

Open Source 20大高手倾力奉献

开源技术选型手册

精选版



《开源技术选型手册》编委会 著

InfoQ 企业软件开发丛书

免费在线版本

(非印刷免费在线版)

[登录China-Pub网站购买此书完整版](#)



电子工业出版社

了解本书更多信息请登陆[博文视点官方博客](#)

InfoQ 中文站出品

InfoQ 中文站
www.infoq.com/cn

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。

本书主页为

<http://infoq.com/cn/minibooks/open-source-tools-choice>

© 2008 C4Media Inc.

版权所有

C4Media 是 InfoQ.com 这一企业软件开发社区的出版商

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。

本书在征得博文视点出版公司许可下制作，以电子文档形式发布。

作者：《开源技术选型手册》编委会

特约策划：霍泰稳

责任编辑：杨绣国

美术编辑：杨小勤

封面设计：吴志民

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。

10 9 8 7 6 5 3 2 1

编委会成员

总策划：霍泰稳

编 委：

(以下排名不分前后，按姓氏拼音排序)

蔡玉宝

程 勇

丁雪丰

高 昂

郭晓刚

何伟平

胡长城

胡 键

霍泰稳

柯自聰

李 剑

林 煦

刘长炯

莫 映

吴 眇

邢波涛

杨 涌

袁 峰

张凯峰

张 远

《开源技术选型手册》编委会成员给灾区人民的祝福：

在本书即将付印的时候，在中国西南的四川汶川县附近发生了 8.0 级的强烈地震，到目前为止已经有五万余同胞遇难。一方有难，八方支援，在这场灾难面前，相信每一个有良知的人都不会无动于衷。经《开源技术选型手册》编委会倡议，作者同意本书的绝大部分版税将通过中国红十字总会捐献给四川地震灾区。另外本书作者谨向灾区的死难同胞表示沉痛的哀悼！对伤者和失去亲人、失去家园的受难者表示最诚挚的慰问和祝福！

你的爱，我的爱，汇成一条爱的河流，愿这爱河中的水能冲淡灾区人民的伤痛。

——霍泰稳

天地无情，人间有爱；重建家园，指日可待。

——吴昕

开源为软件开发做贡献，手册为思想传播而努力，捐款为抗震救灾尽一份力。

——蔡玉宝

此时此刻，任何华丽的词藻都显得苍白无力。你们已经经历了地震带来的苦难，愿你们能早日走出阴影，重建幸福的家园。

——胡键

努力把自己手头的工作做好是我们每个普通的人对灾区人民的最大支持。全国人民永远和你们在一起！

——何伟平

一方有难、八方支援，互帮互助，共度难关！祝愿灾区兄弟姐妹们早日重建家园。

——高 昂

衷心祝福灾区人民早日走出苦难，中国，加油！

——林昊

面对灾难，我们同心协力，永不言弃，祝灾区人民早日重建家园！

——丁雪丰

灾区人民的痛楚，我们感同身受。中国加油，中华民族加油！

——柯自聪

《开源技术选型手册》面市之日，也是四川汶川地震后百废待兴重建家园之时，此为广大软件技术从业者共勉。

——张凯峰

爱让我们在一起。

——李剑

爱的烛光围绕在我们每一个人身旁，因为我们是一家人；愿灾区的兄弟姐妹能够早日走出伤痛的阴影，重建我们的家园。

——胡长城（银狐 999）

把自己的事情做好，做一个有责任心的公民回报社会。

——刑波涛

路在脚下，我们和你在一起。

——袁峰

为了灾区孩子的今天和明天，让我们大家贡献一份爱心！

——程勇

有难同当！

——郭晓刚

我们与你同行，祝愿地震灾区的同胞们早日抚平心灵的创伤，重建美好的家园，希望就在前方。

——莫映

愿灾区人民早日恢复正常的生活。

——张远

地震无情人有情，灾区的同胞，你们并不孤单，全国人民和你们在一起，让我们一起努力，共同渡过这次难关，建设更加美好的家园！

——杨泳

祝福灾区人民早日脱险，重建家园。唯愿天灾永不至，四海九州享平安！

——刘长炯

推荐序

开源在中国——机会和挑战

1999 年，Eric S. Raymond 在自由软件的范畴下提出了 Open Source 开源软件的概念，并出版了《大教堂和集市》(The Cathedral & the Bazaar)一书，这是开源软件发展的标志事件。

近 10 年过去了，开源软件已成为软件行业，特别是互联网行业最重要和发展最快的领域，著名开源项目网站 SourceForge 在 1999 年还只有数百个开源项目，到 2008 年初，其开源项目数已经超过 17 万个，几乎覆盖软件应用的所有领域。大部分产品和技术基于开源平台的 Google 已成为世界上最成功的高科技公司。今天，从全球 500 强企业到中小企业，还有绝大部分的软件公司都在使用开源产品构建自己的信息系统或产品。

为什么开源能取得如此大的成功？

第一：开源社区的协同模式改变和颠覆了软件业的工作方式，可以创造出高质量的软件产品。Raymond 在《大教堂和集市》一书中有关于精彩的阐述。

第二：开源软件的商业模式也得到了验证，RedHat、MySQL、Asterisk 都是非常成功的开源公司。当然最重要的是，开源软件真正释放了软件开发人员的创造力和生产力，全世界有上百万的程序员在参与开源软件的产品开发。国际软件巨头也纷纷加入开源阵营，IBM 投资 10 亿美元支持 Linux，Sun 也斥巨资收购 MySQL，开源在全球的发展成势不可挡之势。

但实际上，中国的开源却是这样一副景象：社区冷，企业热，使用热，开发冷。

使用开源产品的公司和人员众多，但开源社区的发展并不顺利，真正参与开源产品开发和社区贡献的开发人员非常少。

原因是什么呢？《程序员》杂志社组织过专门的研讨，总结下来有多方面的原因：

1. 语言的障碍，阻碍了中国软件开发人员参与国际开源社区；

1. 东西方文化的差异及对开源文化的了解不足；
2. 经济上的快速发展带来的工作和生活压力；
4. 中国软件开发发展的时间还不长，核心开发人员的积累还不够，缺乏开源关键人物；
5. 大学教育在开源领域严重不足，教师也缺乏了解。

开源在中国的发展意义重大：第一，开源软件集合了全世界软件技术的发展精华，可以让我们的开发人员和软件企业充分学习和吸收；第二，开源软件覆盖了软件应用各个领域，中国的软件企业可以在此基础上发展增值应用，只要遵守其商业规则，就能创造出商业价值；第三，软件的用户都更愿意自己用的系统是开源的，这样便于维护升级和系统整合。

开源给中国的软件产业提供了加入国际软件大潮的极好机会，事实上中国互联网的蓬勃发展也是基于开源的产品带来的成效。但更重要的是中国的软件开发人员能够参与开源社区和产品开发，打造中国开源社区的良性生态系统。

美国开源发达的一个重要原因是出版界（特别是 O'Reilly 公司）出版了大量的开源图书，推进了开源技术的普及和发展。中国软件界如果有更多高质量的中文开源内容，就会有更多的人了解开源的技术和价值，就会有更多的开发人员参与开源的使用和开发，就会有企业采用开源的应用，反过来就会刺激更多的开发人员学习和参与开源。

所以，博文视点这次组织中国开源社区的多位专家策划《开源技术选型手册》是非常有意义的。

开源有着美丽的风景，但也非常庞杂，超过 17 万个项目无所不包。《开源技术选型手册》是一本开源技术和产品的导航手册，本书的作者都是涉足开源多年的专家，他们分门别类地为广大读者介绍了 19 类开源技术和产品，开源的最大魅力在于提供给软件开发人员自由选择和使用的权利，读者按图索骥就可以参与开源项目，参与和回馈是开源成功的关键。衷心希望作者的努力能帮助更多读者的参与。

开源在中国有很大的机会和前景，希望我们的开发人员和软件企业能把握机会，去迎接这样的挑战。

CSDN 总裁 蒋 涛

2008 年 5 月

序

经过近半年的努力，也经历了很多先前所没有预料到的困难，《开源技术选型手册》这本书终于告一段落。虽然仍然有诸多的缺憾，但我们把它们称之为“遗憾之美”，就像软件开发中的建模一样，我们相信目前这个版本是可用的，而将来如果有机会会加以完善。

版式定义

针对本书面向中高端技术人员选型手册的定位，我们将本书的版式进行了严格统一，在每一个篇章开始部分你会先看到一个关于该领域技术的综述，从中可以了解到该领域的过去、现在和未来；然后在对该领域单项技术的探讨中，你会看到该技术的活跃度，了解到它在社区中是否很受欢迎，文档是否齐全等；通过简介和上手指南，你可以简单了解到该技术是什么，主要解决什么问题，使用是否方便等；参考资料一般包括网络和图书两部分，你可以了解到目前社区对该技术的支持；最后的社区观点是使用者对该技术的评价，有的评论来自于社区大牛，有的来自一线开发人员，它们可以作为你技术选型的有力参考。

开源共享

从 2007 年开始 Java 就开始了自己的开源之路，而且在 2008 年的 JavaOne 大会前夕，Sun 公司表示他们正在为 Java 的彻底开源做最后努力；在不久前，Sun 还成功完成了对 MySQL 数据库的收购，而且也表示会继续将 MySQL 开源；在 Java 开源技术领域里，IBM 也绝对是一个不能被忽略的角色，在很多开源软件（如 Eclipse 等）里都可以看到它的痕迹，微软现在允许并支持其他语言基于 CLR 平台进行开发，尤其是多用于 Web 开发的动态语言，如 Ruby 和 Python 等，另外微软非官方的开源社区也非常活跃，许多知名的 Java 社区开源软件都有对应的 .NET 产品，如 NHibernate、NUnit 等。当然不能排除上述这些都是大公司的商业策略，但我更倾向于将它们看成是开源社区多年斗争而取得的胜利。

在这些开源软件的开发者中，也不乏中国技术人员的名字，但也应该清醒地认识到我们的很多参与还是在外围打转。在 2008 年 1 月份，Linux 内核的创始人 Linus Torvalds 在接受媒体采访时表示，“虽然亚洲国家有大量的互联网应用，并拥有许多教育和培训机构，但他们并没有为 Linux 内核和其他开源项目做出太多贡献。”近期社区有人说技术人现在也开始玩“政治”了，由于某种原因 PHP 官方网站上的中文文档全部被删除，一时激起民愤无数，但据内部人士透露，真实的原因是这些文档常年没有更新，已经无法跟上 PHP 新版本的需要，官方才决定删除……往事总是能让人警醒，希望在将来我们的开发人员不仅能积极参与技术文档的翻译，还能够参与到核心的技术研发中去。路漫漫其修远兮，激励我们艰难地去求索！

开放合作

曾几何时，在制造业我们习惯了看到别人好的东西，迅速拿过来模仿，然后大规模地复制，将老师“扼杀于摇篮”；曾几何时，在软件业我们习惯了合理地利用开源协议，心安理得地进行二次开发，然后紧紧地将“劳动成果”捂住并申请专利，将开源在自己这儿止步。这种情况应该不是从鲁迅先生当年提出“拿来主义”开始的，我在想如果鲁迅先生生活在今天，而且恰好是一个软件开发人员的话，他对开源会仅仅是“拿来主义”和“捂住主义”吗？他会不会举起开源的大旗，号召大家都来使用开源软件，但更要积极地参与开源软件的核心研发，输出自己的智慧呢？

合作才是永远利己的绝佳策略。在开源软件领域，我们长久以来奉行的是“拿来主义”，我想等我们意识到“输出主义”更能够创造价值，更有利于自己和整个社区的发展时，那就是国内开源技术领域希望的春天！

致谢

这本书之所以能够顺利完成，和每一位参与撰写的专家的辛苦劳动密不可分，这里向他们表示深深的感谢，感谢他们开源共享和开放合作的精神；感谢 InfoQ 中文站的编辑李剑、郭晓刚、胡键、高昂和张凯峰等，他们不仅参与了相关内容章节的撰写，而且在本书遇到困难时给予我很大的精神鼓励；感谢本书责任编辑 Lisa（杨绣国）的耐心与勇气，没有她在我身后坚持不懈的催促，就没有这本书的及时面市；感谢好友孟岩和方舟在内容和版式上必要的指导；感谢周筠老师在后勤上的有力保障；感谢海猫在文案、市场方面的有效协调；感谢那些我到现在还不知道名字的审校人员。没有大家的努力，就没有本书的出版！

再次感谢我的父母和兄弟姐妹，虽然远在家乡，但他们永远是我的精神港湾和前进动力！最后特别感谢我的老婆吴志民同学，在本书紧要的制作过程中，有多少个夜晚我无法给她安静的睡眠，她也很好地忍受了我的那么多无名之火，并在我偷懒的时候不嫌啰嗦地提醒我早日完稿。

霍泰稳

2008 年 5 月于广西阳朔

目录

1	Web框架篇	1
1.1	Struts	3
1.2	Spring	10
1.3	Seam	22
2	动态语言篇	36
2.1	Python	38
2.2	Ruby	43
3	Ajax开发篇	50
3.1	Buffalo	52
3.2	Dojo	60
4	版本控制篇	68
4.1	Subversion	70
5	项目管理篇	81
5.1	Teamwork	84
6	面向方面编程篇 (AOP)	92
6.1	JBoss AOP	94
7	面向服务架构篇 (SOA)	99
7.1	Apache CXF	101
7.2	Apache ODE	109
7.3	Apache Tuscany	117

1

Web框架篇

综述

说到 Web，相信大家一定都不会陌生，可以说它早已融入了人们的生活之中。在这个大背景下，人们对 Web 开发的重视程度也在不断提高，越来越多的项目采用 B/S 方式实现，对开发的质量、速度等诸多方面也有了更高的要求，显然，原始的开发方式在这种形势下就有些力不从心了。

2001 年，Apache 组织推出了大名鼎鼎的 Struts——一个优秀的 MVC 框架，它把开发者从纷繁的细节和重复的劳动中解放了出来，使他们能够专注于真正的业务逻辑，而不用再分心于如何分派请求。Struts 很快就红了，而且还红得发紫，最后成为了事实上的行业标准，这可以说是开源社区的一大成功。

现实中的需求是在不断变化的，如今的 Web 开发可不是做个网页显示一些数据，或者提交一个表单这么简单，它需要处理的问题多种多样。比方说，要方便地在事务环境中操作数据库，要能与外部系统交互，要以不同的方式来呈现最终结果，要结合工作流，等等。这些问题中的绝大部分都可以通过现成的开源产品得以解决，但问题在于它们都是独立的，要让开发者自己把这些东西拼凑到一起还真有些麻烦。都说“懒惰”是开发者的一种美德，如果你也是具有这种美德的人，那就把这些事情交给框架去解决吧。

为了应对这些日益复杂的需求，Web 框架本身也在不断地发展着，目前流行的 Web 框架不再仅是一个 MVC 实现，而是几乎可以满足日常工作中的大多数需求的“一站式”框架，上至表现层，下至持久层，涉及了企业级应用中的方方面面，还真有点包罗万象的感觉。此外，这些“一站式”框架还可以很方便地同其他框架进行整合，比如 Struts、Spring 与 Hibernate 的组合就很不错。

在开源世界中有许多优秀的 Web 框架，选用何种框架，如何进行使用它们都取决于具体的需求，要知道世上没有最好的框架，只有最合适的框架。在了解了这些框架背后的思想后，相信您一定会对 Web 开发有更深入的认识，也许将来的某一天您也能开发出一个属于自己的 Web 框架。

关联信息

开源 Web 框架正处在一个百家争鸣的阶段，不时会有新的框架带着新的想法进入大家的视野，但如果要说目前哪些框架的使用频率比较高，无疑就是 Struts、WebWork、Spring 和 Seam 了，在接下来的文章中会详细介绍其中的 Struts、Spring 和 Seam 三个框架，所以这里仅对 WebWork 做个简单的介绍。

WebWork 是 OpenSymphony 组织开发的致力于组件化和代码重用的 Web 框架，在很多方面为 Web 开发者提供了强大的支持。现在的 Struts 2 正是 Struts 1.x 与 WebWork 结合后的产物，事实上，Struts 2 更多的是基于 WebWork 开发的。

Web 框架更多地关注业务层，让开发者将精力集中在实现业务逻辑上，而在表现层和数据层的处理上则可以结合一些其他的开源产品。例如，表现层除了使用 JSP 外还可选用 Velocity 和 FreeMarker 来呈现；在访问数据时，可以使用 Hibernate、iBatis 和 JDO 来实现 O/R Mapping。在这种时候，Web 框架其实充当着一种“粘合剂”，它帮助开发者以一种相对合理的方式来整合各种现有技术，来最终实现需求。

下面就让我们进入正题，共同探索目前流行的 Struts、Spring 和 Seam 框架，希望这些介绍能让您对 Web 框架有一个整体的了解，如果能帮助您在遇到实际问题时做出合适的选择那就再好不过了。

1.1

Struts

总评

上手容易程度	★★★★★
社区活跃性	★★★★★
应用广泛性	★★★★★
推荐指数	★★★★★

功能和特点

Struts2 将 OSGi 的设计理念贯穿其中，同时也在试验性地提供 REST 架构风格和对 Ajax 的支持，处处不乏精妙之笔。整体给人的感觉，就像一只美丽的白鹤，在 Web 开发的天空下翩翩飞翔。

虽不是十年磨一剑，但 Struts2 也让观众有了太久的等待。下面，还是让我们先来看一下它有哪些令人心动的特性吧。

1. 纯POJO的Action

在使用 Struts1 的时候，很多人都深深痛苦于 Action 的单元测试，因为当时的 Action 是依赖于 Servlet 运行环境的，需要依赖 JMock 或 EasyMock 等 mock 工具。而在 Struts2 中，Action 已经不需要实现任何与容器耦合的接口，或是继承其他类，这便给开发和测试带来了极大的便利。

Struts2 中提供了 ActionSupport 基类，该类实现了一些常用的接口，Action 可以继承该类以自动获取接口带来的便利，不过这也不是必须的，任何带有 execute 方法的 POJO 对象都可以用作

Action。如：

```
class MyAction {  
    public String execute() throws  
        Exception {  
            return "success";  
        }  
}
```

2. 纯POJO的表单

在 Struts2 中，再也不需要 ActionForm 这种贫血的冗余代码，我们可以直接把 Action 拿来使用，而 Action 只需要暴露 setter 和 getter 方法以供容器调用就可以了，从而减少了贫血模型对象的数量。同时，Struts2 中也提供了 WebWork 的模型驱动（ModelDriven）Action，让开发人员可以直接操作领域对象，不需要将重复的属性在 Action 和领域对象之间反复传递。

3. 拦截器（interceptor）和拦截器栈（interceptor stack）

拦截器是 WebWork 最强大的特性之一，同样，在 Struts2 中，它的大多数核心功能也都是通过拦截器实现的，如输入校验、文件上传、异常处理等。从概念上讲，它与 ServletFilter 和 Proxy 颇为相似，通过对 Action 进行预处理和事后处理，把系统中比较分散的关注点进行集中，将通用代码模块化——这也正是 AOP 的目的所在。

图 1-1 是来自 Struts 网站的一张图片 (<http://struts.apache.org/2.x/docs/interceptors.html>)，它形象地描述了在 Action 的生命周期内，拦截器是如何起作用的：

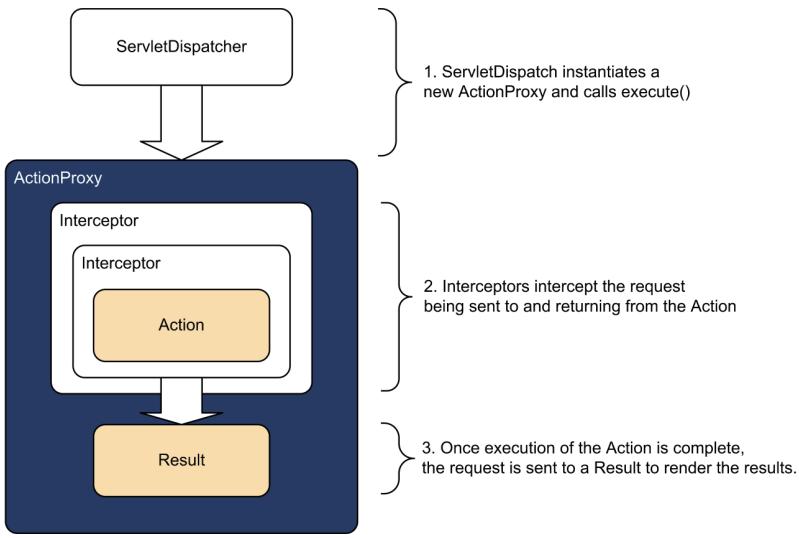


图1-1 Action生命周期

除了拦截器外，开发人员还可以使用内置的拦截器栈来减少配置。按照 Struts 官方网站的说法，Struts2 默认的拦截器栈目的就是为大多数应用的通用功能需求服务，所以开发一般的应用程序时，基本上不需要添加自定义的拦截器，或是修改拦截器栈。

如果能够充分掌握框架所提供的拦截器和拦截器栈，一方面可以大幅度提高开发效率，另一方面则是方便在调试 Action 的时候，对问题进行分析和定位。由于篇幅所限，这里不再一一列举各个默认的拦截器及拦截器栈所提供的功能，有兴趣的读者可以访问 <http://struts.apache.org/2.x/docs/interceptors.html> 或参考《深入浅出 Struts2》一书，该书的免费下载网址为：<http://www.infoq.com/cn/minibooks/starting-struts2>。

4. 惯例重于配置

惯例重于配置（Convention over Configuration）是 Rails 带入 Web 开发中的概念，它假定在大多数情况下，开发人员都会采用同样的模式，如

package 的组织、路径名、文件名等。而由于这种模式具有足够高的通用性，所以被框架采用为默认配置。以降低一部分灵活性为代价，大大减少了在管理配置信息上所投入的时间和精力，从而大幅度地提高了开发效率。

Struts2 也采用了这个概念，为开发人员提供了隐式的配置文件加载、默认的返回值和返回类型、默认的拦截器和拦截器栈、默认的命名方式和 Codebehind 插件（更多信息，请访问 <http://struts.apache.org/2.x/docs/codebehind-plugin.html>）等，力求在灵活性和减少开发人员工作之间作出平衡。如上面 POJO Action 一节所提到的那个例子，就是采用了默认的命名方式，以 execute 来作为 Action 被调用时所执行的方法名。

又例如，在默认情况下，当 `http://example.com/myapp/public/viewProfile.action` 收到请求时，`com.example.myapp.actions.public.ViewProfileAction` 这个对象会被调用，而随后生成的视图则是 `http://example.com/web/public/`

viewProfileAction.jsp。

应用 Rails 进行 Web 开发已经是近年来炙手可热的概念，想必很多读者已对此有所了解，这里不再赘述。

5. 模块化与OSGi

最近 OSGi 正在越来越频繁地进入人们的视野，IBM、BEA、Oracle、SAP、IONA，这些公司都开始在自己的商业产品中采用 OSGi，它为我们带来了规范化的、强制性的模块组织方式，也为动态扩充、修改系统功能提供了支持。

在 Struts2 中，模块化已经成为了体系架构中的基本思想：用户可以通过插件来对框架进行扩展——把与特定应用无关的新功能组织成插件，应用到多个应用中；或是把自包含的应用模块创建为插件——所有用来提供某一种特性的功能组成都可以被打包在一起，并作为独立的插件发布。在这种情况下，采用规范化的模块系统必然是大势所趋。

2007 年 8 月，Struts 的开发成员 Don Brown 宣布，他们已经开始动手开发 Struts 2 的 OSGi 插件，当这项工作完成以后，开发人员就可以把应用拆分成多个独立的 jar 包，只要把它们放到服务器恰当的位置上，该插件就可以把它们——找出并进行部署，而后还可以实时地添加、移除或是升级插件，而无需重启整个应用。Don 还在博客中说道：“我们现在已经开始在内部把越来越多的功能挪到了插件中去，这样我们的代码库就可以只关注核心功能并更加敏捷。”

目前这个插件正处于试验阶段，尚未能用于产品环境中，但现在已经可以将应用程序的 package 分成不同的 bundle，支持 Velocity 与 FreeMarker 模板，以及 Struts2 和 Spring 的集成。读者可以访问 <http://cwiki.apache.org/S2PLUGINS/osgi-plugin.html> 以获得最新信息。

背景介绍

滚滚长江东逝水，浪花淘尽英雄。那个言必称 SSH（Struts，Spring，Hibernate）的年代已经渐渐远去，但我们或许还记得，几年前 Struts 横空出世的时候，那基于 Model 2 的 MVC 设计思想、强大的标签库、优雅灵巧的架构，璀璨四射的光芒几乎充满了整个舞台！一时之间，Struts 几乎成了 Java Web 开发的事实标准，教程、文章漫天飞舞，而各大招聘网站上，只要是与 Java 相关，也鲜见哪个职位不要求精通 Struts 开发的。

随着时间的慢慢推移，笼罩在 Struts 上的光芒散去以后，人们看到了它的笨拙，如 Action 对 Servlet 容器的依赖性导致测试的困难、手工进行类型转换提高了复杂度、线程安全问题、对 Action 执行的控制等。五年一贯的设计思路，让它很难适应时代的发展。而同时诸多的后起之秀，也使得开发人员可以按照具体的应用场景来选择组合适用的框架，这就进一步加速了 Struts 退出历史舞台的脚步。

然而，Struts 也从未放弃过追求辉煌的梦想。从最开始把 Struts 分离成 Struts Action 1 和 Struts Action 2 的构想，再到 Struts Ti，几经波折，Struts 2 终于呈现在了世人的面前！

当时是在 JavaOne 2005 大会上，Struts 的开发者提出了 Struts Ti 的草案，其中对 Struts Ti 的描述是这样的：

“Struts Ti 2.0 是一个简化的 Model 2 框架，旨在为开发 Web 应用程序提供更好的访问底层 servlet/portlet 的方式。它的定位是那些不需要服务端组件所带来的额外复杂度，也不需要冗长的配置文件，只是需要现代 Web 框架的结构和控制器特性的应用程序。Struts Ti 2.0 构建在 Struts1.x 之上，但是需要重新实现整个框架，从而为 Struts

Ti 的下一代发展扫清道路。它的目标是把 RoR 和 Nanoweb 的简洁, WebWork 2 的精致, Beehive 的 Page Flow, 以及从 Struts1.x 中吸取的教训融为一体。”

但 Struts1 的代码库实在不适宜做出大规模调整, 以适应 Struts Ti 的发展方向; 恰好当时 WebWork 的领导者 Patrick Lightbody 也把众多开源项目的领导者邀请到一起, 希望大家能够协力完成一个通用性的框架, 虽然最后由于细节上的困难, 他的愿望并没有得以实现, 但是 Struts Ti 和 WebWork 两个团队却发现了二者的众多共通之处, 从而展开了合作事宜。

2006 年 3 月, IndicThreads 对 Patrick Lightbody 进行了采访。当 IndicThreads 问到 WebWork 与 Struts 合并的进展情况时, Patrick 说道: “我们正在把 WebWork 的代码库放到 Apache 的孵化器中, 与此同时, 在 OpenSymphony 旗下的 WebWork 相关活动大部分都会停止运作。如果一切进展顺利的话, 到今

年 4 月底之前, 孵化器的进展就会完成……”

而就在当月的 23 日, WebWork 官方网站便放出了一条重磅新闻:

“WebWork 2.2.2 正式发布……这将是 WebWork2.2 系列的最后一版, 我们深深感谢整个 OpenSymphony 社区长久以来付出的精力和贡献。我们正在启动向 Apache Struts Action 2.0 进行移植的过程, 它将是下一代的基于 action 的 Web 框架。希望我们的合并工作能够顺利完成。——WebWork 开发组”。不过后来, WebWork 还是推出了几版基于 bugfix 或其他原因的更新, 直至今年 7 月的 2.2.6 版, 此乃后话。

2007 年 2 月, Struts 网站上终于发布了 Struts2 正式版 (2.0.6 GA)! 这款框架完全以 WebWork 为核心, 抛弃了 Struts1 的代码库, 并辅以优秀的插件体系结构, “惯例重于配置”思想的应用, 丰富多彩的 UI 展示和更加强大、灵活的系统架构, 以崭新的姿态呈现在了公众的面前! 这正是, 凤凰集香木自焚, 复从死灰中重生!

参考资料

网站类

1. Struts官方站点: <http://struts.apache.org/>
2. Struts官方文档: <http://cwiki.apache.org/WW/home.html>
3. WebWork&Struts2中文站点: <http://webwork.javascud.org/>
4. InfoQ中文站: <http://www.infoq.com/cn/struts>
5. JavaEye社区Java版面: <http://www.javaeye.com/forums/board/java>

书籍类

1. 《深入浅出Struts2》: <http://www.infoq.com/cn/minibooks/starting-struts2>
作者: IanRoughley 译者: 李剑 出版社: C4Media InC. 出版时间: 2007年
ISBN: 1430320338
2. 《Webwork in Action 中文版》
出版社: 电子工业出版社 出版时间: 2007年3月 ISBN: 7-121-03299-6

快速上手教程

在Struts2官方网站(<http://struts.apache.org/2.x/docs/hello-world.html>)上有一个HelloWorld的入门教程，但可惜仍有不足之处，这里加以补充说明：

1. 从官方网站上下载一个稳定的发行版（当前版本为2.0.11），解压缩以后，里面的lib目录下有4个jar包是开发Struts2必不可少的：struts2-core.jar, xwork.jar, freemarker.jar, ognl.jar（由于各jar包的对应的版本仍在变动中，所以此处未提及）。用IDE创建Web工程以后，将这4个jar包放到WEB-INF/lib目录下。

2. 在web.xml中，声明Struts2的filter和filter-mapping：

```
<filter>
    <filter-name>struts2</filter-name>
    <filter-class>
        org.apache.struts2.dispatcher.FilterDispatcher
    </filter-class>
</filter>
<filter-mapping>
    <filter-name>struts2</filter-name>
    <url-pattern>/*</url-pattern>
</filter-mapping>
```

3. 写一个HelloWorld.jsp，放到项目根目录下，文件内容很简单，只是用了一个struts2标签来获取action的数据并显示给用户：

```
<%@ taglib prefix="s" uri="/struts-tags" %>

<html>
    <head>
        <title>Hello World!</title>
    </head>
    <body>
        <h2><s:property value="message" /></h2>
    </body>
</html>
```

4. 将struts.xml放到WEB-INF/classes目录下（在IDE中，需要在src目录下创建struts.xml，部署后会自动存放到WEB-INF/class目录），内容为：

```
<!DOCTYPE struts PUBLIC
    "-//Apache Software Foundation//DTD Struts Configuration 2.0//EN"
    "http://struts.apache.org/dtds/struts-2.0.dtd">

<struts>
    <package name="example" extends="struts-default">
        <action name="HelloWorld" class="example.HelloWorld">
            <result>/helloWorld.jsp</result>
        </action>
    </package>
</struts>
```

5. 如果你是一个习惯使用测试驱动开发的开发人员，那么这一步就该为Action编写测试代码了，因为在Struts2中的Action是POJO，不存在对HttpRequest、HttpResponse的依赖，所以大大降低了测试复杂度。例如：

```
import static org.junit.Assert.assertEquals;
```

```
import org.junit.Test;

public class HelloWorldTest {
    @Test
    public void testExecute() {
        HelloWorld action = new HelloWorld();
        assertEquals(action.execute(), "success");
        assertEquals(action.getMessage(), "HelloWorld");
    }
}
```

接下来就是编写 Action 的实现类了——HelloWorld.java。在这一步中，官方网站中的例子实际上没有必要继承 xwork 的 ActionSupport 类，完全可以写成如下形式：

```
public class HelloWorld {
    public String execute() {
        setMessage("HelloWorld");
        return "success";
    }

    private String message;

    public void setMessage(String message) {
        this.message = message;
    }

    public String getMessage() {
        return message;
    }
}
```

最后就是运行测试代码，根据测试结果对实现代码进行调整，直至 JUnit 显示绿条为止。

社区视角

Struts2 官方网站上用“构建！部署！维护！”醒目地标注了 Struts2 的出众特性：

“构建！使用我们提供的指南和模板应用可以易于上手，使用与 HTTP 无关的框架接口保证代码的简洁，通过样式表驱动的表单 tag 减少代码，提供了灵活的 Cancel 按钮，使用 AJAX Tag 提供强大的交互性和灵活性，方便与 Spring 集成，POJO 表单，POJO Action……”

“部署！简单的插件体系结构——扔一个 jar 文件进来就可以扩展框架而无需手动配置，集成式的性能分析，精确的异常报告——可以把错误定位到对应的代码行……”

“维护！易于测试的 Actions——无需 Mock

HTTP 对象灵活智能化的默认配置——大多数的框架配置元素都提供了默认值，便于自定义的控制器——在需要的时候可以对每一个 action 的请求处理动作进行自定义，Struts2 只是做了你想让它做的工作而已。集成调试——可以通过内建的调试工具来生成问题报告……”

在没有正式发行 Struts2 之前，InfoQ 英文站上就已经发布了两篇文章：《从 Struts1 迁移到 Struts2》(<http://www.infoq.com/articles/migrating-struts-2-part1>) 和《从 Struts1 迁移到 Struts2——系列 2》(<http://www.infoq.com/articles/migrating-struts-2-part2>)，介绍了 Struts2 的核心概念、特性、请求处理流程、配置语法，并辅以示例讲述了如何从 Struts1 向 Struts2 进行迁移。

JavaEye 上也有着众多有关 Struts2 的讨论，

如《Struts2.0.2 实现零配置文件开发模式!》(<http://www.javaeye.com/topic/34989>),《Struts2 中的零配置与 CoC》(<http://www.javaeye.com/topic/112675>)等,也有不少文章通过示例代码来讲述如何对 Struts2 快速上手。

《Struts2 中的零配置与 CoC》一文的作者 andlu 还对 Struts2 的性能提出了一些建议:

“事实上,我推荐 Struts2 的使用者只用 Struts2 输出 XML 或 JSON, 放弃 UI, 页面这层还是使用标准的 HTML、CSS 和一些 JS 组件来展现……如果我们放弃了 Struts2 的笨拙的 UI, Result 只输出 XML 或 JSON, UI 则使用标准的 HTML+CSS, 若使用 JS 组件 (DOJO、Adobe Spry Framework、YUI-Ext 等) 来操作 Struts2 的输出数据, 情况将会如何? 我们会得到一个高性能、高可配的、UI 和应用服务器的职责分割更为明确、合理的, 更易于静态化部署的开发组合。

这似乎是阉割了 Struts2, 但是这样阉割过的

Struts2 摆脱了性能低下的包袱, 更轻、更现代化。”

在 CSDN 的“Struts2 的专栏”博客中, 马召则正在编写一本开源书籍来讲述 Struts2 应用 (<http://blog.csdn.net.struts2/archive/2007/08/01/1721752.aspx>), 他在博客中写道:

“Struts 2 是一个雅致的, 可扩展的, 用来建立企业级 Java Web 应用程序的框架……不但注重程序的开发过程, 更注重部署和后期维护……融合了 Struts 和 WebWork 的社区力量, 是这两个社区努力的结果。Struts2 非常容易使用。”

作者介绍

李剑, InfoQ 中文站敏捷社区首席编辑,《深入浅出 Struts2》译者, 致力于敏捷软件开发的实施与推广。

1.2 Spring

总评

上手容易程度	★★★★★
社区活跃性	★★★★★★
应用广泛性	★★★★★★
推荐指数	★★★★★★

Spring Framework 是一个优秀的“一站式”Java/Java EE 应用程序框架，提供了大多数传统应用程序所需的基础设施，还有一些其他框架没有涉及的东西。它能帮助你大幅提升开发效率，改善应用程序的质量和性能，让程序更易测试。

功能和特点

作为一个“一站式”的框架，Spring Framework 中包含了很多东西，其中采用了分层架构，你可以仅仅使用你所需要的那部分功能。Spring 的理念中有一条是“不去重新发明轮子”，所以框架中大量地集成了现有的优秀开源项目，并提供了不少辅助工具。

Spring 能为项目带来如下一些好处：

1. Spring可以有效地管理中间层对象；
2. Spring让你能用一致的方式对应用程序和项目进行配置，在控制反转和依赖注入的帮助下，一切都能变得十分简单；
3. 好的编程实践不再高不可攀，Spring大大降低了使用它们所需的代价；
4. 基于Spring的应用程序易于测试；
5. 无论你选择JDBC还是O/R映射工具，Spring

为你提供了一致的数据访问框架；

6. Spring是出色的“粘合剂”，你能简单地通过相近的方式来使用JDBC、JMS、JavaMail等多种API；
7. Spring能帮助你用轻量级的基础设施来解决你所面对的问题。

项目中有了 Spring 就不再需要别的框架了，因为它几乎能满足所有的需要，替你处理各种事情；此外你的代码可以完全不用依赖于 Spring API，这也为你带来了极大的灵活性。

背景介绍

Spring Framework 项目于 2003 年初落户于 SourceForge，至今已有 5 年历史，目前开发团队的核心人员都在 Rod Johnson 的 SpringSource 公司（以前叫 Interface 21）全职进行 Spring 的开发与维护，同时 SpringSource 还提供 Spring 相关的咨询、支持等服务。

Spring 有着深厚的历史背景，项目源于 Rod Johnson 在 2002 年出版的《Expert One-on-One J2EE Design and Development》一书中的代码，书中展现了该框架背后的思想，而其中核心的架构概念则可以追溯到 2000 年，它体现了 Rod Johnson 在一系列成功商业项目开发中的大量经验。那时，B/S 架构被逐渐引入企业级应用的开发之中，Struts、EJB 都是当时流行的选择，可惜 Struts 只提供了 MVC 框架，功能不够强大；而 EJB 在没有工具支持的情况下使用异常复杂，开

发者想要将精力集中在开发业务逻辑上几乎不可能，有太多的工作需要他们来处理，甚至有人认为 EJB 带来的问题远比它解决的要多……Spring

的出现结束了这些噩梦，现在的 Spring 早已成为了 Java 企业级应用开发的事实标准，确定了自己的主流地位。

参考资料

网站类

1. Spring Framework官方网站 (<http://www.springframework.org>)

这是Spring Framework的官方网站，可以从中了解到Spring的最新动态，它提供了Spring及其周边项目的第一手信息，通过该站点你能获得下载、文档及支持等内容。

2. Java视线 (<http://www.javaeye.com>)

JavaEye是一个软件开发人员的深度交流社区，从讨论Hibernate技术起家，现在已经发展为涉及软件开发各方各面的综合性社区，深受广大开发者的好评。

3. Spring Framework中文论坛 (<http://spring.jactiongroup.net>)

这是一个专注于Spring Framework的交流平台，提供了开放、互动的良好环境，让广大中文世界的开发者能够在此分享Spring的知识和经验。同时，它与满江红 (<http://www.redsaga.com>) 共同发起的Spring参考手册翻译项目更是为广大开发者在学习和使用Spring方面带来了极大的便利。

书籍类

1. 《Spring 2.0 核心技术与最佳实践》

作者：廖雪峰 出版社：电子工业出版社 出版时间：2007年6月 ISBN：9787121042621

本书结合理论与实践，由浅入深地介绍了Spring Framework 2.0的几乎所有内容，向读者展示了如何用Spring构建灵活的Java EE应用，并提供了大量实例。

2. 《Spring 2.0技术手册》

作者：林信良 出版社：电子工业出版社 出版时间：2007年4月 ISBN：9787121039850

本书全面深入地介绍了Spring Framework 2.0的很多方面，无论是初学者还是有一定经验的开发者都能从中获益匪浅。

3. 《expert one-on-one J2EE Development without EJB》

作者：（美）詹森（Johnson,R.）、（美）赫鲁（Hoeller, J.） 译者：JavaEye

出版社：电子工业出版社 出版时间：2005年9月 ISBN：9787121016844

Spring Framework作者Rod Johnson的力作，书中介绍了时下流行的一些概念和开源框架，阐述了Spring背后的思想和理念，告诉读者该如何正确地选择一种架构、一种技术。

快速上手教程

Spring Framework 可以被使用在很多场合之中，考虑到目前大多数 Java EE 的项目是 B/S 结构

的，所以这里的快速上手教程会以 Spring MVC 为切入点，用最简单的代码一步一步来实现一个图书列表的页面。

在正式动手之前需要做一些准备工作，先安装并设置好 JDK 1.5 和 Tomcat 5，关于数据库及其访问方式可以根据个人习惯进行选择，教程中使用 MySQL 数据库和 Hibernate（映射由 Hibernate Annotation 实现）。请将实际使用到的 jar 文件复制到 WEB-INF/lib 目录中，整个项目的结构如图 1-2 所示，教程中用到的 jar 文件如图 1-3 所示。

