 BlazeDS security 集成起来 将 BlazeDS 发出的进行认证与授权的调用传递给 Spring 管理的安全上下文。下面的代码片段声明了该 bean 的一个实例 并引用了之前定义的安全上下文 :

```
<bean id="loginCommand"
      class="org.springframework.flex.messaging.security.SpringSecurityLoginCommand">
    <constructor-arg ref="_authenticationManager"/>
</bean>
```

第二个过程需要定义一个安全配置处理器以作为 pointcut advisor , 它定义了保护每个通道及 URL 的机制。 pointcut advisor 是 Spring AOP 的一部分 , 定义了在某个方法调用之前需要调用的 advice 。本质上 , 它会过滤对远程服务的调用然后阻止未授权的调用。

BlazeDS 在内部通过 AMF Filter 来执行消息调用的预处理与后续处理。这些 Filter 的工作方式类似于 Servlet Filter 并遵循着标准的 pipe-and-filter 设计模式。这样 , 每个 Filter 都能阻止对消息的进一步处理。该安全过程会通知通道的 AMF Filter 来增加 Spring 管理的安全。

为了定义安全处理器 , 首先需要向 WEB-INF/flex/services-config.xml 文件中添加两个额外的通道。

```
<channel-definition id="my-protected-amf"
      class="mx.messaging.channels.AMFChannel">
    <endpoint url="http://{server.name}:{server.port}/{context.root}/
                  gorilla/protected/messagebroker/amf"
              class="flex.messaging.endpoints.AMFEEndpoint"/>
</channel-definition>

<channel-definition id="my-protected-by-id-amf"
      class="mx.messaging.channels.AMFChannel">
    <endpoint url="http://{server.name}:{server.port}/{context.root}/
                  gorilla/protected2/messagebroker/amf"
              class="flex.messaging.endpoints.AMFEEndpoint"/>
</channel-definition>
```

接下来我们定义一个端点源（ endpoint source ），它配置了需要保护的端点或通道以及访问它们所需的角色。对于本示例来说，我们只定义一种用户角色，然后配置一个需要保护的 URL 及端点：

```

<bean id="configAttribute"
      class="org.springframework.security.ConfigAttributeDefinition">
    <constructor-arg type="java.lang.String" value="ROLE_USER"/>
</bean>

<bean id="endpointSource"
      class="org.springframework.flex.messaging.security.EndpointDefinitionSource">
    <constructor-arg>
      <bean class="org.springframework.security.util.AntUrlPathMatcher"/>
    </constructor-arg>
    <constructor-arg>
      <map>
        <entry>
          <key>
            <bean
class="org.springframework.security.intercept.web.RequestKey">
              <constructor-arg value="**/protected/
messagebroker/**"/>
            </bean>
          </key>
          <ref bean="configAttribute"/>
        </entry>
      </map>
    </constructor-arg>
    <constructor-arg>
      <map>
        <entry>
          <key>
            <value>my-protected-by-id-amf</value>
          </key>
          <ref bean="configAttribute"/>
        </entry>
      </map>
    </constructor-arg>
  </bean>

```

以上配置保护了匹配于`**/protected/messagebroker/**`路径的 URL。在本示例中，这包括了 `my-protected- amf`（该通道监听/gorilla/protected/messagebroker/amf）与 `my-protected-by-id-amf` 通道。

接下来，我们定义端点拦截器与异常解析器以将所有配置连接在一起：

```
<bean id="endpointInterceptor"
      class="org.springframework.flex.messaging.security.EndpointServiceMessagePointcutAdvisor">
    <constructor-arg>
      <bean class="org.springframework.flex.messaging.security.EndpointInterceptor">
        <property name="accessDecisionManager" ref="_accessManager"/>
        <property name="authenticationManager" ref="_authenticationManager"/>
        <property name="objectDefinitionSource" ref="endpointSource"/>
      </bean>
    </constructor-arg>
  </bean>

  <bean id="exceptionTranslator"
        class="org.springframework.flex.messaging.security.EndpointServiceMessagePointcutAdvisor">
    <constructor-arg>
      <bean
            class="org.springframework.flex.messaging.security.SecurityExceptionTranslationAdvice"/>
    </constructor-arg>
  </bean>
```

上面的代码配置了端点拦截器以将访问与认证管理器应用到端点源上。

最后，我们修改 Spring 管理的 Message Broker 的定义来应用这些配置处理器：

```
<bean id="mySpringManagedMessageBroker"
      class="org.springframework.flex.messaging.MessageBrokerFactoryBean">
  <property name="configProcessors">
    <set>
      <ref bean="loginCommand"/>
      <ref bean="securityConfigProcessor"/>
    </set>
  </property>
</bean>
```

这样，Message Broker 就会通过配置在 security-context.xml 文件中的安全上下文来保护定义在端点拦截器中的通道。现在如果要定义服务，那还需要定义服务通信所需的通道。

```
<bean id="sodaService"
      class="org.springframework.flex.messaging.remoting.FlexRemotingServiceExporter">
  <property name="messageBroker" ref="mySpringManagedMessageBroker"/>
  <property name="service" ref="sodaBean"/>
  <property name="channelIds" value="my-protected-amf,
                                my-protected-by-id-amf"/>
</bean>
```

对于该 soda 服务来说，我们已经定义好了其只能在安全的通道上进行通信。这会阻止未认证的用户（并非来自于正确的角色）对该服务的访问。

保护远程服务——客户端

安全的远程服务的客户端配置相当简单。所有的重头戏都在服务端完成了。在客户端，我们只需修改 remote object 定义使之包含一个安全的通道即可：

```
<mx:RemoteObject id="remoteObject"
    destination="sodaService"
    result="resultHandler(event);"
    fault="faultHandler(event);"
    channelSet="{sodaChannels}"/>

<mx:ChannelSet id="sodaChannels">
    <mx:AMFChannel uri="/gorilla/protected/messagebroker/amf"/>
</mx:ChannelSet>
```

现在，remote object 必须要经过认证方能对 soda 服务进行调用。比如说，如果我们没有认证，同时又调用了 soda 服务来获取 Soda 模型，那么客户端就会收到如下的错误消息：

```
Received fault: [RPC Fault faultString="An Authentication object was not found in theB
SecurityContext" faultCode="Client.Authentication" faultDetail="null"]
```

我们只需将登陆信息传递给管道集就能实现对客户端的认证。如下是个超级简单的示例：

```
var token:AsyncToken = sodaChannels.login(username.text, password.text);
token.addResponder(
    new AsyncResponder(
        function(result:ResultEvent, token:Object = null):void{
            remoteObject.getSodaModel(numAccounts.text);
        },
        function(result:FaultEvent, token:Object = null):void{
            ta.text += "Received fault: " + result.fault + "\n";
        }
    )
);
```

以上代码会从用户名与密码框中获取登陆信息并对用户进行认证。如果认证成功，那么就会调用远程服务并返回 Soda 模型。

小结

Spring BlazeDS Integration 项目通过使用 Spring 开发模型简化了 Java RIA 的开发。通过与 Spring

Bean 及 Spring Security 的集成，它可以轻松实现将远程服务直接公开给 Flex 客户端的过程。总体上来说，该项目主要面向使用 Flex、BlazeDS 及 Java 的企业级应用开发。

Integration 项目的未来版本会进一步增强与 Spring 的集成。计划的特性包括与 Spring Security 及 JMS 的进一步集成。同时还有一个用于定义服务端上远程服务的客户化模式定义。这将极大地简化配置文件的编写。

原文链接：<http://www.infoq.com/cn/articles/spring-blazeds-integration>

相关内容：

- [使用Spring AOP和AspectJ编排工作流](#)
- [对话Spring.NET](#)
- [PHP开发者的BlazeDS和JMS指南，第一部分](#)
- [PHP开发者的BlazeDS和JMS指南，第二部分](#)
- [MyEclipse for Spring 8.6 发布：新增Flex、GWT及iPhone脚手架DSL的演进](#)



CIDC2010 CHINA INTERACTIVE DESIGN COMPETITION

中国交互设计大赛

展示艺术院校交互设计成果 | 表彰优秀的学生及设计团队

CIDC2010 中国交互设计大赛立足于校园，旨在促进中国交互设计教育的发展，推广和表彰通过技术和艺术创造出全新用户体验的交互设计作品或产品。

作品提交及大赛详情 登录CIDC官网：STUDENTART.COM.CN/CIDC/

CIDC2010 中国交互设计大赛 2010.9.10 ~ 2011.1.20 作品提交截止日期：2010年12月20日



INFORMATION DESIGN 信息设计



这个类别中提交的参赛作品可包括网站、交互广告、触摸屏交互内容、应用程序、交互展示或其他交互内容。在本类别的比赛中，要求参赛者在提交的作品中主要体现出良好的用户体验设计和视觉设计等。

ENTERTAINMENT DESIGN 娱乐设计



本类别中提交的作品可包括游戏、电子书、或其他互动娱乐作品。这一类的作品是指通过创新的视图布局、数据可视化和具有吸引力的视觉效果来增强用户体验。

MOBILE DESIGN 移动设计



本类别中提交的作品可包括游戏、互动广告、应用程序、网站等基于掌上移动设备的交互设计。根据移动设备屏幕大小、交互方式和性能特点设计创新性的交互体验。



CIDC大奖



最佳创意奖
最具人气奖
最佳用户体验奖



学生代表计划

每个类别一名大奖获得者。

三项类别分别设立最佳创意奖一名、最具人气奖一名、最佳用户体验奖一名。

所有获奖者都可入选学生代表计划

每人获得5000元现金，证书，奖杯。

每人获得1500元现金，证书，奖杯。

获得CS5一套，并免费参加2011年北京Flash峰会



中央美术学院
China Central Academy of Fine Arts



更多信息，请访问 **Adobe中国开发者社区**

Java 程序员学习 Flex 和 BlazeDS 的十三个理由

作者 [Ryan Knight](#) 译者 [沙晓兰](#)

本文列述了 13 个 Java 程序员应当学习 Flex 和 BlazeDS 的理由，讨论了为什么 Flex 结合 BlazeDS 是开发 RIA 的最佳组合之一。无论是高度交互的网站还是以 Java 为后端的企业应用，这项组合都是最佳选择之一。更重要的是，这项组合能同时为开发员和企业带来高回报（ROI）。

本文列述了 13 个 Java 程序员应当学习 Flex 和 BlazeDS 的理由，讨论了为什么 Flex 结合 BlazeDS 是开发 RIA 的最佳组合之一。无论是高度交互的网站还是以 Java 为后端的企业应用，这项组合都是最佳选择之一。更重要的是，这项组合能同时为开发员和企业带来高回报（ROI）。

在阐述 Java 程序员应当学习 BlazeDS 的 13 条理由时，我以一个假想的苏打分派系统来展示如何让已有的 Java 程序转变为 RIA 应用。通过这个例子，我同时还会讲解到 BlazeDS 在已有 Java 应用或新建 Java 应用中的多种不同用法。

理由一：开源

Flex 软件开发工具箱（SDK）的核心是个开源框架，专门用来开发、维护那些在不同浏览器、不同操作系统下界面都相同的 RIA 应用。Flex 发布采用的是 Mozilla 公共许可证（Mozilla Public License）。编译后的 Flex 应用在 Adobe Flash 平台上运行。

BlazeDS 是连接 Flex 和 Java 的索桥，是项针对远程调用和消息传递的开源技术。在 Java 应用服务器上，它以 servlet 的形式存在，因此可以在任何标准 Java 网络应用中运用它。BlazeDS 以 LGPL（Lesser GNU Public License）公共许可证发布。在发布 BlazeDS 的同时，Adobe 还公布了 AMF（ActionScript Message Format）规格说明，BlazeDS、Java 和 Flex 客户端间以这种简洁的二进制格式实现通信。

