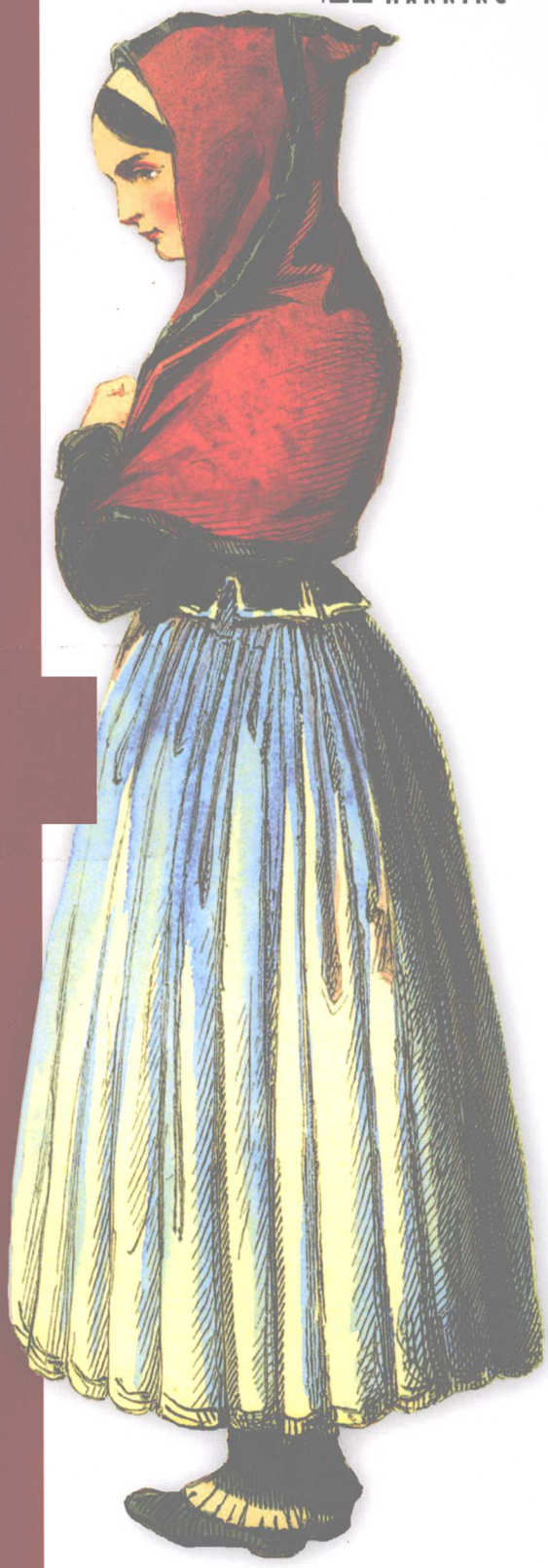
TURING 图灵程序设计丛书 Java 系列

MANNING

Seam in Action

Seam 实战

[美] Dan Allen 著
崔毅 杨春花 俞黎敏 译



人民邮电出版社
POSTS & TELECOM PRESS

TURING 图灵程序设计丛书 Java 系列

Seam in Action

Seam 实战

[美] Dan Allen 著
崔毅 杨春花 俞黎敏 译

人民邮电出版社
北 京

图书在版编目（CIP）数据

Seam实战 / （美）艾伦（Allen, D.）著；崔毅，杨春花，俞黎敏译. —北京：人民邮电出版社，2010.6
（图灵程序设计丛书）
书名原文：Seam in Action
ISBN 978-7-115-22464-4

Ⅰ. ①S… Ⅱ. ①艾… ②崔… ③杨… ④俞… Ⅲ. ①JAVA语言—程序设计 Ⅳ. ①TP312

中国版本图书馆CIP数据核字（2010）第038631号

内 容 提 要

本书深入讲解了JBoss Seam，介绍了Seam如何消除了不必要的层和配置，解决了JSF最常见的难点，建立了JSF、EJB 3和JavaBean组件间缺少的链接。书中也介绍了如何利用Seam进行技术综合，如业务过程、有状态的页面流、Ajax远程处理、PDF生成、异步任务等。

本书适用于Java程序员阅读。

图灵程序设计丛书

Seam实战

◆ 著 [美] Dan Allen
译 崔 毅 杨春花 俞黎敏
责任编辑 傅志红
执行编辑 罗 婧

◆ 人民邮电出版社出版发行 北京市崇文区夕照寺街14号
邮编 100061 电子函件 315@ptpress.com.cn
网址 http://www.ptpress.com.cn
北京艺辉印刷有限公司印刷

◆ 开本：800×1000 1/16
印张：31.75
字数：744千字 2010年6月第1版
印数：1－3 000册 2010年6月北京第1次印刷

著作权合同登记号 图字：01-2009-3808号

ISBN 978-7-115-22464-4

定价：89.00元

读者服务热线：(010)51095186 印装质量热线：(010)67129223
反盗版热线：(010)67171154

版 权 声 明

序

作为Seam项目的开发人员，我们面临的最大挑战不是编写代码，而是怎样向新用户解释并让他们了解Seam。对于初学者来说，要想真正弄明白Seam是什么，必须跨越一道很大的鸿沟。这不是因为Seam过于复杂，或者说学习Seam需要深奥的技术背景，而是因为Seam把主流Java程序员所不熟悉的许多理念综合到了一起。这其中有许多理念将对企业级Java开发常识产生冲击。

首先，Seam填补了一道许多Java程序员并没有意识到的空白。我们习惯了把几种技术组合到一个框架中，对于一套完整的应用程序集成框架却感到陌生。这种解体式的组合在持久层最令人苦不堪言。高速缓存无效和延迟加载困扰着大多数的应用程序，而Seam真正解决了这些问题。不要质疑这一点，要知道Seam的创造者们曾经是Hibernate幕后的智囊团！