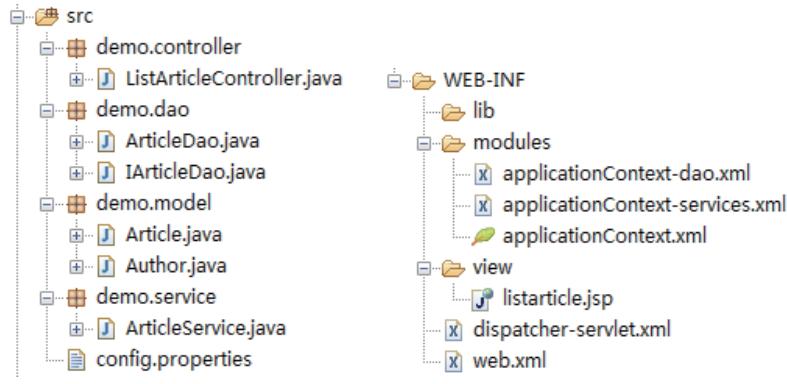


图1-2 项目结构

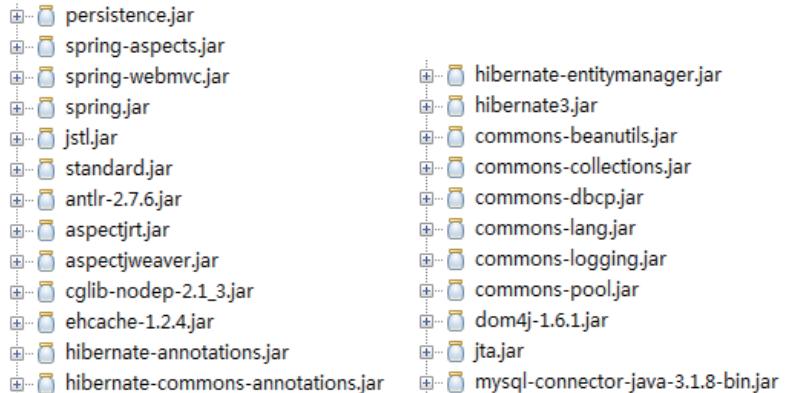


图1-3 使用到的jar文件

项目中的 Bean 定义分散在多个 XML 文件中，每完成一部分代码就给出相应的配置，最后再进行整合和部署。配置中使用 default-autowire="byName" 实现了 Bean 的自动织入，节省了很多工作量，只需注意 Bean 及属性的命名即可。

Step 1 Business Objects & DAO

教程中的例子涉及两个实体对象，代表文章的 Article 类和代表作者的 Author 类，分别对应了数据库中的 article 表和 author 表，一篇文章有一个作者，而一个作者可以有多篇文章。类的代码如下

(省略 getter 和 setter):

代码: Article.java

```
package demo.model;

import javax.persistence.*;

@Entity
public class Article {
    @Id
    @GeneratedValue
    private Long id;

    private String title;

    @ManyToOne
    private Author author;

}
```

代码: Author.java

```
package demo.model;

import java.util.List;
import javax.persistence.*;

@Entity
public class Author {
    @Id
    @GeneratedValue
    private Long id;

    private String name;

    @OneToMany
    private List<Article> articles;

}
```

在 MySQL 中创建数据表的 SQL 语句如下, 数据请自行添加 (如果使用 Hibernate, 表可以根据映射自动生成, 具体做法请参考 Hibernate 文档):

代码: 数据库创建 SQL

```
CREATE DATABASE `articles` DEFAULT CHARACTER SET utf8 COLLATE
    utf8_general_ci;
USE articles;

CREATE TABLE `article` (
    `id` bigint(20) NOT NULL auto_increment,
    `title` varchar(100) NOT NULL default '',
    `author_id` bigint(20) NOT NULL default '0',
    PRIMARY KEY  (`id`)
) ENGINE=MyISAM DEFAULT CHARSET=utf8;

CREATE TABLE `author` (
    `id` bigint(20) NOT NULL auto_increment,
    `name` varchar(100) NOT NULL default '',
    PRIMARY KEY  (`id`)
) ENGINE=MyISAM DEFAULT CHARSET=utf8;
```

考虑到可能会有多种 DAO 的实现，所以在 DAO 层先定义一个 IArticleDao 接口，随后可以自由选择具体的实现方式，此处结合 Spring 的 HibernateDaoSupport 使用 Hibernate 来进行实现：

代码：IArticleDao.java

```
package demo.dao;

import java.util.List;

import demo.model.Article;

public interface IArticleDao {
    public List<Article> loadAllArticles();
}
```

代码：ArticleDao.java

```
package demo.dao;

import java.util.List;
import org.springframework.orm.hibernate3.support.HibernateDaoSupport;
import demo.model.Article;

public class ArticleDao extends HibernateDaoSupport implements
    IArticleDao {

    @SuppressWarnings("unchecked")
    public List<Article> loadAllArticles() {
        return (List<Article>)getHibernateTemplate().
            loadAll(Article.class);
    }

}
```

接下来对 Hibernate 进行相应的配置，如果使用了 JDO 或 iBatis，请参考 Spring 文档。
applicationContext-dao.xml 内容如下：

代码：applicationContext-dao.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<beans
    xmlns="http://www.springframework.org/schema/beans"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://www.springframework.org/schema/beans
        http://www.springframework.org/schema/beans/spring-beans-2.0.xsd"
    default-autowire="byName">
    <!-- DAO 配置于此 -->
    <bean id="articleDao" class="demo.dao.ArticleDao"/>

    <!-- 数据源 -->
    <!-- JNDI 数据源 -->
    <!--
        <bean id="dataSource" class="org.springframework.jndi.
            JndiObjectFactoryBean">
            <property name="jndiName" value="${datasource.jndi.name}"/>
        </bean>
    -->

    <!-- JDBC 数据源 -->
    <bean id="dataSource" class="org.apache.commons.dbcp.BasicDataSource"
        destroy-method="close">
```

```

<property name="driverClassName" value="${datasource.jdbc.driverClassName}" />
<property name="url" value="${datasource.jdbc.url}" />
<property name="username" value="${datasource.jdbc.username}" />
<property name="password" value="${datasource.jdbc.password}" />
</bean>

<!-- 使用Annotation映射的sessionFactory -->
<bean id="sessionFactory" class="org.springframework.orm.hibernate3.annotation.AnnotationSessionFactoryBean">
    <property name="dataSource" ref="dataSource"/>
    <property name="hibernateProperties">
        <props>
            <prop key="hibernate.dialect">${hibernate.dialect}</prop>
            <prop key="hibernate.show_sql">${hibernate.show_sql}</prop>
            <prop key="hibernate.cache.use_query_cache"
                  >${hibernate.cache.use_query_cache}</prop>
            <prop key="hibernate.cache.provider_class"
                  >${hibernate.cache.provider_class}</prop>
        </props>
    </property>
    <property name="annotatedClasses">
        <list>
            <value>demo.model.Article</value>
            <value>demo.model.Author</value>
        </list>
    </property>
</bean>

<!-- 事务管理器，此处为Hibernate的事务管理器 -->
<bean id="transactionManager" class="org.springframework.orm.hibernate3.HibernateTransactionManager" />
</beans>
```

此处如果使用 JNDI 提供数据源，请根据注释进行调整。Spring 的事务管理需要声明事务管理器，由于 Hibernate、JDO、JDBC 的事务管理器都不一样，所以将其与其他事务的配置分开存放。此外，配置中的一些参数使用了占位符（形如 \${}），这些内容将在 Step 4 中进行加载。

Step 2 Service

Service 层只是调用 DAO 中的方法为控制器提供图书列表，Service 最好能先给出接口，随后进行实现，但此处的功能比较简单，就直接进行实现了：

代码：ArticleService.java

```

package demo.service;

import java.util.List;
import demo.dao.IArticleDao;
import demo.model.Article;

public class ArticleService {
    private IArticleDao articleDao;

    public List<Article> loadAllArticles() {
        return articleDao.loadAllArticles();
    }

    public void setArticleDao(IArticleDao articleDao) {
```

```
        this.articleDao = articleDao;
    }
}
```

Spring 通过 setArticleDao 方法为 ArticleService 注入 DAO，也可以选择通过构造方法注入，2.5 中还能用 @Autowired 进行注入。

applicationContext-services.xml 内容如下：

代码： applicationContext-services.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<beans
    xmlns="http://www.springframework.org/schema/beans"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://www.springframework.org/schema/beans
        http://www.springframework.org/schema/beans/spring-beans-2.0.xsd"
    default-autowire="byName">
    <!-- Service配置于此 -->
    <bean id="articleService" class="demo.service.ArticleService" />
</beans>
```

Step 3 Controller & View

Spring MVC 提供了多种实现控制器的方式，此处直接实现 Controller 接口，开发一个单一动作的简单控制器，从 Service 中取得图书列表，提供给视图进行呈现，ListArticleController 内容如下：

代码： ListArticleController.java

```
package demo.controller;

import java.util.List;

import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

import org.springframework.web.servlet.ModelAndView;
import org.springframework.web.servlet.mvc.Controller;

import demo.model.Article;
import demo.service.ArticleService;

public class ListArticleController implements Controller {
    private ArticleService articleService;

    public ModelAndView handleRequest(HttpServletRequest request,
        HttpServletResponse response) throws Exception {
        List<Article> articles = articleService.loadAllArticles();
        ModelAndView mav = new ModelAndView();
        mav.addObject(articles);
        return mav;
    }

    public void setArticleService(ArticleService articleService) {
        this.articleService = articleService;
    }
}
```

ModelAndView 中保存了要传递给视图的对象和具体要使用的视图文件，自 2.0 起， Spring

MVC 提供了 Convention over Configuration 的机制，大大简化了代码与配置。简单地说，名字以 Controller 结尾的控制器类都会被映射为相应的地址，ListArticleController 对应 /listarticle*，如果是 MultiActionController 则会被映射为一个目录。向 ModelAndView 添加对象时可以不用指定键 (key)，单一对象的键取决于类名，比如 x.y.User 的键是 user；而某一类对象的 Set、List 或数组则稍有些复杂，取第一个对象的类名加上“List”作为它的键，比如这里的 articles 是一个存放 Article 对象的 List，它的键就是 articleList。具体的视图会根据请求自动在指定目录中寻找对应的视图文件，本例中就会寻找 listarticle（后缀由配置文件决定）。关于 Convention over Configuration 还有些其他的细节，请参考 Spring 文档的相关章节。

此处的视图比较简陋，只是一张表格，显示了图书的编号、书名和作者，使用 JSTL 的 <c:forEach> 标签来遍历列表，具体代码如下：

代码：listarticle.jsp

```
<%@ page pageEncoding="UTF-8"%>
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>

<html>
    <head>
        <title>Article List</title>
    </head>
    <body>
        <table width="80%" cellspacing="0" cellpadding="0" border="1">
            <thead>
                <tr align="center">
                    <td width="20%>编号</td><td width="50%">
                    书名</td><td width="30%">作者</td>
                </tr>
            </thead>
            <tbody>
                <c:forEach items="${articleList}" var="article">
                    <tr>
                        <td align="center">${article.id}</td>
                        <td>${article.title}</td>
                        <td>${article.author.name}</td>
                    </tr>
                </c:forEach>
            </tbody>
        </table>
    </body>
</html>
```

为了使用 Spring MVC，需要在 web.xml 中配置一个分派器，将一些特定格式的请求交给 Spring MVC 来处理（其实就是一个 Servlet，这和 Struts 有些类似），如果它的名字是 dispatcher，那么 Spring 默认会去寻找名为 dispatcher-servlet.xml 的配置文件，该文件内容如下：

代码：dispatcher-servlet.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<beans
    xmlns="http://www.springframework.org/schema/beans"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://www.springframework.org/schema/beans
        http://www.springframework.org/schema/beans/spring-beans-2.0.xsd"
```

```

default-autowire="byName">
<!-- SpringMVC相关Bean配置 -->

<!-- View Resolver -->
<bean id="viewResolver" class="org.springframework.web.servlet.view.
InternalResourceViewResolver">
    <property name="viewClass" value="org.springframework.web.
        servlet.view.JstlView" />
    <property name="prefix" value="/WEB-INF/view/" />
    <property name="suffix" value=".jsp" />
</bean>

<bean id="viewNameTranslator" class="org.springframework.web.servlet.
    view.DefaultRequestToViewNameTranslator"/>
<bean class="org.springframework.web.servlet.mvc.support.ControllerCla
ssNameHandlerMapping"/>

<!-- 以下为Controller -->
<bean id="listArticleController" class="demo.controller.
    ListArticleController" />

</beans>

```

配置中的 DefaultRequestToViewNameTranslator 和 ControllerClassNameHandlerMapping 就是用来实现 Convention over Configuration 的，而名为 viewResolver 的 Bean 则指定了一些视图的信息。

Step 4 Configuration & Deployment

至此，大部分的工作已经完成了，接下来就是加载 properties 文件和配置事务属性，这些都放在 applicationContext.xml 中：

代码：applicationContext.xml

```

<?xml version="1.0" encoding="UTF-8"?>
<beans
    xmlns="http://www.springframework.org/schema/beans"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns:aop="http://www.springframework.org/schema/aop"
    xmlns:tx="http://www.springframework.org/schema/tx"
    xsi:schemaLocation="
        http://www.springframework.org/schema/aop http://www.springframework.
        org/schema/aop/spring-aop-2.0.xsd
        http://www.springframework.org/schema/beans http://www.
        springframework.org/schema/beans/spring-beans-2.0.xsd
        http://www.springframework.org/schema/tx http://www.springframework.
        org/schema/tx/spring-tx-2.0.xsd">

    <bean id="propertyConfigurer" class="org.springframework.beans.factory.
        config.PropertyPlaceholderConfigurer">
        <property name="locations">
            <list>
                <value>classpath:config.properties</value>
            </list>
        </property>
    </bean>

    <!-- 事务 -->
    <tx:advice id="txAdvice">
        <tx:attributes>
            <tx:method name="get*" read-only="true" />

```

```

<tx:method name="find*" read-only="true" />
<tx:method name="load*" read-only="true" />
<tx:method name="**" />
</tx:attributes>
</tx:advice>

<aop:config proxy-target-class="true">
    <aop:advisor advice-ref="txAdvice" pointcut="execution(* demo.
        service..*.*(..))" />
</aop:config>
</beans>

```

pointcut属性确定了AOP拦截的方法,用的是AspectJ pointcut expression,此处对demo.service中每一个类的所有方法都进行了拦截,也就是它们都在事务中执行。

config.properties中保存了一些与数据库和Hibernate相关的配置信息,它们会代替XML中对应的占位符。

代码: config.properties

```

# DataSource
# JNDI datasource Eg. java:comp/env/jdbc/myds
datasource.jndi.name=
# JDBC datasource
datasource.jdbc.driverClassName=com.mysql.jdbc.Driver
datasource.jdbc.url=jdbc:mysql://localhost/articles?useUnicode=
true&charac terEncoding=utf8
datasource.jdbc.username=root
datasource.jdbc.password=

# Hibernate
hibernate.dialect=org.hibernate.dialect.MySQLDialect
hibernate.show_sql=false
hibernate.cache.use_query_cache=true
hibernate.cache.provider_class=org.hibernate.cache.EhCacheProvider

```

最后要看到的就是web.xml,每个Java EE的Web项目都会有这个配置文件,具体内容如下。

代码: web.xml

```

<?xml version="1.0" encoding="UTF-8"?>
<web-app version="2.4"
    xmlns="http://java.sun.com/xml/ns/j2ee"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://java.sun.com/xml/ns/j2ee
    http://java.sun.com/xml/ns/j2ee/web-app_2_4.xsd">

    <!-- Spring ApplicationContext配置文件的路径可使用通配符, 多个路径用,号分隔,
    此参数用于后面的Spring-Context loader -->
    <context-param>
        <param-name>contextConfigLocation</param-name>
        <param-value>/WEB-INF/modules/applicationContext*.xml</param-value>
    </context-param>

    <!-- SpringMVC 分派器及相关映射 -->
    <servlet>
        <servlet-name>dispatcher</servlet-name>

        <servlet-class>org.springframework.web.servlet.
            DispatcherServlet</servlet-class>

```

```

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

    <servlet-mapping>
        <servlet-name>dispatcher</servlet-name>
        <url-pattern>*.html</url-pattern>
    </servlet-mapping>

    <servlet-mapping>
        <servlet-name>dispatcher</servlet-name>
        <url-pattern>*.do</url-pattern>
    </servlet-mapping>

    <!--Spring ApplicationContext 載入 -->
    <listener>

        <listener-class>org.springframework.web.context.
            ContextLoaderListener</listener-class>
    </listener>

    <!-- Spring 刷新Introspector防止内存泄露 -->
    <listener>

        <listener-class>org.springframework.web.util.
            IntrospectorClean upListener</listener-class>
    </listener>

    <!-- 支持session scope的Spring bean -->
    <listener>

        <listener-class>org.springframework.web.context.request.
            RequestContextListener</listener-class>
    </listener>

    <!-- Character Encoding filter -->
    <filter>
        <filter-name>setCharacterEncoding</filter-name>

        <filter-class>org.springframework.web.filter.
            CharacterEncodingFilter</filter-class>
        <init-param>
            <param-name>encoding</param-name>
            <param-value>UTF-8</param-value>
        </init-param>
    </filter>

    <filter-mapping>
        <filter-name>setCharacterEncoding</filter-name>
        <url-pattern>/*</url-pattern>
    </filter-mapping>

    <!--Hibernate Open Session in View Filter-->
    <filter>
        <filter-name>hibernateFilter</filter-name>

        <filter-class>org.springframework.orm.hibernate3.support.
            OpenSessionInViewFilter</filter-class>
    </filter>

    <filter-mapping>
        <filter-name>hibernateFilter</filter-name>
        <url-pattern>/*</url-pattern>
    </filter-mapping>
</web-app>
```

此处加载了 Spring 的配置文件，并对字符编码进行了处理，*.do 和 *.html 的请求都转交给了

Spring MVC 的分派器。OpenSessionInViewFilter 是用来解决 Hibernate 的 OpenSessionInView 问题的，如果没有使用 Hibernate，或没有遇到 OpenSessionInView 问题，则无需配置此过滤器。

项目的部署和一般的 Web 项目没有任何区别，将项目打成 War 包或直接将目录放到 Tomcat 的 webapps 中即可。假设目录的名字是 SpringDemo，启动 Tomcat 后访问 <http://localhost:8080/SpringDemo/listarticle.html> 就能看到页面的效果了。

版本信息

2006 年 10 月，Spring Framework 发布了 2.0 版本，对之前的 1.x 做了重大更新，提供了新的 Bean 作用域，简化了 Bean、AOP 和声明式事务的配置，增加了对 JPA 的支持。在 Web 层方面，Spring MVC 借鉴了 Ruby on Rails 中 Convention over Configuration 的思想，并提供了一套与 Spring MVC 相类似的 Portlet MVC 框架。此外，2.0 中还引入了对 Java5、动态语言、计划任务、异步 JMS 等方面的支持。

目前 Spring 的最新版本是 2.5.2，强化了 IoC 容器、事务管理和 Web 层的 Annotation 支持，进一步简化了 AOP、异步 JMS 和 JMX 的配置，还加入了对 Tiles 2、JSF 1.2 和 JAX-WS 的支持。最重要的是 2.5 相对 2.0 在性能上有了很大的提升，由于两者可以平滑升级，所以 2.5 会是一个理想的选择。

社区视角

Spring Framework 作为一个流行的开源项目，背后有着庞大的用户社区，大量用户的肯定就是其成功的最好证明。Spring 几乎能被使用在任何 Java 项目中，既可以充当核心框架，也可以只是一个辅助工具。

因为是一个“一站式”的框架，所以 Spring 在一定程度上减少了在决策上花费的时间。如果一个框架能满足所有的需求，并且在这些方面做得都不错，那还有什么必要在项目中用上好几个不同的框架自寻烦恼呢？

此外，“非侵入性”的特点使得应用程序代码能完全不依赖于 Spring API，这也降低了使用 Spring 的风险，在需要时可以方便地从 Spring 切换到别的框架。它的分层结构允许开发者根据实际要求“各取所需”，仅使用需要的部分，例如，可以仅使用 IoC 容器管理对象的生命周期，或者仅用 JavaMail 支持来简化邮件操作，并不强制一定要从头到尾使用 Spring。

Spring 没有重新发明“轮子”，而是充分利用了现有的优秀开源项目，让大量宝贵的经验能继续在项目里发挥作用。比方说曾经使用过 Hibernate 的开发者会发现在 Spring 中依旧能用 Hibernate 实现 ORM，而且比直接使用更方便。Spring 提供了大量辅助方法，类似 JDBC、JMS、JavaMail 的 API 在被封装后都以相近的形式出现在开发者面前，有了 Spring，一切都变得简单了。

Spring 能带来很多看得见摸得着的好处，但更重要的是其背后的思想和理念，它包含了那些经过反复验证的最佳实践，理解了这些东西，即使使用别的框架，甚至是沒有框架，一样能开发出优秀的系统。

作者介绍

丁雪丰，网名 DigitalSonic，编程爱好者，满江红翻译组成员，Spring Framework 2.0 & 2.5 Reference 翻译项目负责人，参与过 Hibernate、Seam 等多份文档的翻译及校对。平时也积极投身开源项目，是 SFTP/SCP 软件 WinSCP 的简体中文汉化者。

1.3 Seam

综合评比

上手容易程度	★★★★★
社区活跃性	★★★★★★
应用广泛性	★★★★★★
推荐指数	★★★★★★

JBoss Seam 是著名开源厂商 RedHat 旗下 JBoss 推出的下一代开源企业级 Java Web 开发平台。之所以称之为下一代，其原因之一是其目标超越传统的 Struts、Webwork、Struts 和 Webwork 业已合并为 Struts2 及 Spring 等 Java Web 框架；原因之二是其整合了 AJAX, JSF, JPA, EJB3 和 BPM 等“下一代”Java EE 标准和规范，以及热门的各种技术。

功能与特点

1. 一站式 (Full Stack) 的统一组件模型

JBoss Seam 为所有业务逻辑提供了统一的组件模型，这种统一也覆盖了 Web 应用的各个分层。开发者不必考虑各个分层之间对象数据的转移问题，这极大简化了开发者的工作（很多开发者厌烦了大量的 DAO 和 DTO）。JBoss Seam 组件在表现层和应用层上没有什么区别。

2. POJO 和 EJB

JBoss Seam 并不强制开发者使用 EJB3.0，虽然新的 EJB3.0 已经完全轻量化了，开发者仍可以使用纯粹的 POJO 来替代 EJB3.0。

3. 声明式的状态管理

JBoss Seam 的组件模型是有状态的，这是与 Struts、Webwork 和 Spring 等无状态的 Web 框架最大的不同之处。多数企业级开发不可避免地要使用状态管理。开发者不得不在先天无状态的 HTTP 协议上开发 Web 应用，小心地构建状态管理机制，处理状态问题。JBoss Seam 提供了声明式的状态管理机制，能够提供完善的状态管理服务。这也是 JBoss Seam 被称为有状态的 Java Web 框架的原因。

4. 扩展的上下文

JBoss Seam 定义并且扩展了 Java Servlet 规范中的上下文模型，首创了对话上下文和业务流程上下文。这两个扩展的上下文模型也是 JBoss Seam 被称为企业级 Web 开发框架的重要原因之一。JBoss Seam 通过扩展的上下文对组件的状态进行管理。

5. 超越依赖注入的双向注射 (Bijection)

JBoss Seam 使用 Java 5 注释实现双向注射，双向注射比传统的轻量的依赖注入（例如 Spring）更进一步，JBoss Seam 中的双向注射是动态的，上下文相关的，当然也是双向的，不仅可以注入 (injection)，还可以注出 (outjection)。

6. 使用注释配置

JBoss Seam 可以使用 Java 5 的注释特性来替代 XML 文件进行配置。在使用注释还是 XML 文件进行配置的问题上，Java 社区还有很多争论，JBoss Seam 采取了折衷的方式将配置问题交给开发者来决定，开发者可以使用注释进行配

置，也可以使用 XML 文件进行配置并且覆盖注释的配置。

7. 增强的表达式语言

JBoss Seam 提供了标准的统一表达式语言的扩展 JBoss EL。通过 JBoss EL，开发者能够在方法表达式中直接传递参数。

8. 现有Java EE规范和标准的补充

JBoss Seam 之所以声称是下一代的 Java Web 框架是因为 JBoss Seam 开发者对于 Java Web 开发规范的前瞻性设计和实现。和传统的 Java Web 框架主要承担胶水（Glue）功能不同，JBoss Seam 设计目标不仅仅是将各个规范实现整合在一起，提供统一的、开箱即用的开发框架，更加深远的目标是推动 Java EE 规范的演化和进步。JBoss Seam 对多个 Java EE 规范进行了补充和扩展，例如 JSF、Servlet 等。JBoss Seam 本身就推动了一个新的 Web Bean 规范（JSR299）来统一 JSF 和 EJB3 的组件模型。JBoss Seam 对于 Web Bean 规范来说，就像 JBoss Hibernate 对 JPA 规范那样影响深远。

9. 强大的胶水功能

虽然 JBoss Seam 志向高远，但对于开发者来说则更为看中胶水功能。JBoss Seam 在此方面极为强大，包括 AJAX、Facelet、JSF、EJB3、Hibernate、Hibernate Search JBPM、Drools、Email、iText、JMS、Cache、Web Service、JUnit、TestNG 等。多数企业级开发需要的特性，开发者都能在 JBoss Seam 中找到身影。

背景介绍

2005 年对 Java Web 开发者是不同寻常的一年。Ruby On Rails 像一颗重磅炸弹在 Java Web 开发者社区爆炸，刚刚出版不到一年的《Expert One-on-One J2EE Development without EJB》

犹如星星之火点燃了 Java Web 开发者对于笨重的 J2EE 开发的抱怨之火药桶。与 Ruby On Rails 开发者的快速开发相比，可怜的 Java Web 开发者常常使用 Struts 这种小米加步枪的装备开发企业级 Web 应用，使用 J2EE 这门大炮打蚊子。Spring 已经成为较为成熟的轻量级的企业 Web 开发框架。它以星火燎原之势占领了大量 Java Web 企业应用的阵地。这是 Java Web 开发者自下而上的、对 J2EE 规范（特别是 EJB）的革命和颠覆。虽然 J2EE 规范的制订者们意识到这个问题，并且已经成立了 EJB3 JCP 的委员会来挽回失去的阵地，但是其漫长繁冗的进度和过程彻底成就了 Spring。一时间，遵循规范开发企业 Web 应用的 Java Web 开发者几乎成为异类。不走寻常路（摒弃 EJB2.X、采用 Spring 轻量开发），几乎是每个 Java Web 开发者坚定不移的开发方针。J2EE 规范制定者似乎成了彻头彻尾的失败者。

2005 年的 9 月，由 JBoss Hibernate 的架构师 Gavin King 和 JBoss Portal 的开发者 Thomas Heute 启动的 JBoss Seam 项目发布了第一版，同年 10 月，在巴塞罗那举办的 JBoss 用户大会上，JBoss Seam（Seam：缝隙之意，意味深远）项目正式登台亮相。这一年年末，JavaNews 杂志发表的一篇文章引起了许多 Java Web 开发者的关注。Thomas Heute 在这篇文章中介绍了 JBoss Seam 项目：将 JSF 和 EJB3.0 整合在一起，弥补其间的缝隙，提供一个全新的 Java Web 框架，和 Spring 不同，JBoss Seam 项目一开始就准备遵循新制定的 Java EE（Java 5）规范。自此，JBoss Seam 进入了广大 Java Web 开发者的视线。紧接着在当年 12 月举办的 Javapolis（欧洲民间 Java 组织盛会）上，Thomas Heute 关于 JBoss Seam 的演讲使拥有 750 个座位的会场被围得水泄不通，而与此同时，包括其他重量级的产品如 JBoss Protal 等也在作介绍，可见 Java Web

开发者对 Java EE 规范还是抱有恨铁不成钢，但又翘首企盼的复杂心情。

JBoss Seam 引起 Java Web 开发者重视的另外一个原因是 JBoss Hibernate 架构师 Gavin King。JBoss Hibernate 获得如此辉煌的成就奠定了 Gavin King 在 Java 开源社区中的地位和口碑。此时，JBoss Hibernate 已经成为广大 Java Web 开发者用脚投票产生的 Java ORM 事实上的标准。为此，Gavin King 受邀加入了 EJB3.0 规范委员会起草新的 EJB3.0 规范。至今，EJB3.0 规范中的 JPA 规范中仍能映射出 JBoss Hibernate 的设计思想。在 Java EE 规范中，这是少有的实现先于规范的例子之一。继 JBoss Hibernate 之后，Gavin King 又将目光转向了 Java Web 开发。此时 JSF 作为事件驱动的 Java Web 表现层规范大受好评，正处于蓬勃发展的阶段；EJB3.0 规范新鲜出炉。然而这两个规范好像鸡犬之声相闻，却老死不相往来，各人自扫门前雪，哪管他人瓦上霜。他们完全忽视了企业级 Java Web 应用需要将两者平滑地结合使用的需求，将这份工作完全抛给了 Java Web 开发者。JSF 在表现层出色的设计掩盖不了应用层的虎头蛇尾（这其实也不能完全责怪 JSF，毕竟 JSF 开宗明义是表现层规范），Gavin King 直指 JSF 的软肋：受管 Bean（Managed Bean）和上下文模型。甚至对自己参与的 EJB3.0 也不放过：EJB 组件无从知晓 Web 上下文范围和状态，EJB 有状态组件和 Web 上下文管理格格不入。Gavin King 此时抛出了他的解决方案：Web Bean 规范（JSR299）。Web Bean 规范目的明确：一统 JSF 受管 Bean 组件模型和 EJB3.0 组件模型，为 Java Web 开发提供统一的简化模型。Web Bean 委员会由 Gavin King

领导，成员包含了 Adobe、RedHat、Apache、Oracle、Sun、Google 等组织和公司。Web Bean 规范初期草案已经在 2007 年 10 月发布。在此期间，JBoss Seam 项目迅速发展，目前，JBoss Seam 2.0.0GA 已经发布，并正向着 2.1.0 版本进军。熟悉 JBoss Hibernate 和 JPA 规范的开发者马上能够意识到，JBoss Seam 和 Web Bean 规范的关系就是 JBoss Hibernate 和 JPA 规范的今日翻版。实际上，JBoss Hibernate 本身已经超越了 JPA 规范使得 JPA 规范成为 JBoss Hibernate 的子集。这种实现大大超越规范的情况，也将在 JBoss Seam 和 Web Bean 身上再次发生。同时，有了 Web Bean 规范在后方支持，JBoss Seam 的发展之路也将越来越顺畅。

常常有人问起 JBoss Seam 和 Spring 的关系及未来的发展（2007 年底，CSDN2.0 大会上曾有人提出这个问题）。从设计的角度上来说 JBoss Seam 吸收了 Spring 的部分思想，例如注入依赖、拦截器。但本质上两者截然不同，尽管两者都意识到了没落笨重的 J2EE 的重大问题，然而 JBoss Seam 更像改良派，而 Spring 更像革命党。尽管 Java Web 开发者有可能不愿意承认，但实际上技术的选择类似于政治游戏，站错队的风险始终存在。至今为止，仍然没有一个 JSR 获得通过 Spring，它将来也不可能成为 Java EE 规范的一部分（当然 Spring 有可能革命到底，不屑于此）。但是和其他技术的发展轨迹一样，Spring 也日益臃肿，庞杂的 XML 配置也同样引起了 Java Web 开发者的抱怨（不知 JBoss Seam 将来会不会殊途同归）。与此同时，Java EE 规范不断的演进和改良，不知 Java EE 弥补缝隙，东山再起之时，Spring 是否依旧是春天？

参考资料

网站类

1. JBoss网站: www.jboss.org
2. JBoss Seam主页: www.seamframework.org
3. IBM DeveloperWorks对JBoss Seam的介绍: <http://www.ibm.com/developerworks/cn/java/j-seam/>
4. InfoQ对JBoss Seam的介绍: <http://www.infoq.com/cn/articles/jboss-seam>
5. InfoQ对JBoss Seam的技术访谈: <http://www.infoq.com/cn/presentations/mayue-seam-framework>
6. JBoss开发者的Blog站点之一（该站点使用JBoss Seam开发）: <http://in.relation.to>
7. 中国满江红开放文档项目中JBoss Seam中文参考（向其致敬）: <http://www.redsaga.com/opendoc/Seam2.0/html/>
8. Web Bean JSR: <http://www.jcp.org/en/jsr/detail?id=299>

书籍类

1. 《JBoss– Seam》
作者: Michael Juntao Yuan and Thomas Heute Paperback: 432 pages
出版社: Prentice Hall 出版时间: May 6, 2007 Language: English ISBN-10: 0131347969
ISBN-13: 978-0131347960
2. 《Practical JBoss® Seam Projects (Practical)》
作者: Jim Farley Paperback: 229 pages 出版社: Apress 出版时间: July 20, 2007
Language: English ISBN-10: 1590598636 ISBN-13: 978-1590598634
3. 《Seam in Action》
作者: Dan Allen Paperback: 500 pages 出版社: Manning Publications
出版时间: June 15, 2008 Language: English ISBN-10: 1933988401
ISBN-13: 978-1933988405
4. 《Beginning JBoss® Seam: From Novice to Professional》
作者: Joseph Faisal Nusairat Paperback: 376 pages 出版社: Apress
出版时间: February 28, 2007 Language: English ISBN-10: 1590597923
ISBN-13: 978-1590597927

快速上手教程

对于一个 Java Web 框架来说，很难写出一个快速上手的教程，特别是对 JBoss Seam 这类与各种各样的技术和规范关联很深的框架。好在 JBoss Seam 的指南提供了很多的例子。为了达到快速上手的目的，我们先选择其中最简单的一个用户注册的例子。

首先你需要确定你的机器上安装了 JBoss AS 4.2，并且 JBoss AS 能够正常运行。一般情况下你可以通过 <http://localhost:8080> 来查看 JBoss AS 是否正常启动。然后你需要下载 JBoss Seam(参

见 www.jboss.org), 将 JBoss Seam 解压到目录中(例如 D:\JBoss\jboss-seam-2.0.0.GA), 这里我们使用 Window XP)。然后确定你的机器上安装并且配置好了 Ant。你可以在 DOS 终端输入 ant -version 命令来查看 Ant 是否安装和配置, 正常情况下终端显示 Ant 的版本信息。接着我们需要进入 D:\JBoss\jboss-seam-2.0.0.GA 目录下的 build 目录中修改 defalut.build.properties 文件的 jboss.home 属性, 将其指向 JBoss AS 的安装位置(例如: jboss.home D:/Software/jboss); 然后我们进入 D:\JBoss\jboss-seam-2.0.0.GA 目录下的 exmaples/registration 目录输入 ant deploy 命令。该命令将编译和部署 JBoss 的用户注册例子。一切正常的话, 你将可以通过 <http://localhost:8080/seam-registration> 来查看这个简单的例子(见图 1-4)。

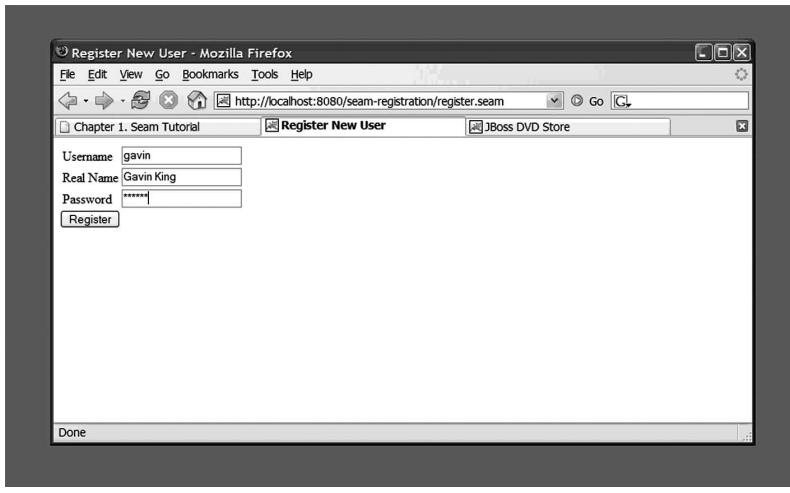


图1-4 Seam注册例子的界面

在 registration 目录下有 3 个目录: view、src、resources。

目录 view 中放置 Web 页面文件: index.html、register.xhtml 和 registered.xhtml。用户通过 register.xhtml 填写注册信息, 完毕后导向 registered.xhtml。

register.xhtml

```
<?xml version="1.0" encoding="utf-8"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml"
  xmlns:s="http://jboss.com/products/seam/taglib"
  xmlns:h="http://java.sun.com/jsf/html"
  xmlns:f="http://java.sun.com/jsf/core">

  <head>
    <title>Register New User</title>
  </head>
  <body>
    <!-- JSF视图 -->
    <f:view>
      <!-- 表单 -->
```

```

<h:form>
    <!--验证 -->
    <s:validateAll>
        <h:panelGrid columns="2">
            Username: <h:inputText value="#{user.username}"
                required="true"/>
            Real Name: <h:inputText value="#{user.name}"
                required="true"/>
            Password: <h:inputSecret value="#{user.password}"
                required="true"/>
        </h:panelGrid>
    </s:validateAll>
    <h:messages/>
    <!--绑定register方法 -->
    <h:commandButton value="Register" action="#{register.
        register}"/>
</h:form>
</f:view>
</body>

</html>

```

registered.xhtml

```

<?xml version="1.0" encoding="utf-8"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
    "http://www.w3.org/TR/xhtml1/DTD/xhtml1-
transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml"
    xmlns:f="http://java.sun.com/jsf/core">

    <head>
        <title>Successfully Registered New User</title>
    </head>
    <body>
        <!--JSF将会要求Seam解析user变量，并显示其属性name和username -->
        <f:view>
            Welcome, #{user.name}, you are successfully registered as #{user.
                username}.
        </f:view>
    </body>

</html>

```

目录src中放置Java源代码。进入包目录中有三个Java文件和一个test目录。三个Java文件分别是: User.java、Register.java、RegisterAction.java。查看源代码会发现这个注册例子使用了EJB3.0。

User类是一个实体类。

```

//$Id: User.java,v 1.10 2006/10/25 20:17:46 gavin Exp $
package org.jboss.seam.example.registration;

import static org.jboss.seam.ScopeType.SESSION;

import java.io.Serializable;

import javax.persistence.Entity;
import javax.persistence.Id;
import javax.persistence.Table;

import org.hibernate.validator.Length;
import org.hibernate.validator.NotNull;

```

```
import org.jboss.seam.annotations.Name;
import org.jboss.seam.annotations.Scope;

@Entity //声明User类是实体类
@GeneratedValue(strategy=GenerationType.AUTO) //自动增加
@Name("user") //声明User类是Seam组件，组件名为user
@Scope(SESSION) //声明user组件的范围是会话
@Table(name="users") //声明user组件在数据库中使用users表来映射
public class User implements Serializable
{
    private static final long serialVersionUID = 1881413500711441951L;

    private String username;
    private String password;
    private String name;

    public User(String name, String password, String username)
    {
        this.name = name;
        this.password = password;
        this.username = username;
    }

    public User() {}

    @NotNull //声明name属性不能为null
    public String getName()
    {
        return name;
    }

    public void setName(String name)
    {
        this.name = name;
    }

    @NotNull @Length(min=5, max=15) //声明密码长度为5-15
    public String getPassword()
    {
        return password;
    }

    public void setPassword(String password)
    {
        this.password = password;
    }

    @Id @NotNull @Length(min=5, max=15) //声明以用户名为数据库ID
    public String getUsername()
    {
        return username;
    }

    public void setUsername(String username)
    {
        this.username = username;
    }

    @Override //声明覆盖toString()方法
    public String toString()
    {
        return "User(" + username + ")";
    }
}
```

Register 类是本地接口

```
//$Id: Register.java,v 1.4 2006/09/28 00:36:56 gavin Exp $  
package org.jboss.seam.example.registration;  
  
import javax.ejb.Local;  
  
@Local //声明本地接口  
public interface Register  
{  
    public String register();  
}
```

RegisterAction 类是无状态的会话 Bean。

```
//$Id: RegisterAction.java,v 1.20 2007/06/20 00:08:47 gavin Exp $  
package org.jboss.seam.example.registration;  
  
import java.util.List;  
  
import javax.ejb.Stateless;  
import javax.persistence.EntityManager;  
import javax.persistence.PersistenceContext;  
  
import org.jboss.seam.annotations.In;  
import org.jboss.seam.annotations.Logger;  
import org.jboss.seam.annotations.Name;  
import org.jboss.seam.faces.FacesMessages;  
import org.jboss.seam.log.Log;  
  
@Stateless //声明是无状态会话Bean  
@Name("register") //声明RegisterAction类是Seam组件，组件名为register  
public class RegisterAction implements Register //实现Register本地接口  
{  
  