理由二：完善的社区支持

Flex 社区非常活跃，社区贡献了大量项目。Flex.org，这个配以社区新闻的 Adobe 站点几乎每天都有新的社区贡献；Yahoo!上的 Flex 用户组的成员也已经超过了 11000。

再比如 Google Code 上的 Flexlib 项目，已经提交了大量的开源 UI 组件。Swiz 和 Mate 项目贡献了优化事件处理的框架；还有 Gorilla Logic 贡献了自动化 UI 测试的 Flex Monkey 项目。

理由三：带来广阔的就业前景

据 Adobe 的 Flex“传道士”——James Ward 看来，Flex 高级开发员的市场需求非常大，学习 Flex 能让你拥有极具市场竞争力的开发技能。

理由四：更高的业务效益回报

总体上，开发企业 web 应用不是个轻松的活，这基本上是众所周知的事实。Flex 和 BlazeDS 提供的不仅仅是功能强大的开发工具，而且开发技术本身相对也非常简单。开发效率可以得到大幅度的提升，产品因此可以很快推向市场。Flex 和 Flash 带来的用户体验也相对更有魅力，对增加流量、提高用户转化率（conversion rate）很有帮助。

很经典的一个例子是 Borders 连锁书店。他们最近发布了带有“魔法书架”的新网站，这个网站采用 Flash 接口来模拟书籍借阅的过程。Borders 发现这一模拟借阅非常明显地提到了用户转换率：“借助这个 Flash 驱动的接口，用户可以浏览书籍、DVD 和 CD 的封面，用户转换率比其他没有此项功能的网站高出 62%”。

理由五：Flex 是第一个专门为创建 UI 而设计的语言

大部分语言都不是在第一时间设计其对 UI 的支持。Java 中 Swing 包的实现刚好是个很好的证明。也就是这个原因，很多像捆绑数据这样的简单动作在 Swing 当中的实现就非常痛苦。用 Swing 最大的问题在于，要想提高开发效率就必须对 API 了如指掌。

Flex 刚好相反，它是专门为创建 web UI 而设计的。正如 Bruce Eckel 所说，Flex 是第一个针对 UI 开发的领域特定语言（DSL）。用 Flex 构建 UI 比其它诸如 JSP、JSF、Swing 等技术简便得多。语言本身糅合了数据绑定、事件处理、控件布局以及其它一些 UI 常用开发技巧，就算对语言没有深刻的理解也不会影响开发效率。

理由六：编程风格近似于 Java

你可以继续使用现有的 Java 开发工具来开发 Flex 应用。当然也可以采用 SDK 中携带的免费命令行工具，Adobe Flex Builder（一个 Eclipse 插件），或最近的 IntelliJ IDEA 8。

Flex 提供的是一个有状态环境，在这个环境中，数据从客户端加载。这种编程模式更像是开发桌面客户端而非 HTML 编程，这种风格对于用过 Java Swing 编程的开发员来说应该是相当熟悉。

Flex 是 MXML（类似 XML 的 UI 标记语言）和 Adobe ActionScript（面向对象的解析语言）的结合体。鉴于这种结合方式，Flex 编程与 Java 非常相似，因为两者用的都是熟知的面向对象的概念。

最理想的开发环境是把 Flex 应用创建在 web 部署文件夹下。这样一来，每次更新应用之后都不需要重新部署，只要在浏览器下刷新一下就可以了。用 Flex 和 BlazeDS 开发后，开发效率绝对比之前有很大的提升。

理由七：BlazeDS 可以在任何 Java 应用服务器上运行

BlazeDS 目前已发布了多个版本，其中的 turnkey 版本还包含了为 BlazeDS 配置的 Apache Tomcat。本文中，我用的是二进制发布版本，其中含有一个 WAR 用来展示如何把应用部署到各种应用服务器上去。不用这个 WAR 的话，你也可以从中提取 JAR 文件放到自己的项目中去。关于安装 BlazeDS 的各种选项内容，可以参见 BlazeDS 的 wiki。

这里举一个简单的例子，比方说要在已有的一个简单的苏打调配系统中应用 BlazeDS。你只要把 JAR 文件放到项目文件夹下，然后就可以在应用里直接用 BlazeDS，可以部署到能够部署应用的任何地方。

在项目中添加 BlazeDS，只需要完成下面两个步骤：

1. 解压缩 BlazeDS WAR 文件的内容：jar xvf blazeds.war。
2. 把 JAR 文件都拷贝到项目的 lib 文件夹下：cp -R WEB-INF/lib /sodaSample。

理由八：可以在已有 Java 应用中运用

比方说这个简单的苏打调配系统，假设你想要扩展这个已开发好的服务，让其它 Flex 应用可以远程调用。在现成的应用中配置 BlazeDS 的基本步骤有：

1. 修改 WEB-INF/flex 文件夹下的 BlazeDS 配置文件
2. 在该应用对应的 web.xml 文件里定义 MessageBrokerServlet 和 session 监听器

配置好 BlazeDS 之后，再把苏打调配服务添加到 BlazeDS 远程配置文件里，Flex 客户就能远

程调用了。这个过程通过在配置文件里定义一个目的地（destination），一个或多个信道（channel）来传输数据。基本的 AMF 信道定义在 services.xml 文件里。下面这段配置在 remoting-config.xml 里定义了目的地（destination）：

```
<destination id="sodaService" channels="my-amf">
    <properties>
        <source>com.gorillalogic.sodaSample.SodaService</source>
    </properties>
</destination>
```

通过在远程调用配置文件里定义端点（endpoint），Flex 客户端就可以调用任何一个基本的 Java 服务。

要是想把 Java 数据模型也传送到 Flex 客户端的话，只要在 ActionScript 类中定义好两者间的映射：

```
[Bindable]
[RemoteClass(alias="com.gorillalogic.sodaSample.SodaModel")]
```

这段代码告诉 Flex，在远程调用的服务返回 SodaModel 的时候，把它映射到 Flex 的 SodaModel。本例中的 Flex 客户端显示的就是如何调用这个 Java 服务。调用返回一个已经填写好预定信息的 SodaModel：

```
public function callSodaService():void {
    var sodaType:String = type.text;
    var sodaCount:int = parseInt(cnt.text);
    var flag:Boolean = preOpen.selected;
    remoteObject.getSoda(sodaType, sodaCount, flag);
}

private function resultHandler(event:ResultEvent):void {
    var sodaModel:SodaModel = event.result as SodaModel;
}
```

Flex 返回的结果是通用的 result 变量，可以直接映射到你的 SodaModel。这里我就不深入讨论怎么实现映射了，但其中值得提到的是要在编译配置里声明 services-config.xml 路径，像这样：

```
-locale en_US -services=/nsource/sodaSample/web/WEB-INF/flex/services-config.xml
-context-root /
```

如果不添加这个路径的话，你的 Flex 客户端就没法找到 Java 服务。同样的方式，你还能把一个对象从客户端传递回服务器端。比如，你可以把一个空的 soda model 发回服务器（审核注：原文这里写的是客户端，根据上下文判断这里应该是服务器端）。

理由九：可以通过 Java 来扩展和修改 BlazeDS

假如你想添加特殊的日志来记录苏打调配服务被调用的情况，那么你可以扩展标准的 Java 适配器来添加日志功能。

首先，添加一个继承了 JavaAdapter 的 Java 类：

```
import flex.messaging.services.remoting.adapters.JavaAdapter.  
public class TimingJavaAdapter extends JavaAdapter {
```

其次，重载 invoke()方法：

```
public Object invoke(Message message) {  
    RemotingMessage remotingMessage = (RemotingMessage) message;  
    String operation = remotingMessage.getOperation();  
    String destination = remotingMessage.getDestination();  
  
    Logger.info("calling " + operation + " on destination " + destination);  
    Object data = super.invoke(message);  
    return data;  
}
```

这个方法中，你可以看到调用之后的操作和调用的目的地（destination）。这种方法也能用来处理其它一些问题，比如记录向服务器发送调用需要多长时间。

理由十：HTML 和 JSP 也能调用 BlazeDS

从 HTML 和 JSP 也能调用 BlazeDS，这种调用有几种不同的实现方式，比如通过 Browser Manager 或 fflashVarsf 来实现。Flex 应用能够读取由 HTML 页面设置的 fflashVarsf。

比方说你想要通过 HTML 页面来发送你的用户名和准备预定的苏打类型，你可以在 HTML 页面这样设置 flashVars：

```
<object id='SodaSample' classid='clsid:D27CDB6E-AE6D-11cf-96B8-444553540000'  
codebase='http://fpdownload.macromedia.com/get/flashplayer/current/swflash.cab'  
height='100%' width='100%'>  
    <param name='src' value='SodaSample.swf'/>  
    <param name='flashVars' value='username=ryan&type=coke'/>  
    <embed name='mySwf' src='SodaSample.swf'  
pluginspage='http://www.adobe.com/go/getflashplayer' height='100%' width='100'%  
flashVars='username=ryan&type=coke'/>  
</object>
```

然后，在 Flex 应用中，你可以通过读取应用参数来获取这些变量：

```

var username:String;
if (Application.application.parameters.hasOwnProperty("username")) {
    username = Application.application.parameters.username;
}
  
```

理由十一：Flex 和 BlazeDS 的数据传输性能远胜于其它 Ajax 解决方案

目前使用的远程过程调用（RPC）都默认选择 AMF 二进制协议。AMF 是个开放的标准，而且相当快。James Ward 曾举例比较过多种远程调用解决方案。尽管其它 Ajax 技术——比如 Dojo——已经能够快速处理几百行的数据，但是用 Flex 和 BlazeDS 的话可以轻松搞定成千上万行。（请参考 James Ward's census，可以了解下各种不同的 RIA 数据加载技术的测评。）

理由十二：Java 客户端能够直接调用 BlazeDS

最新发布的 BlazeDS 当中含有一个 Java 的 AMF 类，通过这个类，你可以在 Java 客户端直接调用 BlazeDS 服务器。对于单元测试和加载测试来说，BlazeDS 的这种调用方式非常实用。

理由十三：Spring 下也能用

Adobe 和 Spring 互相联手，尝试将双方项目集成起来。他们发布的第一 Spring-BlazeDS 集成版本就向大家展示了他们的良苦用心。Spring Bean 能够以远程服务的方式被调用，因此可以清除很多重复的配置文件。更多这方面的相关信息，可以参考该项目的主页。

结论

开源的 BlazeDS 创建在 Java 基础上，无论是对新的还是已有的 Java 服务器项目来说都是个很好的选择。Flex、BlazeDS 技术能够提供高性能的远程通信，支持 Flex 和 Java 间的对象映射，因此是 RIA 开发的理想选择。Flex 和 BlazeDS 的开发新手，如果曾经是 Java 开发员的话，会发现整个开发过程效率非常高，而且很容易掌握。

Flex 加 BlazeDS 还是开发大型 Java 企业应用的理想选择。我们组开发的上个项目中，应用涉及到 50 多个不同的界面，而且服务器和客户端之间需要规律性地互传几千行的代码。这类应用几乎没法通过传统的 Ajax 技术来实现。但是在引入了 Flex 和 BlazeDS 之后，我们在年内就发布了第一个版本。看，这就是这对动态组合为你的应用开发项目带来的过人之处。

原文链接：<http://www.infoq.com/cn/articles/java-flex-blazeds>

微软设计产品市场总监 Forest Key 谈 Silverlight

作者 [温飞飞](#)

在加入微软之前，Forest Key 是 Macromedia 公司 Flash 的产品经理，对用户交互技术有着宗教般的狂热。也是因为对交互技术的喜爱，使他来到了微软。在这次他来中国推广 Silverlight 的时候，InfoQ 中文站的记者有机会就 Silverlight 的性能、与其他平台的比较、客户为什么要采用 Silverlight 技术、开发 Silverlight 所用的工具和 Silverlight 在微软产品线中所扮演的角色等问题进行了探讨。

InfoQ 中文站：Silverlight1.0 相对于其他平台的实际开发效率是否有提升，有那些提升？

Forest Key：首先，微软拥有设计工具、开发工具、以及 Silverlight 运行时插件，它们会组成一个有效的生态系统，让设计人员和开发人员能够有效地工作在一起。当然，今天还不完全具备这些，明年我们将有可能具备这样的能力，提供全套的解决方案。现在最重要的是将 Silverlight 运行时插件部署到更多的电脑上。所以我们把重点放在了媒体应用中，我们有更好的视频质量、更便宜的视频部署方案等一些有效的视频解决方案。

如何将设计和开发的体验统一起来（设计人员设计的产品界面，开发人员能够将其还原成技术实现的产品），这不是一个技术问题，这对任何公司来说都将是一个挑战。在我们的产品中不同的是，我们可以提供更好的设计或者开发的团队体验，更好地将设计理念传达到开发人员中，这点是我们领先竞争对手的。

InfoQ 中文站：Silverlight 和其他平台相比在用户体验上有那些提升，Silverlight 在技术上有什么优势吗？

Forest Key：Silverlight 的突出之处，我认为是将设计和开发统一起来。我们的开发人员可以有效地和设计人员合作。我们在 WPF 中已经提供了很好的用户体验，在最终用户方面 Silverlight 不会有很大不同，Silverlight 利用了 WPF 以后的很多优势技术，而且还将进一步地提升这一优势。由于创建过程更加简单和使用，这也会吸引更多的设计人员来创建更多的丰富的基于浏览器的互联网应用。

InfoQ 中文站：在实际应用中如何说服客户采用 Silverlight 技术？

Forest Key：如果我们的客户已经有 WPF 经验，那么事情就变得很容易，因为他们已经体验到了 WPF 的强大优势和丰富的体验，也认识到我们已经有很多成功的项目经验。当然如果

我去说服客户开发一个全新的项目，我们谈到的首先是编写代码的可维护性，而这些代码中有相当一部分是 JavaScript，所以很容易找到工程师来开发和维护这些代码。

Silverlight 1.0 在网络视频领域是具有相当竞争力的，因为 WMV 是很多公司用的现有媒体技术，而 Silverlight 可以让这些现有的媒体资源拥有更友好的用户界面，可以在不同操作系统中部署（Windows、Mac 和 Linux 等）。因此，现在已经和正在有很多客户转向了 Silverlight 和 Silverlight Video。一旦明年发布 Silverlight 1.1，我们主要的精力就会放到创建丰富互动体验的 Web 应用中，届时会吸引更多开发团队来尝试，并告诉他们如何建设更加健壮的应用程序解决方案。这种 Silverlight 的推广方式会比我们提供的其他任何华丽的辞藻都更有效，更能吸引更多开发商。当然我们也很希望开发人员尝试我们的开发工具 Visual Studio 和 .NET，以及我们的数据库等任何微软的技术，但是如果他们对这些技术不感兴趣，他们仍然可以使用 Silverlight 和其他他们现有的技术进行结合。

InfoQ 中文站：Expression Studio 在设计和开发上有什么优势？

Forest Key：Expression Web 是一个很好的符合 Web 标准的开发工具，也是一个成熟的产品，在很多方面它都是市场上现有产品中非常好的一个，特别是它是一个很好的 CSS 设计工具。在后面的版本中我们将在其中增加很多功能，使它成为市场中更有竞争力的产品。

Expression Blend 带来了新的设计思路，而不是 Flash 的替代者。Flash 开创了动画制作软件的先河，但是它并不是一个很好的交互设计工具，事实上 Adobe 已经发布了一个新的产品 Thermo，用来回应 Blend。它展示了 10 个功能，而这些功能都是 Blend 已经拥有的，这说明 Adobe 已经开始意识到并开始追赶我们——将交互式设计融合到设计工具中。因此 Blend 已经成为这一领域的领导者。而在这一细分的领域，Blend 并不是 Flash 的替代者，两个产品分别处在不同的细分市场。

Expression Design 不打算取代 Photoshop，而是真正的统一工作流程的工具。Expression Media 是一个很好的媒体管理工具，我们认为它优于任何的 Adobe Creative Suite，很多人都喜欢这个产品。它将帮助人们很好地管理自己的媒体资产。Expression Encoder 是迄今为止我所看到的最好的编码工具，当然是就视频编码来说。在 2.0 中我们将加入四个更加强大的功能。

所以，我们认为最重要的是设计师和开发人员的合作以及协同，使设计和开发的整个流程变得更加顺畅。我认为以上几个产品，将使我们的 Expression 产品更具有竞争力。

InfoQ 中文站：在整个微软的产品线中 Silverlight 担当什么角色？

Forest Key：Silverlight 是微软 Server and Tools Business (STB) 的一部分，是一个平台级技术并已经被纳入我们的服务体系。和 Google 做在线应用的方式不太一样，微软虽然也会用

Silverlight 做一些在线应用，我们现在还没有发布任何基于 Silverlight 的在线产品，但是将来你会看见很多。我们现在的重点是增强工具的可用性，做好 API 和 SDK，让人们能更好地用我们的产品来创建更多更有竞争力的 Web 应用，这是我们现阶段所要做的。一个现有的例子就是微软通过 Windows Live 用 Silverlight 来免费发布自己的 Web 应用（这个应用叫 Popfly，一个基于 Silverlight 的网络应用，可以允许用户制作自己的服务，然后放到 Popfly 中，目前还在测试阶段）。我们明年还将在中国设置 CDN，这样在中国访问的速度就会和欧美一样快捷。我们的 MSN 拥有一个 10 亿量级的用户数据库，目前已经开放了这个数据库的 APIs，开发人员可以利用这个 API 开发自己的 Silverlight 应用。

因此，我们希望随着时间的推移，Silverlight 结合 Visual Studio 以及 Expression Studio 可以共同为设计和开发人员提供一个良好的，高效的创建应用软件的生态系统。

关于作者

温飞飞，网名 ai829。知名 Flash/Flex 开发爱好者，Silverlight.cn 名誉管理员，对表现层技术有着多年的研究经验，现就职于搜狐，负责新技术和新产品的研究与推广工作，其博客地址为：<http://blog.80s.net.cn>。参与 InfoQ 中文站内容建设，请邮件至 china-editorial@infoq.com。

原文链接：<http://www.infoq.com/cn/articles/forest-key-silverlight>

相关内容：

- [把WPF作为一种富客户端技术](#)
- [吴磊畅谈Silverlight在中国人寿的应用](#)
- [Adobe Flash平台开发者峰会谈RIA开发现状](#)
- [Silverlight作为Web应用程序技术的角色引发争论](#)
- [JavaFX技术预览](#)

虚拟座谈：HTML5 来了，JavaScript 框架会如何发展

作者 [Dionysios G. Synodinos](#) 译者 [王瑜珩](#)

InfoQ 利用电子邮件的形式，组织了一次虚拟座谈会，针对最近崭露头角的 HTML5 新增加的 Javascript API，邀请了来自主流 JavaScript 框架的代表，主要从 web 人员需要关注点和 JavaScript 的未来发展趋势等方面对这些专家进行了提问。

HTML 5 是万维网核心语言的第 5 个主要版本，早在 2004 年就由[网络富文本应用技术工作组（WHATWG）](#)发起。虽然标准仍在制定之中，但有些浏览器已经能够支持一部分 HTML 5 的特性了，如 [Safari 4 beta](#)。

除了更多的标记以外，HTML 5 还添加了一些[脚本 API](#)：

“新增的特性充分地考虑了应用程序开发人员，HTML 5 引入了大量的新的 Javascript API。可以利用这些内容与对应的 HTML 元素相关联，它们包括：

- 二维绘图 API，可以用在一个新的画布（Canvas）元素上以呈现图像、游戏图形或者其他运行中的可视图形。
- 一个允许 web 应用程序将自身注册为某个协议或 MIME 类型的 API。
- 一个引入新的缓存机制以支持脱机 web 应用程序的 API。
- 一个能够播放视频和音频的 API，可以使用新的 video 和 audio 元素。
- 一个历史纪录 API，它可以公开正在浏览的历史纪录，从而允许页面更好地支持 AJAX 应用程序中实现对后退功能。
- 跨文档的消息传递，它提供了一种方式，使得文档可以互相通信而不用考虑它们的来源域，在某种程度上，这样的设计是为了防止跨站点的脚本攻击。
- 一个支持拖放操作的 API，用它可以与 draggable 特性相关联。
- 一个支持编辑操作的 API，用它可以与一个新的全局 contenteditable 特性相关联。

- 一个新的网络 API，它支持 web 应用程序在本地网络上互相通信，并在它们的源服务器上维持双向的通信。
- 使用 JavaScript API 的键/值对实现客户端的持久化存储，同时支持嵌入的 SQL 数据库。
- 服务器发送的事件，通过它可以与新的事件源（event-source）元素关联，新的事件源元素有利于与远程数据源的持久性连接，而且极大地消除了在 Web 应用程序中对轮询的需求。

最近 InfoQ 利用电子邮件组织了一次虚拟座谈，主题为 JavaScript 框架将会如何发展，以便充分利用这些新的 API。此次座谈邀请了来自主流 JavaScript 框架的代表：

- **Dylan Schiemann**，SitePen 的 CEO，[Dojo](#) 的共同创建者
- **Matt Sweeney** 和 **Eric Miraglia**，来自 [YUI](#) 开发团队
- **Andrew Dupont**，[Prototype](#) 的核心开发者
- **Thomas Fuchs**，[script.aculo.us](#) 的创建者，[Prototype](#) 和 Ruby on Rails 的核心开发人员
- **David Walsh**，[MooTools](#) 的核心开发人员
- **Scott Blum** 和 **Joel Webber**，GWT 的[领头人](#)

下面是我们的问题以及各专家的回答。

InfoQ：由于 HTML 5 标准仍在制定之中，大多数开发人员对它的新特性并不熟悉，您认为对于 web 开发人员，哪些特性是最值得关注的？

Dylan：HTML 5 包含很多东西，其中很多有价值的特性都已经在 Dojo 这样的框架中实现了。例如，内置的富表单控件将包含多文件上传和数据属性，这样人们就不会再抱怨 Dojo 使用非标准的自定义属性了，虽然这些属性也是合法的。最近 Peter Higgins 为 Dojo 解析器写了一个补丁，大概有 1KB 左右的代码量，以便当这些特性添加到浏览器时可以使用它们。对我来说最感兴趣的特性是 WebSocket，它是由 Michael Carter 提出，并由 Orbited 最先实现的。WebSocket 非常适合那些长连接应用，你可以将它看做是 web 安全的 TCP Socket。

Matt & Eric：对那些把浏览器当成是应用平台的开发人员来说，HTML 5 包含了一些很具创新性的组件。但需要注意的是这有点超出文档语义领域，已经到达 DOM API 领域了，这不是 HTML 规范的必须部分。我们希望 HTML 5 规范能够限制在对文档语义的增强和精化上，而把对行为 API 的规定留给其它规范。

一般来说，开发人员应该知道 HTML5 提供的以下 HTML 相关的特性：

- 反对使用仅用于显示的元素和属性
- 更多有语义的元素
- 更多种输入控件和语义
- 自定义数据属性

开发人员看起来对 HTML 5 中的 DOM API 很感兴趣，如果它们最后能够被实现，其中的某些特性确实能够丰富我们的工具箱，成为很重要的工具。最早被浏览器厂商实现的 2D 绘图 API（通过 Canvas 元素）以及客户端存储 API 已经引发了广泛的关注，这些关注与浏览器厂商在标准确定之前就率先实现有关。但是还有很多其他的重要变化目前也在草案之中：