    @In //从Seam上下文中注入user组件实例  
    private User user;  
  
    @PersistenceContext //注入EJB3.0的实体管理器  
    private EntityManager em;  
  
    @Logger //注入日志  
    private static Log log;  
  
    public String register()  
    {  
        //使用EL将user组件的username属性传递给实体管理器创建的查询  
        List existing = em.createQuery("select u.username from User u where  
            u.username=#{user.username}")  
            .getResultList();  
  
        if ( existing.size()==0 )  
        {  
            //实体管理器持久化user组件  
            em.persist(user);  
            log.info("Registered new user #{user.username}");  
            //为简单起见使用硬编码进行页面导航，Seam提供基于工作流的页面导航  
            return "/registered.jspx";  
        }  
        else  
        {  
            //设置Faces消息  
        }  
    }  
}
```

```

        FacesMessages.instance().add("User #{user.username} already
exists");
return null;
}
}

}

```

整个应用的交互如图 1-5 所示：

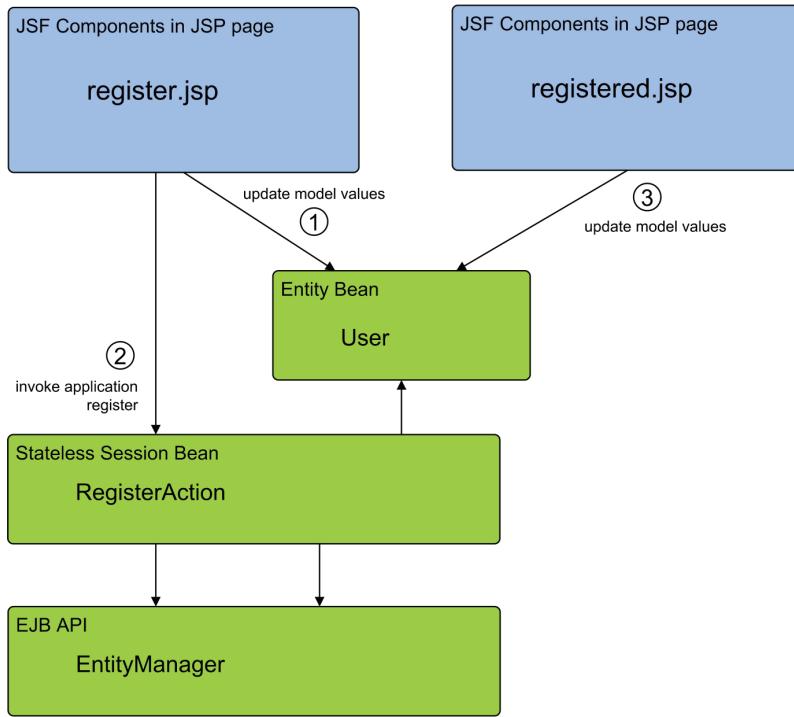


图1-5 Seam注册例子的交互示意

当表单提交的时候（1），JSF 将会要求 JBoss Seam 解析 User 变量，而此时 Seam 的上下文中并没有这个名为 User 的组件实例（Seam 将在不同范围的上下文中依次查询），因此 Seam 将实例化名为 User 的 Seam 组件，将其存储在相应的 Seam 上下文中，然后将 User 实体 Bean 实例返回给 JSF。接下来 Seam 使用 Hibernate 验证器验证在 User 实体类中设置的约束。如果不能满足约束条件，JSF 将会重新显示页面，并且附加上错误消息。如果通过验证，JSF 将表单的值和相应的 User 实体类的属性绑定。

接下来 JSF 会要求 JBoss Seam 解析 register 变量（2）。Seam 在无状态上下文中找到名为 register 的无状态会话 Bean: RegisterAction，并将其返回给 JSF，JSF 调用其 register 监听方法。此时 Seam 会探知该方法调用，并且拦截这个方法，将会话上下文中的 User 实体类的实例注入。register 方法会检查用户名是否已经存在，并且做相应处理。如果已经存在，就将错误消息在 FacesMessage 组件中排队，并且返回 null 值，引起页面重新渲染，将会显示错误消息。如果不存在，register 方法将持久

化 User 实体类的实例，页面将会导向 registered.xhtml (3)。JSF 将会要求 Seam 解析 User 变量，然后将会话上下文中的 User 实体类的实例的属性显示在页面上。

我们再来看看 resources 目录，这个目录放置 JBoss Seam 应用的配置文件。这个目录下有两个目录 META-INF 和 WEB-INF，以及两个文件 components.properties 和 seam.properties。需要特别注意的是 seam.properties 文件。该文件对于 JBoss Seam 应用是必须存在的，除了可以在该文件中配置 JBoss Seam 应用之外，更重要的意义在于告知 JBoss Seam 如何快速加载 Seam 组件 (JBoss Seam 将会扫描 seam.properties 文件的位置利用其定位加载 Seam 组件)。所以，即使 seam.properties 文件为空也必须存在。

WEB-INF 目录下包含三个配置文件：web.xml、faces-config.xml、components.xml。

我们需要在 web.xml 设置 JBoss Seam 监听器。

```
<?xml version="1.0" encoding="UTF-8"?>

<web-app version="2.5"
  xmlns="http://java.sun.com/xml/ns/javaee"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://java.sun.com/xml/ns/javaee http://java.sun.
  com/xml/ns/javaee/web-app_2_5.xsd">

  <!--设置Seam监听器 -->

  <listener>
    <listener-class>org.jboss.seam.servlet.SeamListener</listener-
      class>
  </listener>

  <context-param>
    <param-name>javax.faces.DEFAULT_SUFFIX</param-name>
    <param-value>.xhtml</param-value>
  </context-param>

  <servlet>
    <servlet-name>Faces Servlet</servlet-name>
    <servlet-class>javax.faces.webapp.FacesServlet</servlet-class>
    <load-on-startup>1</load-on-startup>
  </servlet>
  <!--Seam URL后缀 -- >
  <servlet-mapping>
    <servlet-name>Faces Servlet</servlet-name>
    <url-pattern>*.seam</url-pattern>
  </servlet-mapping>

  <session-config>
    <session-timeout>10</session-timeout>
  </session-config>
</web-app>
```

JBoss Seam 鼓励使用 Facelet。

```
<?xml version="1.0" encoding="UTF-8"?>
<faces-config version="1.2"
  xmlns="http://java.sun.com/xml/ns/javaee"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
```

```

xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
http://java.sun.com/xml/ns/javaee/web-facesconfig_1_2.xsd">

<!--配置Facelet视图处理器 -->
<application>
    <view-handler>com.sun.facelets.FaceletViewHandler</view-handler>
</application>

</faces-config>

```

JBoss Seam 除了支持注释进行组件配置之外，也可以使用 XML 文件进行组件配置，这个文件就是 components.xml。

```

<?xml version="1.0" encoding="UTF-8"?>
<components xmlns="http://jboss.com/products/seam/components"
    xmlns:core="http://jboss.com/products/seam/core"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation=
        "http://jboss.com/products/seam/core http://jboss.com/
         products/seam/core-2.0.xsd
        http://jboss.com/products/seam/components http://jboss.
         com/products/seam/components-2.0.xsd">
    <!--配置内置的core:init组件，设置jndi模式，注意@jndiPattern@将由Ant脚本在布署的时候替换为正确的值 -- >
    <core:init jndi-pattern="@jndiPattern@"/>

</components>

```

由于这个例子比较简单且使用注释较多，所以组件的配置信息并不多，你可以参考其他例子来获得更加详细的信息（如 DVD Store 例子）。

在 META-INF 目录下包含 4 个配置文件：ejb-jar.xml、persistence.xml、application.xml、jboss-app.xml。

ejb-jar.xml 文件配置 JBoss Seam 拦截器，拦截会话 Bean，从而整合 JBoss Seam 和 EJB3。

```

<?xml version="1.0" encoding="UTF-8"?>
<ejb-jar xmlns="http://java.sun.com/xml/ns/javaee"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://java.sun.com/xml/ns/javaee http://
        java.sun.com/xml/ns/javaee/ejb-jar_3_0.xsd"
    version="3.0">
    <!--配置JBoss Seam拦截器 -- >
    <interceptors>
        <interceptor>
            <interceptor-class>org.jboss.seam.ejb.SeamInterceptor</interceptor-
                class>
        </interceptor>
    </interceptors>

    <!--将拦截器和所有会话EJB绑定 -- >
    <assembly-descriptor>
        <interceptor-binding>
            <ejb-name>*</ejb-name>
            <interceptor-class>org.jboss.seam.ejb.SeamInterceptor</
                interceptor-class>
        </interceptor-binding>
    </assembly-descriptor>

</ejb-jar>

```

persistence.xml 文件配置持久化单元。

```
<?xml version="1.0" encoding="UTF-8"?>
<persistence xmlns="http://java.sun.com/xml/ns/persistence"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://java.sun.com/xml/ns/persistence
        http://java.sun.com/xml/ns/persistence/persistence_1_0.xsd"
    version="1.0">
    <!--使用Hibernate作为ORM -- >
    <persistence-unit name="userDatabase">
        <provider>org.hibernate.ejb.HibernatePersistence</provider>
        <!--JBoss AS使用Hypersonic DB作为默认数据源 -- >
        <jta-data-source>java:/DefaultDS</jta-data-source>
        <properties>
            <!--自动创建和删除数据库表 -->
            <property name="hibernate.hbm2ddl.auto" value="create-drop"/>
        </properties>
    </persistence-unit>
</persistence>
```

application.xml 文件是标准的 EAR 描述文件。

```
<?xml version="1.0" encoding="UTF-8"?>
<application xmlns="http://java.sun.com/xml/ns/javaee"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://java.sun.com/xml/ns/javaee http://
        java.sun.com/xml/ns/javaee/application_5.xsd"
    version="5">

    <display-name>Seam Registration</display-name>
    <!--用户注册应用war文件 -- >
    <module>
        <web>
            <web-uri>jboss-seam-registration.war</web-uri>
            <context-root>/seam-registration</context-root>
        </web>
    </module>
    <!--用户注册代码jar文件 -- >
    <module>
        <ejb>jboss-seam-registration.jar</ejb>
    </module>
    <!--JBoss Seam jar 文件 -- >
    <module>
        <ejb>jboss-seam.jar</ejb>
    </module>
</application>
```

jboss-app.xml 文件是 JBoss 特定的 EAR 描述文件。

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE jboss-app
PUBLIC "-//JBoss//DTD J2EE Application 4.2//EN"
"http://www.jboss.org/j2ee/dtd/jboss-app_4_2.dtd">

<jboss-app>
    <!--指定应用类加载 -- >
    <loader-repository>
        seam.jboss.org:loader=seam-registration
    </loader-repository>
</jboss-app>
```

至此，我们大致了解了一个 JBoss Seam 应用的结构。JBoss Seam 整合的其他技术和规范没有在此详细讲解，JBoss Seam 自带的例子比较丰富，若有兴趣可以查看 examples 目录下的例子，比较完整的例子有 DVD Store 和 Wiki（现在的 JBoss Seam 站点 www.seamframework.org 就是基于这个 Wiki 开发的）。

为了方便开发者，JBoss Seam 自带了一个 Seam-gen 程序（本身是一个复杂的 Ant 脚本）。通过 Seam-gen，开发者能够迅速架设 Seam 应用的结构，而且 Seam-gen 提供了默认的帮助类和界面来加速开发和调试。关于 Seam-gen 的使用请参见 JBoss Seam 参考文档。

版本信息

目前 JBoss Seam 的稳定版本是 JBoss Seam 2.0.0GA，下一个版本 JBoss Seam 2.0.2GA 将在不久发布。该版本将会修复 30 多处 Bug，并且完成约 20 多项任务。

社区视角

不可否认的一点是，尽管 JBoss Seam 从设计上讲是轻量级 Web 框架（有哪个 Web 框架会说自己是重量级的呢？），但它对 Java Web 开发者的要求和挑战是比较高的。这种要求和挑战不是来源于 JBoss Seam 本身的使用，而是来源于其整合的各个技术和规范。经典的 JBoss Seam 应用以 JSF+EJB3 架构为主。在表现层，当然要求开发者对 JSF 比较了解，然而这还不够，要想达到企业级（该词有点滥用）或者展现丰富的表现层，开发者必须对 Facelet（一种模板语言）、RichFaces、ICEFaces（两者都是 Java Ajax 组件框架）、JBoss Seam Remote 有所了解，这样才能充分展示 JBoss Seam 在表现层的强大；在应用层，开发者需要比较了解 EJB3.0 和 Hibernate，同时要熟练掌握 Java 5 的注释功能，如果企业

应用需要工作流，当然开发者还必须了解 JBoss JBPM 工作流引擎，甚至，当企业安全需要规则来控制的时候，开发者还要了解 JBoss Drools 规则引擎。因此，JBoss Seam 为拥有企业级技术水平的开发者提供了一个一站式的 Web 开发框架。它的目标人群并非 Java Web 开发的新手。对于他们，建议首先学习 JBoss Seam 整合的各个技术和规范，然后再使用 JBoss Seam 开发应用。

JBoss Seam 提供了丰富的范例，有很多都是开箱即用的完整应用，例如 DVD Store、Hotel Booking 应用。这些丰富的范例能够帮助开发者迅速上手。

关于开发工具，JBoss Seam 本身提供了一个命令行的快速开发脚本（其实是较为复杂的 Ant 脚本）seam-gen。开发者还可以使用免费的 JBoss Tools（一个 Eclipse 的插件，通过 seam-gen wrapper 支持 JBoss Seam），或者使用付费的（\$99 美元）JBoss Developer Studio JBoss Seam 进行开发。对于 Netbeans 的爱好者，来自 Sun 的一位开发者开发了 seam-gen 的 Netbeans 插件，该插件可以在 Netbean5.5 上运行，但在 Netbeans6.0 上需要小的修改才能正常运行。无论如何，JBoss Seam 的开发工具还有很大的提升空间。

作者介绍

Sean Wu, 吴昕 博士, 毕业于北京大学生物信息中心。攻读博士学位期间主要研究方向为数字化基因表达谱分析系统和分布式生物信息计算平台的研发。2000 年投身于科研领域的开源运动, 参与以及推广 BioJava(生物信息领域的开源 Java 项目) 项目, 曾参与多项生物信息领域国家 863 项目并且获得多项软件著作权。目前就职于北京中通软联信息科技有限公司, 致力于 Java 开源技术在企业和科学研究中的应用和推广。

2

动态语言篇

综述

如果是在两年前，我们来谈论动态语言，耳边常常听到的是对这个不算新鲜的新鲜事物的质疑之声。人们常常会问，这个语言真的比 Java 强大吗？它的学习曲线一定很陡峭吧？像这样的语言很多，而且大都昙花一现，还是学习 Java 更安全一些吧？……在今天，当你看到关于这些语言的讨论组对它们的热火朝天的讨论，看到出版商们对它们的追捧，看到媒体对它们的及时报道，我想你可能对过去的那些质疑一笑置之。这就是生活，当你处于黎明前的黑暗时，肯定会遭受来自各方的怀疑，当你冲破阻挠见天日时，乌云也会逐渐散去。

在解释动态语言之前，首先我们需要来了解一下什么是静态语言。简而言之，静态语言的数据类型是在编译期间检查的，也就是说在写程序时要声明所有变量的数据类型，C、C++、C# 和 Java 都是静态类型的典型代表。静态语言的主要优点在于其结构非常规范、便于调试、方便类型安全，缺点是为此需要写更多类型相关的代码，最终导致代码不便于阅读。相反，动态语言是指在运行期间才去做数据类型检查的语言，也就是说，在用动态语言编程时，不需要给任何变量指定数据类型。它会在你第一次赋值给变量时，在内部将数据类型记录下来。像 Ruby、Python 和 Groovy 等都属于典型的动态语言。归结来看动态语言的优点是方便阅读，不需要写非常多的类型相关的代码，当然其缺点是不方便调试，当命名不规范时会造成对代码的理解困难等。

从动态语言的发展过程来看，它的产生和互联网应用的兴起有着莫大的关系。比如 Ruby 语言，其实它在很多年前就已经被实现了，但直到 Ruby on Rails 的出现，Ruby 才为更多的人关注。现在利用 Ruby on Rails 构建 Web 应用的组织也越来越多，据说那些在硅谷基于互联网创业的人有近 80% 都选择了 Ruby on Rails。在国内技术社区，我们也可以看到许多新兴的网站在做技术选型时也多选择了动态语言，比如圈内网是基于 Ruby on Rails 的，豆瓣网是基于 Python 的等。和 Ruby 的发展类似，Groovy 也是如此，Grails 的出现让更多原来基于 Java 做开发的人士转投 Groovy，因为 Groovy 很好地继承了 Java 的语法，而 Grails 又能帮助他们方便地搭建所需的 Web 程序。如果我们注意一下现在

社区内举办的一些技术活动，就会发现 Web 开发是一个非常热门的话题，而在这个话题里，动态语言又是不容忽视的主角之一。

每一种语言的背后都蕴藏着作者和该社区自己的哲理，而每一个语言的特性也都包罗万象，要在短短几页之内将它们都介绍清楚非笔者能力所及，也并非笔者的本意。笔者希望将几个典型动态语言的定义、框架结构及简单的应用等整理在这儿，让读者能够对动态语言的过去和现状有一个比较好的感性的认识。

关联信息

根据动态语言的定义，“所谓‘动态语言’，也叫脚本语言，就是说一种在执行期间才去发现数据类型的程序设计语言，主要创建一些需要经常更新的动态系统”，包括 Python、Ruby 和 Groovy 在内，我们可以将 PHP、ActionScript、JavaScript、Erlang、Perl 等都归属于动态语言，关于 Python、Ruby 和 Groovy 我们会在后面详细分析，这里对其他几种语言做简要介绍。

PHP 是一种开源的 HTML 内嵌式语言，在服务器端执行，语言风格类似于 C 语言。PHP 最初是 1994 年由 Rasmus Lerdorf 创建的，当时仅是一个用 Perl 语言编写的简单程序，后来经过几番扩充，增加了访客留言本、计数器等功能，也赢得越来越多的网站的青睐。

ActionScript 是一个遵循 ECMAScript 第 4 版的 Adobe Flash Player 运行时环境编程语言，由 Flash Player 中的 ActionScript 虚拟机来执行，主要在 Flash 内容和应用程序中实现交互性、数据处理及其他的功能。

JavaScript 是一种由 Netscape 公司的 LiveScript 发展而来的脚本语言，主要目的是帮助解决服务器端语言如 Perl 等带来的遗留速度问题。JavaScript 使得网页具有交互性功能，能够及时响应用户的操作，对用户所提交的表单做即时的检查，从而节省了服务器端的验证等。

Erlang 是一个结构化的动态编程语言，内建并行计算支持，这也是其被广泛应用于分布式计算系统的原因之一。Erlang 最初是由爱立信专门为通信而设计的，比如控制交换机或变换协议。

Perl 是 Practical Extraction and Report Language 的简称，距今已经有 20 多年的历史，最初设计者是 Larry Wall。使用 Perl 可以很容易地操作数字、文本、文件、目录、计算机和网络等。Perl 的一个非常独特之处是可以很容易地在任何现代的操作系统上进行移植。

对于 Python、Ruby 和 Groovy 等三种动态语言，通常情况下，都会结合相应的 Web 框架来使用，比如 Python 和 Django，Ruby 和 Rails，Groovy 和 Grails 等，在下面的介绍中我们也是基于这个关系来对它们进行分析的。

2.1 Python

总评

上手容易程度	★★★★★
社区活跃性	★★★★
应用广泛性	★★★★★
推荐指数	★★★★★

一直关注语言流行度变化的网站 TIOBE 宣布将 Python 列为 2007 年度程序语言，入选理由为：在过去的一年里，Python 的占有上升了 2.04%，在 2007 年 12 月更是历史性地首次超过 Perl，位列最受欢迎的程序语言第 6 位。从此数据中可以看到，Python 正在赢得越来越多开发人员的喜爱，尤其是系统管理员和项目构建工程师等。

功能和特点

Python 是一种即译式的、互动的和面向对象的编程语言，另外它还是开源的。作为一种脚本语言，它特别强调开发速度和代码的清晰程度。Python 可以用来开发各种程序，从简单的脚本任务到复杂的面向对象的应用程序都有其发挥作用的地方。另外，因为 Python 免费、面向对象、可扩展性强，以及执行严格的编码标准等特点，它还常被当作一种程序员的入门选择语言。

Python 是免费的。Guido van Rossum 于 1990 年开始开发 Python，最初只是将它作为一个自娱自乐的项目，但随着不断的发展，它逐渐成为可以解决许多问题的优秀工具。Guido 目前

是 Python 软件基金会的主席，该组织按照 GNU 公关许可协议的要求拥有 Python 的知识产权和许可权。目前，Python 可以用在包括 Windows、Mac 和各种常见的 UNIX 系统中，针对 Palm OS 和 Pocket PC 的相应版本也在开发之中。

通过 Python 可用高级编程技术创建脚本。虽然 Python 常常被用来创建简单的脚本，但是它所使用的技术却包括面向对象的、面向套接字的、面向线程的和面向 GUI 的等，这些都是高级语言所必备的特性。学习了 Python 之后，转而去研究像 Java 或者 C++、C# 等语言就相对容易许多，当然多数情况下你压根儿就不用再学习这些语言。相对于其他面向对象的语言，Python 强调空格和编码结构，从而使得开发者的代码具有良好的重用性，另外就是 Python 强调动态性，即执行脚本之前无需对 Python 代码进行编译。

Python 易于扩展，它可以针对语言无法完成相应功能的函数调用函数库，另外，Python 自身还可以嵌入到其他编程语言或平台中，比如用 Java 编写的 Python 解释器，以及基于 .NET 平台的 IronPython 等。开发人员可以用 C、C++ 或者 Java 等为 Python 撰写新的模块，比如函数等。也可直接与 Python 编译在一起，或者采用动态库的装入方式来实现等。

将 Python 用于原型开发工具。软件的构建并不总是能够一步到位的，很多时候我们需要不断地尝试。在 Python 下面创建对象，可以使用较其他语言更少的代码和时间来发挥同样的功能，比

如支持继承、多态性等。一个比较常用的方法是先在 Python 中原型出一个程序，经过调试后认为正确了再切换到 Java 或 C++ 中。

将 Python 作为粘合剂。这个特点对于那些计划或正在集成多个系统的开发人员会比较有用。目前 Python 支持几乎所有的集成关键技术，同时还能很好地与文件、协议、动态链接库及 COM 对象等协同工作。另外，Python 还提供广泛的类库帮助你得到任何种类的数据等。

Python 还有其他许多优秀的特点，不论是初级开发人员还是有经验的高手，都会发现 Python 是一种相当灵活和强大的编程语言，使用它可以开发出各种各样的应用程序。

背景介绍

和许多语言的诞生一样，Python 的出现也可以算得上是一个无心之举。在 1989 年的圣诞节期间，Guido van Rossum 在阿姆斯特丹闲得实在无聊，于是决定开发一个新的脚本解释程序，以替代其最喜爱的 ABC 语言。说到 ABC，这个语言也和 Guido 有着密不可分的联系，在开始的时候，Guido 也参与了这一教学语言的设计。就其本人看来，ABC 语言非常优美和强大，是专门为非专业程序员而设计的。但是美中不足的是，ABC 语言不是一个开放的语言，这让 Guido 一直感觉很不爽。Python 的诞生从某种意义上来说是为了避免 ABC 的不开放特征，同时 Guido 也想实现一些在 ABC 语言中没有被实现的功能。之所以选择用 Python 作为这个语言的名字，也并不是说 Guido 其人非常喜欢这个庞大又不漂亮的动物，而是因为他曾经是 BBC 当时热播的喜剧 Monty Python 的粉丝。

就这样，Python 在那个别人欢闹而 Guido 无聊的圣诞夜晚诞生了，当时 Guido 在 Stichting

Mathematisch Centrum (CWI) 工作，在其离开 CWI 之前 Python 发布的最后版本是 Python 1.2。1995 年，Guido 在 Corporation for National Research Initiatives (CNRI) 继续进行 Python 的工作，期间他又发布了多个 Python 版本，直到 Python 1.6。这一版本发布后，让 Python 和以 GPL 协议发布的软件协同工作的需求逐渐明朗起来。CNRI 和自由软件基金会进行了接触，德奥对 Python 协议改动了许可，所以仔细一些的开发人员会发现，随后发行的 Python 1.6.1 虽然功能和 Python 1.6 相同，但却是以不同的协议发布的。从此以后发布的 Python 版本都兼容 GPL。2000 年，Guido 和 Python 的核心团队转移到 BeOpen.com，形成 BeOpen PythonLabs 团队，Python 2.0 是第一个也是唯一一个由 BeOpen.com 发布的版本。

随后，Guido 和其他的 PythonLabs 的开发人员一起加入了 Digital Creations 公司，也是从那个时候开始，和 Python 相关的所有知识产权都有非盈利组织 Python 软件基金会所有。Python 的发展也引起了许多大公司的重视，其中就包括互联网新贵 Google。在 Google 的盛情邀请下，Guido 在 2005 年进入 Google 公司。而 Google 给他的任务也很简单，就是用 Python 语言开发一些为 Google 程序员所使用的工具，以提高他们的工作效率。这只需要占用 Guido 一半的时间，另一半时间他可以专心开发他的 Python 语言。也许恰因为如此，现在 Python 前进的脚步比从前有力得多，也快速得多。

在进入 Google 公司后，Guido 得以有更多的时间设计和规划 Python 3000。如 Guido 在他的博客中所提到的那样，长久以来，Python 已经积累了许多让其不安的设计和瑕疵，如果不破坏向后兼容性根本就没有办法修正它们，于是为了 Python 日后能够轻装上阵，他开始着手设计第一

个不考虑向后兼容性的 Python 版本——Python 3000。经过近两年的研发，在 2007 年 8 月 31 日，Python 3000 的 Alpha 1 对外发布并提供下载。目前关于 Python 3000 的最新版本是 alpha 4，于 2008 年 4 月 2 日对外提供下载。

参考资料

书籍类

1. 《Python语言入门》

作者：Mark Lutz、David Ascher 译者：陈革、冯大辉 出版社：中国电力出版社

出版时间：2001年4月 ISBN：7508305809

这本书出版的较早，也很经典。对于一名有经验的程序员，快速阅读此书可以大体了解Python语言的核心。另外，本书也讲到了Python语言的许多易学的特性，并辅以翔实的代码解释，在Python的入门级图书中，《Python语言入门》不失为一部经典之作。

2. 《Python编程（第三版·英文影印版）》

作者：Mark Lutz 出版社：东南大学出版社 出版时间：2000年10月 ISBN：7564105704

某种意义上来说，这本书已经成为Python用户的行业标准，而且更加完善。在前两版的基础上，《Python编程（第三版）》进行的更新反映了当前的最佳实践，以及在Python 2.5中所引入的最新改变。在书中，作者清晰简练地解释了Python语言的语法及编程技巧，并辅以大量例子阐明正确的使用方法和通用特性，从而使得读者能够学习到如何将Python运用到实际的工作中，比如GUI编程、并行处理、网络应用程序、Internet脚本编程和数据库管理等。

3. 《Python技术手册（影印版）》

作者：Alex Martelli 出版社：东南大学出版社 出版时间：2006年 ISBN：9787564105761

对于开发人员而言，如果手边没有一本自己常用语言的参考手册，那绝对是一种缺憾。当Python程序员在回忆或解释这种开源语言的语法及它的众多强大但又缺乏文档的模块而获得帮助的时候，本书为他们提供了参考。本书所提供的内容不仅仅是关于Python语言本身，还包括标准库中经常使用的部分，以及最重要的第三方扩展。对于在.NET平台上工作的人们，还可以在本书中看到微软关于IronPython项目的一些信息。

4. 《Python网络编程基础》

作者：John Goerzen 译者：莫迟等 出版社：电子工业出版社 出版时间：2007年6月
ISBN：9787121044953

对于使用Python从事网络编程的朋友而言，本书可能是一个不错的选择。从编排角度来看，这本书非常清晰，几乎涵盖了网络编程的所有方面。从传统的FTP、E-mail到较新的XML、Web服务，以及当前流行的多线程和异步通信等，都做了详细的介绍。比较难得的是，在书中作者还给出了大量直接或稍加修改就可以使用的例子。

网站类

1. Python官方网站：<http://www.python.org>

关于Python的最新新闻、文档、下载等都可以在这里找到，另外，网站还收集了Python在各个领域中的应用案例，对于计划在实际工作中使用Python的人来说一定会有比较大的帮助。如果在使用Python过程中发现Bug等，也可以通过该网站向Python团队提交。

2. Guido的博客：<http://www.artima.com/weblogs/index.jsp?blogger=guido>

Python之父会不定期更新他的博客，来发布关于Python的一些最新进展，虽然很多消息在Python官方站点也能看到，但很多来自大师的第一手资料还是很难得的。另外，你说不定还能看到Guido对那些对Python持有不同意见的人的回应，是不是很酷？

3. Python中文社区：<http://python.cn/>

CPUG，全称是China Python User Group，据说是第一个正式成立的民间Python用户组织，由广大Python爱好者推动而成立。内容包括用于知识分享的Wiki、用于组织项目开发的Trac和用于网志汇集的Planet等。

4. 华蟒用户组：<http://groups.google.com/group/python-cn/>

顾名思义，这是一个中文的针对Python的讨论社区，口号是“简洁就是美”。除了python-cn主站点讨论，该用户组下面还分珠江事务、北京事务、安徽事务、华东南事务和中国事务等，组织性很强。

5. 啄木鸟：<http://wiki.woodpecker.org.cn/moin/>

一个开放的协作组织，关注Python语言的各种应用。

快速上手教程

下面让我们来看一下如何用Python编写运行一个传统的“Hello World”程序。通过它，我们可以了解到如何编写、保存和运行Python程序。和其他语言相比，Python也有两种运行程序的方式——交互式的带提示符的解释器或使用源文件。

如果是使用带提示符的解释器，那么我们先在命令行的shell提示符下键入python，启动解释器。现在输入print 'Hello World'，然后再按回车键。这时，你应该可以看到输出的单词Hello World。对于Windows用户，只要设置了正确的PATH变量，就可以从命令行启动解释器。如果你使用的是IDE工具IDLE，那可能会稍微麻烦一点，首先输入下面的代码段：

```
$ python
Python 2.4.3 (#1, Jul 26 2006, 16:42:40)
[GCC 3.4.2 20050110 (Red Hat 3.4.2-6.fc3)] on linux2
Type "help", "copyright", "credits" or "license" for more information.
>>> print 'hello world'
```

然后将会发现在接下来的一行会出现我们想要的结果：

```
hello world
>>>
```

如果我们使用的是源文件，就需要启动一个编辑器，并在编辑器中输入下面的代码：

```
#!/usr/bin/python
# Filename : helloworld.py
print 'Hello World'
```

保存完毕后，打开 shell（Linux 终端或 DOS 提示符），然后键入命令 `python helloworld.py`。如果你使用 IDLE，就使用菜单 `Edit->Run Script` 或使用键盘快捷方式 `Ctrl-F5`。然后就会看到如下的输出：

```
$ python helloworld.py  
Hello World
```

如果你得到的输出与上面所示的一样，那么恭喜——你已经成功地运行了你的第一个 Python 程序。

版本信息

Python 现在稳定的版本是 2.5，相较于 Python 2.4，这一版本修正了 450 个 Bug 和 350 个补丁。从 Python 官方的文档里我们可以了解到，Python 2.5 比以前速度更快，在错误处理和字符串操作等方面做了很好的优化；另外还增加了一些新的语言特性，比如在产生字节码之前，先在 Python 编译器会将源代码转化成一个抽象的语法树（AST）；修改了 `try/except/finally`，现在可以为同一个 `try` 程序块使用 `except` 和 `finally` 等；还更新了一些模块和包。

现行 Python 项目的测试版是 Python 2.6，Python 3000 日前也已经发布了 alpha 4 版本，预计在 2008 年 8 月发布 Python 3000 的正式版。

社区视角

Python 的诸多优点我们在前文已经做了比较详细的叙述，但是在使用 Python 时还是需要注意它的局限性。比如，Python 程序是强制要求缩进的，虽然这不能说是一个缺陷，但是通过缩进来区分语法关系的方式还是给很多初学者带来困惑。即便非常有经验的 Python 程序员，也常会

陷入这个陷阱之中，最常见的情况就是由 Tab 和空格的混用所导致的错误。另外需要注意的是在 Python 中，很多时候不能将程序连写成一行。

Python 目前的发展，可以说是一直处于稳步上升之中，根据前面我们所提到的 TIOBE 排名，现在 Python 已经在 Java、C、VB、PHP 和 C++ 之后列第 6 位了。对于这样一个产生时间较长，但近几年才焕发青春的语言来说着实不易。另外，微软也已将 Python 纳入到其 .NET 平台，相信 Python 将来的发展会更加强劲。在厂商方面，Google 对 Python 的拥抱也会加速这一语言的普及。

在选择使用 Python 构建 Web 应用时，Web 框架 Django 是不二之选。首先，Django 是用 Python 编写的，和 Python 可以完美地融合。通过使用 Django，可以使得开发人员免去构建动态 Web 站点的痛苦，因为 Django 已经将常见的 Web 开发问题进行抽象化，并提供对使用频繁的编程任务进行简化的功能。在设计 Django 时，作者所秉持的哲学就是竭尽所能地提升开发速度，一个熟悉 Django 的开发人员，可以在几小时之内构建出一个相对完善的 Web 应用。《The Django Book》一书的作者曾经说过一句话——Django 是为行业中 Web 开发团队解决日常问题而构建的。

2.2

Ruby

总评

上手容易程度	★★★★★
社区活跃性	★★★★
应用广泛性	★★★★★
推荐指数	★★★★★

有消息称现在硅谷有近 80% 的项目是基于 Ruby 语言的，而国内的一些新兴网站比如圈内网、技能云，包括技术社区 JavaEye 网站等也都采用了 Ruby on Rails 技术。在社区支持方面，这两年关于 Ruby 的讨论从来就没有停止过，比如 O'Reilly、InfoQ、IBM developerWorks 等知名技术媒体专门开设了 Ruby 专区，而 Rubyforge 上也多有新的项目发布。Sun 公司自从和 JRuby 团队结缘以来，持续加大投入，并在 2007 年 5 月推出了 JRuby 1.0，微软在 MIX 07 大会上也宣布了在 .NET CLR 之上的 Ruby 实现——IronRuby。

功能和特点

Ruby 是一种纯粹、动态的 OOP 语言。下面我们来看一下 Ruby 语言的一些特征：

1. Ruby 是一种敏捷的语言，它是可塑的，能够很容易地进行比较频繁的重构；
2. Ruby 是一种解释性的语言。也许以后可能会出于性能的考虑而实现 Ruby 编译器，但目前来看使用解释器有很多的优点，不仅可以快速实现原型，而且可以有效地缩短整体开发周期；

3. Ruby 是一种面向表达式的语言。在表达式可完成任务的情况下，为什么要使用语句呢？使用表达式意味着代码更简洁，因为可以消除通用的部分，从而减少重复代码的冗余量；
4. Ruby 还是一种非常高级的语言。我们知道语言设计的原则之一是计算机应该为程序员服务而不是相反的情况。Ruby 和那些低级语言相比起来，更容易执行深奥而复杂的操作。

背景介绍

Ruby 语言的发明人是松本行弘，他从 1993 年起便开始着手 Ruby 的研发工作。初衷是他一直想发明一种语言，使自己既能进行高效开发又能享受编程的快乐。经过两年多的努力，松本于 1995 年 12 月推出了 Ruby 的第一个版本 Ruby 0.95。其后不久，Ruby 凭借其独特的魅力横扫日本，现在已经为越来越多的人使用。虽然 Ruby 的诞生要比 1995 年已经为人知晓的 Java 和 PHP 还要早两年，虽然 Ruby 已经是一个成熟的语言，但 Ruby 的普及性却远没有它的历史那么强悍，这也是其社区为什么一直不如 Java、PHP 和 Perl 阵营火热的原因。所以，如果你在前两年想深入研究这门语言，会发现专门针对这一预言的框架、库、书籍、网站、博客和包括讨论组等在内的资源很稀少。

Rails 的出现如同给 Ruby 社区吹来一阵清

风，一扫其多年颓势，有人曾评论说 Rails 框架是 Ruby 诞生以来最受欢迎的应用程序。由 David Heinemeier Hansson(DHH)创造的 Rails 将 Ruby 语言扩展为一种极其适用于 Web 开发的领域特定语言(DSL)，引入了对象关系持久性。说起 Rails 的诞生，可以用“奇遇”二字来描述。DHH 在 37signals 公司工作的时候，帮助开发一个小型的项目管理和交流软件 Basecamp，使用的开发语言是 PHP。但是当他不小心接触了 Ruby 之后，PHP 在他心目中忽然变得让人厌恶起来，于是经过一番考虑，他决定转向 Ruby。而且他还决定基于 Ruby 来开发 Basecamp，有意思的是 37signals 看了他的报告后竟然很快答应了。于是，在接下来的两个月 DHH 先用 Ruby 开发了一个框架，又过了两个月后 Basecamp 便大功告成，并且很快被全世界 40 多个国家的人们广泛使用，当时有人说这是他们所见到的世界上最好的 Web 应用程

序。在这股热潮之下，DHH 决定与大家分享他的这个 Framework，于是将这个框架从 Basecamp 中剥离出来，并取名为 Ruby on Rails。Rails 诞生在社区所引发的结果是，它使 Ruby 书籍的销量在一年内(2005~2006)翻了十几倍，也使 Ruby 成为 2005 年最受关注的语言。

可以说，很多软件开发人员是通过 Ruby on Rails 框架才开始主动或被动地对 Ruby 语言产生兴趣的。最初，人们只是把 Ruby 当作 Ruby on Rails 的一种开发工具，但是随着人们对 Ruby 的学习，很多人逐渐认为 Ruby 就是他们所期望的一门动态语言。从 Ruby 自身的特性来看，相比 Perl 语言，Ruby 更简洁；从面向对象的角度来说，Ruby 又比 Python 语言更加抽象；从开发效率方面看，Ruby 又比 Java、C 和 .NET 平台上的语言高出一个数量级等。

参考资料

网站类

1. Ruby 官方网站：<http://www.ruby-lang.org>
Ruby 语言的最新资讯，包括版本更新、大会信息发布和 Ruby 最新版本发布等消息。
2. Ruby 中文社区论坛：<http://www.ruby-lang.org.cn>
一个非官方的 Ruby 中文社区，绝大部分信息翻译自官方网站，有利于英文基础不好的读者获取 Ruby 相关的资讯。但本网站资讯有时不及时。
3. Ruby on Rails 官方网站：<http://www.rubyonrails.com>
有关 Ruby on Rails 的版本更新、发展路线图、下载信息等都可以在这里直接找到。对更多做技术选型的架构师而言，网站中的那些网站的使用案例可能更有帮助。
4. 开源项目 RubyForge：<http://rubyforge.org>
顾名思义，RubyForge 是 Ruby 开源项目的一个集市，许多有名的 Ruby 项目都可以在这里找到。如果要看社区里有哪些关于 Ruby 的好用的插件信息等，不妨到这里逛逛。
5. InfoQ 中文站 Ruby 社区：<http://www.infoq.com/cn/ruby>
InfoQ 中文站是一个关注企业软件开发领域变化与创新的网站，Ruby 社区有多位技术专家撰写相关的新闻和文章，并有精彩视频和迷你书。这是一个新兴的网站，值得关注。

书籍类

1. 《The Ruby Way中文版（第二版）》

作者：Hal Fulton 译者：陈秋萍、赵子鹏 出版社：人民邮电出版社 出版时间：2007年11月
ISBN：9787115166692

Ruby借鉴了LISP、Smalltalk、Perl、CLU和其他语言的最佳特性。在本书第一版面世后的5年内，Ruby便日益流行。本书采用“如何解决问题”的方式阐述Ruby编程，书中包含400多个按主题分类的示例，每个示例都回答了“如何使用Ruby来完成”的问题。首先对要完成的任务进行了描述，并讨论了技术方面的约束条件，然后逐步地阐述了一种优秀的解决方案。在此过程中，作者辅以说明和解释来帮助读者理解。

2. 《Ruby Cookbook（影印版）》

作者：Lucos Carlson、Leonard Richardson 出版社：东南大学出版社
出版时间：2006年11月 ISBN：9787564105969

你想深入探究Ruby吗？Ruby Cookbook就是关于这一当今最热门编程语言的最全面的问题求解指南。本书使用清晰的阐述和数千行可以在你的项目中使用的源代码，来为你在实际应用中可能碰到的数百个问题提供解决方法。从数据结构到集成前沿技术的算法，Ruby Cookbook为每一位编程人员都准备了一些专题。