- Iframe 沙箱
- 支持"getElementsByClassName()"
- "classList" API
- 音频和视频（通过 Audio 和 Video 元素）API
- 离线 web 应用 API
- 编辑（通过 contentEditable 属性）API
- 跨文档消息 API
- 服务器端事件 API

HTML 5 试图解决一些重大的问题，而且解决的不错，比我们上面列出的还要多。

Andrew：“web 存储”（客户端数据库）会被广泛使用——很多网站已经开始好好地利用它了。新的 form 控件（Web Forms 2）从一开始就存在于标准之中，我都等不及赶快使用了。服务器发送事件可以用来构造纯粹的“推”服务程序，而不需要依赖 Ajax 的不断轮询或不稳定的长连接。我还喜欢自定义数据属性，虽然相比之下，这只是一个不太重要的特性。

Thomas：除了离线应用特性（主要指持久化存储），我认为 VIDEO 和 AUDIO 标签是最新的新特性，因为我们终于可以离开烦人的 OBJECT 和 EMBED 了。当然这还需要一段时间，直到所有浏览器都支持，不过这确实是朝着正确的方向在发展，而对所有非正式标准的标准化工作也会让浏览器表现得比现在更好。一个容易被忽略的特性是 web 应用程序能够注册协议和媒体类型，以使它们自己成为默认处理程序，就像桌面应用程序一样（但是从 UI 的

角度看，仍有很多阻碍，因为对计算机不甚了解的用户不懂什么是“协议”或者“媒体类型”）。

David：除非被广泛支持，否则没有哪个 HTML 5 的特性是非常值得关注的。这些概念值得去实现，但在只有少数几个浏览器支持的情况下使用是不明智的。这就是我们不使用 querySelectorAll 的原因：浏览器厂商的不同实现会导致更多针对浏览器的 hack，而不是简单的避免使用 QSA。

Scott & Joel：在 HTML5 中我最关注的是那些现在就可以使用，而不需要开发人员额外创建不同版本程序的特性。我对其 中的数据库和缓存 API 特别感兴趣，程序可以真正支持在线和离线两种模式了。另外对于移动 web 开发人员来说，HTML5 提供了大量特性来处理低速和连接 不稳定的移动网络。

另外一些 HTML5 的特性也很让人激动，例如<canvas>、<audio>和<video>，这些功能现在还需要插件来实现。目前这些标签还没有被主流浏览器广泛支持，但是随着浏览器支持的增强，使用的人也会增加。

InfoQ：您认为现有的 JavaScript 框架该如何发展，以支持像内置媒体、离线浏览以及更高级的服务器进程通信这样的新特性？能粗略说一下您所在项目的路线图么？

Dylan：我们的目标一直是弥补浏览器的不足。我们希望我们的框架能够越来越不重要（例如，实现统一的事件系统或者是 QuerySelectorAll）。我们发现在某些情况下，我们会随着时间的推移一点一点的删除代码，但是 Dojo 的用户并不会感到有任何的不同，他们 还是可以继续使用相同的 API，只是这些 API 会简单的调用本地实现。对于你提到的更高级的特性，Dojo 已经可以支持媒体和离线程序，并且支持 JSON-PRC 和 REST，甚至可以在 Perservere 这样的服务器端 JavaScript 环境中运行！

Matt & Eric：JavaScript 框架的作用是利用更丰富的 API 和透明的跨浏览器支持来改善编程环境。YUI 将会遵循 HTML 5 标准（特别是那些已经对浏览器产生影响的），并添加对老版本浏览器的支持，以便让新的功能可以在标准实现和推广之前就得以应用。客户端存储 API 是一个 例子，YUI 将要实现它以消除 HTML 5 和现有浏览器之间的不同。

Andrew：像 Prototype 这样的“中间件”框架，主要是把难用的、不一致的 API 包装为统一的、完善的接口，HTML5 并不会（为 Prototype）带来太大的影响。HTML5 的特性会让某些工作变得简单，但是 Prototype 会一直保留“困难的方式”，直到最后一个非 HTML5 浏览器不再得到支持。

另一方面，建立在 Prototype 上的库 - script.aculo.us，显然需要做出选择。Script.aculo.us 提供了一个“slider”控件，HTML5 也有。是否应该使用浏览器内置的 HTML5 slider？还是使用

现有的纯 JavaScript 版本，以保正在所有浏览器中的外观一致？

HTML5 提供了一些特性的本地实现，而之前的多年我们一直用 HTML 和 JavaScript 来实现这些特性，这是一个进步。如果我们能够全部转移到 HTML5 而不管老版本浏览器，大多数的（JavaScript）库都可以移除大量的代码。但是在很长的一段时间里，我们还需要保留这些旧代码。

Prototype 和 script.aculo.us 并没有长期的路线图，但是我知道，当四种主流浏览器中至少有两个支持某一特性时，我就会认真考虑是否实现它。记住，HTML5 不会一次就全部实现的。

Thomas：是的，它们会进化的，最终当浏览器支持这些新特性时，它们也会支持。这对于框架来说并不是什么新鲜事，当新版本浏览器发布时，通常需要关注的是 bugs 而不是新特性。目前，对于 script.aculo.us 来说，最新的“舞台”将是 Canvas 元素，它可以平衡 Prototype 的功能。例如，insertAdjacentHTML() DOM API 可能会比我们目前插入 HTML 的方法更快。

David：我认为我们会像以前那样：从浏览器实现中抽象出 API，对那些不一致的实现进行修复。在某种程度上，我们为老版本浏览器开发解决方案，以提供跨浏览器支持，但当我们无法实现时，我们会提供检测功能以处理这种情况（或许使用欺骗手段）。我们还不能忘记 IE6 拒绝灭亡的事实，还要过很长的时间，我们才能完全依赖这些特性。

对于路线图，我们将会利用这些特性的优点，首先创建更小、更快的库。如果我们可以为支持 HTML5 的浏览器构建更快的选择器引擎，我们会将精力集中在那 里，而不是一些非广泛支持的特性。将规范集成到我们的 API 将会提升性能并减少代码量，这在短期内是对我们的最大好处，直到更高级的特性被浏览器市场普遍 实现。

Scott & Joel 对于 GWT 来说，我们会集中在那些被主流浏览器实现的特性上，但是我们也 可能会根据用户浏览器的不同而提供不同的实现方式。例如我们提供了抽象的存储 API，可以根据用户的环境使用 Google Gears 或 HTML 5。GWT 的好处是，最终用户只需要下载他们浏览器支持的特定实现，因为我们已经事先考虑了所有的可能。

InfoQ：HTML 5 为客户端添加了非常多的重量级特性，有些人认为目前的 JavaScript 和 DOM 编程模型已经走到了尽头。我们是否需要为不久的将来考虑一个完全不同的编程模型？

Dylan：jQuery 和 Dojo 的一个主要不同是，jQuery 本质上是以 DOM 为中心的，而 Dojo 还试图改进 JavaScript 的不足。像 Mozilla Lab 的 Bespin 这样的程序用于非 DOM 为中心的开发，我一直认为 DOM 是 JavaScript 开发人员的工具，而不是全部。如果有些事不能通过修改 DOM 来解决，那就不应该在浏览器中解决。坦白地说，我们已经通过不同的工具提供了不同的开发方式。

Matt and Eric：浏览器环境允许不同的模型，而且事实上也有很多模型被开发出来。Flash/Flex/AIR 就是一个很好的例子，它与 HTML/DOM/CSS（同时）都是 web 成功的关键部分。当你使用 iPhone 上的 Facebook 程序而不是 Facebook 的移动网站时，你就在使用一种新的模型。到目前为止，还没有哪个其他的模型可以在访问性和简便性上超过它。我们应该在以后“考虑其它模型”么？作为开发人员，我们曾经有过争执，每次我们构建新程序时，都会考虑其他的模型。如果今天的大多数程序是 web 程序，那么就已经说明浏览器仍然有价值。我们认为浏览器中仍有许多未被发现的价值，聪明的改进规范（就像 ECMA3.1 那样）将会有对目前的平台产生长远的改善。

Andrew：这很难说。有些人希望浏览器解析标记；有些人希望浏览器在窗口上绘制像素。这取决于你构建的程序类型。这并非是要代替 DOM，而是寻找其他的模型来补充 DOM，这样你就可以使用最合适的工具来工作。我觉得 Canvas 和 SVG 可能就是这一类模型。

Thomas：我不这样认为，HTML、CSS 和 JavaScript 的组合已经被证明是非常实用和通用的，每一项技术都在积极的进步，没有必要替换掉它们。

David：怎么会在“不久的将来”呢？任何 HTML5 中能够改变这个模型的东西，都会在多年后才能出现。而且目前的模型已经很完善、很易于理解，更被成千上万的网页所使用。我认为目前还不会有任何改变。

Scott & Joel：我认为目前的方式还没有任何走到尽头的迹象，反而发展的更快、更有组织了。一方面，有很多理由去制造更好的浏览器（带有 v8 的 Chrome、带有 TraceMonkey 的 Firefox、带有 SquirrelFish Extreme 的 Safari，当然还有 IE8）。不管你喜欢单个浏览器，开发者都有一个更强大的平台。

同时，开发者在这个平台上可使用的工具也越来越好。当 GWT 出现时，我认为它是革命性的。而几周前我们刚刚发布了 GWT 1.6，它比最初的 GWT，甚至比一年前的 GWT 更强大。你可以看到，它的内部其实还是那些东西，只不过随着它的成熟而增加了一些工具。

因此我认为还有很大的发展空间。

InfoQ：有人建议使用另一种客户端语言，例如 Ruby。您认为 JavaScript 目前是否足够强大？是否需要做出大的修改？是否应该使用更多的 DSL 技术？

Dylan：我想我们很可能会看到 DSL，甚至在服务器端也会有 JavaScript。有一个服务器端 JavaScript 讨论组对此有着极大的兴趣。JavaScript 本身并不是问题所在，真正的问题是浏览器对 DOM 和 JavaScript 交互的实现方式，以前的 JavaScript 引擎比现在 Mozilla、Google、

Apple 和 Opera 都要慢很多。我曾经认真考虑过如果 Python 或 Ruby 成为客户端语言意味着什么，坦白说我觉得这并不能解决问题。

Matt & Eric : JavaScript 在 ECMA 3.1 中做出了彻底的改变，这就是目前我们真正需要的。

浏览器可以自行选择实现其它的脚本引擎。不过既然它们按照规范实现了 DOM API，使用什么脚本语言也就无所谓了。一些人——包括 Yahoo 的 Douglas Crockford - 曾经争论过将来的 Web 是否需要一种新的内建安全机制的语言。

Andrew : 完全的语言替换是不会发生的 - JavaScript 背后的力量很强大。我喜欢已经流产的 ES4 提案中的很多新特性——运算符重载、Ruby 风格的 catch-all methods 等等。不幸的是，ES4 包含的太多了。我很庆幸在“妥协”后，ES3.1 包含了 getter 和 setter。但是我认为 ES 3.1 做的还不够。简单来说，我觉得应该尽量让 JavaScript 更加动态。

Thomas : 是的，我觉得 JavaScript 就应该是现在这样。它不应该成为一个“真正的面向对象编程语言”，它强大的基于原型的机制已经很好了。这可以让我们使用不同的开发风格，并根据个人喜好进行定制。JavaScript 和 Ruby 有时候非常相似 (JavaScript 框架 Prototype 中的某些部分就是在向 JavaScript 中添加 Ruby 风格的元素)。更多的 DSL 将会很好——我最希望未来在 JavaScript 中能有两样东西，一样是捕获未定义函数名 (就像是 Ruby 的 method_missing)，另一样是内联函数 (blocks) 以避免总是需要写 function(){...}。

David : JavaScript 是这个星球上最成功的编程语言之一。尽管有些语言没有那么多的问题(我们知道 Valerio 喜欢写 Lua 代码，他已经爱上 Lua 了)，JavaScript 真正的问题一直是浏览器的实现。框架为我们解决了其中的大量问题，但是显然 JavaScript 规范应该得到更新。框架的目的有 3 个：