3. 《应用Rails进行敏捷Web开发》

作者：Dave Thomas、David Hansson等 译者：林芷薰 出版社：电子工业出版社 出版时间：2006年7月 ISBN：7121028727

2006年3月，本书荣获Jolt大奖的“最佳技术类图书”奖。全书主要内容分为两大部分，在“构建应用程序”部分中，读者将看到一个完整的“在线购书网站”示例。在随后的“Rails框架”部分中，作者深入介绍了Rails框架的各个组成部分。

快速上手教程

笔者认为通过Rails的应用更能理解Ruby语言，所以下面我们通过一个创建在线通讯录的经典案例来解释一下如何使用Ruby on Rails。本案例主要步骤如下：

1. 生成模型（在此步骤中创建MySQL数据库和表）；
2. 生成应用程序（包括生成基本代码和目录）；
3. 启动Rails（并配置数据库的访问）；
4. 创建一些内容（包括生成支架模型和控制器，并告知控制器去使用哪个支架）。

下面让我们仔细研究一下各个步骤：

1. 生成AddressBook模型

对于任何应用程序，我们需要做的第一件事就是为它创建一个存放数据的数据库。技术上这个步骤不必最先进行，不过需要在早期完成；应该在编写任何应用程序代码（甚至是自动生成的代码）之前创建数据库，这是显然的。所以，让我们在MySQL数据库中创建一个数据库，并在此数据库中创建第一张表。这里我们假定MySQL已经安装并且可用。创建MySQL数据库和表的代码如下：

```
[~/Sites]$ cat AddressBook.sql
CREATE DATABASE IF NOT EXISTS AddressBook;
USE AddressBook;
CREATE TABLE IF NOT EXISTS contacts (
    id smallint(5) unsigned NOT NULL auto_increment,
    name varchar(30) NOT NULL default '',
    created_on timestamp(14) NOT NULL,
    updated_on timestamp(14) NOT NULL,
    PRIMARY KEY (id),
    UNIQUE KEY name_key (name)
) TYPE=MyISAM COMMENT='List of Contacts';
[~/Sites]$ cat AddressBook.sql | mysql
```

2. 生成AddressBook应用程序

既然已经拥有了一个能够交互的数据库，就可以创建 AddressBook 应用程序了。第一个步骤是简单地运行 Rails 来生成基本目录和支架代码：

```
[~/Sites]$ rails AddressBook
create
create app/apis
create app/controllers
create app/helpers
create app/models
create app/views/layouts
create config/environments
create components
[...]
create public/images
create public/javascripts
create public/stylesheets
create script
[...]
create README
create script/generate
create script/server
[...]
```

3. 运行Rails

创建了 AddressBook/ 目录和所需要的子目录后，您需要执行一次惟一的初始配置。首先，通过修改 YAML 配置文件来设置数据库，代码如下：

```
[~/Sites]$ cd AddressBook
[~/Sites/AddressBook]$ head -6 config/database.yml # after editing
development:
  adapter: mysql
  database: AddressBook
  host: localhost
  username: some_user
  password: password_if_needed
```

最后，我们需要提供数据。Rails 附带了它自己的单一功能的 Web 服务器，即 WEBrick，非常适合于这个试验。我们可能也会遵循 Ruby on Rails Web 站点上的说明来配置 Apache 或其他服务器，以通过 FCGI（或者普通的 CGI，但是普通的 CGI 会比较慢）向 Rails 应用程序提供服务。下面是启动 WEBrick 服务器的代码：

```
[~/Sites/AddressBook]$ ruby script/server -d
=> Rails application started on http://0.0.0.0:3000
[2005-03-21 17:57:38] INFO  WEBrick 1.3.1
```

```
[2005-03-21 17:57:38] INFO  ruby 1.8.2 (2004-12-25) [powerpc-darwin7.8.0]
```

4. 创建一些内容

要在 WEBrick 端口上看到一个欢迎页面，先前的步骤就足够了。例如，在我的本地系统中，现在可以访问 <http://localhost:3000/>。不过，为了操作定制数据库，需要多生成一些代码。可以使用脚本 generate 来完成此任务，这个脚本创建在 AddressBook/ 应用程序目录中：

```
[~/Sites/AddressBook]$ ruby script/generate model contact
exists app/models/
exists test/unit/
exists test/fixtures/
create app/models/contact.rb
create test/unit/contact_test.rb
create test/fixtures/contacts.yml
[~/Sites/AddressBook]$ ruby script/generate controller contact
exists app/controllers/
exists app/helpers/
create app/views/contact
exists test/functional/
create app/controllers/contact_controller.rb
create test/functional/contact_controller_test.rb
create app/helpers/contact_helper.rb
```

注意，在相应的表名中，这里应该使用单数的 contact，而不是复数的 contacts。现在需要编辑一个或多个生成的文件（只需稍加编辑）来让控制器去使用支架：

```
[~/Sites/AddressBook]$ cat app/controllers/contact_controller.rb
class ContactController < ApplicationController
  model :contact
  scaffold :contact
end
```

现在可以通过类似于 <http://rails.server/contact/> 的 URL（在我的测试用例中是 <http://localhost:3000/contact/>）来查看和修改数据库的内容。输入一些数据后，它看起来如图 2-1 和图 2-2 所示：

Name	Created on	Updated on	Show	Edit	Destroy
Uche Ogbuji	Mon Mar 21 03:02:06 EST 2005	Mon Mar 21 23:55:02 EST 2005	Show	Edit	Destroy
Tom Young	Mon Mar 21 03:02:38 EST 2005	Mon Mar 21 03:02:38 EST 2005	Show	Edit	Destroy
Dethe Elza	Mon Mar 21 23:37:39 EST 2005	Mon Mar 21 23:37:39 EST 2005	Show	Edit	Destroy

New contact

图2-1 列出联系人

Name	<input type="text" value="Dethe Elza"/>
Created on	
Updated on	
<input type="button" value="Update"/>	
Show Back	

图2-2 编辑联系人

小功告成，到此我们已经创建了一个具有查看和修改数据库的功能完全的 Web 应用。可以看出，Ruby on Rails 为我们提供了开发灵活的 Web 应用程序的一种非常简单快速的途径。这里只是对 Ruby on Rails 做肤浅的介绍，要体验 Ruby 和 Rails 强大的功能，请浏览其网站和阅读相关书籍。

版本信息

目前，Ruby 最新的版本是 2007 年圣诞节（12月 25 日）发布的 1.9 版。新的版本较从前有了大幅的改进，更新记录达 4 万行之多。尤其值得关注的是以前被反对 Ruby 的人引以为诟病的执行效率慢的问题，这在 1.9 版本里有了很大的进步，其所提供的 YARV 虚拟机，因为支持本地线程库而能够大幅度提升运行效率。网上有人就 Ruby 1.9 和 Ruby 1.8.6，以及 Python 2.5.1 等进行了比较，在 Mac OS X 10.5 下使用斐波那契递归函数对三者测试后发现，同样的循环次数中，Ruby 1.8.6 用时 158.869 秒，Python 2.5.1 用时 31.507 秒，而 Ruby 1.9.0 仅用时 11.934 秒。目前，Ruby 创始人松本行弘正在带领团队忙于 Ruby 2.0 版本的研发。

也就是在 Ruby 1.9 发布的一周内，Rails 2.0 发布，在 InfoQ 中文站采访 DHH 的一篇文章中，DHH 评价说其认为 Rails 2.0 中最重要的特性是对于 RESTful 应用的开发，用 RESTful 的方式开发 Web 应用是一个让人欢欣鼓舞的转变，虽然理解这个转变需要花些时间，但过了这一阶段，你就会适应并享受它。

社区视角

和 Ruby 最经常搭配使用的自然是 Rails，因为 Ruby on Rails 使用了严格的 MVC 体系架构，而这一架构目前已经被证实是设计 Web 应用的不二之选；另外，使用 Rails 可以非常方便地构建基础系统，这也是它之所以能够吸引大量 PHP 开发人员投奔于此的主要原因；在开发效率上，Ruby on Rails 框架的支持者说 Ruby on Rails 开

发人员的生产率最多是使用传统 J2EE 框架的 10 倍。虽然这句话引发了 Rails 和 J2EE 社区相当大的争议，但争论中却很少谈及如何比较 Rails 和 J2EE 这两个架构，Ruby on Rails 现在已被视为现有企业框架的一个良好的替代品。但是，Rails 并非一点缺憾也没有，从数据库角度来看，Rails 不支持双步提交，目前只能用于单个数据库后端情况等。

《程序员》杂志的记者在采访松本行弘时，就评价 Ruby 技术进行 Web 开发的优势与劣势时，他回答说“开发时效高是 Ruby 进行 Web 开发的优势所在。使用 Ruby 和 Rails，我们可以在几分钟之内创建 Web 服务。而且，在适应需求的变化上，采用 Ruby 语言远比 Java 及其相关技术要容易得多。用 Ruby 进行 Web 开发的劣势在于它的运行效率不够高。我们不能期望它能达到像 C++ 或 Java 所能达到的最高性能。不过我认为，网络和数据库是目前应用程序的发展瓶颈，所以 Ruby 在性能上的问题也无大碍。在业务逻辑中性能的重要性正在减弱。”

众所周知，O'Reilly 多年以来一直狂热地支持 Java，但是现在它已经毫不犹豫地开始出版 Ruby 方面的书籍，带领人们从 Java 阵营离开。Java 之父 Gosling 也表示，Java 将会继续处于顶峰，并在企业应用上保持良好的表现，但是时间不会停滞不前，Java 在将来也肯定会被替代，我们需要一个更高级别的抽象。虽然 Gosling 没有明确说取代 Java 的应该是动态语言，但是他希望在 JVM 上做充分的投入以更好地支持动态语言的观点，无疑可以被人理解为这位 Java 巨人的目光已经聚焦在动态语言上，而 Ruby 恰是动态语言阵营中的一个优秀学生。

作者介绍

霍泰稳，混迹于国内技术社区，目前以为中国技术社区提供一流的资讯为己任，现为 InfoQ 中文站总编辑。关注团队管理，相信“合作是最佳的利己策略”。欢迎访问 <http://www.infoq.com/cn> 及个人博客 <http://blog.csdn.net/futurelight>。

3

Ajax开发篇

综述

从 2005 年 Web2.0 的兴起开始算起，Ajax 伴随着国内的 Web 开发社区走过了近三年的成长时间。三年的时间，对于一项 Web 技术不算短了，这不仅是对当初对 Ajax “旧瓶装新酒” 持不屑态度的人的反驳，更把更多观望中的开发者拉入了 Web 开发的行列。我们开始看到越来越多的软件企业期待专业的 Web 软件开发人员来加盟，来增强自己产品的可用性。这都是由 Ajax 的终极目标来决定的：改善用户的体验。原来 Web 前端的开发也可以做得如此精彩，原来前端也有 MVC 架构存在，原来 Ajax 不是想当然的奇技淫巧。

会用 Javascript、XML、CSS、DOM 和 XMLHttpRequest 对象，就可以写出你的 Ajax 代码，是不是很简单？可惜的是，因为几大浏览器厂商对 Web 标准的支持不够统一，有的甚至加入了一些自己的扩展，导致 Ajax 开发者在编写代码时不得不加入一些特定于浏览器的兼容性代码（除非你的产品只支持一种浏览器，而且你的客户也同意这么做），于是代码中浏览器类型判断的语句随处可见，结果可想而知：难以维护，噩梦不断。

现在，Ajax 开发者们有福了，有 Ajax 框架可以把这一切浏览器兼容的代码为你封装起来，你面对的仿佛是完美无瑕的浏览器，不需要再判断当前是什么浏览器，你的代码可以在任意浏览器中运行。不仅如此，这些 Ajax 框架还提供了许多更为强大的功能：可以即用并扩展的 UI 工具包、简单便捷的 API、与 Server 端框架的集成，甚至是离线存储这样高级的功能。

这些框架的名单很长，但本篇涉及的 Buffalo、Dojo、Prototype、jQuery 和 DWR 无疑是其中的佼佼者。它们相比较其他框架而言，拥有广泛的社区基础，文档翔实，功能接口稳定，口碑甚好。

本篇对于每一种框架从功能特点、背景介绍、参考资料、快速上手、版本信息、社区视角和综合评比七个方面进行介绍，既然是选型手册，功能特点和社区视角必然是读者可以参考的主要内容，相信每一位期待选型的读者都会从中得到满意的选择。

有理由相信，随着 Web 标准的发展和浏览器厂商对标准的支持，Ajax 的明天会更好。

关联信息

Buffalo 是现任 ThoughtWorks 中国咨询师陈金洲（Michael Chen）的作品。可以说，Buffalo 是一款真正意义上的 Ajax 远程调用框架，它没有包含那些繁杂而华丽的 UI 支持，而是代码短小精悍，在提供 Ajax 基本功能的同时也提供一些极为方便使用的高级功能。目前最新版本是 2.0，依靠社区的力量，Buffalo 已经在越来越多的产品和项目中得到了广泛应用。国人出品，文档充分，更难得的是你可以直接联系到 Michael 来获取技术支持。

Dojo 是目前来看最为强大，功能也最为全面的 Ajax 库。无论是 IBM 这样的国际一流的软件企业的产品线，还是普通开发爱好者的工具箱中都为 Dojo 留有一席之地，应用不可谓不广泛。Dojo 从 0.4 和 0.9 分别开始提供两个版本，0.4 是为了和之前的版本兼容，而 0.9 则改善了整个架构，层次更为清晰，接口调用更为简便，更是加入了许多高级功能，如离线存储。

Prototype 无疑是 Ajax 开发的基础类库，包括 Buffalo 在内，很多 Ajax 库都是以 prototype 为基础发展而来的。一如其名，prototype 提供的是 Ajax 调用所需的最基本的基础设施，此外还提供众多方便快捷的函数封装供开发者在开发 JavaScript 时使用。Prototype 是 Web 开发者提升水平，研究 Ajax 不可或缺的参考。

jQuery 的出现可以用技惊四座来形容。归根到底，它提供了精悍而强大的接口功能，令人赏心悦目的闭包调用方式，俘获了大批一心追求完美的程序员的心。除了提供基本的动画和显示效果，以及 Ajax 应用封装外，jQuery 还提供了类似 Eclipse 的插件机制，原理就是为全局的 jQuery 对象提供扩展的方法。已经有非常多的插件可供选用，包括制表、圆角、滑动显示、工具提示、日期选择器、表单处理排序和拖拽等。

DWR 最大的亮点就是，可以在浏览器端使用 JavaScript 直接调用 Web 服务器上的 Java 类暴露出来的方法。DWR 采取了一个类似 Ajax 的新方法来动态生成基于 Java 类的 JavaScript 代码。这样，Web 开发人员就可以在 JavaScript 里使用 Java 代码，就像它们是浏览器的本地代码（客户端代码）一样。

3.1

Buffalo

总评

上手容易程度	★★★★★
社区活跃性	★★★★★
应用广泛性	★★★★
推荐指数	★★★★★

功能和特点

作为中国的 Web 开发者，如果没有听说过 Buffalo，只能说是孤陋寡闻了。可以说，Buffalo 是一款真正意义上的 Ajax 远程调用框架，它没有包含那些繁杂而华丽的 UI 支持，而是代码短小精悍，在提供 Ajax 基本功能的同时也提供一些极为方便使用的高级功能。

更加难能可贵的是，上手 Buffalo 对于开发者来说实在是一件再轻松不过的事情，它提供一系列入门教程及最佳实践来帮助开发者，而这一切，都是由现为 ThoughtWorks 工程师的陈金洲和一些热心的贡献者来完成的。

如果在你的项目中急需能在半个小时内上手并熟练使用的 Ajax 框架，甚至期望来自于框架作者的直接支持的话，还犹豫什么呢？选择 Buffalo 吧！

让我们一起来看看 Buffalo 提供了哪些激动人心的特性：

1. 轻量级xml协议的JavaScript实现。

Buffalo 采用一个轻量级协议（burlap 的子

集，并对孩子集做了小幅修改），它非常利于 Web remoting，而且足够简单。Buffalo 实现包含了 JavaScript 对象的序列化和反序列化。

2. 完全支持Java到JavaScript序列化/反序列化。

任何调用在 Java 端方法后的结果都将透明地序列化到 JavaScript 端。Buffalo 能处理原始数据类型和对象类型（List、Map 乃至你自己的业务域对象）的序列化。你可以在 JavaScript 端访问同样的属性，这就像在 Java 端一样轻松。

3. 基于回调的编程模型。

这种 API 是易学易用的，几乎每个用户都能在半小时内轻松学会使用。

4. 支持异步事件。

5. 简单易用的API。

服务器端任意 POJO 都能暴露为 Buffalo 服务，无需编写 Buffalo 特定的 Java 文件，客户端用户仅需使用一个 Buffalo 对象和为数不多的方法。

6. 集成Prototype JavaScript库。

Buffalo 客户端脚本建造在这个著名的原型库之上，你可以从这个提供便利的基础组织——“Prototype” 中直接获益。

7. Spring集成。

每个被 Spring 管理的 Bean 都能通过简单的配置成为 Buffalo 服务。

8. 兼容性。

远程调用特性支持 IE5.5+/Firefox1.0+/

- Safari/Opera9+。
- 9. 支持浏览器前进/后退。
 - 10. 支持数据绑定。

Buffalo 对大多数常用的 HTML 元素提供自动化的绑定功能，可以绑定 JavaScript 对象到 HTML 中的成员。

背景介绍

早在 2004 年 11 月，Buffalo 的作者陈金洲就在当时兴起的 Web 异步应用——诸如 Google Gmail 的影响下，开始了对 Web 应用程序开发的重新思考。根据多年的企业级 Web 应用开发的经验，他提出了面向异步消息的 Web 应用（Asynchronous-Messaging Oriented Web Application，简称 AMOWA）的概念，并在 2004 年 11 月 13 日的 Javaparty 聚会上进行阐述。这恰恰和后期传入国内的 Ajax 概念不谋而合。

2005 年他更是提出 One Page,One Application（后面缩写为 OPOA，或者 1P1A）的概念。OPOA 含义很简单：一个页面就是一个应用。在众多的基于 Web 的 MIS 系统中，没有人关心页面的组织形式；大多数稍微复杂的 MIS 系统，都采用分帧（Frame）的方式来组织页面，这样，在进行业务操作的时候，url 的变化表现在一个框架页面内，从浏览器的地址来看，只有一个地址；更有甚者，一些应用干脆弹出一个去掉了浏览器菜

单、工具条、地址栏、状态栏的窗口（比如招商银行、民生银行的网上银行系统），连地址都看不见。因此，一个页面就是一个应用，从用户的角度来说，对于操作型系统，这是一种非常自然的体现。用户无需了解每一个具体的操作对应的地址是什么。

这种设计背后的含义实际是在权衡：是由程序来控制用户的行为，还是由用户的行为来控制程序。在操作型系统中，每一步的操作都被业务含义严格定义，无论是应用的设计者，还是其使用者，都希望在一种受控的状况下来进行操作。例如，一个审批动作，用户更希望是通过一个按钮来触发，而不是采取访问类似于 /approve.action?itemid=123 的方式。

同年，他发布了 Buffalo 1.0 测试版本。具体的日程可以在 Buffalo 变更历史里看到。从 Buffalo 产生至今，整整两年了，目前最新的版本是 2.0 正式版。

最早的时候作者希望 Buffalo 成为一个全功能、全新的、Web 2.0 时代的 Web 框架，异步传输只是其中最基础的一部分。后来发现凭借自己现有的资源（时间、能力等）不太可能在可预见的时间内完成这一目标。目前，作者想达到的目标是：使其成为最简单、最容易使用的 Ajax 框架。从使用者的反馈来看，这一目标完成得比较不错，大多数用户都能在很短的时间内配置 Buffalo 并开始使用。

参考资料

网站类

- 1. Buffalo 官方站点
<http://Buffalo.sourceforge.net/>
- 2. InfoQ 关于 Buffalo 的报道：
<http://www.infoq.com/cn/news/2007/04/Ajax-framework-Buffalo2>

3. Buffalo Wiki
<http://confluence.redsaga.com/display/BUFFALO/Home>
4. Buffalo作者对于Amowa概念的阐述:
<http://michael.nona.name/archives/78>
5. Buffalo作者对于OPOA概念的阐述:
<http://michael.nona.name/archives/117>
6. Buffalo作者经常会在JavaEye回答开发者的问题，就Buffalo的版本和使用问题做出详细的回答。
<http://www.Javaeye.com>
7. Buffalo中文论坛
<http://groups.google.com/group/amowa>

快速上手教程

准备工作

请先下载最新版本的 Buffalo，然后按照以下关系建立文件夹：

```
Buffalo-example
  Web-INF/classes
  Web-INF/lib
  script
```

拷贝 commons-logging.jar 和 Buffalo-<version>.jar 到 Web-INF 下的 lib 目录，拷贝 Prototype.js 和 Buffalo.js 到 script 文件夹。

修改 Web.xml

在 Web-INF 目录下建立或修改 Web.xml 文件，填写以下内容：

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE Web-app PUBLIC "-//Sun Microsystems, Inc.//DTD Web Application
  2.3//EN" "http://Java.sun.com/dtd/Web-app_2_3.dtd">
<Web-app>
  <display-name>Buffalo Example Application</display-name>
  <servlet>
    <servlet-name>bfapp</servlet-name>
    <servlet-class>net.Buffalo.Web.servlet.ApplicationServlet</servlet-class>
  </servlet>
  <servlet-mapping>
    <servlet-name>bfapp</servlet-name>
    <url-pattern>/bfapp/*</url-pattern>
  </servlet-mapping>
</Web-app>
```

修改 Buffalo-service.properties

在 Web-INF 下的 classes 目录中创建一个名为 Buffalo-service.properties 的文本文件，并填写以下内容：

```
# 示例服务
helloService=example.HelloService
```

修改 JSP 文件

在 Web 根目录创建 example.jsp，并填写以下内容：

```
<HTML>
<head>
    <meta http-equiv="Content-Type" content="text/HTML">
    <title>Example::Hello</title>
    <script language="JavaScript" src="script/Prototype.js"></script>
    <script language="JavaScript" src="script/Buffalo.js"></script>
    <script language="JavaScript">
        var END_POINT=<%=request.getContextPath()%>/bfapp";
        var Buffalo = new Buffalo(END_POINT);
        function hello() {
            var p1 = $("myname").value;
            Buffalo.remoteCall("helloService.hello", [p1], function(reply) {
                alert(reply.getResult());
            });
        }
    </script>
</head>
<body>
    <p>Buffalo Hello World</p>
    <p>&nbsp;</p>
    <form name="form1" method="post" action="">Your name:
        <input name="myname" type="text" id="myname">
        <input type="button" name="Submit" value="Hello" onclick="hello()">
    </form>
</body>
</HTML>
```

增加一个服务

```
package example;
public class HelloService {
    public String hello(String name) {
        try {
            // to see the loading status
            Thread.sleep(2000);
        } catch (InterruptedException e) {
            e.printStackTrace();
        }
        return "Hello, " + name;
    }
}
```

将其编译并拷贝至 Web-INF 下的 classes 目录。

最后一步：运行此程序

拷贝 Buffalo-example 目录到 TOMCAT_HOME 下的 Webapps 中，启动 tomcat，用你喜爱的浏览器浏览 <http://localhost:8080/Buffalo-example/example.jsp>，在页面中输入你的名称并点击“Hello”按钮，查看效果。

与 Spring 集成

集成 Spring 非常简单。首先你需要确定 Spring 能被成功装载（使用 config servlet 或 context listener，请参考 Spring 的相关文档），此后，增加如下一个 config bean 到 Spring 配置文件中：

```
<bean name="BuffaloConfigBean" class="net.Buffalo.service.  
    BuffaloServiceConfigurer">  
    <property name="services">  
        <map>  
            <entry key="testSpringService1"><ref bean="yourBeanId"/></entry>  
            <!-- other entries... -->  
        </map>  
    </property>  
</bean>
```

就是这些了！当应用被启动后，Buffalo 将找到这个 Bean 并装载所有定义在服务属性中的服务。来自于 Buffalo-service.properties 和 Spring 的服务并没有区别，在 Buffalo 客户端看来都是一样的。

把Web页面中的from序列化为对象

这里用到了 Buffalo1.2.1 版本的 Buffalo.Form.formToBean 方法，它能把一个 form 直接序列化为一个对象。你可以看在 demo 应用中的 Form demo。在这里有一个非常简单的例子，展示了我们是如何把一个 form 转换为 net.Buffalo.demo.form.User 对象的：

```
var userObj = Buffalo.Form.formToBean("form1", "net.Buffalo.demo.form.  
    User");  
Buffalo.remoteCall("userService.createUser", [userObj], function(reply) {  
  
    alert(reply.getResult().username);  
})
```

序列化原则：

对于简单的 form 域（field），如 text、password、radio、select（单选），Buffalo 将 field 赋值给对象中名为 fieldName 的属性。

对于 select（多选）、checkbox，Buffalo 将 List<String> 赋值给对象中名为 fieldName 的属性。

绑定

Buffalo 支持将对象值绑定到 DOM 成员上，包括 form 成员（text、password、hidden、radio、checkbox、select、textarea）、table、form、div/span，它也能转换 form 到具体 Java 类型的对象。它只提供了一种能绑定所有类型值到不同 form 成员的便捷方法。

你可以如此轻松地使用绑定：

```
Buffalo.bindReply("yourService.method", [parameters], "elementId")
```

上面的代码片段尝试把 “parameters” 作为参数调用 “yourService.method”，然后绑定调用结果给 “elementId”。

如果你并不想做远程调用，也可以按下面的方式使用绑定：

```
Buffalo.Bind.bind(elementId, bindValue)
```

它将绑定 “bindValue” 到 “elementId”。

注意：这里无需在意 form 成员的不同类型。Buffalo 将检查类型并完成正确的绑定行为。

将form转换为Bean

一些用户恨透了在进行一次远程调用时从 form 中构造参数, Buffalo.Form.formToBean 会使这一切变得简单。

```
Buffalo.Form.formToBean(form, BuffaloObjectClass, ignoreButton)
```

上面的代码片段将把 form 转换为 BuffaloObjectClass 类型的 Bean。

参数说明

1. form: form id或form对象, 这是必需的参数;
2. BuffaloObjectClass: 你要转换成为的类, 不是必备参数。如果没有提供, Buffalo默认认为是Java.util.HashMap类;
3. ignoreButton: 转换过程中是否忽略掉form中的button。不是必备参数, 默认为true。

转换风格

1. 所有的值都是字符串或字符串的列表;
2. 单个字符串值将被转换为字符串, 多个字符串值将被转换为字符串的列表。服务器端将使用 Java.util.List类来存放字符串的列表。

版本信息

截止到笔者交稿时, Buffalo 官方站点提供的版本是 2.0 正式版。

2.0 版的最大亮点在于性能的提升和完全自行实现的 Java 到 JavaScript 协议的转换。根据评测, 2.0 版本要比前一阶段版本最高提升 30% 的性能。这得益于新的协议实现, 以及为大规模 Ajax 调用而进行的优化。

Buffalo 1.x 版本的用户都知道, 之前的版本都基于 Burlap 协议, 而 Burlap 协议的维护者 Caucho 公司已经很久没有更新这个库了, 很多在 Resin 上使用 Buffalo 的用户发现这样或那样基于 Burlap 的问题。种种考虑之后, 2.0 版本正式使用了完全自行编写的协议解析和转换。

对于一直使用 alpha 版本的用户, 此次升级很简单, 只需要将相应的 jar 和 js 进行替换即可。从 1.2.x 版本升级的用户, 升级也很简单:

1. 删除所有burlap*.jar、Buffalo*.jar, 将它们替换为Buffalo – 2.0.jar;
2. 替换成新的Buffalo.js;
3. 将继承自BuffaloService的类, 对session等信息的使用替换为对RequestContext.getContext().getXXX的使用(注意, 目前有开发者报告在resin 2.1服务器上偶尔会出现丢失session的现象)。

2.0 正式版本的发布意味着完全自我独立的协议实现, 为后续特性的开发打下了基础。

下载地址: http://sourceforge.net/project/showfiles.php?group_id=178867

Buffalo 作者陈金洲在社区不止一次地表达过 Buffalo 的发展方向: 只会更简单, 不会更复杂。Buffalo 在添加新特性方面一直很慎重, 最不希望看到的是 Buffalo 成为一个庞大的怪物。

当 Buffalo 的用户遇到问题时, 可以访问 Buffalo 邮件列表, 绝大多数问题都可以在那里找

到答案，其他的开发者也会帮助你解决问题。

社区视角

Buffalo 官方站点对为什么要选择 Buffalo 有着如下的观点：

Buffalo 并不是一个新生婴儿。但在过去的两年里，它从 Java 到 JavaScript（以及 js 到 Java）的复杂的序列化和反序列化机制已经在许多实际的项目中得到应用和印证。API 的使用非常简便直接，以至于每个用户都可以在不到半小时内开始开发。

当前的不足在于，部分用户报告说在 resin2.1 下使用 session feature 时存在一些问题。如果你的确可以重现这个问题，请在 Buffalo 的论坛上留帖。

Buffalo 作者陈金洲也对 Buffalo 和 DWR 做了一番深入的比较。

Buffalo特性

1. 基于Prototype。如果你的Ajax应用也是基于Prototype的，那么可以减少重复加载 Prototype的带宽，并且获得相当一致的编程概念。
2. Bind。提供了对结果数据的处理，直接将数据绑定到页面对象并展示，这是一个动人的特性（DWR在Util.js中也提供了一些方法来简化数据的展示，但不比 Buffalo 做得更多）。在2.0版本中，Bind功能更加强大，能够将值直接绑定到表单元素、表格、DIV/Span、甚至整个表单上。关键是这种绑定是无侵入的，并且与Buffalo 整体结构完全整合，对外表现则只有一个简单的{{Buffalo.bindReply}}或{{Buffalo.Bind.bind}}。<http://Buffalo.sourceforge.net/binding.HTML>上有一些描述。

3. 序列化。Buffalo支持任意对象、任意深度、任意数据结构的Java到JavaScript及JavaScript到Java的双向序列化，并且支持引用。由于文档和演示不充分，很多人以为 Buffalo不支持任意对象。

4. 生命周期对象访问。1.2.4版本之前，Buffalo需要继承一个BuffaloService，从1.2.4版本开始，Buffalo就不需要继承了，引入了线程安全的BuffaloContext对象，只需要通过BuffaloContext.getContext()即可获得一个线程安全的引用，并且对Request的各种属性进行操作。更方便的是通过：

```
Map BuffaloContext.  
    getContext().getSession()  
Map BuffaloContext.  
    getContext().getApplication()  
Map BuffaloContext.getContext().  
    getCookie()
```

即可获得session/application/cookie的Map，操作这些 Map 即可对这些生命周期的各种变量进行查询和更新。这个特性参考了 Webwork 中 ActionContext 的设计。

5. 对Collection/Array的模糊处理。Buffalo提供了对Collection/Array对象的模糊识别能力。例如：服务器端有一个方法需要调用List参数，客户端传递过去一个JavaScript 数组就可以了，不需要费心地组装对象。Buffalo通过这些细节来提高程序员生产力。
6. 客户端组装对象。Buffalo支持在客户端组装对象，甚至可以直接将整个表单序列化为一个对象，作为参数传给远程客户端。
7. 对重载方法的处理能力。由于 Java 与 JavaScript 之间类型的不匹配，DWR 的代码生成无法对重载方法进行处理。例如：sum (double, double)、sum (int, int)，DWR很可能不知道你要调用哪一个。从2.0版本开始，Buffalo支持了对重载的处理。

DWR特性

1. 支持Batch，可以将多个Service函数调用放在一个XMLHttpRequest请求中完成。

作者：我一直认为这不是一个好的实践。在客户端发起多个请求并获得响应，除了使编程变得更复杂外，还增加了服务器端设计Service的自由度。这种方式感觉上更鼓励为远程调用设计细粒度的服务，将组装逻辑放在客户端。这种设计风格我不太喜欢，因此Batch也一直没有考虑实现它，虽然实现不太麻烦。
2. Converter。可以转换任意类型的Java对象到JavaScript，并允许直接使用。例如：Customer类包含一个address变量，当AjaxCall返回Customer对象的时候，可以直接在JavaScript中使用customer.address来获得Address的信息。
3. 允许Expose部分函数和属性（Buffalo无限制，可以访问Service中的任意函数）。

作者：这个我也考虑过……DWR的代码生成机制使得它不得不通过这种方式减小流量。Buffalo如果想实现这个特性也不是不行，只是我觉得，既然辛辛苦苦实现了Service，还需要通过这种方式来让别人不能用么？况且Buffalo没有代码生成，无论你暴露多少方法，流量都是一样的。考虑到实际情况，Buffalo没有实现这个特性。

网友jimguo：想像这样的需求，对于一个AddressService，通过Ajax只能添加和更新，不允许删除。如果没有Expose限制，就只能写在两个Service中了，添加和更新的功能在一个Service里被Expose，而删除

的功能在另一个Service里被Expose，或者再创建一个Facade类？

4. DWR2.0中提出了Reverse Ajax，提供在Java代码中处理页面上元素的功能。

作者：这个东东……也还是代码生成的trick……然而我的态度是JavaScript与Java同样重要，因此不会让代码生成之类的东西破坏JavaScript的整体性。

来自 JavaEye 社区的开发者的观点：

“一个项目使用一个 Ajax remote 框架已经足够，至于是 DWR 还是 Buffalo，就看你们自己的技术评估了。我个人认为 Buffalo 更为简单易用，而且它支持的表单元素绑定功能很舒服。”

“Buffalo 可比 DWR 简单多了，也不需要 XML 的配置文件，只要在属性文件中定义需要开放的服务就可以了。我这里举 Buffalo 和 DWR 的例子，只是在思考我们到底该如何选择？同样是文件上传，可以有那么多种方式，开源之后，太多的框架出来了，这是好还是坏呢？每个人都需要用大量的精力去比较、选择一个适合自己的方式，为什么大家不把精力只放在一个东西上面呢？就像 JavaEye 的圈子一样，劲要往一处使才能发挥它的威力。”

“我也比较喜欢 Buffalo，现在做的东西也是基于它的。但是 Buffalo 相对来说功能还是比较简单的，比如安全性、服务器端执行 js 等。别人写的很多很好的用户体验都是基于 DWR 的，这样都需要自己再转成 Buffalo 来实现，确实浪费很多精力。”

3.2 Dojo

总评

上手容易程度	★★★
社区活跃性	★★★★★
应用广泛性	★★★★★
推荐指数	★★★★★

功能和特点

您是不是在选择一款 Ajax 开发库，但是面对琳琅满目的开源项目却无从下手？

您是不是在苦恼因为采用几种不同的 Web UI 库，而导致用户界面风格严重不一致？

您是不是在厌烦某种 UI 库提供给您的 html 部件的使用方式太过僵硬，而希望能对其进行定制，甚至能创建出自己的可以重复使用的页面部件？

您有没有体会过在 Web 页面端进行 AOP 编程的乐趣？

现在告诉您，您只需在项目中只采用一个开源软件就可以实现上述所有的功能！

这就是 Dojo。

Dojo 是 Academic Free License 下的一个开放源码项目，正处于 Dojo Foundation 的开发之

中。Dojo 是一个非常活跃的项目，很多大公司，例如 IBM、Sun Microsystems、AOL 等都提供了对 Dojo 的支持。它是一个开源的 JavaScript UI 工具包。它使开发 JavaScript 更加简单，使构建并部署强大的 UI 更加迅速。

Dojo 的基础是个叫做 “Dojo Base” 的东西，一个很小的代码库，其中包含着 Ajax、事件处理、特效、炫目的 CSS 查询、语言工具，以及更多内容。在这个 Base 之上，Dojo Core 的其他部分增加了用于拖拽的高质量设施、Ajax 和 I/O 的扩展形式、JSON-RPC、国际化和回退按钮处理。在 Dojo Core 之上的是一个小部件系统，叫做 Dijit，它能够很容易地开发和重用 UI 的部件。

Dojo 的实际构架是建立在包（package）的基础上的，这是一个类似于 Java 语言中类包的概念。使用内含不同功能库的包能使您更方便地构建动态站点。它提供了一个丰富的小部件库，您可以使用它组成页面。您可以使用基于 Dojo 方面的事件系统将事件附加到组件，以创建丰富的交互体验。此外，您还可以使用几个 Dojo 库进行异步服务器请求、添加动画效果和浏览存储实用工具等。

Dojo 的包结构如图 3-1 所示：



图3-1 Dojo的包结构概览

下面是对 Dojo 中几种重要的类型的包作简单的功能使用介绍。

1. Dojo小部件。Dojo提供了可以用于构建页面的一组丰富的小部件。您还可以在JavaScript中创建、访问、修改和删除小部件，从而实现动态行为。
2. Dojo事件系统。Dojo事件系统使用AOP的技术将事件附加到小部件，这可以将小部件与实际的事件处理分离。Dojo不是将硬代码JavaScript事件添加到html标记上，而是提供允许您将事件附加到小部件的 API。
3. Dojo 异步服务器请求。Dojo通过抽象的特定于浏览器的详细信息，提供了对服务器进

行异步请求的简单方法。Dojo允许您创建数据结构来隐藏详细信息。此外，Dojo使用JSON-RPC标准支持RPC。

4. 另外，Dojo还包含下列丰富的功能：
 - ◆ 处理 html、字符串、样式、dom、正则表达式和若干其他实用工具的通用库。
 - ◆ 包括字典、ArrayLists、队列、SortedList、设置和堆栈的数据结构。
 - ◆ 用于添加动画效果、验证、拖放和若干其他功能的可视化 Web 实用工具。
 - ◆ 数学和加密库。
 - ◆ 存储组件。
 - ◆ XML解析。

背景介绍

Dojo 最早开始于一些 JavaScript 的开发者对 DHTML 的未来展开的讨论。Alex Russell (NetWindows 的最初创建者) 在 Informatica 寻求帮助,用以“DHTML 的未来”为标题的邮件和许多 DHTML 社区的成员进行讨论。接着, David Schontzler 在 Informatica 工作了一个夏天, 而 Dylan Schiemann 后来也加入了 Informatica。贡献给 Dojo 的最初几行代码就是由 Alex 和 Dylan 在 Informatica 的支持下完成的。还有很多其他的社区成员是 Dojo 发展方向的知名的积极参与者, 包括 Joyce Pard (mod_pubsub)、Tom Trenka (f(m) 的创建者)、Mark Anderson (bustlib)、Leonard Lin (Dojo 的命名建议者)、Aaron Boodman (现在在 YoungPup, 之前在 GreaseMonkey 和 Google)、Simon Willison (之前在 Django, 现在在 Flickr)、Cal Henderson (Flickr) 和 Dan Pupius (负责 dojo 的动画实现, 现在在 Flickr)。

在 ng-dhtml(现在是 Dojo-developer)邮件列表中对 license、命名、编码规范、构建工具、服务器配置和需求 (Dojo 应该是怎样的) 讨论了几个月之后, 我们开始编写代码, 并建立了一个拥

有 Dojo 代码版权的基金会。在编写完一些最初的规范之后, 第一部分 Dojo 的代码由 Alex Russell 和 Dylan Schiemann 在 2004 年 9 月编写完并迁入 Dojo 中。除了新的代码之外, 还有早期的贡献和来自于 NetWindows、burstlib 和 f(m) 的精髓。在 2005 年 3 月, 我们开始接受来自于其他 Dojo 成员的贡献, 到 2006 年底, 我们已经完成了 4 个主要的版本, 软件下载量超过 300 000 份, 并且接受了超过 60 个开发者和公司的贡献。

Dojo 基金会建立于 2005 年, 目前, Alex Russell 任主席, 而 Dylan Schiemann 是 Secretary 或 Treasurer, 投票是通过贡献者邮件列表中 +1/-1 来完成的。

在最初的邮件讨论后, Dojo 在开始两年一直是以我们梦想的方式发展。通过发布几个主要的版本, 以及广泛的社区贡献, Dojo 在远远短于它的前辈所经历的时间内, 变成一个更加全面和灵活的工具包。我们最初仅仅希望能创建一个避免当时所有工具包缺点的新工具包, 能够充分利用支持 DOM 的浏览器, 并通过现实能力和真实的软件工程方法构建出我们想要的工具包。我们的目标是避免重新发明轮子, 这样我们才能继续前进, 避免犯下新的错误。

参考资料

网站类

1. Dojo 官方站点: <http://Dojotoolkit.org/>
2. Dojo 手册 (The Dojo Manual) : <http://manual.Dojotoolkit.org/>, 系统地介绍了 Dojo 的机制、使用方法、示例代码等;
3. Dojo 邮件列表 (The Dojo Mailing Lists), 可以交流使用 Dojo 中遇到的问题或提出改进的建议;
4. Dojo 错误跟踪系统 (Bug Tracking) : <http://trac.Dojotoolkit.org/>, 可以提交有效地发现 Bug 的方法或直接提交 Bug, 以便能够及时消除 Dojo 中的 Bug;
5. 代码仓库 (The Dojo Subversion Repository): <http://trac.Dojotoolkit.org/browser>, 可以很方便地获得最新版本的 Dojo 程序包;

1. The Dojo Wiki: <http://Dojo.jot.com/>, Dojo Wiki的内容相对比较丰富, 涉及Dojo的各个方面;
2. Ajaxian.com/Dojo: <http://Ajaxian.com/by/topic/Dojo/>;
3. Dojo中文社区站点: <http://www.Dojocn.cn/apage/amushen/archives/2007/15.html#4>。

书籍类

1. 《征服Ajax: Dojo、Prototype、script.aculo.us框架解析与实例》

作者: 施伟伟, 张蓓 出版社: 人民邮电出版社 出版时间: 2007-3 ISBN: 9787115158031
<http://www.china-pub.com/computers/common/info.asp?id=34089>

快速上手教程

本部分会向读者展示 Dojo 在 Ajax 方面几种特定功能的实现, 同时为读者讲解使用 Dojo 编程基础知识。

文件上传

在默认情况下, Dojo 使用 XMLHttpRequest 对象作为调用远程页面的方法, 但是 XMLHttpRequest 对象是存在一定局限性的, 它不能用于传输文件, 不能访问跨域的页面, 也不支持对本地文件 (file:/// 协议) 的访问。

除了 XMLHttpRequest 对象, Dojo.io.bind 还支持另外两种传输介质 (transports): IFrame I/O 和 ScriptSrcIO。IFrame I/O 的特点是可以用于传输文件, 如果需要实现 Ajax 方式的文件上传功能, 就必须选择 IFrame I/O 作为传输介质。下面给出的是使用 IFrame I/O 传输介质上传文件的示例:

```
<script language="JavaScript" type="text/JavaScript">
    Dojo.require("Dojo.io.IframeIO");
    function upload(){
        var bindArgs = {
            formNode: document.getElementById("uploadForm"),
            mimetype:"text/JavaScript",
            content:{
                fileFields:"u11"
            },
            handler:function(type, data, evt){
                .....
            }
        };
        var request = Dojo.io.bind(bindArgs);
    }
</script>
.....
<form action="upload.cgi" id="uploadForm"
method="post" enctype="multipart/form-data">
    <input type="file" name="u11">
    <input type="button" onclick="upload();" value="Upload">
</form>
```

在使用 IFrame I/O 作为传输介质的情况下, Dojo 支持的 mimetype 参数包括 text/JavaScript、text/plain 和 text/html, 而对于 mimetype 参数为 text/xml 的情况 (即 XML 格式的返回结果), Dojo

是不予支持的。

跨域页面调用

由于安全性方面的原因，`XMLHttpRequest` 对象是不支持跨域页面调用的，因此 Dojo 提供了 `ScriptSrcIO` 传输介质用于跨域的页面调用。使用 `ScriptSrcIO` 进行跨域页面调用有以下 4 种基本方式。

1. 简单跨域页面调用

要在Dojo中实现最简单的跨域页面调用，只需要在`Dojo.io.bind`方法的参数中指定`transport`属性为“`ScriptSrcTransport`”即可，例如：

```
Dojo.io.bind({
    url: "http://del.icio.us/feeds/json/Dojoman",
    transport: "ScriptSrcTransport"
});
```

上面这种最简单的实现方式在功能上存在一些局限，它仅仅是请求一个跨域的页面，由于不会返回任何数据，当然也不可能实现响应回调和超时的处理。

2. Polling

第二种跨域页面调用的方法是使用Polling。在`Dojo.io.bind`方法的参数中增加一个`checkString` 属性，Dojo会判断`checkString`指定的变量是否有效，如果有效则执行相应的`load`回调函数。这种方式支持超时机制，如果超过`timeoutSeconds`指定的时间后仍然没有响应，Dojo会结束对远程页面的请求，执行`timeout`参数指定的超时处理函数。下面是用Polling方法跨域请求页面的示例，页面加载后会执行`mytest.js`中的代码，并且在页面中显示`tag[0]`的值。

```
<script language="JavaScript" type="text/JavaScript">
    Dojo.require("Dojo.event.*");
    Dojo.require("Dojo.io.*");
    Dojo.require("Dojo.io.ScriptSrcIO");
    Dojo.addOnLoad( function() {
        Dojo.io.bind({
            url: "mytest.js",
            mimetype: "text/JavaScript",
            transport: "ScriptSrcTransport",
            checkString: "tags",
            load: function(type, data, event, kwArgs) {
                document.getElementById("div").innerHTML =
                "tags[0] = "+tags[0];
            },
            error: function(type, data, event, kwArgs) {},
            timeout: function() {},
            timeoutSeconds: 10
        });
    });
</script>
<div id="div">
</div>
```

其中 `mytest.js` 的代码如下。

```
tags = new Array();
tags[0] = "Ajax";
tags[1] = "Java";
tags[2] = ".NET";
```

3. JSONP/JSON回调

Dojo.io.bind可以直接使用其他网站公开的JSONP/JSON服务，例如网络书签站点del.icio.us上提供的JSON服务，访问`http://del.icio.us/feeds/json/Dojomaster`会得到用户Dojomaster提交的书签列表，结果如下：

```
if(typeof(Delicious) == 'undefined') Delicious = {};
Delicious.posts = [
{
    "u": "http://manual.Dojotoolkit.org/WikiHome/DojoDotBook",
    "d": "The source for Dojo documentation",
    "t": ["Dojo", "documentation", "Ajax", "JavaScript"]
},
{
    "u": "http://Dojotoolkit.org",
    "d": "Cross Browser JavaScript Platform",
    "t": ["Ajax", "JavaScript", "dhtml"]
}
]
```

在 Dojo 中通过 Dojo.io.bind 请求 “`http://del.icio.us/feeds/json/Dojomaster`”，可以直接使用 posts 对象，代码如下：

```
<script type="text/JavaScript">
Dojo.require("Dojo.event.*");
Dojo.require("Dojo.io.*");
Dojo.require("Dojo.io.ScriptSrcIO");
Dojo.addOnLoad(getBookmarks);
function getBookmarks(){
    Dojo.io.bind({
        url: "http://del.icio.us/feeds/json/Dojomaster",
        transport: "ScriptSrcTransport",
        jsonParamName: "callback",
        load: function(type, data, event, kwArgs){showBookmarks(data);},
        mimetype: "text/json",
        timeout: function() {alert('timeout');},
        timeoutSeconds: 10
    });
}
//列表显示书签内容
function showBookmarks(posts){
    var div=document.getElementById("container");
    for (var i=0, post; post = posts[i]; i++){
        div.innerHTML += "<a href='"+post.u+"'>" +post.d + "</a><br/>";
    }
}
</script>
.....
<div id="container" class="bookmarks"></div>
```

4. DSR/Multipart

第4种跨域页面调用的方法使用DSR/Multipart，DSR是Dynamic Script Request的缩写，它解决了由于参数长度较长而导致URL超长的问题。通常情况下，使用URL传递参数会受到URL最大长度（1KB）的限制。使用DSR/Multipart方式请求页面时则可以将一个请求分成多次进行发送，

将URL中的参数分多次发送到服务器端，从而保证URL的长度不会超出限制的范围。使用这种跨域页面调用的方式并不需要特别设置，默认情况下Dojo会自动判断URL的长度来确定是否需要将请求分多次发送。以下是使用DSR/Multipart方式进行跨域页面调用的示例代码：

```
Dojo.io.bind({
    url: http://the.script.url/goes/here?para=.....,
    transport: "ScriptSrcTransport",
    forceSingleRequest: false, //是否强制使用一次请求
    useRequestId: true, //是否在请求中附加_dsrId参数
    load: function(type, data, event, kwArgs) {},
    error: function(type, data, event, kwArgs) {},
    timeout: function() {},
    timeoutSeconds: 10
});
```

版本信息

截止笔者交稿时，Dojo 官方站点提供的版本有 0.4 和 0.9。

Dojo 0.4 在原先 0.3.x 的基础上又提供了很多激动人心的特性，并修改了 529 个 Bug。这些新特性包括图形和动画方面的增强功能和一些新的 html 部件。

而 Dojo 0.9 是 Dojo 创建者对 Dojo 架构的一次重新思考和改造。

“……我们对 API 接口从底层开始往上进行重新思考，并把它划分成三个不同层次的项目，0.9 版本变得更快、更小、更快、更加容易理解和使用……”

这三个层次分别是：

1. Dijit。提供html部件的定义、数据绑定，部件的a11y和i18n特性。
2. Core。Dojo 0.9的核心，提供了一个裁剪后的Dojo.js文件，使用户能够更容易记住和使用API，以及新的构建系统等。
3. DojoX。提供了一种类似孵化器的实现，能够更高质量地实现以前实验的特性，包括离线存储、cometd等。同时还实现了DojoX.gfx对Silverlight的支持。

社区视角

Dojo 官方对为什么选择 Dojo 有这样的宣传：
<http://Dojotoolkit.org/book/Dojo-book-0-9/introduction/why-Dojo>

“宽度和深度。Dojo 是 full-stack 式的解决方案，不需要集成任何来源的组件……提供集成的基础架构和大量可选的模块……从通常的用户体验来说，这些组件是很棒的解决方案……”

“质量。国际化支持和可访问性交织构成整个 Dojo 的基础……提供正确的击键提示……所有组件都可以通过 CSS 进行自定义……界面风格一致……”

“性能。Dojo 每天都运行在高交易量和高并发量的网站上……它的 package 系统使开发大规模的 UI 项目变得很容易……Dojo 0.9 版本尤其注重 Dojo 性能的提升……”

“社区。Dojo 是个开放社区。许多个人和公司都加入进来创建一个对所有人都有好处的工具……license 会尽可能地与政治无关，并确保每个人都能容易地加入进来……”

另外，Dojo 网站上的大量详实的文档也是对开发者强有力的帮助。

1. Jon Aquino 使用过 Dojo 和 Prototype，他曾经对此做了一个简要的比较：(<http://Ajaxian.com/archives/comparing-Dojo-and-Prototype>)

“总的来说，两个我都喜欢。Prototype 就像保时捷，而 Dojo 更像是悍马。Prototype 能让人体验到编程的快感（感觉非常像 Ruby），而 Dojo 更加工程化（感觉像 Java）——但在某些地方（基于 IFrame 的解决方案）过度工程化了，从而才能提供其他工具包没有提供的功能（异步文件上传、Ajax 回退按钮处理、JavaScript “包含” 机制）。”

他同时提到 Dojo 的事件处理工具包是他遇到过的第一个真正有用的 AOP 应用程序。

2. 在 RaibleDesigns 站点上有人这样比较 Dojo 与 Script.aculo.us：(http://raibledesigns.com/rd/entry/script_aculo_us_vs_Dojo)

对于 Script.aculo.us 和 Dojo 来说，它们背后所隐藏的思想是我最欣赏之处，经过权衡，我心中的天平向 Script.aculo.us 发生了倾斜。Why？原因在于 Script.aculo.us 的创建者是一个设计者 (Designer) 而非一个开发者 (Developer)，它更多的灵感来自于大量的 Web framework 和真实的 Web application，而不是像 Dojo 一样是一个 Web library 的什锦拼盘。

另一方面，我们无法否认的是，Dojo 详实的文档可以说是它的一个亮点。不过我在这里不得不对 JotSpot 提出善意的批评，在大名鼎鼎的 Jot.com 上面，Dojo Wiki (<http://Dojo.jot.com/WikiHome>) 向我们展示了 Dojo toolkit 的方方面面，令人遗憾的是无论是排版，还是在 IE、FireFox 上的表现，它都让人大跌眼镜，毫无美感可言。

3. 国内的开发者对于 Dojo 开发也有自己的理解：(<http://www.Dojocn.cn/apage/amushen/archives/2007/2.html>)

Dojo 也许是将来的一个趋势，但是 appolo 可

能会比它更有竞争力。Dojo 到底是夕阳技术，还是朝阳技术，等待历史的验证吧。

不过现在我们小组决定，下一期的项目主要采用 Dojo 来做界面，所以，还是先研究透了再说吧，况且，对我来说，Dojo 源代码比较亲切，研究起来也不怎么费劲。

widget 也就是 0.9 版本中的 dijit，很有竞争力，还有一些底层的方法，比如 xml、加密、io、事件、异常、调试、图表（一般）、页面特效等，这些方法如果能比较熟练地使用，会给页面开发人员带来很多方便，同时，开发人员能够以比较接近 cs 的方式开发 bs 软件。

现在可惜的是：没有足够的 JavaScript 和 dhtml 的开发经验，很少有人能看懂 Dojo 的源代码；没有一个很好的 IDE 支持 Dojo；Dojo 本身也在不断成长，所以，很多方法变化比较大。比如从 0.4.3 版本到 0.9 版本，变化太大了，导致我们要修改所有的页面。io 的变化也太大了，令人汗颜；Dojo 需要的 js 文件比较大，如果在互联网上使用，加载速度是一个问题，局域网内使用效果还不错；Dojo 的代码比较好理解，但是不够简练，这点是优势也是劣势啊。

Dojo 的官方网站上有很多不错的资源，不过我想很少人愿意去研读它，大家希望的是，我想实现什么功能，马上就有现成的方案或例子供我使用，只要简单修改一下，就可以立即使用。

Dojo 的 mail list 有很多答疑和讨论，不过我想很少人会主动订阅而且仔细分析。

作者介绍

张凯峰，在 Java 和 Web 开发方面略有心得，现在从事软件测试、Automation 相关工作，曾参与译著《Ajax 实战》。

4

版本控制篇

综述

随着软件业的迅速发展，软件项目正变得日益庞大和复杂，在软件产品具备更为强大特性和功能的同时，也为开发者带来了一些不容忽视的问题，比方说，项目代码及文档和开发环境日益复杂化且难以管理，无组织的代码开发环境将导致潜在问题的产生，甚至影响整个项目的进程，造成历史数据丢失，降低产品质量与可靠性，以及维护困难，增大开发风险等问题。这些情况在没有得到有效管理的软件开发过程中，出现的问题和风险将越来越突出。

软件工程思想从理论体系建立到软件版本管理工具的普及，在技术和产品更新换代的同时，也一直为软件源代码版本管理带来新的方法和思路。通过软件源代码版本控制来记录开发过程中的代码变更，以及与其他开发者分工协作，建立规范化的软件开发环境，已被实践证明是解决上述问题的有效途径并将在软件开发过程中得到进一步普及。

目前，在进入软件工业化生产时代的项目开发，已经很难单靠一个人的努力来完成。项目通常是由一个研发小组来共同分析、设计、编码和维护，并有专门的小组对已完成编码调试的软件进行细致的测试。在软件开发这个复杂的过程中，需要涉及各个方面的人员，信息与资源的交流不仅存在于研发小组的成员及各个研发小组之间，还存在于客户和研发者之间。这些交流反馈信息都可能导致对软件的修改，小到对具体代码源文件的改动，大到重新设计程序模块甚至变更对最初需求的描述。在这个过程中，在反复的代码变更中可能形成不同的软件版本，这需要开发者对项目各个版本进行统一管理，并且保证项目开发小组成员之间以一种有效的机制进行协调，之后对研发小组各成员所做的修改进行统一汇总并纳入到存储库管理之中。

随着版本管理机制的采纳，目前版本管理系统已经成为软件开发者必备的辅助工具之一，包括源代码、文档在内所有需要进行协作的内容都可以纳入到存储库的统一管理中。在众多的版本管理产品之中，有很多已经非常成熟的商业或开源社区提供的解决方案，在此笔者将从开放源代码软件领域里

面选出三款优秀且具有广泛用户群体的源代码版本管理工具，如 CVS、Subversion 及 Git，来逐一了解开源版本控制系统的功能与特性，让这些优秀的开源程序在软件版本管理中起到有力的辅助作用。

关联信息

下面是常见的代码存储库管理工具。

1. **CVS**：老牌的代码存储库管理工具，帮助开发者将项目存储在资源存储库中，并记录版本存储库中源文件的改动历史及不同版本之间的差异；
2. **Subversion**：免费、开放源码的版本控制系统，帮助开发者记录文件和目录的每次改动，允许用户随时恢复以前历史版本的数据；
3. **Git**：Linux之父Linus Torvalds为Linux内核开发而专门打造的版本控制软件，是一个强大的分布式版本管理系统；
4. **ClearCase**：提供给开发者独立的工作空间管理设施，可以实现动态评估及选择指定用户版本和透明访问等多种版本管理配置功能；
5. **StarTeam**：采用CS架构，使用数据库在后台进行历史版本的管理，安全性好，同时具有配置管理、缺陷管理、需求管理等功能。

对于软件版本管理系统来说，其主要功能一般包含以下内容：

1. 对软件项目代码和文档进行统一管理；
2. 帮助项目开发小组成员进行有效的分工与协调；
3. 将团队成员对代码和文档做的修改进行统一汇总；
4. 保存历史修改记录，以便随时回馈到之前的版本之中；
5. 对开发过程中形成的软件的各个版本分支进行标识和管理；
6. 必要的权限和安全机制来保证版本存储库易于管理控制；
7. 为代码版本冲突提供有效解决方式。

4.1

Subversion

总评

上手容易程度	★★★★★
社区活跃性	★★★★★
应用广泛性	★★★★★
推荐指数	★★★★★

Subversion 是一个免费、开放源码的版本控制系统。在 Subversion 的存储库中记录文件和目录的每次改动，允许用户随时恢复以前历史版本的数据，或者是检查数据变化的历史。Subversion 允许使用者通过互联网访问存储库，存储库可以随时被位于互联网各处的开发者访问并在此基础上进行分工协作。

Subversion 是一个能够管理任何文件集合的通用系统，存储库中代码的变更可以被 Subversion 的触发器感知并进行快速的响应和管理，当开发者提交代码时，通过设定的触发器能够自动给开发者发送邮件通知，相关的开发者可以立即审查特定代码的改动是否符合要求，使得在开发者之间的编码协作更为顺畅。

功能和特点

1. 方便地对历史问题进行检索；
2. 改善了CVS的设计，实现了一个虚拟的版本化文件系统，可以跟踪整个文件目录树随历史记录的改变；
3. 真正的版本历史记录，可以增加、删除、复

制、重命名文件和目录；

4. 提交处理的原子性，一组改动要么完全进入资料库，要么完全不进入，防止只有部分改动成功发送到资料库所带来的问题；
5. 版本化源数据，每个文件和目录由一个键值对组成的属性集与之相关联；
6. Subversion可以作为扩展模块加入到Apache Http Server中，增强了Subversion在稳定性和互操作性上的优势，并具有可利用Apache 服务器认证、授权、压缩传送等特性；
7. 由二进制差别算法表示的一致数据处理，所有类型文件同样压缩存放于存储库中；
8. 高效的分支和标签，使用类似硬链接的机制来复制项目，操作的代价相对较少；
9. Subversion提供一组定义良好的API，使其可以方便地被作其他应用和语言利用。

背景介绍

分布在不同地域的软件开发者，希望通过相互协作把各自的代码最终集成在一起。在 Subversion 出现之前，开源社区的开发者相互协作时最常用的工具就是 CVS，但是 CVS 就像一台老爷车，架构陈旧并且经常出现问题，不过开发者需要一直忍受 CVS 的种种缺点并不断对其进行修补。开发者们大都希望有更有趣的方式，以相互协作编写代码。

在 2000 年初，CollabNet 公司打算让开放源代码的版本管理系统可以适应企业级的应用

需求，并希望开源软件同样可以在企业领域得到推广和应用。CollabNet 此前提供一个称作 SourceCast 的协作软件包，版本控制是其中组件之一。并且 SourceCast 在使用 CVS 作为最初版本管理系统时就感受到了 CVS 的局限性，CollabNet 明白必须找到一个更好的替代系统。于是开始与 CVS 的开发者们取得联系，探讨如何对 CVS 版本控制系统进行修补和更新，但是随后发现 CVS 的开发社区并不活跃，甚至还会试图说服开发者满足于使用当前的版本控制系统。

可在当时，CVS 已经成为开放源码社区中的事实标准，并且没有其他免费且更好的系统来替代它。因此 CollabNet 决定从头开发一个新的版本控制系统，在 CVS 社区的开发者中，恰巧开发者 Karl Fogel 也在考虑设计一个新的版本控制系统，因为 Karl 在使用 CVS 时受到的束缚已经让他开始仔细思考数据管理版本的更好路子，并且他已经想好了新的名称“Subversion”，之后 Karl 加入 CollabNet 公司，与一小群开发者一起在 CVS 思想的基础之上重新构建了新的工具 Subversion，为基于版本控制系统的协作编程带来革命性的变革，而不是简单地重复构建一个复制品。

由于很多开发者同样有受限制于 CVS 的感受，Subversion 很快成为一个活跃的开发者社区。

大约用了 12 个月的时间，Subversion 就有了一个原型系统。在第 18 个月时，Subversion 就可以支持自身的代码更新了，开发者停止使用 CVS 作为 Subversion 的代码存储库，转而使用 Subversion 本身。在项目开始的 4 年之后，Subversion 的 1.0 版本终于问世，支持所有 CVS 现有的功能，并在此基础之上增加了事务提交、分支管理、变动跟踪等众多新的特性。

Subversion 仍然像大多数开源项目一样运作，建立在一个松散透明的开发者社区之上。CollabNet 的版权许可证和 Debian FSG 完全兼容，任何人都可以免费下载、修改，按自己的意愿重新发布 Subversion，而不必得到来自 CollabNet 或其他任何人的许可。

目前的 Subversion，已经可以在带宽有限的情况下，稳定可靠地服务于分布在世界各地的开发者。并且开发者可以很方便地将 Subversion 作为插件集成在各种工具平台中，比如 Eclipse、Netbeans、Visual Studio 等开发平台都提供插件集成了对 Subversion 的支持。目前使用 Subversion 的并不仅仅限于程序开发者，版本控制系统如同时间机器一般，为其他如文档管理等方面的应用都提供了很好的支持。

快速上手教程

在开始之前，我们先简单看一下 Subversion 官方文档中提供的总体架构图，在架构图中清晰地展示了 Subversion 的设计模式（见图 4-1）。

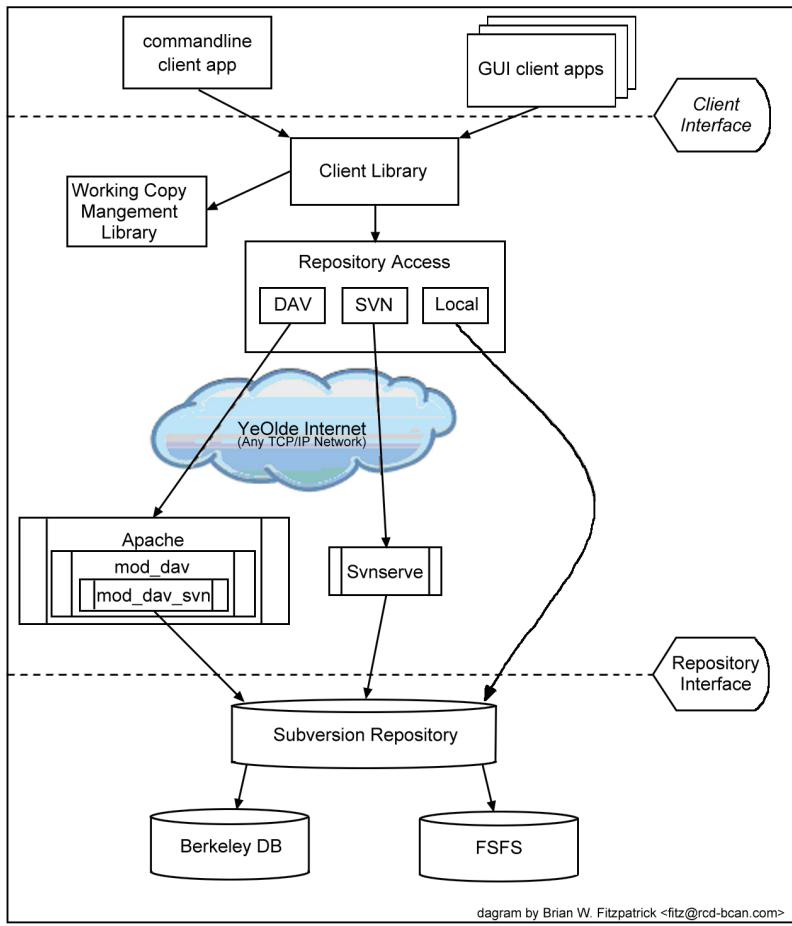


图4-1 Subversion总体架构图

架构图的底端是 Subversion 存储库，负责管理所有的版本化数据，顶端是 Subversion 客户端程序，包括命令行工具和客户端 GUI 工具，负责管理存储库数据的本地映射及与服务器的交互。在服务端与客户端之间，有多种存储库的访问通道，Subversion 可以作为 Apache 2.0 的一个模块工作在 Web 服务器上，通过 Web 服务器来建立对存储库的访问，也可使用 Subversion 自带的小型服务器程序和自带的通信协议，通过 SSH 以 Tunnel 方式使用来直接访问存储库。

Subversion 服务器与客户端在 Windows、Linux、Mac OS X、Solaris 上都可以顺利运行。Subversion 安装包中包含了基于命令行 SVN 工具，在图形化客户端方面，面向 Windows 平台发布的 TortoiseSVN (<http://tortoisेस्वन.net/>) 是目前使用起来最为方便的。TortoiseSVN 是扩展 Windows Shell 的一套工具，可以看作 Windows 资源管理器的插件，安装之后 Windows 就可以识别 Subversion 本地存储库的工作目录。

下面我们以 Windows 环境为例来看 Subversion 服务器的部署与客户端使用。Subversion 最新版

的下载地址为：<http://subversion.tigris.org/>。

Subversion 服务器和 TortoiseSVN 只需要选择适当的安装路径，其他根据默认选项，安装完成重启后即可加载 Subversion 服务。安装后在资源管理器的空白处单击右键，可以在 TortoiseSVN 设置中进行外观、网络或拓展程序的设置（见图 4-2）。

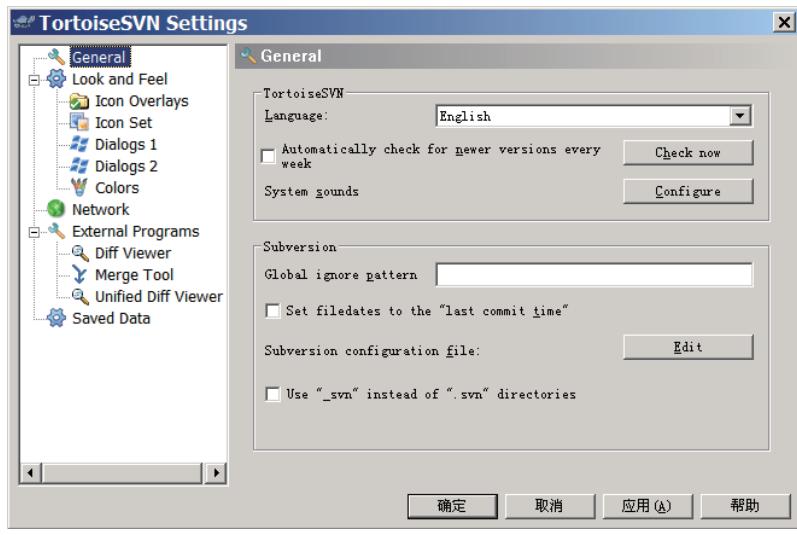


图4-2 TortoiseSVN设置窗口

在使用 Subversion 存储库之前，首先需要建立一个版本库，可以将其看作服务器上数据存放的仓库，下面我们在命令行中通过 svnadmin 命令来创建新的版本库，版本库位于 D 盘根目录下的 demosvn 文件夹中（见图 4-3）：

```
svnadmin create d:\demosvn
```

A screenshot of a Windows Command Prompt window titled 'C:\WINDOWS\system32\cmd.exe'. The command 'svnadmin create d:\demosvn' is entered and executed. The output shows the creation of the repository: '驱动器 D 中的卷没有标签。 卷的序列号是 1835-12F5' (The drive D has no label. The volume serial number is 1835-12F5). Then it lists the directory structure of the newly created repository: 'D:\demosvn' 的目录 (Content of D:\demosvn). It shows a file 'README.txt' and several empty directories ('dav', 'locks', 'hooks', 'conf', 'db', 'format'). At the bottom, it provides summary statistics: 2 个文件 (2 files), 236 字节 (236 bytes), 7 个目录 (7 directories), and 1,293,549,568 可用字节 (1,293,549,568 available bytes).

图4-3 新建的SVN存储库

可以看到，svnadmin 命令为我们生成了存储库管理所需的目录。SVN 存储库的创建同样也可以使用 TortoiseSVN 图形化工具来完成，在资源管理器中点击右键，点击 TortoiseSVN 操作在此创建存储库选项。然后在版本库模式选择中，选用默认的本地文件系统（FSFS），就可以创建 SVN 存储库相应的目录和文件（见图 4-4）。

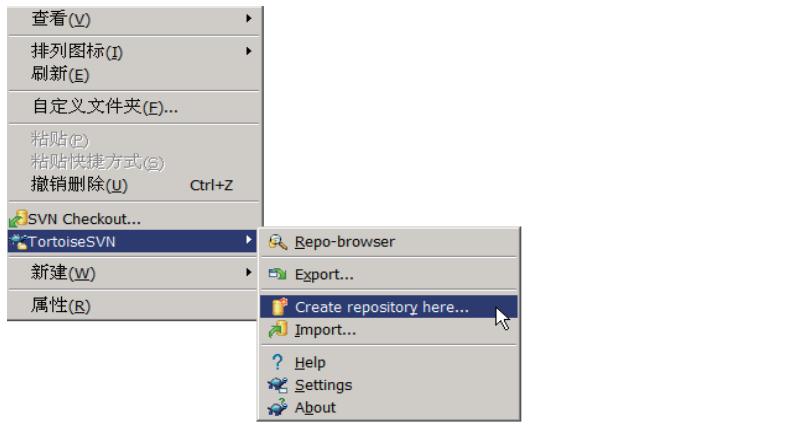


图4-4 创建SVN存储库相应目录和文件

在存储库创建之后，我们需要运行服务器，来使得 SVN 存储库可以接受外部的访问请求。进入命令行模式并运行命令：

```
svnserve -d -r d:\svndemo
```

这样，SVN 服务器已经通过命令行启动。

或者我们也可以把 svnserve 作为系统服务添加进操作系统服务启动项，使得 SVN 服务在开机时启动。

Subversion 从 1.4 版本开始，可以通过 Windows 系统服务的形式在开机时自动运行。所以最为简便的方式就是对 Subversion 的 Windows 服务进行手动安装注册，我们打开命令行窗口，输入下面的命令串：