- 1) 抽象出浏览器的不同，并支持老版本浏览器；
- 2) 提供丰富的、更方便的 API；
- 3) 提供规范中没有的功能 (例如效果、可排序表格、图册)。

对于浏览器的错误实现，我们需要 1)，对于仍不好用的 API，我们需要 2)，对于规范无法提供丰富的功能，我们需要 3)。我们只是没有发现这些东西在近期有任何改变。

Scott & Joel : 我认为 JavaScript 作为一种语言非常强大，甚至有些时候太强大了。你有 64 位整型数或为金融程序而内建的 BigDecimal，但是 JavaScript 面临的最大挑战在与如何构建和管理庞大的手写源代码库。当我们最初创建 GWT 时，我们打赌 JavaScript 为人们喜欢的巨大灵活性也可以使它成为一个优秀的编译器目标语言，因此也可以将它当成是 Web 上的某

种汇编语言。

你可以在 [JavaScript frameworks](#) 和 [Rich Internet Applications](#) 找到更多信息。

原文链接：<http://www.infoq.com/cn/articles/js-for-h5>

相关链接：

- [虚拟研讨会：HTML5 的新 JavaScript 框架](#)
- [JavaScript 多线程编程简介](#)
- [使用 JsUnit 和 JSMock 的 JavaScript 测试驱动开发](#)
- [HTML5 案例研究：使用 WebSockets、Canvas 与 JavaScript 构建 noVNC 客户端](#)
- [HTML5 Web Sockets 与代理服务器交互](#)

RichClient/RIA 原则与实践

作者 [陈金洲](#)

作者根据自己参加的多个不同的 RichClient 项目的开发工作，总结了自己在使用/尝试过的语言包括 Java Swing、Flex/Adobe Air、.NET WinForm/.NET WPF 等，对于不同平台之间的种种体会。对自己的富客户端开发的实践和原则进行了总结。

Web 领域的经验在过去十多年的不断的使用和锤炼中，整个开发领域的技术、理念、缺陷已经趋于成熟。JavaEE Stack, .NET Stack, Ruby On Rails 等框架代表了目前这个技术领域的所有经验积累。这样我们在开始一个新的项目的时候，只需要选择对应语言的最佳实践，基本上不会犯大的错误。例如，如果使用 Java 开发一个新的 Web 应用，那么基本上 Spring/Guice+Hibernate/iBatis/+Struts /SpringMVC 这种架构是不会产生重大的架构问题的；如果使用 RoR 那么你已经在使用最佳实践了；系统的分层：领域层，数据库层，服务层，表现层等等；为了保证系统的可扩展性，服务器端应当是无状态架构，等等。总而言之，web 开发领域，它丰富的积累使得开发者逐渐将更多的精力投入到应用本身。

来看富客户端，或者富互联网应用。在我看来，今天的 RichClient 与 RIA 已经没有分别：只要代表着丰富界面元素和丰富用户体验，需要与服务器进行交互的应用都可以称为 RichClient 或者 RIA，虽然感觉上 RichClient 更“企业化”一些（服务器往往在企业内部），RIA 更“个人化”一些（服务器往往处于公网）。从最小的层面来说，我现在正在使用的离线模式的 GoogleDoc 就是一个 RichClient 应用——虽然它没有那么 Rich，采用和 microsoft office 一样的界面；我现在正在听音乐的 Last.fm 客户端显然是一个非常典型的 RIA——它所有的个人喜好信息、音乐全都来自远在美国的服务器。本地的这个界面，只是提供收集个人和音乐信息，以及控制音乐的播放和停止；目前拥有 1150 万玩家的魔兽世界，则是一个挣钱最多的，最“富”的客户端，10 多 G 的客户端包含了电影 品质的广阔场景，华丽的魔法效果和极其复杂的人机交互。

如今的用户需求已经达到了一个新的高度，那些灰色的，方方正正的界面已经逐渐不能够满足客户的需求。从我们工作的客户看来，他们除了对“完成功能”有着基本的期待外，对于将应用做得“酷”，也抱有极大的热情。我工作的上一个项目是一个 CRM 系统，它是基于.NET Framework 3.5 的一个 RichClient 应用。它的主窗口是一个带着红色渐变背景的无边框窗口，还有请专业美工制作的图标，点击某一个菜单还有华丽的二级菜单滑动效果。我们在这个项

目中获得了很多，有些值得借鉴，有些仍然值得反思。我仍然记得我们在项目的不同阶段，做一个技术决定是如此的彷徨和忐忑：因为在当时的 RichClient 企业开发领域，几乎没有任何丰富的经验可以借鉴，我们重新发明了一些轮子，然后又推翻它；我们偏离了 UI 框架给我们提供的各种便利而自己实现种种基础特性，只是因为他们偏离了我们所倡导的测试性的原则。在写下本文的时候，我尝试搜索了一下，仍然没有比较深入的实践性文章来介绍企业环境下 RichClient 开发。大多数的书，如 Swing、JavaFX、.NET WPF 开发等等，偏向于小规模特性介绍，而在大规模的企业应用中，这些小的技巧对于架构决策往往帮助很小。

我的工作经历应当是和大多数开始进行 RichClient 开发的开发者类似：有着丰富的 Web 开发的经验之后开始进行 RichClient 开发。加入 ThoughtWorks 之后参加了多个不同的 RichClient 项目的开发工作，使用/尝试过的语言包括 Java Swing, Flex/Adobe Air, .NET WinForm/.NET WPF。对于不同平台之间的种种有些体会。在这里我将这些实践和原则总结如下。例子很可能过时，毕竟华丽的界面框架层出不穷，但原则应当通用的。使用和遵循这些原则将会帮助你少犯错误—至少比我们过去犯的错误要少。如果你拥有一定的 web 开发经验，那么这篇文章你读起来会很亲切。

这些原则/实践往往不是孤立的，我尝试将他们之间用图的方式关联起来，帮助你在使用的过程中进行选择。例如，你遵循了“一切皆异步”的原则，那么很可能你需要进行“线程管理”和“事件管理”；如果你需要引入“缓存与本地存储”，那么“数据交互模式”你也需要进行考虑。希望这张图能够帮助读者理解不同原则之间的联系。



下面列出的这些原则或者实践没有严格意义上的区分。按照上面的图，我推荐是，一旦你考虑到了某一个实践，那么与它直接关联的实践你最好也要实现。它会使得你的架构更全面，经得起用户功能的需求和交互的需求。

为了让这些实践更加通用，我采用伪代码书写。相信读者能够转化成相应的语言—Java, C#, ActionScript 或者其他。这些实践并非与某一种语言相关。在某些特定的例子中，我会采用

特定语言，但大多数都是伪代码描述的。

1 一切皆异步

所有耗时的操作都应当异步进行。这是第一条、也是最重要的原则，违背了这条原则将会导致你的应用完全不可用。

考虑这样的一个功能：点击一个“更新股票信息”按钮，系统会从股票市场（第三方应用）获得最新的股票信息，并将信息更新到主界面。丝毫不考虑用户体验的写法：

```
void updateStockDataButton_clicked() {
    stockData = stockDataService.getLatest(); // 从远程获取股票信息
    updateUI(stockData); // 这个方法会更新界面
}
```

那么，当用户点击 `updateStockDataButton` 的时候，会有什么反应？难说。如果是一个无限带宽、无限计算资源的世界，这段代码直观又易懂，而且工作的非常好：它会从第三方股票系统读到股票数据，并且更新到界面上。可惜不是。这段代码在现实世界工作的时候，当用户点击这个按钮，整个界面会冻结——知道那种感觉吗？就是点完这个按钮，界面不动了；如果你在使用 Windows，然后尝试拽住窗口到处移动，你会发现这个窗口经过的地方都是白的。你的客户不会理解你的程序实际上在很努力的从股票市场获得数据，他们只会很愤怒的说，这个东西把我的机器弄死了！他们的思路被打断了。于是他们不再使用你的程序，你们的合作没了。你没钱了。你的狗也跑了。

出现界面冻结的原因是，耗时操作阻塞了 UI 线程。UI 线程一般负责着渲染界面，响应用户交互，如果这个线程被阻塞，它将无法响应所有的用户交互请求，甚至包括拖拽窗口这样简单的操作。所有的界面框架，无论是 Java/.NET/ActionScript/JavaScript，都只有一个 UI 线程，这个估计永远都不会变。

用户看到的应用通常与程序员大相径庭。用户对应用的期待级别分别是：能用、可用、好用、好看。而我观察到的大多数程序员停留在第一阶段：能用。“一切皆异步”这个原则说来简单，做起来也不会很难。把上面的代码稍作改动，如下：

```
void updateStockDataButton_clicked() {
    runInAnotherThread( function () {
        stockData = stockDataService.getLatest(); // 从远程获取股票信息
        updateUI(stockData); // 这个方法在UI线程更新界面
    }
}
```

注意加粗部分。runInAnotherThread 是跟语言平台特定的。对于.net C#，可以是一个 Dispatcher+delegate 或者 ThreadPool.QueueUserWorkItem；对于 Java，可以干脆是一个 Runnable。对于 AJAX，可以是 XMLHttpRequest 或者把这个计算扔到一个 IFrame 中；对于 ActionScript，似乎没有什么好的方法，把获取数据的部分交给 XML.load 然后通过事件回调的方式来进行界面刷新吧。

耗时操作一般两种来源产生：网络带来的延迟以及大规模运算。两者对应的异步实现方式有所不同。前者往往可以通过特定语言、平台的获取数据的方式来进行异步，特别是缺乏多线程特性的动态语言。例如典型的 AJAX 方式：

```
xhr = new XMLHttpRequest()
xhr.send("POST", '/stockData/MSFT', function() {
    doSomethingWith(xhr.responseText); // 只有当数据返回的时候，才会调用
})
```

大规模运算带来的耗时在 Java/C# 等支持多线程的语言环境中很容易实现，而对于 JavaScript/ActionScript 等很难，折衷的方式是 将复杂运算延迟到服务器端进行；或者将复杂运算拆解成若干个耗时较少的小运算，例如 ActionScript 的伪多线程实现方式。

“一切皆异步”这个原则说来容易，但要在企业应用中以一种一致的方式进行实现很难。上例中 runInAnotherThread 的方式貌似简单，也可能出现在各种 GUI 框架的介绍中，但绝不是一个稍具规模的 RichClient 应当采用的方式。它很难作为一种编程范式被遵循，你绝不会希望看到在你的代码中 所有用到异步的地方都 new Runnable(){...}。这样带来的问题不仅仅是异步被不被管理的到处乱扔，还带来了测试的复杂性。为了解决这些只有在至少有点规模的 RichClient 中才出现的问题，你最好也实现了“4 线程管理”，能够实现“3 事件管理”（见下篇）更好。终极方式是将这些抽象到应用的基础框架中，使得所有的开发人员以一种一致的方式进行编程。

2 视图管理

2.1 视图生命周期管理

视图这个概念在 WEB 开发中几乎被忽略。这里所说的视图是指页面、页面块等界面元素。在 WEB 开发中，视图的生命周期很短：在进入页面的时候创建，在离开页面的时候销毁。一不小心页面被弄糟了，或者不能按照预期的渲染了，点下刷新按钮，整个世界一片清净。

WEB 下的视图导航也是如此自然。基于超链接的方式，每点击一次，就能够打开一个新的

页面，旧的页面被浏览器销毁，新的页面诞生。（这里不考虑 AJAX 或者其他 JavaScript 特效）

如果把这种想法带入到 RichClient 开发，后果会很糟糕。每当点击按钮或者进行其他操作需要导航到新的窗口，你不加任何限制的创建新窗口或者新的视图。然而 CPU 不是无限的。创建一个新的视图通常是很耗 CPU 和内存的。系统响应会变慢。用户会抱怨，拒绝付钱，于是因为饥饿，你的狗再次离开了你。