```
sc create svnserve binPath= "\"d:\Program Files\Subversion\bin\svnserve.exe\" --service --root d:\demosvn" displayname= "Subversion Repository" depend= Tcpip start= auto
```

其中，sc 是 Windows 自带的服务配置程序，参数 binPath 表示 svnserve 可执行文件的安装路径，笔者的安装路径是 D 分区的 Program Files 目录下，由于路径中的 Program Files 带有空格，因此路径需要用双引号引起。--service 参数表示以 Windows 服务的形式运行，--root 指明我们刚才创建 SVN 存储库的位置。displayname 表示在 Windows 服务列表中显示的名称，而接下来的一项表示 svnserve 服务的运行需要 Tcpip 协议支持，start=auto 表示开机后自动运行。

svnserve 服务安装并重启后，服务列表中会显示出已经随开机自动启动的 svnserve 服务（见图 4-5）。svnserve 服务的卸载仅在命令行执行 sc delete svnserve 即可。

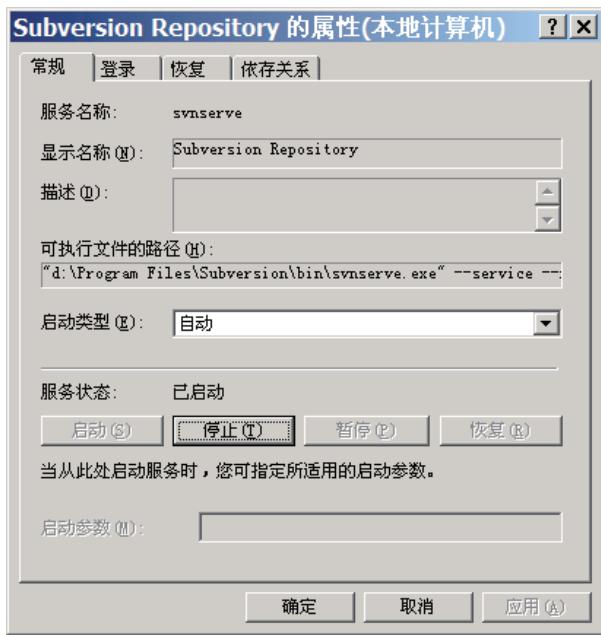


图4-5 自启动的svnserve服务

下面我们打开资源管理器，在空白处点击右键，选中 TortoiseSVN 菜单中的 SVN Checkout，在 URL 对话框中输入刚才启动的 SVN 服务器地址 svn://localhost，点击确定之后，就可以将新建的存储库同步到本地的目录中（见图 4-6）。

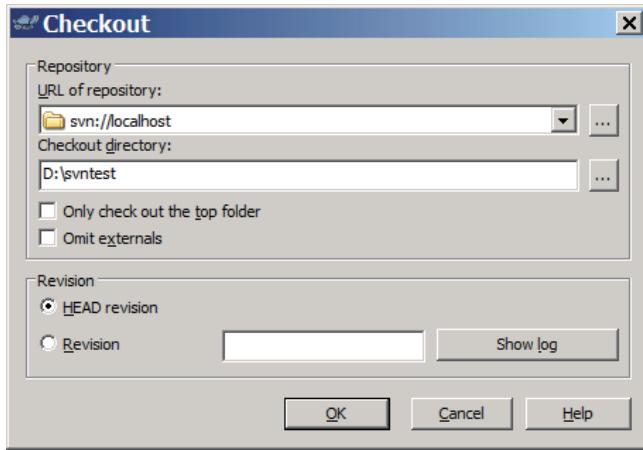


图4-6 使用TortoiseSVN取出新建的SVN存储库

在向存储库提交文件之前，一般需要先创建授权的访问用户。在 SVN 服务端目录中，conf 文件夹下的文件负责存储库的配置。其中 conf 下面有 3 个文件——authz、passwd、svnserve.conf，其中的 svnserve.conf 是存储库的配置文件，决定了使用什么形式的认证和授权文件，配置文件中的

password-db 指定了用户密码文件，authz-db 是授权文件。下面打开文本编辑器，我们将 svnserve.conf 文件中 password-db = passwd 前面的 # 注释去掉，以使用户密码文件生效，这样用户的认证信息就从默认的文件 passwd 中读取。之后我们打开 passwd 文件，在其中添加存储库的访问用户名和密码，如 svn = svn_admin，这样我们就可以使用用户名 svn 和密码 svn_admin 来对刚才建立的存储库进行访问。

下面我们加入两个测试使用的文本文件，之后通过 TortoiseSVN 的菜单选择提交，在提交的同时加入相应的注释，并点击确定（见图 4-7）。

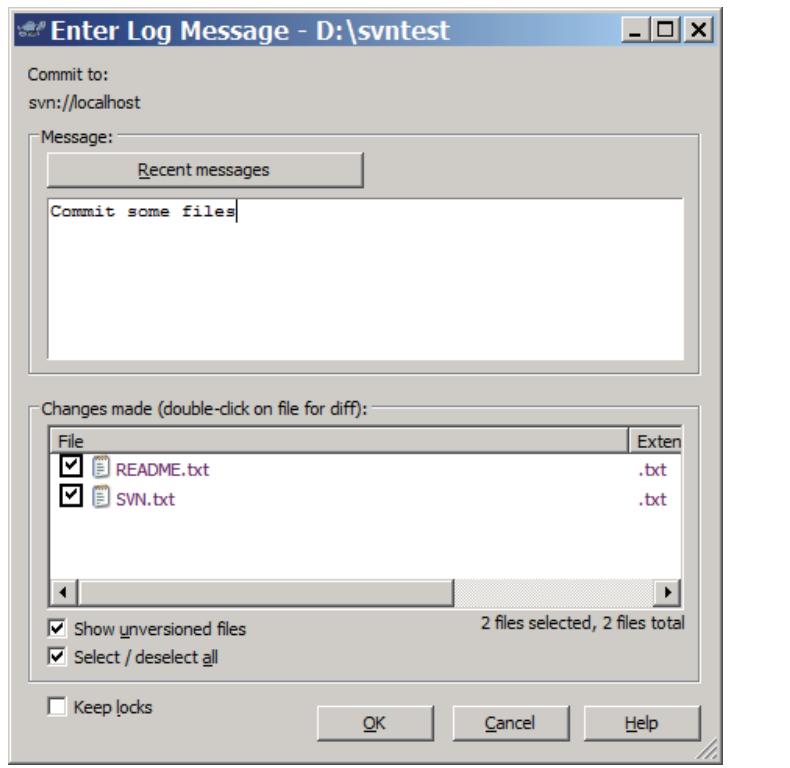


图4-7 选择需要提交的文件并添加注释

在接下来的对话框中输入我们刚才在 passwd 文件中加入的用户名和密码，并点击确定提交更改（见图 4-8）。

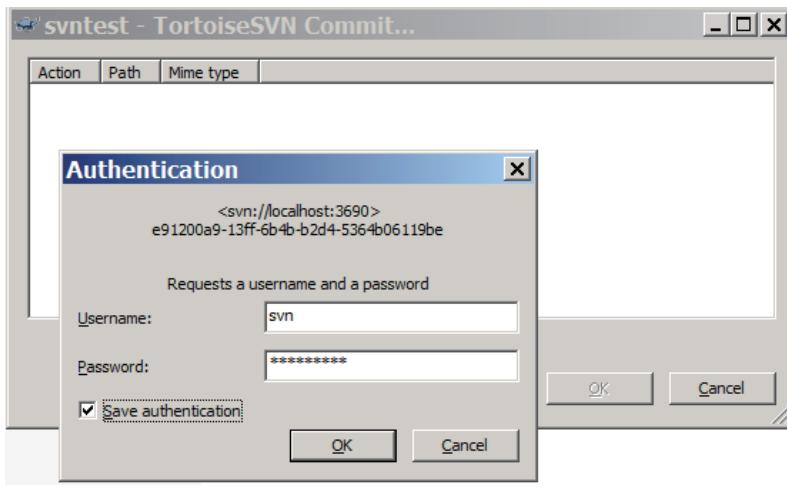


图4-8 输入用户认证授权

在 Subversion 的版本更新记录中，可以看到，我们为存储库新增加了两个文本文件，并且将文件提交到存储库之中（见图 4-9）。

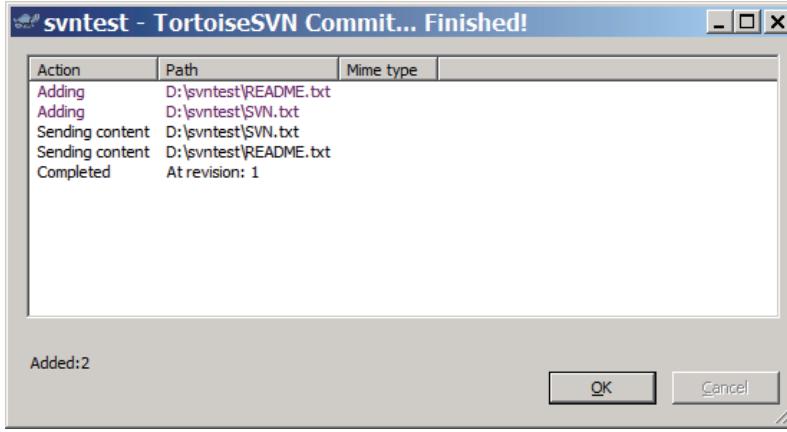


图4-9 已经提交到存储库中的文件

在文件提交之后，可以通过 TortoiseSVN 客户端提供的存储库浏览器查看存储库中已有的文件、版本以及提交者等相关的信息（见图 4-10）。

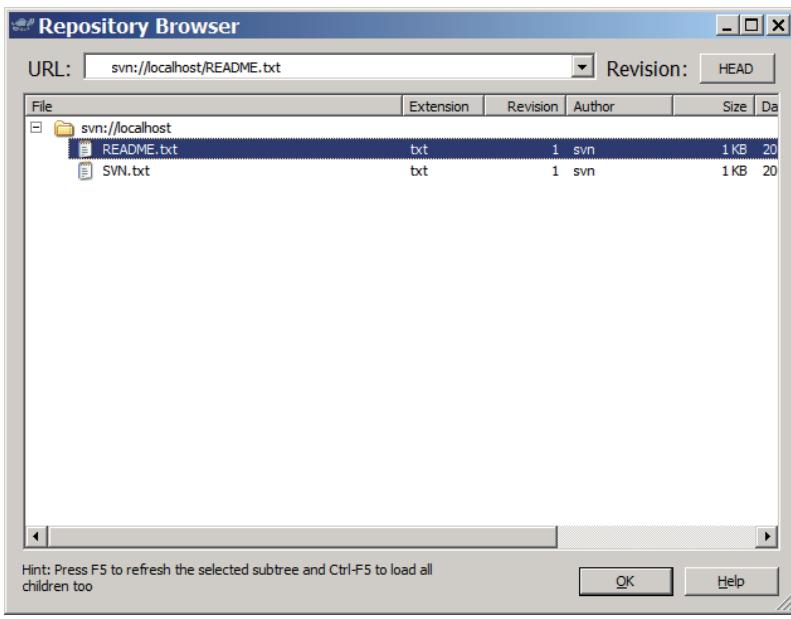


图4-10 存储库浏览器中的当前存储库版本

与资源管理器集成的 TortoiseSVN 客户端，在文件提交之后，会对文件和文件夹更换具有不同含义的图标。使用者可以根据不同图标的含义来获得本地文件与远程存储库之间的关联关系。文件的 Subversion 状态的不同，对应的图标也不同，表 4-1 显示出了不同图标与存储库文件当前状态的对应关系。

表4-1

	与版本库中文件保持同步的图标，表示Subversion状态正常
	用户在编辑文件之后，状态变成已修改，图标更改为红色的感叹号。这些文件在上次更新之后进行了修改，需要再次被提交到版本库之中
	如果在提交的过程中出现了冲突，图标变成黄色感叹号
	在用户给文件设置了加锁属性之后，Subversion会让此文件属性为只读，直到获得文件锁之后才可进行提交
	如果用户拥有文件的锁，并且Subversion状态是正常的，该图标就提醒用户如果不使用该文件则应该释放锁，来允许别人提交对该文件的修改
	该图标表示当前文件夹下的某些文件或文件夹已被计划从版本控制中删除，或是该文件夹下某个在存储库控制中的受控的文件丢失了
	加号提示用户有一个文件或目录已经计划加入到版本控制之中

为了方便开发者，Subversion 也为不同的开发工具提供了相应的插件，比如 Microsoft Visual Studio、Netbeans、Eclipse 等，都可以直接将 Subversion 集成在开发环境之中，方便开发者随时提交改动并对存储库进行管理。

Subversion 服务器也可以在为 Apache HTTP 服务器添加 mod_dav_svn.so 模块，使 Subversion 服务随 Apache 的启动同时加载，之后可以通过 HTTP 的方式来访问 Subversion 存储库。但需要注意

的是，在使用Apache作为服务器时，用户许可和访问权限需要根据Apache服务器文件来进行配置。

更多的 Subversion 使用资料，可以参看 Subversion 官方站的教程（<http://subversion.tigris.org/servlets/ProjectDocumentList>），以及中文文档“采用 Subversion 进行版本控制”（<http://www.cn-java.com/subversion.pdf>）中的配置与使用介绍。

版本信息

目前最好的 Subversion 稳定版本是 1.4.6。在这个版本中，增加了一个被称为 svnsync 的新的存储库镜像工具，svnsync 上一个实现可维护只读 Subversion 镜像功能的工具，svnsync 帮助用户提供了版本库备份，这样便于使用者进行灾后恢复和软件升级。svnsync 工作方式是将一个版本库的修订版本重新放到另一个版本库之中，即镜像版本库和主版本库使用相同的规则，对镜像版本库执行操作的用户需要具备完全的写权限。在 svnsync 中，同步用户需要对整个版本库有读/写权限，同步用户需要能够修改特定修订版本属性，镜像版本库需要对除同步用户以外的用户只读。

除此之外，Subversion 的 1.4.6 稳定版本实现了大型工作拷贝的性能提升，并且为 BerkeleyDB 数据库及其自动恢复特性提供了支持，同时更新了新的 API 函数，以及对 40 多处的 Bug 进行修复。

CollabNet 公司准备发布的 Subversion 1.5 版本将会出现许多新的亮点，增加被称作 Merge Tracking（合并跟踪）的新功能，为代码比较与合并增加新的特性。除此之外，Subversion 1.5 中还包含称作 Sparse Directory（稀疏目录）的新增特性，详细的功能文档可以在版本发布时得到。对于使用者来说，Subversion 1.5 将是一个值得期待的产品。

社区视角

在开源世界里，长久以来 CVS 一直都是版本控制的首选。但随着 Subversion 的诞生与普及，更多的用户逐渐倾向于使用 Subversion 来替代 CVS 进行版本管理。大多数使用者都认为拥有自由、开放的源代码并且可用来管理任何类型文件的版本控制系统 Subversion 比 CVS 更好用。

的确，和 CVS 相比，Subversion 拥有更多优点，例如目录版本控制、提交的原子性、一致的数据处理方式和更有效率的分支与标记等，而且国内外也有很多 Subversion 技术站点为其提供了文档和讨论组支持。这些都为 Subversion 的使用带来了更多的便捷，同时也为开发者赋予了更为强大的版本功能控制能力。

同时拥有版本控制工具 Subversion 和 CollabNet 开源软件协作平台的 CollabNet 创始人 Brian Behlendorf 在接受采访时也曾说道：Subversion 是轻量级的，相对于商业的解决方案非常容易安装和部署。而且 SVN 的适用范围广泛，从文档管理人员到需要复杂操作的程序开发者，都可以满足其各自版本管理的需求。而类似于 Rational ClearCase 的版本控制系统仅是针对于程序开发者设计，并且大多是针对企业网内部的协作开发，对分布在不同地点的跨 Internet 编程协作，并没有提供有效的支持。Brian 认为，目前软件领域的创新和最有趣的想法，大多集中在开源领域之中，而不是出现在商业的开发者讨论组之中。

作者介绍

高昂，博士研究生，关注开源软件与动态语言的发展。目前在资源与环境信息系统国家重点实验室从事网格 GIS、空间数据库研究工作，同时也是 OSGeo 中国和 InfoQ 中文站成员。业余时间里，喜欢徒步旅行并担当过数次北京近郊登山活动的领队，常写旅行游记。个人站点为开源网格 GIS 试验田 (<http://www.gaoang.com/>)。

5

项目管理篇

综述

项目管理和软件的缺陷跟踪始终是软件开发生命周期之中不容忽视的话题，随着国内软件产业的发展，指导软件开发的过程与方法已经在项目实践中得到广泛的应用。软件缺陷也就是软件的 Bug，指软件中存在的设计错误、编码漏洞、需求或设计变更以及种种缺陷、疏忽和过失等。对软件缺陷的跟踪与管理，贯穿于整个软件开发生命周期的始终，是软件开发中不可缺少的环节，同时也是保证软件产品质量的重要内容。

缺陷涵盖的范围延伸到软件开发的整个生命周期，一旦软件 Bug 被发现，需要立即进行记录并分派团队成员去追踪解决产生的软件缺陷，并确定各种 Bug 的优先级别，然后按照优先程度进行处理。

在实际的软件开发过程中，软件缺陷常常是难以管理和控制的，这主要是因为开发者无法准确地预知缺陷会出现在软件开发的哪个阶段。有些情况下，缺陷往往是在产品开发的中后期发现，这些问题很可能是由最初架构设计缺陷造成的，解决这样的缺陷需要投入大量的人力和时间，不但会增加项目开发的人力和时间成本，同时可能会为项目引入新的 Bug。

软件缺陷不仅局限于通常意义上所说代码级别的错误，同时包括在设计和测试阶段发现的缺陷。但目前大多的缺陷管理工具，都是根据软件编码阶段进行分类的，因为编码阶段是产生缺陷最集中的时期。无论在软件开发的哪个阶段，在程序中任何潜在的错误，需要引起注意的地方及值得跟踪的问题，都需要作为开发过程中的 Bug 来认真记录并在开发过程中予以解决。

在所有应用程序中，都不可避免地会存在各种缺陷，但这并不意味着我们要忽视软件设计的重要性，存在缺陷相对较多的程序，尽管有时并不会影响软件的正常功能，但会给正常运行留下难以预测的隐患。设计良好的程序包含有的 Bug 会相对较少，或许这些缺陷不会妨碍程序的正常应用，但在某些情况下也可能会出现问题。

在参与的项目中，如果测试人员提交的缺陷还没有汇总和处理；软件错误报告仍旧没有及时提

交，开发人员与测试人员难以通过错误的反馈进行相互沟通；软件变更信息无法在短时间内顺利传达；软件项目交付使用时由于缺陷太多而遭用户抱怨，那么，软件缺陷管理的实施计划与应用缺陷管理软件到项目之中，将是开发团队的当务之急。

笔者将在开放源代码软件领域里面选出四款优秀的缺陷管理与项目管理工具，让我们逐一了解 Mantis、Bugzilla、Scarab、Teamwork 这些开发辅助工具的功能与特性，希望这些优秀的开源程序可以真正在您的软件项目之中起到适当的作用。

关联信息

基于 Web 的软件缺陷管理方面的优秀开放源代码项目很多，几乎各种 Web 开发语言都有相应的缺陷管理工具，常见的软件缺陷与项目管理工具的列表如下：

1. Mantis：一款基于Web的软件缺陷管理工具，配置和使用都很简单，适合中小型软件开发团队；
2. Bugzilla：一款老牌的软件缺陷管理工具；
3. Scarab：基于Java的问题管理系统；
4. Teamwork：基于Java的软件缺陷管理系统，安装使用非常便捷；
5. Trac：基于Python的包含Wiki在内的问题跟踪系统；
6. GNATS：GNU维护和使用的 Bug 跟踪系统；
7. ITracker：基于JavaEE的issue/bug跟踪系统；
8. TrackIt：基于Web的项目追踪工具；
9. BugFree：国产的开源Bug管理系统。

对于一款软件缺陷与问题跟踪系统来说，主要功能将会包含在以下的条目之中：

1. 支持多个项目管理；
2. 问题录入和反馈；
3. 问题查询和关键词检索；
4. 问题更新订阅；
5. 问题讨论面板；
6. 个人显示和E-mail通知设定；
7. 集成版本控制工具；
8. 消息发布；
9. 统计分析、报表生成和输出；
10. 用户分级别管理；
11. 自定义的面板；
12. 系统设置。

上述各个软件缺陷管理工具基本覆盖了缺陷管理的生命周期中的主要环节，典型的软件缺陷管理的生命周期如图 5-1 所示。

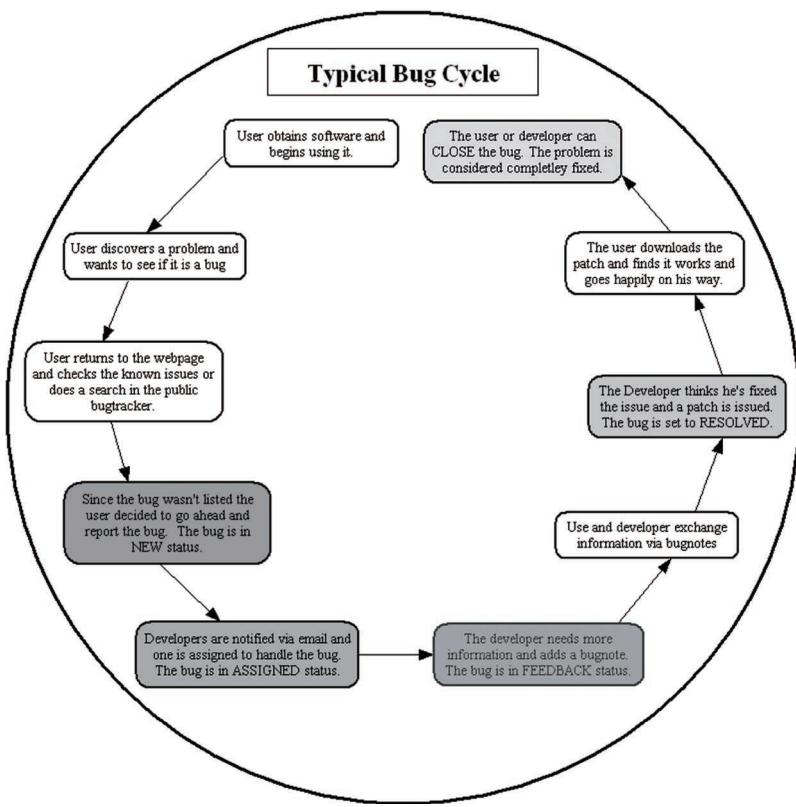


图5-1 软件缺陷管理的生命周期

5.1

Teamwork

总评

上手容易程度	★★★★★
社区活跃性	★★★★
应用广泛性	★★★★
推荐指数	★★★★

Teamwork 是一款非常优秀的软件项目管理工具。随着敏捷开发、统一过程方法等概念慢慢为人所熟悉，团队合作已经逐渐被认识和接受，并且逐步应用到实际软件项目开发管理当中。为保证团队合作的顺利进行，良好的团队协同管理工具是必不可少的。其实项目管理工具并不鲜见，为数不少的商业产品一直占据市场主流，如微软公司的 Project 和 Primavera 公司的 P3 等，但是小巧而功能强大的 Teamwork 同样具备不少令人称赞的特性。

Teamwork 是源自软件仓库 SourceForge 的开放源代码项目，针对团队合作的整个管理周期而设计。Teamwork 集成了项目进度、团队成员、工作日志及问题追踪等多项管理模块，并且将项目管理和文档管理很好地结合在了一起，大大方便了项目团队的分工与协同，提高了整个团队的工作效率。

自从 Teamwork 项目于 2005 年 1 月份在 SourceForge 上启动以来，经历了 1 年多的发展，整个应用架构已经非常稳定和成熟。从 SourceForge 上的统计报表可以看出，Teamwork 的用户数量和下载量一直保持在同类项目管理工

具的前列，以其先进的技术构建和完善出众的功能，深受广大开源爱好者的欢迎。

Teamwork 项目注册启动时，使用 LGPL 开源协议，允许用户免费使用并自由拓展其功能。由于 Teamwork 在团队管理实践中的出色表现，随着版本的不断更新，在 Teamwork 3 版本发布时，已经由原先的 LGPL 开源协议逐步转化为商业运作模式，但 Teamwork 开发团队仍旧在 SourceForge 上提供了程序的源代码下载，供爱好者研究与学习。

功能和特点

在用好 Teamwork 之前，先让我们从实现技术上认识一下这个项目管理工具。Teamwork 的开发者 Open Lab 擅长提供 Java 领域内的解决方案，在开源框架方面广泛地使用 Hibernate 框架和 JBoss 项目组提供的业务流程解决方案 JBPM。同时，Open Lab 拥有自己的开发框架 JBlooming，并且一直在不断地将其拓展和完善。

Teamwork 基于 J2EE 体系实现并且按照 MVC 的分层结构进行组织，从技术架构上来看，Teamwork 可谓是集众多的优秀开源项目于一身。Teamwork 的持久层设计使用了著名的 Java 对象关系映射框架 Hibernate，对 JDBC 进行轻量级的对象封装，用对象管理的方式来读写数据。从 3.0 版本开始，Teamwork 使用 Open Lab 自己出品的 Java Web 开发框架 JBlooming 来进行表示层设计，JBlooming 使用标准 Java API 和面

向对象的 Jsp 模板技术实现，简化表示层组件调用及 Ajax 实现的代码量，提高了 Web 应用的开发效率。Teamwork 的内容索引则使用了 Apache Jakarta 的 Lucene 项目实现。并且在 Teamwork 内部集成了一个简单的工作流引擎，可以和 Open Lab 的 FlowWork 项目很好地协同来进行工作流程交互与管理。

如果深入 Teamwork 的源代码，可以进一步研究 Teamwork 的各种 Java 类功能实现，以及 teamwork.hbm.xml、common.hbm.xml 等 Hibernate 持久层映射配置文件，以了解程序结构和后台数据库的组织方式。本文的主要目的是把 Teamwork 这个优秀工具的使用方式介绍给大家，暂时不把太多笔墨放在 Teamwork 代码的研究中，有兴趣的读者可以在 SourceForge 下载 Teamwork 代码，来进行深入的学习和研究。

背景介绍

Teamwork 的开发者 Open Lab 是一家来自于佛罗伦萨的软件公司。佛罗伦萨是意大利的历史名城，被称为意大利文化的首都，是意大利乃至整个欧洲文艺复兴的发源地。在意大利语中，佛罗伦萨是“鲜花之城”的意思，正如这座城市美丽的名称一样，佛罗伦萨是意大利传统和艺术的宝库，整个城市保留着文艺复兴时的风貌，到处弥漫着当时文化繁荣的气息。

Open Lab 于 2001 年成立在这样一个具有浓烈艺术气息的城市里，除了软件开发主项之外，Open Lab 还在全球范围内提供软件解决方案和

服务，借助于开源项目 Teamwork 的影响力，Open Lab 已经逐渐成为具有一定国际声誉的软件公司。在 Teamwork 官方站列出的客户名单中，我们不难发现很多知名企业的身影，Teamwork 的客户传统上多集中在欧洲，如法国、意大利、德国的不少企业，随着 Open Lab 业务的不断拓展，其用户也不乏像美国通用电器这样的跨国巨头。

Teamwork 项目的两位主要开发成员 Pietro Polzinelli 和 Roberto Bicchierai，同时也都是 Open Lab 旗下另一个 Java 开源项目 JBooming 的开发者。作为 Open Lab 的创始人，Pietro Polzinelli 本身就是 Java 开源项目的狂热追随者，经常在知名的 Java 技术站点 The Server Side 发布项目的信息，每当 Teamwork 或 JBlooming 项目有新的版本发布，他会第一时间将版本更新消息在社区中进行发布。Pietro 对著名的实体关系映射框架 Hibernate 的设计理念非常赞赏，曾对其进行过深入的技术研究，感兴趣的读者不必费多少力气就可以在 Google 上搜索到 Pietro Polzinelli 关于 Hibernate 运作机制的 PDF 格式分析文档，这也可能是为什么 Teamwork 的持久层设计会理所当然地选用 Hibernate 的原因。

Pietro Polzinelli 这位开源爱好者不仅对编程相当痴迷，还对哲学和政治方面的内容感兴趣，在 Pietro 的博客 (<http://ppolzinelli.blogspot.com/>) 上，可以读到他所关注的内容。当然，Pietro 的博客还是以他所热衷的技术为主，在这里可以读到 Teamwork 与使用 Ruby on Rails 开发的在线项目管理站点 Basecamp 的功能比较。

快速上手教程

由于具有跨平台特性，Teamwork 可以部署运行在不同的操作系统之上，Teamwork 官方发布了 Windows、Linux、Unix 及 Mac OSX 等平台下的安装包。在 Teamwork 3.0 版本发布之后，它由先前

版本使用的 LGPL 开源协议转变为商业模式，当然，Teamwork 的源代码仍然可以在 sourceforge.net 下载，供爱好者使用学习。

Teamwork 3.0 的安装包集成了 tomcat 5.5.17 作为解析 JSP 页面的容器，安装中可以根据实际情况选择 MySQL、SQL Server、Oracle 及 PostgreSQL 等数据库作为后台数据库支持。当配置生效之后，Teamwork 会自动在数据库中生成系统运行所需的各种表结构。在国际化方面，Teamwork 已经将英语、法语和意大利语等多国语言包集成在产品中，可以根据需要切换不同语言，随着 Teamwork 项目影响力的不断扩大，相信中文在不久之后也会加入到 Teamwork 的多国语言支持中来。

安装完成之后，Teamwork 会启动 Apache Tomcat 来加载应用。默认的管理员账户是 Administrator，初始密码为空，利用这个账户，就可以对 Teamwork 的各个模块进行设置。安装之后的初次登录界面如图 5-2 所示。

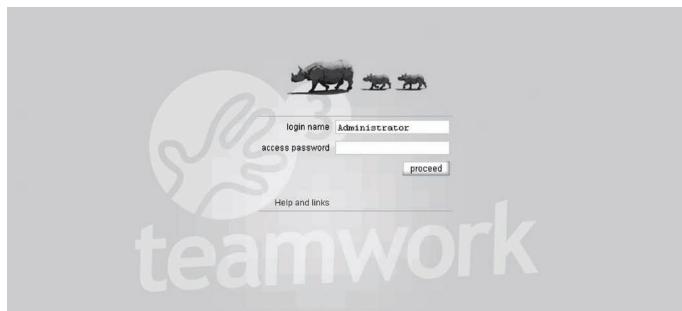


图5-2 安装完成之后启动的Teamwork登录界面

对于团队项目开发的生命周期，美国宾夕法尼亚州立大学技术援助项目主任 Jack Gido 认为，一般项目的生命周期大致可以划分为 4 个阶段，分别是识别需求、提出解决方案、执行项目和结束项目。具体的项目应用，可以根据实际情况对这 4 个阶段进行合理的扩充，比如软件开发项目可以再细化为需求分析、概要设计、详细设计、系统开发、系统测试、运行维护等不同阶段。针对 Jack Gido 给出的项目生命周期 4 个阶段，阶段的资源投入和时间投入数量关系可以由图 5-3 清晰地表示出来。

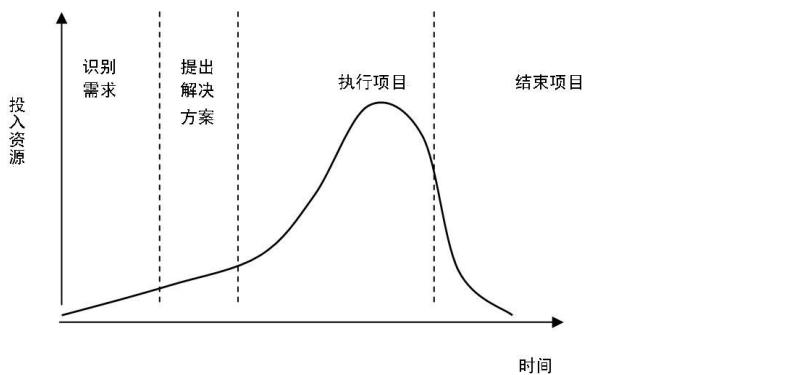


图5-3 项目生命周期及其资源投入模式

Teamwork 可以对整个项目周期做出详细的计划，并且可以将项目划分为不同的实施阶段进行跟踪，目的就是帮助整个团队充分利用现有资源，尽量在有限的时间及成本预算之内保证项目的顺利推进。Teamwork 的组成模块主要包含 4 个部分，即项目（Projects）、人员（Resources）、文档（Documents）及问题（Issues）模块。Teamwork 将各部分的管理进行有机结合，4 个部分合理地渗透在项目周期各个阶段。同时，Teamwork 将一些新的概念引入到团队合作中：区域（Areas）相当于对象的容器，Teamwork 中的所有资源、项目等都被各自的特定区域统一管理；角色（Roles）定义了一组对象管理的读写许可，每个成员在项目里的角色决定了成员在项目中对不同对象的控制权限；小组（Groups）则定义隶属于不同分组成员的级别及安全控制权限。

由于 Teamwork 提供了类似于富客户端的项目管理界面，Ajax 技术被广泛地应用在 Teamwork 的各个模块当中。在需要引导使用者进行下一步操作，以及用户期望立即得到服务器反馈的场合，Teamwork 都很好地应用了 Ajax，使其具备非常友善并且很有现代感的管理界面。登录 Teamwork 之后，出现在用户面前的项目总体梗概页面如图 5-4 所示。

The screenshot shows the Teamwork interface for a project named 'Grid GIS'. The top navigation bar includes links for 'file', 'edit', 'projects', 'resources', 'documents', 'team', 'tools', 'bookmarks', 'help & info', and a search bar. Below the navigation is a breadcrumb trail: 'projects / Grid GIS'. The main content area is divided into several sections:

- general:** Shows the project name 'Grid GIS' and a status dropdown set to 'active'.
- status:** Displays start date (05/07/2006), end date (25/08/2006), and checkboxes for 'is milestone' and 'auto-synch periods from children'.
- notes / deliverables:** Contains notes like 'grid gis project' and 'include ogsa-dai and ogsa-dqp'.
- tree:** A small tree diagram showing project structure.
- relevance:** Set at 50%.
- progress:** Set at 0.0%.
- by worklog:** An empty checkbox.
- estimated worklog required (hours):** Set to 144.
- milestones:** A section stating 'No milestones defined.'
- bottom footer:** Includes creation and modification logs ('created by System Manager on 02/07/2006-15:19:11 area - last modified by System Manager on 02/07/2006-17:13:00'), and buttons for 'delete', 'back', 'reset', and 'save'.
- calendar:** A small calendar at the bottom showing dates from 14 Jun to 06 Sep.

图5-4 项目总体梗概页面

人员是项目团队的基本要素，在 Teamwork 中，人员管理对应 Resources 资源模块。项目团队里的每个成员都将被指派负责一项或几项工作，成员需要严格按照项目规定的进度展开工作。

Teamwork 管理员首先为团队成员建立账户，指定项目经理，并且为成员设置权限级别及添加项目中出现的各种角色。项目经理具有安排项目进度和负责每个团队成员任务分配的权限，项目经理在 Teamwork 中为每个小组成员分配任务，使其明确各自的进度和目标。在新的任务分配之后，小组成员在登录 Teamwork 后会出现提示便签，并且会在任务分配栏中出现详细的项目计划和进度安排等信息。当任务被项目管理者指派到各个成员后，团队成员登录时就会收到关于项目任务指派的提示信息，任务信息的提示方式如图 5-5 所示。

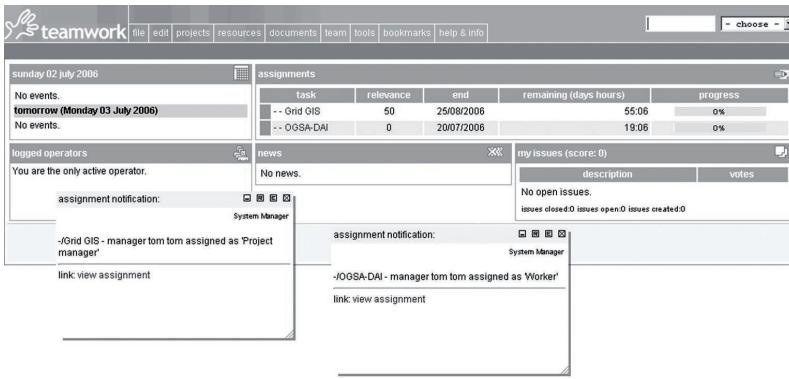


图5-5 项目任务指派提示信息

在确定了项目团队成员之后，接下来就是项目的定义和启动。在 Teamwork 的项目概括中可以定义项目名称、类型、起止日期、描述和进度安排等内容。Teamwork 按照树型结构定义项目和子项目，每个项目允许包含若干个子项及多层次子项节点嵌套。

在定义子项目的操作中，Teamwork 允许以树状结构定义多个层次的子项，不同子项目之间可以进行文档共享和交互访问。稍大一些的项目往往分成不同的实现阶段和里程碑，在 Teamwork 的分阶段协调之下协同完成。包含两个子项目的 Teamwork 项目管理界面如图 5-6 所示。



图5-6 Teamwork项目管理界面

详细的项目进度计划可以使团队成员明确责任与分工，保证任务按部就班地开展。团队的领导者需要根据实际情况对工作量、工作难度作出较为准确的估计，然后在 Teamwork 中为每个成员定义较为详尽的进度计划，设定项目里程碑和预计完成时间，在 Teamwork 里内置了便捷的工作量换算功能，方便统计成员每个时间段的工作量。

详细的工作计划帮助团队成员明确自己的任务和工作量，恰当安排好工作时间。当任务分配之后，所有的队员就应该按照计划表的要求开展工作。在工作过程中对任务执行情况的掌握和了解是相当重要的，清晰的日程不但可以帮助团队成员掌握每天的工作任务，同时还可以帮助团队避免重复性

的工作，通过 Teamwork 的日程进度表，可以详细地监视到整个团队的开发推进速度和每天完成的工作量。Teamwork 的项目进度日程表如图 5-7 所示。



图5-7 Teamwork项目进度日程表

Teamwork 提供了高效的文档管理模块来帮助团队进行资源的共享和流通。Teamwork 使用统一的存储库（Repository）对文档进行集中管理，并且任何笔记、文档、各种类型的文件甚至 URL 地址都可以作为项目文档，在 Teamwork 中提供统一的检索与管理。Teamwork 允许团队成员修改没有被锁定的文档，并且在每次修改之后保存修改记录和版本更新信息，在存储库中统一进行备份和管理。为了减轻整个团队的工作量，同时避免重复劳动，项目管理者应该于团队进度工作开展之前在 Teamwork 文档存储库中发布统一明确的标准文档格式，作为成员提交文档的模板。Teamwork 提供的文档提交与管理界面如图 5-8 所示。



图5-8 Teamwork项目文档存储库

在团队开发中，交流与沟通是不可缺少的环节，团队成员相互沟通工作情况，对于整体的工作进度和问题解决很有帮助。交流的内容可以为各自的工作进度、遇到的问题、是否需要得到团队支援或探讨新的工作方法等。Teamwork 提供了便捷的消息传递机制，无论项目管理者还是项目组成员，都可以随时通过便签条、新闻发布或邮件等方式与整个团队或特定成员展开讨论与联系。在 Teamwork 中，消息发布和收到便签通知的界面如图 5-9 所示。

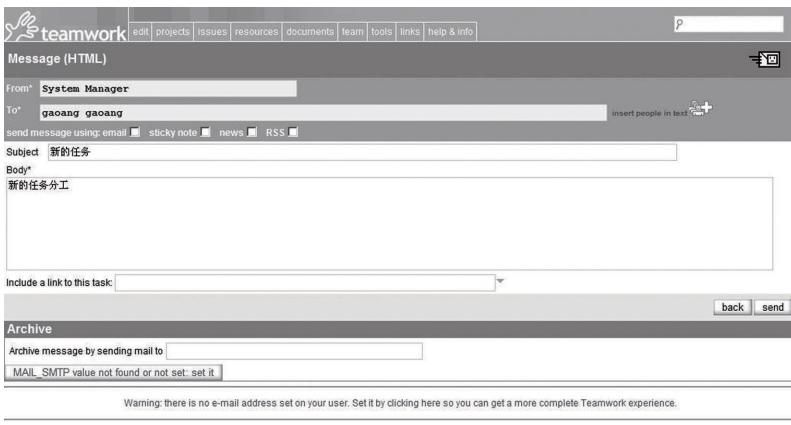


图5-9 通过Teamwork进行团队成员间的交流沟通

当团队成员发现工作不能按照计划完成时，可以通过 Teamwork 与团队领导者进行沟通与协调，以便及时在 Teamwork 中根据情况调整工作计划。同时项目领导者也要根据 Teamwork 动态更新的甘特图，来掌握整个项目和团队成员的工作进度，以及对进度安排和任务分工进行调整。有效的沟通有利于激励团队的合作氛围与积极性，同时还可以对工作进行及时的反馈和检查，调整计划中存在的不合理之处，避免在期限临近时才发现因为分工的问题而延误了项目进程。

项目进展的过程中，团队领导者需要始终关注项目和子项目中各项内容的完成情况。在团队成员接受到任务分配之后，Teamwork 会根据项目周期自动生成详尽的甘特图来监控项目进展情况。项目管理者可以通过甘特图掌握信息，随时进行恰当的任务调整，达到最佳的执行效果。团队成员也应该随时查看根据新任务动态更新的甘特图，以便了解项目整体和各个子项的进度。Teamwork 生成的动态项目甘特图如图 5-10 所示。

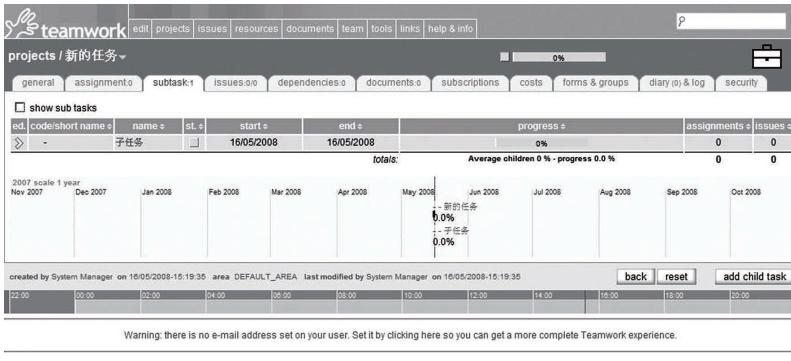


图5-10 Teamwork根据任务和周期动态生成的项目甘特图

作为团队管理的新一代产品，Teamwork 带给我们的不仅是多了一种新的管理工具可以选择，更重要的是把一套团队合作理念通过 Teamwork 平台引入到实际项目之中，帮助每个成员都树立起团队合作观念，明确每个阶段的目标与成就，调动整体的战斗力，更好地推动整个集体快速、稳步发展。

版本信息

Teamwork 一直由 Open Lab 公司负责开发和维护,目前的 Release 版本是 Teamwork 3.2,在新版本中,Open Lab 修正了前一个 Teamwork 版本中多处需要完善的地方,并且针对任务分级增添了新的功能。除此之外,新版本还完善了使用 Teamwork 进行团队合作管理中问题列表的在线打印支持功能和工作量计算功能。每一个 Teamwork 新版本发布之前,都会认真汲取使用者在反馈列表中提出的建议,并针对问题和建议对 Teamwork 的功能作进一步完善,可见 Teamwork 开发者在为用户打造一个成熟的项目管理系统上,是非常细致用心的。

社区视角

在收集 Teamwork 项目资料的过程中,笔者与 Open Lab 公司的创始人 Pietro Polzinelli 通过邮件取得了联系。在邮件联系中得知,Open Lab 公司目前有 15 人的规模,专注于提供基于 Java

的解决方案和对象关系映射方面的模型,除了软件产品开发与技术服务之外,Open Lab 还可以为特殊的用户提供定制的 Teamwork 应用,使 Teamwork 方便地应用于特定的工程项目中。作为一个成长中的软件公司,Open Lab 的软件产品曾在 2006 年的国际软件博览会里亮相,同时 Teamwork 还获得了第 17 届的年度 Jolt 奖项。项目组在 BlogSpot 上建立了 Teamwork 的项目博客,发表关于 Teamwork 开发最新进度和功能变化,感兴趣的读者可以随时通过 Teamwork 项目博客进一步了解 Teamwork 开发的最新进展。

作者介绍

高昂,博士研究生,关注开源软件与动态语言的发展。目前在资源与环境信息系统国家重点实验室从事网格 GIS、空间数据库研究工作,同时也是 OSGeo 中国和 InfoQ 中文站成员。业余时间里,喜欢徒步旅行并担当过数次北京近郊登山活动的领队,常写旅行游记。个人站点为开源网格 GIS 试验田 (<http://www.gaoang.com/>)。

6

面向方面编程篇 (AOP)

综述

面向方面编程 (AOP) 作为看待问题的另一种角度，是 OOP 的一种补充。它打破对象层次的界限，将一些共通的关切点集中到一起，让程序的结构更加清晰。在 OOP 中我们会运用“单一职责”原则去剥离不同目的的代码，而 Aspect 让你即使在关注面跨越了语言中的自然抽象单元（在 OOP 中是类）的情况下，也能够比较容易地完成剥离的任务。在某种程度上，AOP 可以看作是对一些设计模式的修正，比如 Observer 模式、Decorator 模式、职责链模式，AOP 在达到相同目的的同时，去除了由于使用设计模式而引入的复杂性。

AOP 的作用还体现在能有效地减少重复代码，也因此降低了测试的负担。出于以上理由，AOP 也被用作一种重构的手段。另外，由于 AOP 在重组系统流程上的灵活性，它也被用来改造遗留系统。

AOP 为系统在面对某种特定情况时应采取何种行为提供指示（即 Advice），当某个触发事件（即 Joint Point）时，比如当某个方法调用发生的时候，Advice 就会被自动执行。而 Pointcut 表达式被用来匹配 Join Point 和 Advice。以事务管理为例，你不需要在代码里到处调用事务管理器，而只需要编写一条 Pointcut 表达式去定义所有需要事务管理器介入的地方，并为之提供适当的 Advice。总之，AOP 的用途非常广泛，并不局限于日志、跟踪这几个小领域，而且它的作用也不是其他技术能够取代的。

目前几个主流的 AOP 框架具有不同的实现形式：AspectJ 是对 Java 语言的扩展，依靠特殊的编译器来实行编译时织入，或者通过对类加载过程的介入实现加载时织入；另一类是所谓的“动态”AOP，它们通过拦截器去拦截目标代码，并使用自动代理去改变目标代码的行为，Spring AOP 和 JBoss AOP 都属于这一类。显而易见，AspectJ 这样的静态实现在构建时较为麻烦，但却没有运行时的负担，代码执行效率是最高的。而依赖拦截器的 JBoss AOP 和 Spring AOP 在运行时需要做更多的工作，但像 Spring AOP 在构建时完全不需要什么特殊工具或步骤。虽然三者的起点不同，但实际上现在已经比较趋同，比如 JBoss AOP 也可以通过特殊编译器去实现所谓的 Aspectized Application，而 Spring

AOP 和 AspectJ 就更加是携手合作，互相取长补短。

不管是哪种实现方式，实质上都需要使用者学习新的语言。虽然有不同的编写方式，比如代码方式、XML 配置文件方式或 Annotation 的方式，它们本质上都是一种 DSL。因此无论如何选择，都有语言学习的负担。

除了这些，我们还要考虑它们在语义和表现力上的区别。AspectJ 作为一种语言扩展，其表现力最强，但它为了表达完善的语义，也因此变得复杂，要想全部掌握不是一件容易的事情。AspectJ 能做的事情，JBoss AOP 大多都能做得到，但由于 JBoss AOP 的实现和用法上比较接近于 API 的风格，不是很像一种语言，因此表达上会不那么清晰，但同时也让熟悉 Java 的人学习起 JBoss AOP 时容易不少。Spring AOP 以简洁和动态为目标，因此放弃了一些不好实现的语义，但它通过与 AspectJ 的紧密集成，弥补了功能上的不足。

关联信息

工具支持也是衡量一个框架生产力的重要因素。AspectJ、Spring AOP、JBoss AOP 都各自提供了 Eclipse 插件，分别是 AJDT、Spring IDE 和 JBoss IDE，这几个也都是它们各自最完善的开发环境。这 3 个工具的开发也很活跃，能够紧紧跟上各自框架的进展。

至于 NetBeans 平台，可以找到 AspectJ for NetBeans 插件 (<http://aspectj-netbeans.sourceforge.net/>)，但目前开发已经停滞，而且只支持到 AspectJ 1.1.1。虽然 Spring Netbeans Module (<http://spring-netbeans.sourceforge.net/>) 可以支持到最新的 Spring 2.5，但却没有 AOP 支持。

IntelliJ IDEA 对 Spring AOP 有不错的支持，功能包括 Pointcut 表达式和 @AspectJ 的自动完成、语法高亮、代码检查、快速修复等。AspectJ 也可以找到相应的插件 (<http://intellij.expertsystems.se/aspectj.html>)。

6.1

JBoss AOP

总评

上手容易程度	★★★★★
社区活跃性	★★★★
应用广泛性	★★★★★
推荐指数	★★★★

JBoss AOP 属于 JBoss 框架的一员，它是一个完全动态的 AOP 框架。JBoss AOP 的着眼点与其他 AOP 实现颇有不同，它的目标是一个“面向方面的框架，用于开发可重用的服务，并在运行时将其应用到预编译的代码上”。它为 JEE 开发提供了一套预定义的 Aspect “服务”，包括缓存、异步通信、事务、安全性、Remoting 等。

功能和特点

- 偏向于 API 风格，甚至近似事件监听的模型，适合熟悉 Java 开发的人。
- 随 JBoss AS 发布，部署方便。
- 多种书写方式，除了 XML 和 Annotation，还可以用编程的方式。
- 动态的实现模型，降低开发过程中的负担。

- 可独立使用，不限于 JEE 应用。
- 纯 Java 的实现，不需要额外的工具。

背景介绍

JBoss 最初是 Marc Fleury 建立的一个中间件研究项目，他随后成立了 JBoss 公司实行商业化发展。由于 JBoss 应用服务器的设计非常优秀，而且以开源的形式实行社区化发展，从而吸引了众多用户。JBoss 因此被认为是能够与 BEA 和 IBM 在中间件市场上抗衡的力量，也因此吸引了 Oracle 去试图收购 JBoss，但最终由 Red Hat 收购成功。收购后不久，Marc Fleury 就辞去 JBoss 部门总经理的职位，离开了 Red Hat。

JBoss AOP 在众多 JBoss 组件中一直都扮演着一个基石的角色。自从在 JBoss 4 中加入 JBoss AOP 框架之后，很多组件都利用它来简化实现或增强功能。比如 JBoss Cache 就用它来实现 POJO 的管理及事务等。JBoss 的 EJB3 实现也严重依赖于 JBoss AOP。新的 2.0 版大大加强了 JBoss AOP 与 Microcontainer 的集成。

参考资料

网站类

- JBoss 网站：<http://jboss.com/>, <http://labs.jboss.com/>

JBoss 所有产品的下载、文档、工具、社区等都在此处。由于 JBoss 开发是社区驱动的，很多最新的资源可以在 Labs 网站上找到。JBoss AOP 的文档包括用户指南、参考文档和快速参考，都

已经包括在JBoss AOP的下载包里。

书籍类

1. 《Foundations of AOP for J2EE Development》

作者: Pawlak.Renaud 出版社: Apress 出版时间: 2005年 ISBN: 9781590595077

这本书的作者也是JAC AOP框架的作者。他在书中比较了多种AOP实现的特点，包括AspectJ、Spring AOP、JBoss AOP和JAC。虽然是2005年出版的，但JBoss AOP的基本理念和实现手法等一直没有太大变化，因此这本书仍然值得一读。

快速上手教程

如果把JBoss AOP当作类库来单独使用，那么并不需要什么安装过程。如果是在应用服务器上部署，除非你是要升级旧版JBoss AS上的JBoss AOP，否则只需简单的复制操作就可以了。

下面我们简要说明JBoss AOP的用法。且看下面的代码（这里所用的示例代码取自JBoss AOP的文档）：