每次新创建视图产生的严重后果并不仅仅是非功能性的，还包括功能性的缺失。如果你用过 Skype，当你在给张三通话的时候，再次点击张三并且进行通话，你会发现刚刚的通话界面会弹出来，而不是开启新窗口。在我们的一个项目中，有一个功能：点击软件界面上的电话号码就能开启一个新窗口，并直接连到桌上的电话 拨号通话。可以想象，如果每次都会弹出新的窗口，软件的逻辑是根本错误的。

如何解决这个问题？最简单的方式是将所有已知的视图全都保存到本地的一个缓存中，我们命名为 ViewFactory，当需要进行获取某个视图的时候，直接从 ViewFactory 拿到，如果没有创建，那么创建，并放到 Cache 中：

```
class ViewFactory {
    cache = {}
    View getView(Object key) {
        if cache.contains(key) {
            return cache[key]
        }
        cache[key] = createView(key)
        return cache[key]
    }
}
```

需要注意的是，ViewFactory 中 key 的选择。对于简单的应用，key 可以干脆就是某个单独窗口的类名。例如整个系统中往往只有一个配置窗口，那么 key 就是这个类名；对于需要复用的窗口，往往需要根据其业务主键来创建相应的视图。例如代码中只有一个 UserDetailWindow，需要用来展示不同用户的信息。当需要同时显示两个以上的用户信息的时候，用同一个窗口实例显然不对。这时候 key 的选择可以是类名+用户 ID。

2.2 视图导航

上面的方案并没有解决导航的问题。导航需要解决的问题有两个，如何导航以及如何在导航时传递数据。这时候不得不羡慕 WEB 的解决方式。我要访问 ID 为 1 的用户信息，只需要访问类似于 users/1 的页面就好；需要访问搜索结果第 5 页，只需要访问

/search?q=someword&page=5 就好。这里/search 是视图，q=someword 和 page=5 是传递的数据。目前我还没有发现任何一本书来讲述如何进行视图导航。我们的方式是实现一个 Navigator 类用来导航，Navigator 依赖于前面提到的 ViewFactory：

```

class Navigator {

    Navigator(ViewFactory viewFactory) {
        this.viewFactory = viewFactory;
    }

    void goTo(Object viewKey) {
        this.viewFactory.getView(viewKey).show()
    }

}

```

(这个类看起来跟 ViewFactory 没什么大的差别，但他们逻辑上是完全不同，并且下面的扩展中会增强)

这样是可以解决问题的。如果要在不同的视图之间传递数据，只需要对 Navigator.goTo 方法稍加扩展，多添加一个参数就能够传递参数了。例如，在用户列表窗口点击用户名，发送一条消息并打开聊天窗口，可以写为：

```

void messageButton_clicked() {
    Navigator.goTo("ChatWindow#userId", "聊天消息")
}

```

然而这种方式并不完美。当你发现大量的数据在窗口之间交互的时候，这种将主动权交给调用方控制的方式，会给状态同步带来不少麻烦；如果你使用了本地存储，它越过存储层直接与服务器交互的方式也会带来不少的不便之处。更好的方式是使用“3 事件管理”。当然，如果窗口之间导航不存在数据传递，基于 Navigator 的方式仍然简单并且可用。

3 事件管理

事件管理应当是整个 RichClient/RIA 开发中的最难以把握的部分。这部分控制的好，你的程序用起来将如行云流水，用户的思维不会被打断。任何一个做 RichClient 开发的程序员，可以对其他方面毫无所知，但这部分应当非常熟悉。事件是 RichClient 的核心，是“一切皆异步”的终极实现。前面所说的例子，实际上可以被抽象为事件，例如第一个，获取股票数据，从事件的观点看，应该是：

- 开始获取股票数据

- 正在获取股票数据
- 获取数据完成
- 获取数据失败

看起来相当复杂。然而这样去考虑的时候，你可以将执行计算与界面展现清晰的分开。界面只需要响应事件，运算可以在另外的地方悄悄的进行，并当任务完成或者失败的时候报告相应的事件。从经验看来，往往同样的数据会在不同的地方进行不同的展示，例如 skype 在通话的时候这个人 的头像会显示为占线 ,而具体的通话窗口中又是另外不同的展现 ;MSN 的个人签名在好友列表窗口中显示为一个点击可以编辑控件 ,而同时在聊天窗口显示为一个不能点击只能看的标签。这是 RichClient 的特性，你永远不知道同一份数据会以什么形式来展现，更要命的是，当数据在一个地方更新的时候，其他所有 能展现的地方都需要同时做相应的更新。如果我们仍然以第一部分的例子，简单采用 runInAnoterThread 是完全不能解决这个问题的。

我们曾经犯过一些很严重的错误，导致最终即便重构都积重难返。无视事件的抽象带来的影响是架构级别的，小修小补将无济于事。

事件的实现方式可以有很多种。对于没有事件支持的语言，接口或者干脆某一个约束的方法就可以。有事件支持的语言能够享受到好处，但仍然是语法级别的，根本 是一样的。观察者模式在这里很好用。仍然以股票为例，被观察的对象就是获取股票数据对象 StockDataRetriver，观察的就是 StockWindow：

```

StockDataRetriver {
    observers: []

    retrieve() {
        try {
            theData = ...// 从远程获取数据
            observers.each { |o| o.stockDataReady(theData)} // 触发数据获取成功事件
        } catch {
            observers.each { |o| o.stockDataFailed()} // 触发事件获取失败事件
        }
    }
}

StockDataRetriver.observers.add(StockWindow) // 将StockWindow加入到观察者队列

```

```

StockWindow {
    stockDataReady(theData) {
        showDataInUIThread(); // 在UI线程显示数据
    }
    stockDataFailed() {
        showErrorInUIThread(); // 在UI线程显示错误
    }
}
  
```

你会发现代码变得简单。UI 与计算之间的耦合被事件解开，并且区分 UI 线程与运算线程之间也变得容易。当尝试以事件的视角去观察整个应用程序的时候，你会更关注于用户与界面之间的交互。

让我们继续抽象。如果把“获取股票数据”这个按钮点击，让 StockDataRetriver 去获取数据当作事件来处理，应该怎么写呢？将按钮作为被观察者，StockDataRetriver 作为观察者显然不好，好不容易分开的耦合又黏在一起。引入一个中间的 Events 看起来不错：

```

Events {
    listeners: {}

    register(eventId, listener) {
        listeners[eventId].add(listener)
    }

    broadcast(eventId) {
        listeners[eventId].observers.each{|o| o.doSomething()}
    }
}
  
```

Events 中维护了一个 listeners 的列表，它是一个简单的 Hash 结构，key 是 eventId，value 是 observer 的列表；它提供了两个方法，用来注册事件监听以及通知事件产生。对于上面的案例，可以先注册 StockDataRetriver 为一个观察者，观察 start_retrive_stock_data 事件：

```
Events.register('start_retrive_stock_data', StockDataRetriver)
```

当点击“获取股票数据”按钮的时候，可以是这样：

```
Events.broadcast('start_retrive_stock_data')
```

你会发现 StockDataRetriver 能够老老实实的开始获取数据了。

需要注意的是，并非将所有事件定义为全局事件是一个好的实践。在更大规模的系统中，将事件进行有效整理和分级是有好处的。在强型的语言（如 Java/C#）中，抽象出强类型的 EventId，能够帮助理解系统和进行编程，避免到处进行强制类型转换。例如，StockEvent：

```
StockDataLoadedEvent {
```

```

        StockData theData;
        StockDataLoadedEvent(StockData theData);
    }

    Event.broadcast(new StockDataLoadedEvent(loadedData))

```

这个事件的监听者能够不加类型转换的获得 StockData 数据。上面的例子是不支持事件的语言，C#语言支持自定义强类型事件，用起来要自然一些：

```
delegate void StockDataLoaded(StockData theData)
```

事件管理原则我相信并不难理解。然而困难的是具体实现。对一个新的 UI 框架不熟悉的时候，我们经常在“代码的优美”与“界面提供的特性”之间徘徊。实现这样的一个事件架构需要在项目一开始就稍具雏形，并且所有的事件都有良好的命名和管理。避免在命名、使用事件的时候的随意性，对于让代码可读、应用稳定有非常大的意义。一个好的事件管理、通知机制是一个良好 RichClient 应用的根本基础。一般说来，你正在使用的编程平台如 Swing/WinForm /WPF/Flex 等能够提供良好的事件响应机制，即监听事件、onXXX 等，但一般没有统一的事件的监听和管理机制。对于架构师，对于要使用的编程平台 对于这些的原生支持要了熟于心，在编写这样的事件架构的时候也能兼顾这些语言、平台提供给你的支持。

采用了事件的事件后，你不得不同时实践“线程管理”，因为事件一般来说意味着将耗时的操作放到别的地方完成，当完成的时候进行事件通知。简单的模式下，你可以在所有需要进行异步运算的地方，将运算放到另外一个线程，如 ThreadPool.QueueUserWorkItem，在运算完成的时候通知事件。但从资源的角度考虑，将这些线程资源有效的管理也是很重要的，在“线程管理”部分有详细的阐述。另外，如果能将你的应用转变为 数据驱动的，你需要关注“缓存以及本地存储”。

4 线程管理

在 WEB 开发几乎无需考虑线程，所有的页面渲染由浏览器完成，浏览器会异步的进行文字和图片的渲染。我们只需要写界面和 JavaScript 就好。如果你认同“一切皆异步”，你一定得考虑线程管理。

毫无管理的线程处理是这样的：凡是需要进行异步调用的地方，都新起一个线程来进行运算，例如前面提到的 runInThread 的实现。这种方式如果托管在 在“事件管理”之下，问题不大，只会给测试带来一些麻烦：你不得不 wait 一段时间来确定是否耗时操作完成。这种方式很山寨，也无法实现更高级功能。更好的方式是将这些线程资源进行统筹管理。

线程的管理的核心功能是用来统一化所有的耗时操作，最简单的 TaskExecutor 如下：

```

TaskExecutor {
    void pendTask(task) { //task: 耗时操作任务
        runInThread {
            task.run(); // 运行任务
        }
    }

    RetrieveStockDataTask extends Task {
        void run() {
            theData = ... // 直接获取远程数据，不用在另外线程中执行
            Events.broadcast(new StockDataLoadedEvent(theData)) // 广播事件
        }
    }
}

```

需要进行这个操作的时候，只需要执行类似于下面的代码：

```
TaskExecutor.pendTask(new RetrieveStockDataTask())
```

好处很明显。通过引入 TaskExecutor，所有线程管理放在同一个地方，耗时操作不需要自行维护线程的生命周期。你可以在 TaskExecutor 中灵活定义线程策略实现一些有趣的效果，如暂停执行，监控任务状况等，如果你愿意，为了更好的进行调试跟踪，你甚至可以将所有的任务以同步的方式执行。

耗时任务的定义与执行被分开，使得在任务内部能够按照正常的方式进行编码。测试也很容易写了。

不同的语言平台会提供不同的线程管理能力。.NET2.0 提供了 BackgroundWorker，提供了一系列对多线程调用的封装，事件如开始调用，调用，跨线程返回值，报告运算进度等等。它内部也实现了对线程的调度处理。在你要开始实现类似的 TaskExecutor 时，参考一下它的 API 设计会有参考价值。Java 6 提供的 Executor 也不错。

一个完善的 TaskExecutor 可以包含如下功能：

- Task 的定义：一个通用的任务定义。最简单的就是 run()，复杂的可以加上生命周期的管理：start()、end()、success()、fail()..取决于要控制到多么细致的粒度。
- pendTask，将任务放入运算线程中
- reportStatus，报告运算状态

- 事件：任务完成
- 事件：任务失败

写这样的一个线程管理的不难。最简单的实现就是每当 pendTask 的时候新开线程，当运算结束的时候报告状态。或者使用像 BackgroundWorker 或者 Executor 这样的高级 API。对于像 ActionScript/JavaScript 这样的，只能用伪线程，或者干脆将无法拆解的任务扔到服务器端完成。

5 缓存与本地存储

纯粹的 B/S 结构 浏览器不持有任何数据，包括基本不变的界面和实际展现的数据。RichClient 的一大进步是将界面部分本地持有，与服务器只作数据通讯，从而降低数据流量。像《魔兽世界》10 多 G 的超大型客户端，在普通的拨号网络都可以顺畅的游戏。

缓存与本地存储之间的差别在于，前者是在线模式下，将一段时间不变的数据缓存，最少的与服务器进行交互，更快的响应客户；后者是在离线模式下，应用仍然能够完成某些功能。一般来说，凡是需要类似于“查看 xxx 历史”功能的，需要“点击列表查看详细信息”的，都会存在本地存储的必要，无论这个功能是否需要向 用户开放。

无论是缓存还是本地存储，最需要处理的问题如何处理本地数据与服务器数据之间的更新机制。当新数据来的时候，当旧数据更新的时候，当数据被删除的时候，等等。一般来说，引入这个实践，最好也实现基于数据变化的“事件管理”。如果能够实现“客户机-服务器数据交互模式”那就更完美了。

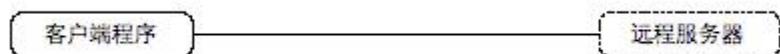
我们犯过这样一个错误。系统启动的时候，将当前用户的联系人列表读取出来，放到内存中。当用户双击这个联系人的时候，弹出这个联系人的详细信息窗口。由于 没有本地存储，由于采用了 Navigator 方式的导航，于是很自然的采用了 Navigator.goTo('ContactDetailWindow', theContactInfo)。由于列表页面一般是不变的，因此显示出来的永远是那份旧的数据。后来有了编辑联系人信息的功能，为了总是显示更新的数据，我们将调用更改为 Navigator.goTo('ContactDetailWindow', 'contactId')，然后在 ContactDetailWindow 中按照 contactId 把联系人信息重新读取一次。远在南非的用户抱怨慢。还好我没养狗，没有狗离开我。后来我们慢慢的实现了本地存储，所有的数据读取都从这个地方获得。当数据需要更新的时候，直接更新这个本地存储。

本地存储会在根本上影响 RichClient 程序的架构。除非本地不保存任何信息，否则本地存储一定需要优先考虑。某些编程平台需要你在本地存储界面和数据，如 Google Gears 的本地

存储，置于 Adobe Air 的 AJAX 应用等，某些编程平台只需要存储数据，因为界面完全是本地绘制的，如 Java/JavaFX/WinForm/WPF 等。缓存界面与缓存 数据在实现上差别很大。

本地存储的存储机制最好是采用某一种基于文件的关系数据库，如 SQLite、H2 (HypersonicSQL)、Firebird 等。一旦确定要采用本地存储，就从成熟的数据库中选择一个，而不要尝试着自己写基于文件的某种缓存机制。你会发现到最后你实现了一个山寨版的数据库。

在没有考虑本地存储之前，与远端的数据访问是直接连接的：



我们上面的例子说明，一旦考虑使用本地存储，就不能直接访问远程服务器，那么就需要一个中间的数据层：



数据层的主要职责是维护本地存储与远程服务器之间的数据同步，并提供与应用相关的数据缓存、更新机制。数据更新机制有两种，一种是 Proxy (代理) 模式，一种是自动同步模式。

代理模式比较容易理解。每当需要访问数据的时候，将请求发送到这个代理。这个代理会检查本地是否可用，如果可用，如缓存处于有效期，那么直接从本地读取数据，否则它会真正去访问远端服务器，获取数据，更新缓存并返回数据。这种手工处理同步的方式简单并且容易控制。当应用处于离线模式的时候仍然可以工作的很好。



自动同步模式下，客户端变成都针对本地数据层。有一个健壮的自动同步机制与服务器的保持长连接，保证数据一直都是更新的。这种方式在应用需要完全本地可运行的时候工作的非常好。如果设计得好，自动同步方式健壮的话，这种方式会给编程带来极大的便利。



说到同步，很多人会考虑数据库自带的自动同步机制。我完全不推荐数据库自带的机制。他们的设计初衷本身是为了数据库备份，以及可扩展性 (Scalability) 的考虑。在应用层面，

数据库的同步机制往往不知道具体应用需要进行哪些数据的同步，同步周期等等。更致命的是，这种机制或多或少会要求客户端与服务器端具备类似的数据表结构，迁就这样的设计会给客户端的缓存表设计带来很大的局限。另外，它对客户机-服务器连接也存在一定的局限性，例如需要开放特定端口，特定服务等等。对于纯粹的 Internet 应用，这种方式更是完全不可行的，你根本不知道远程数据库的结构，例如 Flickr, Google Docs.

当本地存储+自动同步机制与“事件管理”都实现的时候，应用会是一种全新的架构：基于数据驱动的事件结构。对于所有本地数据的增删改都定义为事件，将关心这些数据的视图都注册为响应的观察者，彻底将数据的变化于展现隔离。界面永远只是被动的响应数据的变化，在我看来，这是最极致的方式。

结尾

限于篇幅，这篇文章并没有很深入的讨论每一种原则/实践。同时还有一些在 RichClient 中需要考虑的东西我们并没有讨论：

- 纯 Internet 应用离线模式的实现。像 AdobeAir/Google Gears 都有离线模式和本地存储的支持，他们的特点是缓存的不仅仅是数据，还包括界面。虽然常规的企业应用不太可能包含这些特性，但也具备借鉴意义。
- 状态的控制。例如管理员能够看到编辑按钮而普通用户无法看见，例如不同操作系统下的快捷键不同。简单情况下，通过 if- else 或者对应编程平台下提供的绑定能够完成，然而涉及到更复杂的情况时，特别是网络游戏中大量互斥状态时，一个设计良好的分层状态机模型能够解决这些问题。如何定义、分析这些状态之间的互斥、并行关系，也是处理超复杂
- 测试性。如何对 RichClient 进行测试？特别是像 WPF、JavaFX、Adobe Air 等用 Runtime+ 编程实现的框架。它们控制了视图的创建过程，并且倾向于绑定来进行界面更新。采用传统的 MVP/MVC 方式会带来巨大的不必要的工作量（我们这么做过！），而且测试带来的价值并没有想象那么高。
- 客户机-服务器数据交互模式。如何进行客户机服务器之间的数据交互？最简单的方式是类似于 Http Request/Response。这种方式对于单用户程序工作得很好，但当用户之间需要进行交互的时候，会面临巨大挑战。例如，股票代理人关注亚洲银行板块，刚好有一篇新的关于这方面的评论出现，股票代理人需要在最多 5 分钟内知道这个消息。如果是 Http Request/Response，你不得不做每隔 5 分钟刷一次的蠢事，虽然大多数时候都不会给

你数据。项目一旦开始，就应当仔细考虑是否存在这样的需求来选择如何进行交互。这部分与本地存储也有密切的关系。

- 部署方式。RichClient 与 B/S 直接最大的差异就是，它需要本地安装。如何进行版本检测以及自动升级？如何进行分发？在大规模访问的时候如何进行服务器端分布式部署？这些问题有些被新技术解决了，例如 Adobe Air 以及 Google Gears，但仍然存在考虑的空间。如果是一个安全要求较高的应用，还需要考虑两端之间的安全加密以及客户端正确性验证。新的 UI 框架层出不穷。开始一个新的 RichClient 项目的时候，作为架构师/Tech Lead 首先应当关注的不是华丽的界面和效果，应当观察如何将上述原则和时间华丽的界面框架结合起来。就像我们开始一个 web 项目就会考虑 domain 层、持久层、服务层、web 层的技术选型一样，这些原则和实践也是项目一开始就考虑的问题。

感谢

感谢我的同事周小强、付莹在我写作过程中提供的无私的建议和帮助。小强推荐了介绍 Google Gears 架构的链接，让我能够写作“本地存储”部分有了更深的体会。

这篇文章是我近两年来在 RichClient 工作、网络游戏、WebGame 众多思考的一个集合。我尝试过 JavaFX/WPF/AdobAir 以及相关的文章，然而大多数的例子都是从华丽的界面入手，没有实践相关的内容。有意思的是《大型多人在线游戏开发》这本书，给了我在企业 RichClient 开发很多启发。我们曾经犯了很多错误，也获得了许多经验，以后我们应当能做得更好。

参考

- [.NET 的 BackgroundWorker 定义](#)
- [ActionScript 的伪线程](#)
- [Google Gears 架构](#)

相关阅读

[ThoughtWorks 实践集锦 (1)] [我和敏捷团队的五个约定。](#)

[ThoughtWorks 实践集锦 (2)] [如何在敏捷开发中做好数据迁移。](#)

[ThoughtWorks 实践集锦 (3)] [RichClient/RIA 原则与实践 \(上 \)](#)

关于作者

陈金洲，Buffalo Ajax Framework 作者，ThoughtWorks 咨询师，现居北京。目前的工作主要集中在 RichClient 开发，同时一直对 Web 可用性进行观察，并对其实现保持兴趣。

原文链接：<http://www.infoq.com/cn/articles/thoughtworks-practice-partiii>、
<http://www.infoq.com/cn/articles/thoughtworks-practice-partiii-ii>

相关内容：

- [简化异步操作（上）：使用CCR和AsyncEnumerator简化异步操作](#)
- [简化异步操作（下）：构建AsyncTaskDispatcher简化多个异步操作之间的协作调用](#)
- [敏捷实践的秘密——ThoughtWorks文集II](#)
- [雾里看花：微软的前端技术战略何去何从？](#)
- [节选与书评：巨富客户端—桌面Java应用的动画与图形特效开发](#)

设计者-开发者工作流中的迭代模式

作者 [Doug Winnie](#) 译者 [罗小平](#)

设计者-开发者工作流 (designer-developer workflow) 这个词已经流行了好几年。它描述了设计人员、开发人员在为 Web 或桌面应用创造交互体验过程中的关系，而没有表达出设计者、开发者之间的交互和协作。工作流这个术语让我们觉得这种关系是线性的，但实际上它不是。身为 Adobe 的产品部门经理的作者根据自己的实际工作经历，讲述了一些可在开发和设计工作中应用的迭代模式和如何利用这些迭代模式实现团队内的高效沟通。

设计者-开发者工作流 (designer-developer workflow) 这个词已经流行了好几年。它描述了设计人员、开发人员在为 Web 或桌面应用创造交互体验过程中的关系，而没有表达出设计者、开发者之间 的交互和协作。工作流这个术语让我们觉得这种关系是线性的，但实际上它不是。

在项目的整个生命周期中，我们会不停地为项目增加内容。项目本身从开始到结束可能是线性的，但项目参与者之间的协作不是。需要协作的项目，不会变成一个装配流水线；在项目结束之间，我们每个参与者都可能要往项目中不停地添加各种组件、功能片、代码和设计方案。这个过程是有机的，而且——更重要是迭代式的。

作为 Adobe 的产品部门经理，我经常要和构建各种交互应用、内容的设计和开发人员一起工作。在此过程中我常听到的一点，就是“团队成员间的工作流程，是项目成功的关键；有效减少团队可能遇到的困难的方法是保证项目中每个人之间的清晰、高效沟通”。