```
public class BankAccountDAO {  
    public void withdraw(double amount) {  
        long startTime = System.currentTimeMillis();  
        try {  
            // 实际的方法体...  
        } finally {  
            long endTime = System.currentTimeMillis() - startTime;  
            System.out.println("withdraw took: " + endTime);  
        }  
    }  
}
```

这里也同样存在代码分散、不容易修改等毛病，我们用JBoss AOP来改造它。首先我们把测量时间的代码分离出来，放进一个拦截器（JBoss AOP中的拦截器可看作是一种特殊的Aspect）：

```
public class Metrics implements org.jboss.aop.advice.Interceptor {  
    public Object invoke(Invocation invocation) throws Throwable {  
        long startTime = System.currentTimeMillis();  
        try {  
            return invocation.invokeNext();  
        } finally {  
            long endTime = System.currentTimeMillis() - startTime;  
            java.lang.reflect.Method m = ((MethodInvocation)invocation).method;  
            System.out.println("method " + m.toString() + " time: " + endTime  
                + "ms");  
        }  
    }  
}
```

原先的对象也就只剩下纯粹的业务代码，变成：

```
public class BankAccountDAO {  
    public void withdraw(double amount) {  
        // 实际的方法体...  
    }  
}
```

我们还需要为 JBoss AOP 提供指示，让它将两者联系起来：

```
<bind pointcut="public void com.mc.BankAccountDAO->withdraw(double amount)
">
    <interceptor class="com.mc.Metrics"/>
</bind >
```

在 JBoss AOP 的支配下，每当发生对 withdraw() 的调用，AOP 框架就会通过反射，将这个方法包装进一个 Invocation 对象（等于是委托），并暂停它的执行，插入 Aspect 的执行代码。当 Aspect 的这部分工作完成后，JBoss AOP 框架就会通过下面这句代码调用原来被拦截的方法：

```
invocation.invokeNext();
```

JBoss AOP 也可以使用 Annotation 来绑定 Aspect。被作为 Aspect 使用的类不需要继承什么父类，也不需要实现什么接口，但它必须有一个空的构造器和至少一个签名为如下形式的方法：

```
public Object <any-method-name>(org.jboss.aop.joinpoint.Invocation)
```

定义一个 Aspect 就类似于下面的样子：

```
package com.mypackage;
import org.jboss.aop.PointcutDef;
import org.jboss.aop.pointcut.Pointcut;
@Aspect (scope = Scope.PER_VM)
public class MyAspect {

    @PointcutDef ("(execution(* org.blah.Foo->someMethod()) OR \
                  execution(* org.blah.Foo->otherMethod()))")
    public static Pointcut fooMethods;

    public Object myAdvice(Invocation invocation) {
        return invocation.invokeNext();
    }
}
```

我们前面所写的拦截器也可以用 @InterceptDef 改写成下面的样子：

```
@InterceptDef (scope = Scope.PER_VM)
@Bind (pointcut="execution(* com.mc.BankAccountDAO-> withdraw(..))")
public class Metrics implements Interceptor {
    public Object invoke(Invocation invocation) throws Throwable {
        ...
    }
}
```

各种 Annotation 都有相应的等价 XML 配置风格的写法。比如前面的 MyAspect 中的 Annotation 就和下面的 XML 配置等价：

```
<aop>
    <aspect class="com.mypackage.MyAspect" scope="PER_VM"/>
    <pointcut
        name="com.mypackage.MyAspect.fooMethods"
        expr="(execution(* org.blah.Foo->someMethod()) OR \
               execution(* org.blah.Foo->otherMethod()))"
    />
</aop>
```

JBoss AOP 提供了很丰富的 Pointcut 表达式语法，该语法和 AspectJ 的语法很相似。JBoss AOP 曾经只提供一种 Advice 类型，即 Around 类型。虽然仅支持 Around 并不会在功能上有所欠缺（其他

类型都可以看作是 Around 的变形), 但为了便利起见, JBoss AOP 现在已经提供了 5 种 Advice 类型: Around、Before、After、After-Throwing 和 Finally。

JBoss AOP 的用法可以非常灵活, 比如用户手册中就给出了用它来对异常处理进行测试、注入 Mock 对象等用法。JBoss AOP 对对象属性的拦截能力也可以被用来实现依赖注入。假设有这么一个 Aspect, 里面封装了获取事务管理器的代码:

```
import javax.transaction.TransactionManager;
import org.jboss.tm.TxManager;

public InjectTMAAspect {
    private TransactionManager tm = TxManager.getInstance();

    public Object access(FieldReadInvocation invocation) throws Throwable {
        return tm;
    }
    ...
}
```

然后是配置。当遇到某个属性拥有 @Inject 标记的时候, 就触发上面的 Aspect, 对该属性进行注入:

```
<aop>
    <aspect class="org.jboss.aspects.InjectTMAAspect"/>
    <bind pointcut="field(javax.transaction.TransactionManager
        *->@org.jboss.aspects.Inject)">
        <advice name="access" aspect="org.jboss.aspects.InjectTMAAspect"/>
    </bind>
</aop>
```

使用起来非常简单:

```
public class MyTransactionalCache {
    @Inject private TransactionManager tm;
    ...
}
```

JBoss AOP 也同样具备 Introduction、Mixin、CFlow 等功能, 甚至还有它独门的在 JBoss AS 上进行热部署的能力, 即在运行时改变 Advice 或者拦截器的绑定。

版本信息

目前 JBoss AOP 有两个主要版本分支, 1.5.x 是稳定版, 最新的是 1.5.6 版, 2.0.x 是开发中的版本, 但刚刚进行了社区发布 (2.0.0.CR1), 已经接近完成。2.0.x 版的主要改进包括增加了 Before、After、Throwing 等 3 种 Advice, 调整了实例化模型, 类加载器的可见范围, 还有与 Microcontainer 的整合, 比如自动管理 Aspect 的依赖项等。JBoss AOP 采用 LGPL 许可。

由于 Red Hat 正在进行开发工具产品线的整合, 所以有些混乱。他们的新 IDE 产品叫做 JBoss Developer Studio, 是由收购不久的 Exadel 公司的商业产品 Exadel Studio Pro 修改而来。JBoss Developer Studio 是开源产品, 采用 GPLv2 许可, 但并不免费。另有一个免费的社区版本 JBoss Tools 作为新功能的孵化器, 但这个产品不是完整的 IDE, 只是一组 Eclipse 插件的集合。

令人非常头晕的问题是, 新的 JBoss Developer Studio 还没有整合进 AOP 功能, 开发

者只能继续使用已经停止开发的 JBoss IDE，最后的 JBoss IDE 版本是 2.0.0 Beta2，可在 <http://sourceforge.net/projects/jboss/> 下载。

社区视角

JBoss AOP 的首要任务是为 JBoss AS 服务，它随着 JBoss 应用服务器一起发布，但也可以单独下载。虽然可以在其他应用服务器中部署，但常常会遇到一些小麻烦。当然，JBoss AS 也是一款相当优秀而且流行的应用服务器，如果采用 JBoss AS，就没有必要引入其他 AOP 框架了，直接使用 JBoss AOP 不但免除了部署的麻烦，而且完全可以满足你对功能的要求。还应该指出，在非 JEE 应用中单独使用 JBoss AOP 是完全没问题的。

由于其实现方式的原因，JBoss AOP/JBoss

IDE 在静态的检查上有所欠缺，比如万一写错了 Advice 的名字或签名，就没有办法在编译时立即发现。

JBoss AOP 有不少社区提供了现成的 Aspect，它涵盖了各种设计模式实现、缓存、事务管理、集群等方面，这是 JBoss AOP 独有的优势。不过在文档质量和社区活跃程度上可能比不过 AspectJ 和 Spring AOP，在选择一种不熟悉的技 术时，社区支持也是非常重要的考虑因素。

作者介绍

郭晓刚：InfoQ 中文站架构社区的首席编辑。多年从事企业开发工作，目前的主要方向是 RIA 和 UX 的研究。在技术翻译方面亦颇有心得。欢迎来信讨论：

hiugong.gwok@cn.infoq.com

7

面向服务架构篇 (SOA)

综述

第一次听说 Web 服务是在 2002 年刚刚接触 .NET 的时候，当时 .NET 所标榜的优越性之一就是宣称其是专为 Web 服务而设计的。的确，在试用过 VS.net 的 Web 服务工具之后，感觉真的很方便，几乎和创建一个普通的类没什么区别。不久又接触到了 SOA 的概念，因为有 Web 服务先入为主的观念，以为其中的 S 就是专指 Web 服务。只是阅读了更多 SOA 的资料之后，才知道 Web 服务只是 SOA 的实现技术之一。

现在，SOA 已不再是什么新鲜的概念。在各大厂商、媒体的狂轰乱炸之下，几乎连学校中计算机专业的学生也知道世界上还有 SOA 这种东西。但是，与这些宣传攻势相比，SOA 的落地就没有那么光鲜了，给人一种光打雷不下雨的感觉。这种情形和最初的 ERP 的情形几乎如出一辙。于是乎，SOA 在一些开发者的心目中几乎已经成为一个典型的负面教材，甚至对其有一种莫名的厌恶。就其本身而言，SOA 并没有什么问题。但是，是什么使之沦落到如此地步呢？这其中又有几方面的原因：

1. 首先，SOA 的目标非常宏大，但是与之配套的标准规范制定的速度并没有跟上宣传的节奏。这其中有不少政治因素，但是复杂性也是原因之一；
2. 其次，SOA 的实施方法论和一般的软件开发实施的方法论并不完全一样，而且尚处于不断的发展完善之中。很少有企业会一开始就非常先知先觉地将其信息系统建立在 SOA 的风格之上，大都是在 IT 成为阻碍其业务发展时才求助于 SOA。此时，企业的 IT 环境已经错综复杂，在这种环境下去实施 SOA，必须有一套方法论来指导；
3. 第三，人的因素尤其突出。只有在整个企业范围内建立 SOA，这样才能显现出 SOA 的价值，实现部门一级的 SOA 的意义并不是太大。虽然，SOA 的普及可以从局部着手，但是从大局出发却是至关重要。这就要求参与 SOA 项目的客户方代表必须有足够的影响力，否则最后结果无疑只能是空中楼阁；
4. 最后，缺乏高生产力的工具。不管 SOA 被吹嘘得如何天花乱坠，但是最后的活还得人来完成。这时工具的作用就体现出来了。如果工具还需要开发者过于陷入到具体的技术细节之中，那么

4. 在工期一定的前提下，只能以减小解决业务问题上的时间为代价。

作为开源工具的选型手册，本章只能关注于最后一点。随着技术的发展，SOA 工具也在不断地进化。各厂商的工具如此，开源的工具也是如此。此外，由于开源运动的广泛深入，现在有一种趋势，就是厂商直接参与到开源项目中，并将自家的产品建构在开源项目之上。如 IBM、IONA、RedHat 等。

本章所选的工具都带有 Java 背景，这主要是因为 Java 是目前企业开发的主流，相比起 .NET 来说，它在企业环境的应用更广。如主要的 ERP、CRM、EAM、HR 等大型系统几乎无一例外均采用 Java。这其中既有 Java 更开放的原因，也有 Java 的主要厂商（如 IBM、Oracle、SAP 等）早在微软专注于个人软件领域的时候，就已经有很深的企业软件开发的背景原因。除了以上原因之外，选择 Java 背景的工具，也是因为 Java 的开源项目更多，而且笔者的大部分项目主要以 Java 为工具，比起 .NET 来说更熟悉。

本章所涉及的工具主要来自 Apache，这其中除了这些工具本身就很优秀之外，也是出于 Apache 的许可证对于商业更友好的角度考虑的。本章涉及的工具是 Apache CXF，服务基础设施；Apache ODE，BPEL 引擎；Apache Tuscany，SCA/SDO 的第一个参考实现。

关联信息

SOA 的范畴非常广泛，开源工具也绝非只有此处所列的这 3 个。在 Apache、ObjectWeb、Codehaus、Eclipse 等主要开源网站上有很多的 SOA 相关工具可供选择，而且不少工具的功能是相似的。如与 CXF 类似的工具就有 Axis2，与 ODE 类似的就有 ActiveBPEL，与 Tuscany 类似就有 Fabric3。

工具的选择依赖于对工具背后的技术或方法的理解，对于 SOA 工具而言，自然是 SOA 本身。但是，由于 SOA 本身就错综复杂，难免会让初学者不知所措，不知如何下手。在此，笔者期望以下的学习曲线会有助于对 SOA 的学习和研究：

1. 了解 SOA 的作用和目的，对自己学习和研究 SOA 有个交待，不至于连自己都无法说服；
2. 了解 SOA 的参考架构，对其有个整体把握。这方面的资料在各大厂商的技术网站上都有，基本上大同小异；
3. 寻找 SOA 的着陆点。例如，IBM 提出人员、流程和信息 3 个方面。如果套用三层架构来理解的话（虽然不是太准确，但是可以类比，方便理解），人员对应于表现层，流程对应于中间层，信息则对应于数据层；
4. 掌握 SOA 的方法论。这方面可以参考一些厂商或组织的实施方法论，这些资料一般也会出现在他们的网站上。此外，案例学习也是掌握方法论的一个重要素材。对于案例，经验和教训应该并重，不要只听从厂商的一面之辞，从客户方面去了解也是非常重要的；
5. SOA 相关的标准和规范，这是选择工具最直接的因素。

7.1

Apache CXF

总评

上手容易程度	★★★★★
社区活跃性	★★★★★
应用广泛性	★★★★★
推荐指数	★★★★★

在其官方网站上，Apache CXF 被定位为一个开源的服务框架。它是两个著名的开源项目——ObjectWeb Celtix 和 Codehaus XFire——于 2006 年合并后的产物，前者是开源的 ESB 产品，后者则是著名的 SOAP 堆栈。强强联手之后的 CXF 自然是集两者的长处于一身，给开源 SOA 工具领域带来不小的冲击。除了支持众多的标准、数据绑定、传输协议，CXF 还是开源社区继 JAX-WS RI 之后第一个通过 JAX-WS TCK 的 SOAP 堆栈。

选择 CXF 的理由很多，但是主要集中在：

1. 简单易用，支持众多编程模型；
2. 与Spring集成；
3. 提供了对RESTful服务的支持；
4. 可插拔的框架设计对其他数据绑定和遗留系统提供了支持；
5. 商业友好的Apache许可证。

目前，CXF 尚处于 Apache 的孵化器中。但由于其活跃度很高，且用户基础非常好（主要来自 XFire 和 Spring），后势不可小视。加之对 JAX-WS 的支持，关注 CXF 的进展是非常有意义的。

功能和特点

CXF 的当前版本是 2.0.3，该版本的功能特性如下。

1. 支持众多的标准，包括：
 - ◆ JAX-WS、JAX-WSA、JavaWeb服务元数据规范 (JSR-181) 和SAAJ；
 - ◆ SOAP 1.1、1.2、WS-I BasicProfile、WS-Security、WS-Addressing、WS-RM和WS-Policy；
 - ◆ WSDL 1.1；
 - ◆ MTOM。
2. 支持多种传输协议、绑定、数据绑定和格式
 - ◆ 绑定：SOAP、REST/HTTP；
 - ◆ 数据绑定：JAXB 2.0、Aegis；
 - ◆ 格式：XML、JSON；
 - ◆ 传输协议：HTTP、Servlet、JMS和Local（即JVM内部消息通信机制）；
 - ◆ 可扩展的API允许开发者方便地对绑定和消息格式进行扩展。
3. 灵活的部署
 - ◆ 轻量级容器：Tomcat或基于Spring的容器；
 - ◆ JEE集成：可部署于JEE应用服务器中，如：Geronimo、JOnAS、JBoss、WebLogic和WebSphere；
 - ◆ 单独运行的客户机/服务器。
4. 支持多种编程语言
 - ◆ 完全支持JAX-WS 2.0客户机/服务器编程模型；

- ◆ JAX-WS 2.0同步、异步和单程API;
 - ◆ JAX-WS 2.0动态调用接口（DII）API;
 - ◆ 支持包装和非包装风格;
 - ◆ XML消息传递API;
 - ◆ 客户端和服务器编程都支持使用JavaScript和ECMAScript 4 XML（E4X）;
 - ◆ 通过Yoko提供对CORBA的支持。
5. 代码产生
- ◆ Java到WSDL;
 - ◆ WSDL 到Java;
 - ◆ XSD到WSDL;
 - ◆ WSDL到XML;
 - ◆ WSDL到SOAP;
 - ◆ WSDL到服务。

背景介绍

作为两个项目结合体，CXF 最初的名字叫CeltixXFire。明眼人一看就知道这表示该项目是Celtix 和 XFire 的后继者。但是这个名字实在没有什么创意，而且社区也有人表示该名字难听，在被Apache 接受后不久就被更名为 CXF。虽然从名字上已经很难看出它的源头，但是了解一下它的历史还是有助于我们对 CXF 的更深入的理解，更何况 Celtix 和 XFire 本身的来头都不小。尤其是后者，在近年的 Web 服务开发工具内更是声名鹊起，几乎成为 Java 开发 Web 服务的首选。

在开源 ESB 领域，Celtix 与 Mule、ServiceMix 等著名 ESB 项目相比，知名度要差了不少。但这也并非意味着 Celtix 就是无名之辈。Celtix 背靠大名鼎鼎的 IONA 公司，在该公司捐献给 ObjectWeb 的代码基础上发展而来。虽然自称为 ESB，但是 Celtix 更专注于对 JAX-WS 和 WS-* 提供支

持。这难免会给人一种名不副实的感觉，因此当Celtix 和 XFire 合并之后，新项目的名字并没有特意强调 ESB。或许也正是这个原因，使得IONA于 2007 年收购了 LogicBlaze。由于 LogicBlaze 为许多著名的开源项目——包括 ActiveMQ 和 ServiceMix——作出贡献，不难想象，这些技术有可能会被融入到 CXF 中。

“前途是光明的，道路是曲折的”，这句话形容使用 SOA 来进行开发是再贴切不过的了。由于工具的支持还很不到位，即使创建一个简单的服务，开发者也需了解大量的协议细节，进行大量的手工配置。而且，最重要的是不易测试。在 Spring 大行其道之后，开发者一直在期盼相似的开发工具，以给他们黑暗的 SOA 开发生涯带来光明。此时，XFire 出现了。支持 POJO 开发服务、与 Spring 集成很快就使 XFire 抓住了开发人员的心。此外，灵活的绑定机制，出色的性能更是令使用者难以割舍。

至于两个项目合并的原因，XFire 创始人 Dan Diephouse 在其博客中进行了交代。在他思考 XFire 2.0 的时候，遇到了 IONA 的 Debbie Moynihan。在围绕 Celtix 和 XFire 进行了一番讨论之后，他们发现两个项目存在很多共同的目标，如实现 WS-* 和 JAX-WS。接下来，故事就很平常了。自然是英雄惜英雄，都觉得是到了一起工作的时候了。2006 年 6 月 20 日，两个社区给 Apache 软件基金提交了一份提议，建议将两个社区合并。于是，CXF 项目诞生了。详细的内容，可以访问 <http://netzoid.com/blog/2006/08/29/celtixfire-20-xfire-20-celtix-20/>。虽然目前 CXF 尚在 Apache 的孵化器中孵化，但这并不代表代码质量低劣。在孵化器中孵化的目的主要是出于法律和社区等问题的考虑，关于这些内容请参见 CXF 官方网站的 FAQ。

参考资料

网站类

1. CXF官方网站: <http://incubator.apache.org/cxf/>;
2. CXF的Groovy插件GroovyWS网站: <http://docs.codehaus.org/display/GROOVY/GroovyWS>;
3. XFire官方网站: <http://xfire.codehaus.org/>;
4. Celtix官方网站: <http://celtix.objectweb.org/>;
5. JAX-WS规范: <http://jcp.org/en/jsr/detail?id=224>;
6. IBM DeveloperWorks中文站 SOA与Web服务专区: <http://www-128.ibm.com/developerworks/cn/webservices/>;
7. BEA dev2dev中文站点: <http://dev2dev.bea.com.cn>。

书籍类

《SOA Using Java Web Services》

作者: Mark D. Hansen 出版社: Pearson Education, Inc. 出版时间: 2007年
ISBN: 9780130449689。

文章类

Web Services Using Apache CXF: http://java.sys-con.com/read/452355_p.htm#。

快速上手教程

基础架构

CXF旨在为服务创建必要的基础设施，它的整体架构主要由以下几个部分组成：

1. Bus

它是CXF架构的主干，为共享资源提供了一个可配置的场所，作用非常类似于Spring的ApplicationContext。这些共享资源包括WSDL管理器、绑定工厂等。通过对Bus进行扩展，可以方便地容纳自己的资源，或替换现有的资源。默认Bus实现是基于Spring的，通过依赖注入，将运行时组件串起来。Bus的创建由BusFactory负责，默认是SpringBusFactory，对于默认Bus实现。在构造过程中，SpringBusFactory会搜索META-INF/cxf（就包含在CXF的Jar中）下的所有Bean配置文件，根据它们构建一个ApplicationContext。开发者也可提供自己的配置文件来定制Bus。

2. 消息传递和拦截器 (Interceptor)

CXF建立于一个通用的消息层之上，主要由消息、拦截器和拦截器链 (InterceptorChain) 组成。CXF是以消息处理为中心的，熟悉JSP/Servlet的开发者可以将拦截器视为CXF架构中的“Filter”，拦截器链也与“FilterChain”类似。通过拦截器，开发者可以方便地在消息传递、处理的整个过程中对CXF进行扩展。拦截器的方法主要有两个：handleMessage和handleFault，分别对应消息处理和错误处理。在开发拦截器的时候需要注意两点：

- ◆ 拦截器不是线程安全的，不建议在拦截器中定义实例变量并使用它。这一点跟JSP/Servlet中对于Filter的处理是一样的；
- ◆ 不要调用下一个拦截器的handleMessage或handleFault，这个工作由InterceptorChain来完成。

3. 前端 (Front End)

它为CXF提供了创建服务的编程模型，当前主要的前端就是JAX-WS。

4. 服务模型

CXF中的服务通过服务模型来表示。它主要有两部分：ServiceInfo和服务本身。ServiceInfo作用类似WSDL，包含接口信息、绑定、端点（EndPoint）等信息；服务则包含了ServiceInfo、数据绑定、拦截器和服务属性等信息。可使用Java类和WSDL来创建服务。一般是由前端负责服务的创建，它通过ServiceFactory来完成。

5. 绑定 (Binding)

绑定提供了在传输之上映射具体格式和协议的方法，主要的两个类是Binding和BindingFactory。BindingFactory负责创建Binding。

6. 传输 (Transport)

为了向绑定和前端屏蔽传输细节，CXF提供了自己的传输抽象。其中主要有两个对象：Conduit和Destination。前者是消息发送的基础，后者则对应消息接收。开发者还可以给Conduit和Destination注册MessageObserver，以便在消息发送和接收时获得通知。

开发方法

CXF可以创建的Web服务应用有两种：服务提供者和服务消费者。这种结构可类比客户端 / 服务器结构，服务消费者类似于客户端，服务提供者类似于服务器。使用 CXF 创建应用时，服务提供者和服务消费者并不需要同时出现，因为有可能应用只是作为服务提供者或服务消费者单独出现。

为了说明使用 CXF 如何创建这两种类型的应用，本教程将同时给出它们的例子。另外，由于 Groovy 在 Java 世界中变得越来越流行，本教程会给出使用 Groovy 的 CXF 插件 GroovyWS 的实现例子。例子使用 JDK 1.5.X 和 Groovy 1.0 完成，包含以下几部分：

1. User，用户对象，在消费者和提供者之间传递；
2. UserService，用户管理服务，它提供增加和获取所有用户的功能；
3. Client，服务消费者，它向UserService发起服务请求。

Java 实现的步骤包括以下几点。

1. 服务端包含UserService、UserServiceImpl和User。其中，UserService是接口定义，UserServiceImpl是它的实现，并负责服务的发布，服务只有发布之后才能被消费。例子使用了JAX-WS，它们的主要内容如下：

UserService

```
package server;
import javax.jws.WebService;

@WebService
public interface UserService {
    void add(User user);
```

```
        User[] findAllUsers();
    }
```

@WebService 指明接口是 Web 服务

UserServiceImpl

```
import java.util.List;
import java.util.Vector;

import javax.jws.WebService;
import javax.xml.ws.Endpoint;

@WebService(endpointInterface = "server.UserService",
            serviceName = "UserService",
            portName="UserServicePort")
public class UserServiceImpl implements UserService {
    static List<User> UserRepository= new Vector<User>();
    public void add(User user) {
        UserRepository.add(user);
    }
    public User[] findAllUsers() {
        User[] users= new User[UserRepository.size()];
        UserRepository.toArray(users);
        return users;
    }
    public static void main(String[] args){
        UserServiceImpl userService= new UserServiceImpl();
        Endpoint.publish("http://localhost:9000/userService", userService);
    }
}
```

@ WebService中的serviceName、portName，分别指定了其产生的WSDL中的服务名和端口名。endpointInterface为接口的类名。服务发布代码也可以放在另一个类中。

User

```
package server;
public class User {
    String first;
    String last;
    public String getFirst() {
        return first;
    }
    public void setFirst(String first) {
        this.first = first;
    }
    public String getLast() {
        return last;
    }
    public void setLast(String last) {
        this.last = last;
    }
}
```

2. 客户端只有一个类: Client，其他的User、UserService引用server包中的对象。

```
package client;

import javax.xml.namespace.QName;
import javax.xml.ws.Service;
import javax.xml.ws.soap.SOAPBinding;

import server.User;
```

```

import server.UserService;

public class Client {
    public static void main(String[] arg){
        Service service = Service.create(
            new QName("http://server/", "UserService"));
        service.addPort(new QName("http://server/", "UserServicePort")
            , SOAPBinding.SOAP11HTTP_BINDING
            , "http://localhost:9000/userService");
        UserService userService= service.getPort(UserService.class);

        User user= new User();
        user.setFirst("James");
        user.setLast("Bond");
        userService.add(user);

        User[] users= userService.findAllUsers();
        for(User u : users){
            System.out.println(u.getFirst()+"."+u.getLast());
        }
    }
}

```

注意，QName的第一个参数指明了WSDL的目标名字空间，可以通过“服务地址 ?wsdl”方式获取服务的WSDL，来查看服务的目标名字空间。对于由CXF创建的服务，目标名字空间的默认构造规则是：http://包名的倒序/。即如果包名是a.b.c，那么名字空间就为http://c.b.a/。

GroovyWS 的实现步骤包括以下几点。

1. 服务端包括：User、UserService、User.aegis.xml。由于Groovy类包含一个metaClass属性，该属性不应该被序列化在服务端和客户端之间传递，User.aegis.xml用来指定将该属性忽略掉。

User

```

package server;

class User {
    String first
    String last
}

```

UserService

```

package server;
import groovyx.net.ws.WSServer

class UserService {
    private static List users= new Vector()

    void add(User user){
        users.add(user)
    }

    User[] findAllUsers(){
        User[] u= new User[users.size()]
        users.toArray(u)
        return u
    }

    static void main(args) {
        def server = new WSServer()
        server.setNode("server.UserService",
                      "http://localhost:9000/UserService")
    }
}

```

```
}
```

```
}
```

注意它的发布。

User.aegis.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<mappings xmlns:sample="http://DefaultNamespace">
    <mapping name="sample:User">
        <property name="metaClass" ignore="true"/>
    </mapping>
</mappings>
```

2. 客户端包含：Client和User.aegis.xml，User.aegis.xml的内容和服务端的一样。Client的内容如下：

```
package client;
import groovyx.net.ws.WSClient

class Client {

    static void main(args) {
        def proxy = new WSClient(
            "http://localhost:9000/UserService?wsdl", Client.class.classLoader)
        def user= proxy.create("defaultnamespace.User");
        user.first="James"
        user.last="Bond"

        proxy.add(user)
        def result= proxy.findAllUsers()
        result.users.each{
            println it.first+"."+it.last
        }
    }
}
```

相关建议

CXF的功能特性非常多，要熟练使用它非得花些功夫才行。笔者在此给出一些建议，期望能对读者在今后学习和使用CXF的过程中有所帮助：

1. 熟悉工具涉及领域的协议是个不错的主意。虽然CXF提供了简化服务创建的编程模型，但是如果不懂WS-*协议，在遇到问题调试时必然会花不少时间。尤其是在SOA的环境中，客户端和服务不一定是使用同一语言、同一工具实现的情况下，互操作问题经常是由于对协议的不同支持造成的；
2. 作为CXF实现内容的一个重点，JAX-WS是值得关注的；
3. 在Java的环境中，Spring几乎已经成为开发服务器端应用的首选，应重点关注CXF和Spring的配合使用；
4. 近些年来，Java世界的动态语言旋风愈演愈烈。Groovy由于其语法和Java兼容且提供了不少方便的语法，吸引了不少Java开发者。更何况新兴的Grails框架逐渐引人注目，其前途不可限量。GroovyWS专为Groovy开发，且底层就是CXF，作为CXF的开发者，没有理由不去使用可以使自己生活过得舒适的工具；
5. CXF携带了大量的例程，它们是熟悉和了解CXF的大门的；
6. 参与社区，参与讨论，往往比起自己单干要有用得多。

版本信息

在官方网站上，CXF 公布了其 2.0.4 版和 2.1 版的开发计划。2.0.4 版于 2008 年 1 月 15 日发布，2.1 版则是 2 月 28 日发布。2.0.4 版的计划主要是修正 2.0.3 版的错误，以及工具迁移的问题。2.1 版则引入一些新特性，包括：

1. 支持 JAX-WS 2.1，包括 JAXB 2.1、API 中的 WS-A、SEI 接口方法的 JAXB 标注、WebServiceFeature 标注；
2. XMLBeans、JiBX 数据绑定；
3. 新的 java2ws 工具；
4. 更好地支持 REST (JSR-311)；
5. 支持 js；
6. OSGi bundling。

除了上述计划，有可能包含在 2.1 版，但是肯定会在 2.2 版的特性包括：

1. 通过继承 Yoko 的代码来支持 CORBA；
2. 更好地集成 Acegi；
3. WS-SecureConversation/Trust；
4. 其他 WS-* 协议。

社区视角

在 Celtix 和 XFire 宣布合并的同年，另一个著名开源 Web 服务框架 Axis 的后继者 Axis2 也诞生了。Axis2 并非 Axis 的 2.0 版，而是完全重写了 Axis 的新项目。作为功能和背景都极其相似的两个项目，人们难免会将它们相提并论。在著名的 Java 企业开发站点 TheServerSide 上就有一篇名为“Axis, Axis2 and CXF: Survey the WS Landscape”（地址：<http://www.theserverside.com/tt/articles/content/AxisAxis2andCXF/article.html>）的文章对这两个项目进行了比较，主要内容如下。

1. 在特性方面：

- ◆ CXF 支持 WS-Addressing、WS-Policy、WS-RM、WS-Security 和 WS-I BasicProfile。Axis2 支持除了 WS-Policy 之外的所有这些标准，WS-Policy 预计会在未来版本中得到支持；
- ◆ CXF 可以方便地和 Spring 集成在一起，Axis2 不行；
- ◆ Axis2 支持范围更广的数据绑定，包括 XMLBeans、JiBX、JaxME、JaxBRI，以及它自己的数据绑定 ADB。在 Axis2 1.2 版中，JaxME 和 JaxBRI 尚处于试验阶段。目前，CXF 只支持 JAXB 和 Aegis，对 XMLBeans、JiBX 和 Castor 的支持将在 CXF 2.1 版中实现；
- ◆ Axis2 支持多语言，除了 Java 版本，尚有 C/C++ 版本。

2. 在开发方面：

- ◆ Axis2 更像一个微型服务器。Axis2 被打包成一个 WAR，可部署到任何 Servlet 容器中，为了更方便地在运行中管理和部署服务进行专门的设计。
 - ◆ CXF 更专注于对开发人员友好及可嵌入性。大部分配置只需使用 API 即可完成，与 Spring 紧密集成。CXF 强调代码优先的服务开发方式。
3. 建议：如果需要多语言支持，那么就采用 Axis2；如果考虑到使用 Java、与 Spring 集成，或将服务嵌入到其他程序中，那么 CXF 更好。

当然，并不是所有人都说好。例如，在国内的一些论坛上，就有开发者抱怨 CXF 的入门比起 XFire 来要复杂得多。这是可以理解的，毕竟 CXF 本身也比 XFire 要复杂得多。为了帮助 Celtix 和 XFire 的开发者向新工具的迁移，其官方网站也提供了相应的迁移指南。另外一个常见的问题是和 Spring AOP 相关的（如事务、安全），这在官方网站的 FAQ 中也有说明。

7.2

Apache ODE

总评

上手容易程度	★★★★★
社区活跃性	★★★★★
应用广泛性	★★★★★
推荐指数	★★★★★

Apache ODE (Orchestration Director Engine, 编制指导引擎)是基于 Java 的开源 WS-BPEL (简称 BPEL) 引擎, 它于 2007 年 7 月 18 日从 Apache 的孵化器中诞生成为一个顶级项目。它的主要功能就是执行使用 BPEL 描述的业务流程, 实现业务流程自动化, 它支持长期运行和短期运行的过程。与另一著名的开源 BPEL 引擎 ActiveBPEL 相比, 它具备以下特点:

1. ODE 的许可证是 Apache 2.0, 而 ActiveBPEL 则是 GPL。前者对于商业用途较后者要友好;
2. ODE 可以更方便地与 ESB (如 ServiceMIX、Mule) 进行集成。

考虑到商业上的限制, 以及 ESB 几乎成为了当今 SOA 基础设施的必选件, 在开源 BPEL 引擎领域, 我们建议关注 Apache ODE。

功能和特点

Apache ODE 的当前版本是 1.1, 当前版本的功能特性包括如下几点:

1. 同时支持 WS-BPEL 2.0 OASIS 标准和遗留的 BPEL4WS 1.1 厂商规范;

2. 支持两种通信层: 一种基于 Axis2 (Web Services http transport), 另一种则基于 JBI 规范 (使用 ServiceMix);
3. 提供了高层次的 API, 允许任何通信层与内核进行集成;
4. 支持过程的热部署;
5. 在命令行或部署提供了详细分析和验证 BPEL 的方法;
6. 为过程、实例和消息提供了管理接口。

背景介绍

在 SOA 的世界中, WS-BPEL (简称 BPEL) 被用来描述业务流程, 它是 Web Services Business Process Execution Language (Web 服务业务过程执行语言) 的缩写。BPEL 是基于 XML 的描述语言, 为了描述业务逻辑, 它提供了一些控制结构 (如条件、循环), 以及一些调用 Web 服务和接收服务消息的元素。它依赖 WSDL 来表示 Web 服务接口。可以操纵消息结构, 将部分或全部赋给那些要被发送到其他消息的变量中。BPEL 于 2003 年提交给 OASIS 标准组织, 当前的最新版是 WS-BPEL 2.0, 在此之前的一个重要版本是 BPEL4WS 1.1。

BPEL 在 SOA 中的作用类似工作流领域中的 XPDL。但是与后者不同的是, BPEL 主要关注自动任务的交互, 而没有涉及人工任务的交互, 最近发布的 WS-BPEL4People 规范正是对 BPEL 缺失内容的补充。BPEL 和 BPEL4WS 1.1 的详细

情况可从它们相关的文档中获得。

Apache ODE 是目前唯一商业友好且支持全部 BPEL 规范（1.0、1.1 和 2.0）的开源 BPEL 引擎。ODE 由 Intalio 公司于 2006 年 6 月捐献给 Apache 软件基金，当时该公司刚刚收购 FiveSight Technologies。Intalio 是一家领先的开源 BPMS 厂商，成立于 1999 年 6 月，公司位于加州的 Palo Alto。该公司的产品 Intalio Server 建构于 Apache ODE 之上，该产品也是目前市面上最快和最具伸缩性的过程引擎，支持在同一服务器部署几十万过程模型，以及在单 CPU 上并行运行上千万的过程实例。除了 Intalio Server，ODE 还在以下几个

项目中得到应用：

1. Apache ServiceMix，开源ESB，支持JBIG规范；
2. Coghead，创建基于Web应用的在线平台；
3. Tuscan SCA，SCA参考实现；
4. SUPER，综合EU研究项目。

在开源 BPEL 领域，不能不提另一著名的产品：ActiveBPEL。与 ActiveBPEL 相比，ODE 主要表现在许可证友好和易于与 ESB 集成方面。至于开发环境方面，ActiveBPEL 提供了基于 Eclipse 的图形设计器，ODE 并没有提供自己的设计器，可以使用 Intalio Designer 或 Lomboz 作为替代选择。

参考资料

网站类

1. ODE官方网站：<http://ode.apache.org/>;
2. OASIS WS-BPEL文档站点：<http://docs.oasis-open.org/wsbpel/>;
3. ODE论坛：<http://www.nabble.com/Apache-Ode-f16254.html>;
4. Lomboz官方网站：<http://lomboz.objectweb.org/index.php>;
5. IBM DeveloperWorks中文站 SOA与Web服务专区：<http://www-128.ibm.com/developerworks/cn/webservices/>;
6. BEA dev2dev中文站点：<http://dev2dev.bea.com.cn>;
7. Intalio公司网站：<http://www.intalio.com/>;

书籍类

1. 《Web Services Platform Architecture: SOAP, WSDL, WS-Policy, WS-Addressing, WS-BPEL, WS-Reliable Messaging, and More》
作者：Sanjiva Weerawarana等 出版社：Pearson Education, Inc
出版时间：2005年 ISBN：0131488740
2. 《BPEL Cookbook: Best Practices for SOA-based integration and composite applications development》
作者：Jeremy Bolie等 出版社：Packt Publishing 出版时间：2006年 ISBN：9781904811336

文章类

Apache ODE简介：<http://www.infoq.com/articles/paul-brown-ode>

基础架构

ODE 以创建一个可靠、简洁和可嵌入的组件为其主要设计目标。因此，它关注于创建一些具有最小依赖的模块，这些模块通过装配形成一个功能齐全的 BPMS。它的主要组件包括：

1. ODE BPEL编译器

负责将BPEL相关资源（即BPEL描述文档、WSDL和Schema）编译成适合执行的描述。编译成功之后会产生一个扩展名为.cbp的文件，该文件是BPEL运行时唯一需要的文件。编译失败时，编译器会列出与资源有关的错误信息。

2. ODE BPEL引擎运行时

负责执行编译后的BPEL过程，主要的功能包括：过程实例的创建、销毁、消息的收发等。它还为用户工具与引擎交互提供了过程管理API。执行过程中的实例状态跟踪和消息传递都会需要利用持久化机制，这由DAO组件来完成。

引擎运行时利用Java并发对象（Java Concurrent Objects，Jacob）来完成过程实例的状态表示和并发性管理。Jacob提供了应用级的并发机制，它不依赖于线程，这样就降低了系统的开销。ODE的官方网站提供了一份Jacob的教程，有兴趣的读者可去了解一下。

3. ODE DAO

DAO组件负责BPEL引擎运行时和底层数据存储的交互，数据存储一般使用关系数据库。此时，DAO由OpenJPA来实现。可以自定义不使用JDBC的DAO实现，目前ODE并不提供这种DAO实现。对于自定义DAO实现，需要处理以下持久化问题：

- ◆ 激活实例，跟踪哪些实例已经被创建；
- ◆ 消息路由，哪些实例在等待哪些消息；
- ◆ 变量，每个实例的BPEL变量的值；
- ◆ 合作伙伴链接（partner links），每个实例的BPEL合作伙伴链接值；
- ◆ 过程执行状态，序列化Jacob的状态。

4. ODE集成层（IL）

ODE BPEL引擎运行时依赖集成层与外界进行通信。在执行环境中，集成层内嵌了运行时。除了通信，集成层还给运行时提供了线程调度机制，并管理运行时的生命周期。当前版本提供了两种集成层实现：Apache AXIS2，使用Web服务进行交互；ServiceMix，通过JBI消息总线进行交互。

作为总结，以上组件的关系如图 7-1 所示：

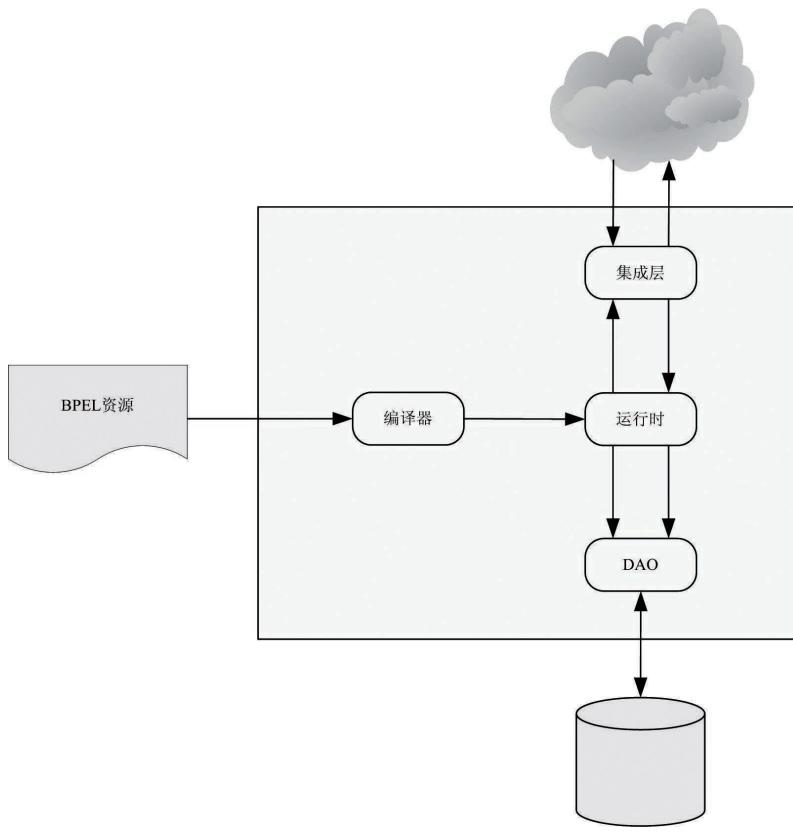


图7-1 ODE的主要组件

安装

ODE 的官方网站上提供了两种二进制的包，分别对应于 Axis2 和 ServiceMix 两个集成层的实现。在本教程中使用 AXIS2 作为例子，JDK 版本是 Java 1.5.x，安装步骤如下：

1. 下载对应Axis2的二进制包，解压到目标目录；
2. 将其中的ode.war复制到Tomcat的webapps目录，Tomcat的版本为5.5.x；
3. 运行Tomcat，访问<http://localhost:8080/ode>，出现Axis2的欢迎页面即表示安装成功。

ode.war 默认使用内嵌数据库 Derby，用户也可换成其他的关系数据库。官方网站的用户指南给出了在 Tomcat 下使用 MySQL 的例子。大致做法是：首先，在 Tomcat 中创建一个指向 MySQL 数据库的 JNDI 数据源；其次，在目标数据库中导入 ode 的 Schema；再次，在 %Tomcat_Home%/common/lib 放置 MySQL JDBC 驱动；最后，修改 ode/WEB-INF/conf 目录下的 ode-axis2.properties，将配置的 JNDI 名字作为数据源的名字。详细过程请参见用户指南。

开发方法

使用 ODE 进行开发还是相当简单的，只需 3 步即可：定义 BPEL 资源，部署，使用。本节通过

建立一个简单的 Echo 例子来说明整个开发过程。该例子实现了常见的 echo 功能，它返回的消息内容就是它所接收的消息内容。在进行开发时建议使用 Lomboz，它提供了对 ODE 的支持，并带有一个图形化的设计工具，降低了开发的难度。步骤如下：

1. 定义BPEL资源，在本例只需两种。

- ◆ WSDL，定义暴露的Web服务，定义如图7-2所示：

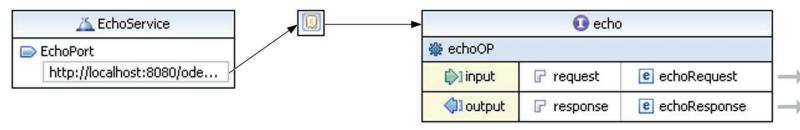


图7-2 WSDL定义

限于篇幅在此不列出全部 WSDL 文件的内容，只对关键点进行说明：

- ◆ portType为echo，它只有一个操作echoOP。
- ◆ echoOP的输入和输出分别对应两种消息：echoRequestMessage和echoResponseMessage。
- ◆ portType的绑定风格使用“document”，协议使用soap。
- ◆ 服务名称为EchoService，对应的端口是EchoPort，地址为：<http://localhost:8080/ode/processes/echo>。
- ◆ partnerLinkType如下：

```
<plnk:partnerLinkType name="echo">
  <plnk:role name="echoProvider" portType="tns:echo"/>
</plnk:partnerLinkType>
```

2. BPEL，定义了过程描述，定义如图7-3所示。

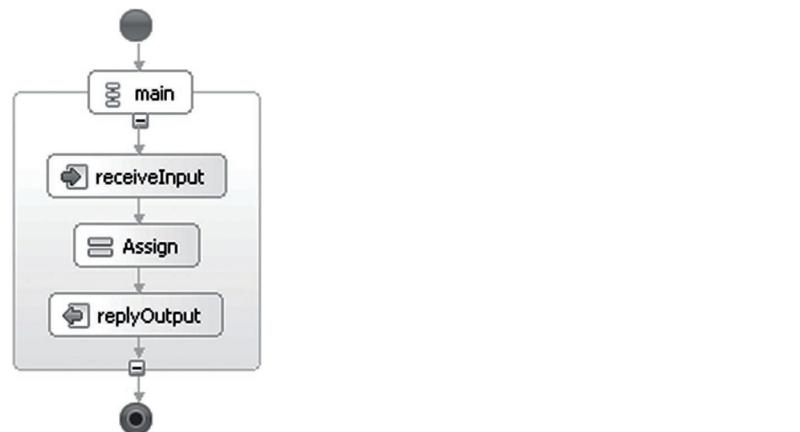


图 7-3 BPEL流程定义

Echo 过程只包含 3 个主要步骤：

- ◆ 接收消息；
- ◆ 将接收到的消息内容复制到变量中；

- ◆ 返回消息。

文件的内容见下：

```
<!-- echo BPEL Process [Generated by the Eclipse BPEL Designer] -->
<process name="echo" targetNamespace="http://ode/sample/echo"
    suppressJoinFailure="yes"
    xmlns:tns="http://ode/sample/echo"
    xmlns="http://docs.oasis-open.org/wsbpel/2.0/process/executable">

    <import location="echo.wsdl" namespace="http://ode/sample/echo"
        importType="http://schemas.xmlsoap.org/wsdl/" />

    <partnerLinks>
        <partnerLink name="client" partnerLinkType="tns:echo"
            myRole="echoProvider"/>
    </partnerLinks>

    <variables>
        <variable name="input" messageType="tns:echoRequestMessage"/>
        <variable name="output"
            messageType="tns:echoResponseMessage"/>
    </variables>
    <sequence name="main">
        <receive name="receiveInput" partnerLink="client"
            portType="tns:echo" operation="echoOP"
            variable="input" createInstance="yes"/>
        <assign>
            <copy>
                <from variable="input" part="request"/>
                <to variable="output" part="response"/>
            </copy>
        </assign>
        <reply name="replyOutput" partnerLink="client" portType="tns:echo"
            operation="echoOP" variable="output"/>
    </sequence>
</process>
```

3. 部署

ODE 支持两种部署模式：

- ◆ 将BPEL资源使用zip压缩，然后用部署Web服务或JBI部署进行部署。部署后，ODE将压缩文件解压到一个目录，目录名格式为压缩文件名+版本号。该版本号是ODE自动生成的；
- ◆ 在WEB-INF/processes下新建目录，然后将定义好的BPEL资源手工地放到该目录中。该种方式下，ODE仍然会记住过程的版本号，但是不会显式地出现在文件系统中。

本例使用第二种方式。

- ◆ 使用，启动Tomcat，访问由WSDL暴露的Web服务即可。

为了测试过程，ODE 提供了一个 sendsoap 命令。通过它可以给一个 Web 服务发送 SOAP 消息，通过返回结果可以了解过程是否正确工作。本例使用的 soap 消息如下，它保存在 echo.soap 文件。文件内容如下：

```
<?xml version="1.0" encoding="utf-8" ?>
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/
    envelope/">
    <SOAP-ENV:Body>
        <ns1:echoRequest xmlns:ns1="http://ode/sample/echo">
```

```
Hello ODE!  
</ns1:echoRequest>  
</SOAP-ENV:Body>  
</SOAP-ENV:Envelope>
```

测试命令：

```
sendsoap http://localhost:8080/ode/processes/echo echo.soap
```

输出结果如图 7-4 所示：

```
<?xml version='1.0' encoding='UTF-8'?><soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"><soapenv:Body><axis2ns1:echoResponse xmlns:axis2ns1="http://ode/sample/echo" xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/" xmlns:ns1="http://ode/sample/echo" xmlns="http://ode/sample/echo">Hello ODE!</axis2ns1:echoResponse></soapenv:Body></soapenv:Envelope>
```

图7-4 输出结果

相关建议

- 熟悉相关的规范和标准：WSDL、SOAP和BPEL。
- BPEL通常是应用中性能的瓶颈，在使用前需要考虑这个BPEL是否是必需的。
- 与熟悉ODE相比，熟练应用BPEL对业务过程建模的重要性则要高得多。
- 如果BPEL的移植性要求高，那么就应该避免在过程定义中使用工具特定的BPEL扩展。

版本信息

ODE 的当前版本是 1.1，这也是它从孵化器中毕业后的第一个重要版本。该版本的功能特性已在“功能特点”中有所说明，对于未来版本的计划，官方网站上已经公布了相关的路线图。

- 对于BPEL 2.0兼容性工作：
 - 支持<toParts>和<fromParts>；
 - 使用那些不是消息的变量进行调用；
 - 隔离范围（Isolated Scope）。
- 使用OpenJPA实现所有的管理查询。
- 使用对性能感知更加智能的调度器代替Quartz。
- 以RESTful方式暴露管理API。
- 实现一个SCA集成层（基本的集成已发布在Tuscany 1.0中）。
- 增加更多的BPEL 2.0测试用例。
- 提供管理工具。

社区视角

从架构上看，ODE 的集成层是一个非常不错的设计，这也是它与其他开源 BPEL 引擎最大的区别之一。通过实现不同的集成层，可以使 ODE 方便地扩展到其他的环境中，为之提供执行 BPEL 过程的能力。ODE 默认提供的两种集成层实现 Axis2 和 ServiceMix，分别给 Web 服务环境和 JBI 环境提供了 BPEL 的能力。前者使用的是通常的 SOAP/HTTP 方式，后者则使用 JBI 规范中的 NMR。此外，随着 SCA 受到关注，在开发者列表中也有人建议引入新的集成层实现：SCA。

除了集成层，ODE 使用的 Jacob 框架是另一个亮点。通过它，在引擎内部的消息传递并不使用重量级的 WSDL 消息（只在引擎与外界通信时才用上），而是使用轻量级的内部消息（它只是一个 Java 类型）。并发性的管理是建立在

每个过程实例单个线程基础之上的，这避免了创建过多的线程而带来的线程调度对系统性能的影响。有人曾对 ODE、ActiveBPEL 和 Oracle BPEL 做过性能比较，结果排名顺序依次是 ODE、ActiveBPEL、Oracle BPEL（参见 <http://twit88.com/blog/2007/09/11/comparisons-of-open-source-bpel-engine-versus-oracle-bpel/>）。但是需要了解的是，性能除了受软硬件平台的限制外，影响最大的还是过程本身的设计。

对于这个新开源 BPEL 引擎的到来，不少人的首先反应就是“既然已经有了 ActiveBPEL，为什么还需要 ODE”。该问题也出现在 ODE 的用户邮件列表中。对此，邮件列表中的讨论主要集中在：

1. ODE的许可证商业更加友好；
2. 虽然ActiveBPEL是开源的，但是对于商业使用一般还是需要购买商业许可证。因为开源版本的功能有限；

3. ODE与ESB集成更方便。

其中最主要的原因集中在许可证上。著名的开源内容管理软件 Alfresco ECM 的创建者 John Newton 表示，尽管 ActiveBPEL 非常诱人，但是其严格的许可证最后让他放弃了将它包含到他们的 ECM 套件中的主意。

不管怎样，多一种选择总是好事。在做出决定之前，了解一些有关的风险还是有必要的：

1. ODE才刚刚出道，英文和中文资料都非常的有限，社区参与必不可少；
2. 不论选择哪种工具，SOAP/WSDL/BPEL都是核心，熟悉它们是熟练运用工具的前提。不要幻想有了工具就可以替代对这些标准规范的了解，掌握它们有一定的难度；
3. BPEL并不提供对人工活动的直接支持，如果需要人工活动且必须使用BPEL完成，就需要WS-BPEL4People扩展。目前ODE并不支持它。

7.3

Apache Tuscany

总评

上手容易程度	★★★★★
社区活跃性	★★★★★
应用广泛性	★★★★★
推荐指数	★★★★★

随着 SCA（服务组件架构）和 SDO（服务数据对象）最终落户 OASIS，国内对它们的关注也呈上升趋势。与以往 SOA 开发部署方式相比，SCA 所提供的编程模型和装配模型大大减轻了这些工作的难度。加之有 OASIS 标准组织和以 IBM 为代表的业界巨擘的保驾护航，SCA 和 SDO 的前途不可限量。

Tuscany 是 SCA 和 SDO 的第一个参考实现，旨在提供基于 SCA 和 SDO 的 SOA 基础设施。Tuscany 几乎是伴随规范成长起来的，这样做的目的是对规范进行验证并获得有益的反馈。Tuscany 当前版本实现了 SCA 1.0 规范和 SDO 2.1 规范，它目前处于 Apache 的孵化器中。

鉴于 SCA 成长性良好，以及 Tuscany 的首个参考实现的地位，我们强烈建议研究和关注 SOA 的人们去了解它。

功能和特点

Tuscany 项目包含 3 个子项目：SCA 项目、SDO 项目和 DAS 项目。其中 SCA 和 SDO 分别对应于 SCA 和 SDO 规范，DAS 项目暂时没有相关的规范对应。以上 3 个子项目的功能特点如下：

Tuscany SCA子项目

实现了 SCA 1.0 规范，目前有 Java 和 C++ 两个实现版本。以下为 SCA-Java 的功能特点。

1. 实现了 SCA 主要规范，包括：
 - ◆ SCA 装配模型 1.0
 - ◆ SCA 策略框架 1.0
 - ◆ SCA Java 公共标准和 API 1.0
 - ◆ SCA Java 组件实现 1.0
 - ◆ SCA Spring 组件实现 1.0
 - ◆ SCA BPEL 客户端和实现 1.0
 - ◆ SCA Web 服务绑定 1.0
 - ◆ SCA EJB 会话 Bean 绑定 1.0
2. 支持许多尚未在 SCA 规范中定义的特性：
 - ◆ 支持 DWR、RSS 和 ATOM Feed 的 SCA 绑定
 - ◆ HTTP 资源、JSON-RPC、PUB/SUB 通知和 RMI
 - ◆ 支持 OSGI、XQuery、BPEL 及包括 Groovy、Javascript、Python 和 Ruby 在内的动态语言的 SCA 实现类型
 - ◆ 支持 SDO、JAXB、XmlBeans、Axis2 的 AXIOM、JSON、SAXON、DOM、SAX 和 StAX 的数据绑定
3. 支持多种部署方式：
 - ◆ 单独运行
 - ◆ 跨多 JVM 的分布式节点运行
 - ◆ 内嵌于 Jetty 或 Tomcat 中运行
 - ◆ 作为标准的 Web 应用运行
 - ◆ 作为 Geronimo 中的一个模块运行（实验阶段）

Tuscany SDO子项目

实现了 SDO 2.1 规范，目前只有 Java 实现版本。以下是其功能特点：

1. 简化和统一 SOA 中异构数据的访问和代码；
2. 为数据访问提供了静态和动态编程模型；
3. 基于离线数据集，并能跟踪数据集中数据的变化；
4. 提供了冲突的乐观锁检测机制。

Tuscany DAS子项目

实现了访问关系数据的 SDO 接口，只有 Java 实现版本。以下是其功能特点：

1. 将对数据源的查询转化成一个数据对象图；
2. 将数据对象图的更新转化为对数据源的一系列的 CUD 操作；
3. 基于 Command 模式；
4. 支持 MySQL；
5. 静态数据对象；
6. 用于静态图的动态根；
7. 关系的“唯一性”属性；
8. 定义结果集格式；
9. 日志功能改进；
10. 配置可通过编程完成；
11. 用于空 SDO 图的帮助类；
12. COC (Convention over configuration)；
13. “ID”列是主键；
14. “xxx_ID”是“xxx”表的外键。

背景介绍

长期以来，SOA 被视为解决业务和 IT 之间映射的灵丹妙药。在各大厂商、机构和媒体的包装下，SOA 被标榜为具有加速业务实现、快速应对业务变更，以及重用现有投资等种种好处。如今，从事企业开发的开发者如果没有听说过 SOA，在开发者中他会被视为异类，在客户心目中，就有可能对他的水平打上大大的问号。毕竟，在如此猛烈的宣传攻势下，从开发者到客户没有谁能充耳不闻。

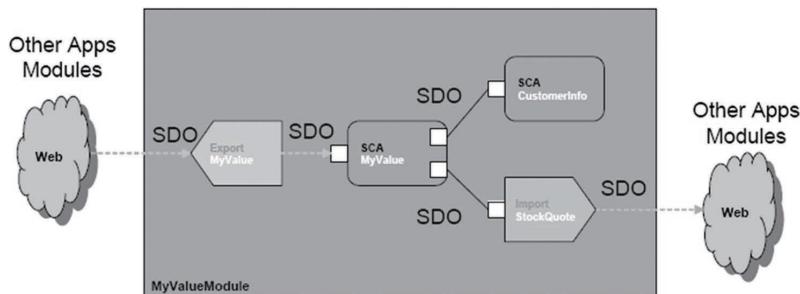
但是，现实中的 SOA 真的是那么美好么？个中奥秘，恐怕只有实践者才能体会得到。由于面对过多的技术细节，缺少清晰的编程模型，加上部署环境复杂多变，使得 SOA 项目危机重重。而这，正是 SCA/SDO 规范大显身手之处。

由 IBM、BEA 等业界巨头共同开发制定的 SCA/SDO 规范，是近年来影响最大的 SOA 规范。SCA 于 2005 年面世，旨在为服务网络的构建、装配和部署定义一个简单、基于服务的模型。SDO 则早在 2003 年就已经存在，它要解决的问题是为 SOA 环境下，异构数据的访问提供一个统一简单的方式。为了促进规范的制定、开发和推广，这些厂商成立了 OSOA (Open Service Oriented Architecture) 组织。目前 OSOA 组织的厂商有 18 家，旨在为 SOA 起草一系列的规范，并以免版税的许可方式提供给业界使用，值得一提的是国内厂商普元也是该组织的成员之一，这也是该组织唯一的国内厂商。目前 OSOA 下只有两个项目：SCA 和 SDO。但是 OSOA 毕竟不是标准组织，而且有强烈的厂商背景。为了进一步促进 SCA/SDO 的推广和采纳，2007 年 3 月，OSOA 联盟宣布了 SCA 和 SDO 规范中关键部分的完成，并发布了 SCA 1.0 和 SDO 2.1 的时候，将其正式提交给 OASIS。同年 9 月，SCA/SDO 正式成为 OASIS 下的标准，同时成立 OpenSOA (Open Service Oriented Architecture) 工作组负责标准化。

与 WS-* 等 SOA 标准规范最大的不同点在于，创建 SCA/SDO 并非为了解决 SOA 领域中的某一技术问题。它更多地偏向于创建一个明确的 SOA 编程和装配模型，帮助 SOA 实实在在地落地。与组件技术相结合，为开发者提供一个简单的 SOA 编程运行时；同时提供一个装配模型，将一些需要在部署时才确定的绑定协议、策略等信息推迟。这些都使得开发者在开发时无需面对具体的技术细节，而将更多的精力投入到业务问题的解决之中。这无疑大大地提高了 SOA 开发实施的效率。

SCA 和 SDO 均可以独当一面，并非一定要联合使用，但联合使用有助于将两者的优势结合。联合使用时，SCA 对应服务网络结构，其每个节

点由具体的业务组件组成；而 SDO 则对应于在服务网络上各个节点间传递的业务数据。它们之间的关系如图 7-5 所示：



(摘自：SDO: Service Data Objects, An Overview, 该教材在 OSOA 网站上可下载)

图7-5 SCA和SDO

目前，基于 SCA 的产品不少，但是开源的可选择的并不多。在 Java 领域，主要是 Apache Tuscany、Codehaus Fabric3、Eclipse STP。其中，Tuscany 是第一个实现 SCA 1.0 和 SDO 2.1 规范的开源软件，它注重提供一个 SOA 的基础设施，项目本身并不提供 SOA 开发和管理方面的 IDE 插件。据说 Fabric3 项目本身是 Tuscany 项目内讧的结果，这次的内讧导致了 Bea 的开发者

全部撤出 Tuscany，而成立了 Fabric3。目前该项目的主导是 Bea，目前还没有 1.0 版本的发布。Eclipse STP，它的目的是为 SOA 软件的设计、配置、装配、部署、监视和管理提供支持，它利用 SCA 作为其模型。目前项目还在 Eclipse 的孵化器中。从了解 SCA/SDO 本身来说，Tuscany 显然是一个最佳的选择。

参考资料

网站类：

1. OSOA的官方网站：<http://www.osoa.org/>
2. IBM DeveloperWorks中文站 SOA与Web服务专区：<http://www-128.ibm.com/developerworks/cn/webservices/>
3. BEA dev2dev中文站点：<http://dev2dev.bea.com.cn>
4. Apache Tuscany官方站点：<http://incubator.apache.org/tuscany/>
5. Eclipse plugin的更新站点：<http://people.apache.org/~jsdelfino/tuscany/tools/updatesite/>
6. 构客网：<http://gocom.primeton.com/>

该网站是 OSOA 组织成员，国内厂商普元创办的 SOA 和面向构件社区。从该网站可获得不少关于 SCA/SDO，以及 Tuscany 的相关讨论。

书籍类：

1. 《Understanding SOA with Web Services中文版》
译者：徐涵 出版社：电子工业出版社 出版时间：2006年 ISBN：7121028018
2. 《SOA原理、方法、实践》
作者：毛新生等 出版社：电子工业出版社 出版时间：2007年 ISBN：9787121042645

文章类：

1. SCA入门的绝佳材料：http://www.davidchappell.com/articles/Introducing_SCA.pdf
2. SOA编程模型：<http://www.infoq.com/cn/articles/SOA-programming-models>
3. SDO介绍
第一部分：<http://java.sys-con.com/read/313547.htm>
第二部分：<http://java.sys-con.com/read/358059.htm>

快速上手教程

本节就 Tuscany 的 3 个子项目分别给出相关的入门介绍，为读者在今后的使用和研究铺平道路。
所需环境如下：

1. Java: JDK 1.5.X
2. 如果用Ant: 1.7.0
3. 如果用Maven: 2.0.6

Tuscany 的安装非常简单，从官方网站下载解压即可。

Tuscany SCA

1. 基本概念

◆ 基础

组件（Component），组件是SCA中最基本的功能单元，可以类比于Java中的类。需要注意的是，由于SCA语言中立的特性，SCA组件可能是Java代码，也可能是C++代码，甚至是BPEL描述语言。

组合（Composite），组合是一个逻辑概念，类似功能模块。它可以包含多个组件，也可以包含多个组合。在物理上组合并没有对应物，它可能是单机单进程的，也可能是多机多进程的。组成组合的组件可以是由单个技术实现的，也可以是多个不同技术实现的。在SCA中，组合使用SCDL（服务组件定义语言）来定义，保存到composite文件中。

域（Domain），域代表了一个完整的配置运行时，它可包含多个组合。单个域定义了所有SCA机制的可视化边境，如在同一个域内，可以使用域特定的通信机制。在域外则使用如Web服务等互操作机制与其他客户端进行通信。

◆ 组件相关概念

服务，代表组件对外提供的功能。

引用，代表组件对外界的依赖。

属性，代表组件的配置信息。

绑定，定义组件与外界的通信机制（一般就是通信协议），当这发生在同一域中时，可以不显式指定。

- ◆ 组合相关概念

连线（Wire），代表组合内组件间的依赖关系。

提升（Promotion），组合暴露其包含组件的服务或引用的过程。

- ◆ 策略，对应QoS方面的设定，如安全、可靠性、事务等。

2. 开发方法

SCA 的开发方法非常简单，一句话即可概括：先实现，再装配。在此，通过实现一个简单的组合来说明使用 Tuscany SCA 的过程。在这个组合中包含有两个组件——GreetingService 和 IsWhoService，前者调用后者，并打印出相应的信息。整个实现步骤非常简单：

定义和实现 GreetingService

```
public interface GreetingService {  
    String greeting();  
}  
.....  
import org.osoa.sca.annotations.Reference;  
public class GreetingServiceImpl implements GreetingService {  
    private IsWhoService isWho;  
    @Reference // 定义引用  
    public void setIsWho(IsWhoService isWho) {  
        this.isWho = isWho;  
    }  
    public String greeting() {  
        return "Hello " + isWho.who();  
    }  
}
```

定义和实现 IsWhoService

```
public interface IsWhoService {  
    String who();  
}  
.....  
public class IsWhoServiceImpl implements IsWhoService {  
    public String who() {  
        return "SCA";  
    }  
}
```

装配（hellocomposite.composite）

```
<composite xmlns="http://www.osoa.org/xmlns/sca/1.0" name="Greeting">  
    <component name="GreetingComponent">  
        <implementation.java class="hellocomposite.  
            GreetingServiceImpl"/>  
        <!-- 与Spring非常类似 -->  
        <reference name="isWho" target="IsWhoComponent" />  
    </component>  
  
    <component name="IsWhoComponent">  
        <implementation.java class="hellocomposite.IsWhoServiceImpl"/>  
    </component>  
</composite>
```

测试运行

```
import org.apache.tuscany.sca.host.embedded.SCADomain;
.....
SCADomain scaDomain= SCADomain.newInstance("hellocompositecomposite");
GreetingService greeting= scaDomain.getService(GreetingService.class,
                                                "GreetingComponent");
System.out.println(greeting.greeting());
```

从整个步骤来看，它与 Spring 的开发方式极其相似。其中，在代码中使用 @Reference 来定义引用；在 .composite 文件中，使用 implementation.java 指明组件的实现是 Java，使用 <reference> 定义组件间的引用关系。组件间的依赖关系采用依赖注入的方式由 SCADomain 对象完成，从这个角度来看，SCADomain 与 Spring 的 BeanFactory 非常类似。

3. 相关建议

这个例子毕竟只能用来演示 Tuscany SCA 的使用方式，对于实际的开发应用作用却十分有限。为了能在实际的生产环境中使用它，有几点建议可以参考：

- ◆ 以David Chappell的《Introducing SCA》为入门材料（下载地址见上述的参考资料）。
- ◆ SCA目前的资料并不多，它的白皮书是难得的学习材料，而且官方网站也提供了一些教程和文章的链接。
- ◆ 学习例子代码。Tuscany的文档虽少，但是例子代码却相当丰富。但是全部把例子代码都看一遍显然是不合适的。有些例子代码可以在实际需要时再看不迟。阅读例子以相应涉及的知识点为优先考虑的因素。
- ◆ 参与论坛，多与同行进行交流，共同学习，共同成长。

Tuscany SDO

1. 基本概念

- ◆ DataObject，它包含实际数据。其中的数据可能是一些原始类型的数据，如int；也有可能是指向其他DataObject的引用。DataObject还有指向它的元数据的引用。
- ◆ DataGraph，它是一个多根的DataObject集合，会记录所有数据的变化，包括DataObject的CRUD。
- ◆ 元数据，记录DataObject的数据类型、约束和关系等信息。
- ◆ 数据访问服务（DAS），负责由数据源产生DataGraph，并将DataGraph的变化应用到数据源中。

另外需要注意的一点就是，SDO 是一个脱机的数据集合，这是它与 JDBC 的结果集最大的区别。SDO 这 4 个基本组件之间的协作关系如图 7-6 所示：

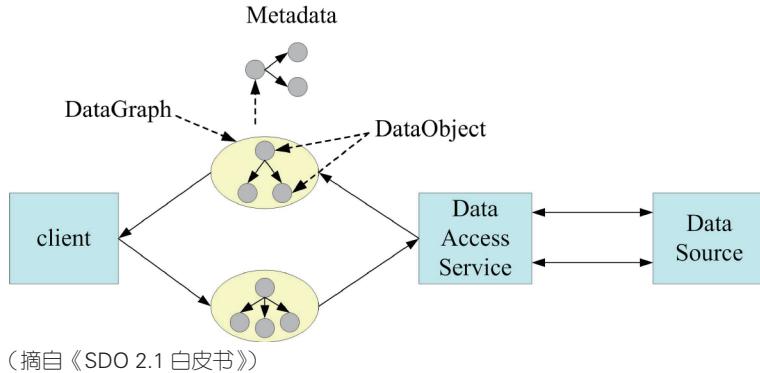


图7-6 SDO主要组件

2. 开发方法

SDO 提供了静态和动态两种编程模型。在静态模型中，不需要使用太多的 SDO API，而是使用相应的工具，根据定义好的数据模型（一般保存在 xsd 文件中）直接产生强类型结构，然后如普通对象一样使用它就行了。动态模型则相反，它不会产生一个强类型结构，而是根据数据模型在程序运行时动态创建 DataObject，整个过程会大量使用 SDO API。

以下分别给出静态模型和动态模型的例子，这两个例子都使用以下的 XSD (name.xsd)：

```
<xsd:schema xmlns:name="name.xsd" targetNamespace="name.xsd"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <xsd:element name="name" type="name:Name"/>
  <xsd:complexType name="Name" mixed="true">
    <xsd:sequence>
      <xsd:element name="first" minOccurs="0" type="xsd:string"/>
      <xsd:element name="last" minOccurs="0" type="xsd:string"/>
    </xsd:sequence>
  </xsd:complexType>
</xsd:schema>
```

- ◆ 静态模型

使用org.apache.tuscany.sdo.generate.XSD2JavaGenerator产生对应的Java对象，书写对应的Ant脚本：

```
<target name="generate-sdo" depends="init">
<java classnames=
  "org.apache.tuscany.sdo.generate.XSD2JavaGenerator" fork="true">
  <arg value="-targetDirectory"/>
  <arg value="target/sdo-source"/>
  <arg value="-javaPackage"/>
  <arg value="hello"/>
  <arg value="-prefix"/>
  <arg value="Name"/>
  <arg value="-noNotification"/>
  <arg value="-noContainment"/>
  <arg value="-noUnsettable"/>
  <arg value="src/main/resources/name.xsd"/>
```

```
<classpath refid="project.classpath"/>
</java>
</target>
```

使用：

```
Name name = NameFactory.INSTANCE.createName();
name.setFirst("David");
name.setLast("Haney");
```

其中，XSD2JavaGenerator 的 prefix 参数指定 xxxFactory 的前缀，此处是“Name”，最终 Factory 的名字就是 NameFactory；产生的 Java 对象的名字与 XSD 中元素类型的名字是一致的，此处是“Name”。

- ◆ 动态模型

获取类型信息

```
HelperContext scope = SDOUtil.createHelperContext();
XSDHelper xsdHelper = scope.getXSDHelper();
URL url = getClass().getResource("src/main/resources/name.xsd ");
is = url.openStream();
xsdHelper.define(is, url.toString());
```

使用DataObject

```
DataObject name = scope.getDataFactory().create("name.xsd ", "Name");
name.setString("first", "David");
name.setString("last", "Haney");
```

整个过程实际与静态模型一致，只不过一种是以强类型的方式产生再直接使用，另一种是在内存中创建再使用。两种方式各有优缺点：前者的耦合性高，但是简单易于使用，且可利用实现环境的编译器在运行前发现语法错误；后者的耦合性低，但是使用相对麻烦，而且数据类型的错误要在运行时才能发现。

3. 相关建议

- ◆ 通过例子了解API的使用，Tuscany SDO的例子给出了低、中、高三级的使用例子。但这些都是动态模型的例子，对于静态模型的一个例子可以参考Tuscany SCA的helloworld-ws-sdo。
- ◆ 着重了解它的应用场景。虽然在小场景中它的使用方式与JDBC之类的数据方开发包非常类似，但是如何在一个异构的环境中去使用它，是一个需要考虑的大问题。

Tuscany DAS

1. 基本概念

与Tuscany SCA/SDO不同，Tuscany DAS并没有对应的规范，它只提供了对访问关系数据库的支持。它的最大特点就是使用Command模式，关于该模式的介绍请参考GOF的《设计模式》。

2. 开发方法

Tuscany DAS的使用步骤分为两步：

- ◆ 在配置文件中定义连接、Command名字、表的信息和它们之间的关系等信息；
- ◆ 由配置文件创建一个DAS对象，然后由DAS创建相应的Command对象去使用。

具体过程请参见Tuscany DAS自带的例子。

版本信息

Tuscany 项目由 3 个子项目组成，当前每个项目中涉及语言实现和版本号如下：

SCA	SDO	DAS
C++, 1.0 M3	C++, 开发中	C++, 开发中
Java, 1.01	Java, 1.0	Java, 1.0 beta2

关于当前版本的功能特点，在上述文字中已有交待，在此就不再赘述。至于未来的走向，在新版本发布后不久，项目的主要开发人员 Jean-Sebastien Delfino 曾在邮件列表中征求过大家的意见。这些邮件的内容最后形成了 Tuscany SCA Java 的路线图讨论稿，该讨论稿 (<http://cwiki.apache.org/confluence/display/TUSCANYWIKI/Roadmap+Discussion>) 保存在 Tuscany 的 Wiki 中。其中对开发者来说，最关心的莫过于开发工具和文档。幸运的是，在该讨论稿中已经非常明确地提出，要改进对基于 Eclipse 的工具的支持，以及网站的文档。

社区视角

SCA 和 SDO 中的概念和模型并不是新鲜的面孔，它们也借鉴了许多已经在业界成功应用的经验和成果。如 SCA 的编程模型和装配模型，对于熟悉 Spring 的开发者来说并不陌生。其中的依赖注入、标注、分离关注点和声明式的配置方式，早已在 Spring 中广泛使用。而 SDO 本身，其实也就是一种 DTO（数据传输对象）的应用，并且它的离线结果集、元数据、数据变更记录等概念，也可在 .Net 的 ADO.Net 中的 DataSet 中找到相应的影子。此外，DAS 应用了 Command 模式，而这也与 Ado.Net 的数据访问方式不谋而合。

SCA 和 SDO 并非是一种彻底的革命，而是技术发展道路上的一次演变。这样降低了失败的风险，方便其推广和被接受。而且需要注意的是，

SCA 和 SDO 并不是替代现有的 SOA 相关规范和标准。它的目的是提供一个编程和装配模型，以方便地使用它们。因此，与现有的标准规范之间的关系更多的是一种互补，而非竞争关系。2007 年 9 月，SCA 和 SDO 最终成为 OASIS 下的标准，并成立了 OpenCSA（开放组合服务架构）工作组负责它们的标准化。这一事件无疑为 SCA 和 SDO 的推广使用起到了推波助澜的作用。

业界对于 Tuscany 的评价基本是正面的，而且作为 SCA/SDO 的牵头人之一的 IBM 已经准备将 Tuscany 融入其 WebSphere 产品线中。研究和使用作为 SCA/SDO 参考实现的 Tuscany，对于了解规范的意义重大。但是，在使用之前尚需了解潜在的风险：

1. 社区尚未有大规模应用的经验，这意味着遇到新问题的几率非常大。此时，应多求助于它的邮件列表；
2. Tuscany 官方网站上的资料和文档非常有限，在其他地方要找到相关的文档和资料也非常不易；
3. SCA 虽然简化了开发人员的编程任务，但是对于装配来说，相关的标准规范仍需了解。而这些内容涉及广泛，有一定的学习难度；
4. 由于规范正处于标准化的进程中，API 变化频繁是件常事，因此要有心理准备。

作者介绍

胡键，2000 年硕士毕业后从事软件开发工作，在实际的项目中长期担任项目经理和技术经理。关心软件技术和相关工具的动态，将其中相对成熟的技术和工具应用到实际的项目之中。已在 IBM developerWorks 中国网站发表数篇文章。目前醉心于服务器端软件的设计和开发，并致力于研究 SOA 方面的规范、技术和工具。

编后记

记得博文视点出版公司最近出过一本书，名为《编程之美》，很是畅销。但是在我撰写《开源技术选型手册》的编写手记的时候，首先浮在脑海中的词汇是“遗憾之美”。当 Lisa（本书责任编辑）告诉我，书稿终于拿去出片了，那一时刻我能感受到在电脑的那端，她肯定也在长舒一口气，虽然也许还有些忐忑。然后我们又聊起书中的一些遗憾，比如可能会有某些地方标点符号不正确，有些句子可能不够顺畅等等。这种感觉，我在从前做杂志期间经常有过，自己感觉非常费心费力，本认为非常完美的东西，在印刷之后常会又找到这样或者那样的 Bug。对于类似的问题，我们可以无限小地避免，但要完全消灭，可能就违背了“万事无绝对”的“真理”，我就开玩笑地告诉 Lisa——遗憾有时也是很美的，相信我们下次会做的更好。

这也许有点为自己开脱的意思，可是回想整本书从策划到撰写到成书的经过，这种感觉逐渐为“平和”所代替。想起在策划之初，周筠老师几次找到我，再拉上孟岩和方舟，进行沟通。先是总结这本书的前身《开源大本营》中的经验和教训，然后列出值得继续发扬和需要避免的地方，有的放矢地开始第二轮的迭代。而后将能联系上，可能会参与本书写作的作者拉到五道口蓝旗营的万圣书园开始头脑风暴。凯峰、李剑、玉宝等人先根据讨论的图书结构撰写样章，然后大家以此为靶子进行“攻击”和完善，最后总结出一个相对可行的框架结构，这也是现在本书的骨骼主体。单是万圣书园的几次聚会，我想就令几多参与人员回味的了，另外，要知道，有两次博文的朋友们还是专程从武汉赶到北京来参加的……还记得当时方舟为了让这本书的版式更加完美，还专门买了几本日本同行出版的砖头书，然后我们两个猫在丽都饭店附近的雕刻时光进行研究，详谈甚乐。因为种种原因，最终这些版式可能没有最终采用，但这个“遗憾”不也是一个很美好的记忆吗？

如果要列出这半年来值得回忆的事情，我想躺在摇椅上摇上几宿也不一定能叙述完毕，包括责任编辑 Lisa 的耐心催促，包括每个作者对自己作品的精益求精，包括对本书的针对性宣传，包括后期制作过程中的各种误会，包括封面的设计，包括作者之一袁峰前些天发邮件给我说，因为父亲的去世使他没有更多精力将稿件写到最佳，很是抱歉等等。尤其值得一提的是，在这次汶川地震灾难面前，我们的所有作者表现出了自己诚挚的关注和爱心，一致同意将自己

的稿费全部捐献给灾区，在本书的文前页每人还送上一句祝福，向灾区的人民表达问候。从这些等等一切的“琐事”中，我想每一个参与其中的人都会收获到自己的独特感受。

从一开始，《开源技术选型手册》这本书的定位就是面向项目经理、技术经理、高级软件工程师包括 CTO 等的，我们将每个领域的专家汇集在一起，将他们的经验和智慧汇集在一起，力求在尽量少的篇幅内给这些技术选型决策者提供最有价值的参考。本书是多人协作完成，在写作过程中，也许会有这样或者那样的偏差，如果读者有所体察和发现，我们都欢迎大家不吝赐教。因为如果还有下次的话，这本书在反馈的基础之上肯定会做更新，肯定会更加完美。我想让后来者看到更完美的作品对每个作者和读者都是理应担当的责任，如果这也算是一种“责任”的话。

每一个作品，哪怕是不完美的作品的背后都蕴含了无数人的努力，《开源技术选型手册》也是一样。不做书不知道做书难，这次的经历让我实实在在感受到了做书人的艰辛，借写手记的机会也向那些所有从事出版行业的“苦命人”们表示敬意！

InfoQ 中文站总编辑 霍泰稳

<http://www.infoq.com/cn>

[登录China-Pub网站购买此书完整版](#)



清华大学出版社
http://www.tsinghua.org.cn

了解本书更多信息请登陆[博文视点官方博客](#)