在本文中，我将讨论一些可在开发和设计工作中应用的迭代模式，并说明如何利用这些迭代模式实现团队内的高效沟通。

迭代

所谓迭代开发，可定义为全程功能构建。比如在一个项目中，逐步增加各种新特性、交互体验、特性提升和新功能——每次只增加一项。

我们以开发一个简单的游戏为例。游戏的一个关键特性是记录玩家的分数。而这个特性的最

初版本，可能就只是通过调用计分系统的功能为用户增加分数。计分系统除了按给定点数未用户加分，没有其他任何功能，目的非常简单。在项目全过程中，我们可以逐步演进和完善计分系统，每次增加一个小的特性或功能，即每次就是一个迭代。对于这里的计分系统而言，我们可以通过迭代逐步增加的各项功能和特性大致有：

- 通过对计分系统的调用，实现每次按一个点数为某玩家计分
- 通过对计分系统的调用，实现每次按个数不定的一组点数为某玩家计分
- 分数达到预先固定的满分时，让计分系统通知游戏
- 在计分系统创建时，可自定义满分分数
- 为多个玩家创建多个计分系统实例

作为开发人员，我可以此作为开发某个组件的“特性路标图”，也可用于在完成独立的每个步骤后，标示下一步骤的工作。这样，即使项目周期很长，无论何时我们都能知道自己身在何处，从而可专注于每个独立特性的构建。以这种方式设计整个开发过程，可保证我们不会在未来某个时候遗漏任何工作。

对于大型项目，我可能没有足够时间去定义一份完整的应用规格文档。但我知道大体上可分为哪些部分，并从中选择简单的入手，比如一套 Adobe Flex 组件，我可以先将这些最基础的组件开发出来。在接下来的每个独立的开发步骤中，我可以扩展已完成功能片的功能，并按我最终希望的形式和作用的要求将它们整合在一起。在此之后，我可以如法炮制对付这个大应用中的其他类似或差别不大的组件。

如此这般，最终我可以构建出所有功能模块，然后将它们集成，从而形成整个应用。这些模块的组织集成也可以用迭代方法完成，通过事件、监听器，逐步在这些模块间构建出通讯桥梁。

设计

迭代如何应用于设计呢？最简单的答案是有两种应用途径——你或许不能立刻想到。

首先，在做系统的整体设计，我从基本构建块（building block）入手：在哪里实现导航？主要内容放在哪里？应用中的这个或那个功能安排什么地方？所有基本构建块一起组成了套件的整体框架（有关终端用户如何使用套件、应用或内容的结构、方法的总体设计）。

当你将系统的主内容块（如导航、部件 A/B/C）定下来后，设计过程就开始了。在这个过程

中，每完成一个基本构建块的设计，就是一次迭代。在整个过程中，你可以逐步演进和完善自己的设计意图。

第二个方法应用在我已经完成了应用的整体结构，准备在此基础上进行可视化设计的时候。此时用户界面已经确定，结构中每个离散的元素都可以取出来独立设计。在设计了整体结构后再设计其组成元素是非常重要的，因为你需要知道每个组成元素在整个大套件或应用中的运行环境。每个独立的功能模块与其他模块如何交互，将决定该套件或应用的用户界面设计是相对固定还是相对可变化。

比如前面举到的导航例子，我将迭代设计此组件，最开始使用其缺省状态，然后再扩展引入其他元素。假设在我的游戏中需要一个菜单栏。在设计它时，我可以迭代式设计各种特性，具体可包括：

1. 设计初始导航状态，不支持鼠标交互。
2. 支持鼠标悬浮移动。
3. 支持鼠标悬浮移动和提示框。
4. 支持鼠标点击一级导航元素。
5. 支持二级导航。
6. 在二级导航中支持鼠标悬浮移动。
7. 实现完整导航结构。

在每一步骤中，组件（本例中即菜单）的设计，都需要考虑与导航结构协作；同时，它需工作在整个套件或应用的环境中，因此必须保证其设计和交互流程与该环境适配。

导航结构完成后，你就可以想办法对付下一个组件元素了。

回退

迭代式开发和设计的一大好处是当我走远或走歪的时候，可以清晰回溯到还原点纠正错误，继续前进。

比如，我们现在需要游戏中的计分系统支持减分，就可以倒回去增加这项功能。对于更复杂的项目来说，即使某些基础性的东西需要变化，我们也能将项目回退可实现此变化的状态，然后修改、重新将它引入原模块，并修复任何可能出现的集成问题。

以前面的导航结构为例，如果发现需要增加三级导航，我可以回到构建二级导航的地方，并添加三级导航。如果发现三级导航不能和该套件或应用的整体环境较好的协同工作，我可以回溯到更远点，将二级导航返工，然后再引入三级导航，并使其生效。

对于多数设计和开发人员而言，这些听起来有点老生常谈，但的确都是指导我们如何工作、如何向项目持续增加内容的基本方法。即使在软件开发领域之外的设计准则中，它往往也是适用的。例如视频编辑就是一个典型的迭代式设计实践。编辑人员每段时间只处理一个视频片段，然后是场景，再然后才是整个视频。从单独元素开始，逐步让其成长，容纳更多元素，这是我们在大型项目中采用的最基本的工作方法。

迭代离不开沟通

在迭代模式中，无论是设计人员还是开发人员，都会面临一个难题：每个成员如何与正在构建套件或应用中的小组中其他专业的成员实现有效沟通？

答案之一是在每步迭代中向全体成员广播信息——但无疑只有每次迭代在粒度上得到了充分划分，这种沟通才会产生作用。此外还要求它与项目存在相关性。当然在项目的当前时候，它不可能总是相关的；但在开发过程中的未来某个时候，它必定又是有价值的。这样，团队成员在整个过程中都可以获得他们需要的信息。

灵活性，是迭代开发和设计的一大要点。团队所有成员都向迭代过程贡献自己的成果，因此他们的工作必须是灵活的、开放的，只有这样，过程产出的各种组件、设计方案和代码最终才能有效集成。要想迭代设计和开发最后取得成功，那么所有参与者就必须在项目中取得共识。

很多时候，应用的设计和开发人员最开始都会认真制定详细规格书，并以此为基础开始工作。但不久，他们就会按照每个人自己的想法“私奔”了。多数情况下，规格书并不能容纳应用中全部设计和功能用例，因此在执行过程中，他们有为满足需求而自行改编的倾向。但问题是设计和开发彼此分离，互不依赖，那么最后碰头时，必然发现互不兼容，必须予以修改才能解决二者之间的差异。

规定统一的沟通语言，是项目第一步。在项目开始的时候，设计和开发团队必须就项目的主要组件达成一致。详细规定这些组件的功能和设计或许并非必需，但在它们的主要方面达成一致至关重要。例如某应用程序包含一个菜单、某种聊天功能和一些对象。团队必须就应用中的命名规则、基本组件的作用、以及每个主要组件的主要目标达成一致。

完成了这些要素的定义后，团队成员就可以开始迭代工作了。开发和设计人员可以将这些定

义作为他们不断开展迭代的基础，在每步工作的同时，规划出下一步骤。如果功能发生变化，可以在未来的迭代步骤做出调整以适应变化了的需求。

如在一个迭代步骤中，不同部件间出现了冲突，参与者可以通过沟通解决这些问题并继续前进，无需将整个组件、套件或应用返工。

组织与管理

有多种技术和系统可帮助设计和开发人员组织和管理迭代。对开发人员而言，必不可少的工具就是代码版本控制系统，它负责将每个迭代过程形成的代码予以保存，供未来使用，开发人员可以根据要求返回到先前的任意迭代阶段。

对于设计人员而言，像 Adobe Version Cue 这样的设计档案管理系统，能为迭代开发提供重要的版本管理能力。此外，在团队范围内采用某种统一的文件命名规范也会大有好处。像 Subversion 这样的代码存储系统，也可用于设计迭代过程。

对于团队来说，建立 Wiki、内部开发博客或其他类似的沟通工具，帮助团队成员自动实现信息的收集和分发，对整个团队的沟通效果也有极大帮助。

不过，最关键的一点，还是无论你们选择什么工具，整个团队都必须要使用这些工具。否则，工具将不存在任何意义。如果一个团队成员只顾干自己的，不利用这些工具跟踪其他成员的迭代成果，他们最后开发出来的模块将被弃用，或在集成时出现问题。

总结

研究并和你的团队成员讨论文中提到的技术，并确定哪项技术最适用于你的团队，这是要做的第一步工作。要找到这项技术，需要打开心扉，仔细思考各种可能和以前可能从未入过你眼的工具。再次强调，这些技术的最终目标是让你的团队良好工作，就像我前面提到过的那样，团队成功沟通和协作，是项目走向成功的秘诀。

若需了解设计人员-开发人员工作流的更多信息，请参考 Fireworks Developer Center 中 [design/development](#)、[iterative prototyping](#) 等相关文章，以及 Adobe Labs 的 [Flash Catalyst](#)、[Flash Builder](#)。

作者简介

Doug Winnie：Adobe Systems 公司工作流产品部门经理，致力于 Adobe 产品、平台和技术间

的流程协作。在加入 Adobe 之前，Doug 负责过设计人员和用户体验开发组织的领导工作。他热衷于 Flash、Flex 和 Dreamweaver 等 Web 应用和平台方面的开发工作。他的博客是：<http://www.adobe.dougwinnie.com/>。

原文链接：<http://www.infoq.com/cn/articles/designer-developer-workflow>

相关内容：

- [剖析短迭代](#)
- [RIA领域的设计开发流程](#)
- [Java应用开发中代码生成工具的作用](#)
- [用于Flex和Java快速开发的Flamingo项目](#)



富互联网应用之美

——Rich Internet Application

策划：霍泰稳

执行：葛明见

美术编辑：胡伟红

本迷你书主页为

<http://www.infoq.com/cn/minibooks/beautiful-ria>

本书属于 InfoQ 企业软件开发丛书。

如果您打算订购 InfoQ 的图书，请联系 books@c4media.com

未经出版者预先的书面许可，不得以任何方式复制或者抄袭本书的任何部分，本书任何部分不得用于再印刷，存储于可重复使用的系统，或者以任何方式进行电子、机械、复印和录制等形式传播。

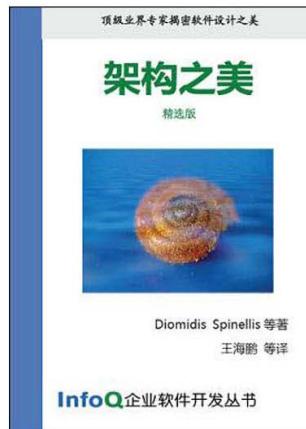
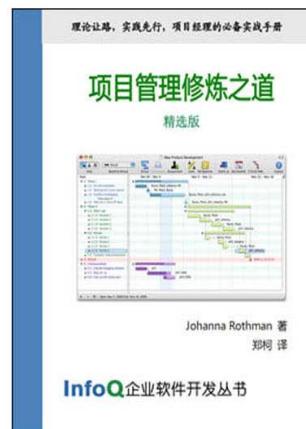
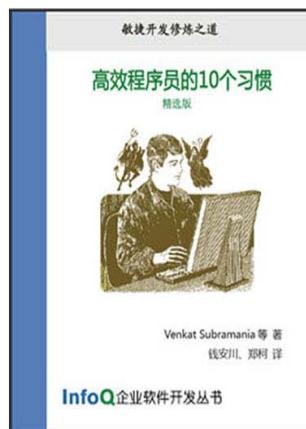
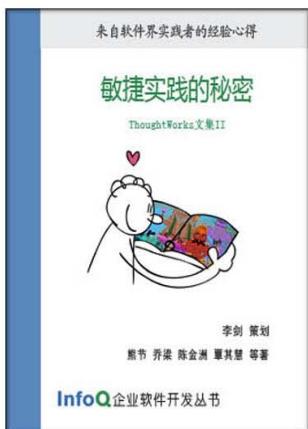
本书提到的公司产品或者使用到的商标为产品公司所有。

如果读者要了解具体的商标和注册信息，应该联系相应的公司。

欢迎共同参与 [InfoQ 中文站](#) 的内容建设工作，包括原创投稿和翻译等，请联系 editors@cn.infoq.com。

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