现在Seam提供了动态双向注入（bijection），它与流行的依赖注入框架所提供的静态注入截然不同。我们要告诉你：当主流的技术迫使所有应用程序都进入多层次的无状态架构中，而不管该架构是否适合正在开发的应用程序时，其实还有更适合的有状态组件可用。

我们刚接触到最表面的内容，就已经可以看出Seam所带来的景象与现状之间的差距有多大，因而指导Seam的新用户就变成了一个巨大的挑战。市面上关于Seam的入门书几乎都只介绍基础知识，它们介绍了这门技术的ABC，却没有介绍如何用这些字母组词和造句。本书是第一本把握住了Seam精髓的Seam图书，它介绍了如何把这些词语和句子组合起来，这也是Seam团队期望Seam被使用的方式。

本书的独特之处在于作者Dan Allen没有局限于Seam的固有结构，而是对其进行了一番剖析，提取了它的核心概念，并以新颖独特的方式将它们重新组合在一起。本书并不是其他Seam图书内容的简单重复，而是很好的补充。它介绍了如何理解Seam，以及如何有效地将Seam应用到你自己的应用程序中去。

Seam可以帮助你编写出更好的代码、功能更加丰富的应用程序，也可以帮助你提高工作效率，用更简单、更易于管理的架构来编写应用程序。但前提是，你必须真正花时间去学习如何最好地应用这门技术。本书是最好的指导书，可以帮助你将Seam的优越功能发挥到极致。

如果你愿意接受这个挑战，希望能够对书中介绍的技术做到举一反三，就请开始行动吧。

Norman Richards
Red Hat高级工程师

致　谢

在写这本书时，我就向自己和朋友们承诺，交稿的时候要做些什么。其中最重要的一条承诺就是感谢每一位为本书付出努力的人们。当然，我要感谢你成为我的读者。但也请你感激那些让这本书面世并传递到你手里的人们。

首先要感谢的是我的妻子Sarah。如果没有她的帮助，就没有这本书的面世。对她的感激之情，我无法用言语表达。是她激励我要自信，当我觉得终点似乎遥遥无期时，是她让我保持斗志。她满脑子都是Seam，我不停地问她全书的结构问题，但她总是不厌其烦。她帮助我编辑草稿、调整索引、提供解决办法。她让我衣食无忧，并包揽了其他一切杂事。最令我感动的是，她放下了自己的事情来支持我，现在我真希望为她做点什么。读者朋友，也请你感谢我的妻子！

写书的过程中，会疏远朋友和家人，感谢所有支持我的朋友和家人，他们坚信我最终会完成使命，会再次回到他们身旁，与他们小聚，聊些与写书无关的话题。对于我的双亲：James和Mary Allen，我永怀感恩之心，因为他们给了我人生中每一次成功的机会。人生只有一次童年，他们却让我的童年变得终生受益并且充满了美好的回忆。爸爸妈妈，感谢你们坚持不懈的支持，在我奋斗的路途中一路相随。

说起本书的由来，我要感谢Andrew Glover将我介绍给IBM developerWorks的Jennifer Aloi，他倡办Seamless JSF系列，继而鼓励我开始技术写作生涯。这个系列的成功有一部分要归功于Athen O'shea，他们的编辑工作做得很棒，并且帮我找到了合适的词汇。当时我根本没有想到，我的想法会很快成为一本书。

我要感谢Marjan Bace和Michael Stephens的信赖，允许我将交稿日期一拖再拖。我知道，他们其实有一份真正的进度规划表，原本预计这个项目可以在15个月之前完成。我还要感谢CodeRyte公司的Andy Kapit和Andrew Van Etten，他们在本书的最初阶段，就认可了它。

接下来，我得感谢Cynthia Kane，她帮助我大开眼界，在我做白日梦的时候，提醒我还有一本书正等着我去写。感谢那些雄心勃勃的天才技术审核人员，他们奉献出自己的时间和学识，使本书成为学习Seam的最佳资源，他们是：Peter Johnson、Doug Warren、Peter Pavlovich、Devon Hillard、Nikolaos Kaintantzis、Hung Tang、Michael Smolyak、Benjamin Muschko、Kevin Galligan、Judy Guglielmin、Valentin Crettaz、Carol McDonald、Ara Abrahamian、Horaci Macias、Norman Richards、Ted Goddard、Costantino Cerbo、Mark Eagle、Carlo Bottiglieri以及Jord Sonneveld。感谢Karen Tegtmeyer，他找到了这些技术审核人员，安排审核工作，并“恐吓”志愿者要反馈他们的意见。特别要感谢Benjamin Muschko、Pete Pavlovich和Ray Van Eperen，他们通读了全书，并

逐行进行了校正和点评。感谢Michael Youngstrom，他审核了第15章[①]。感谢Ted Goddard和Judy Guglielmin，他们审核了第12章，并开发了ICEfaces范例的源代码。感谢Valerie Griffin和Daniel Hinojosa，他们为本书做了最终校订和反馈。还要感谢非正式版的所有真诚的读者，以及参与论坛讨论的读者，尤其要感谢那些自始至终在耐心等待本书出版的读者。

这个项目的英雄是Mary Piergies领导下的整个制作团队，他们的诱导，让我觉得改写不是那么的可怕，使我注意力高度集中，本书才得以印刷出版。在这个过程中承受最大压力的是编审Lis Welch。我由衷地向Liz表达谢意。他帮我纠正了书中所有矛盾的地方，成就了我对完美的追求。我还要感谢技术编辑Norman Richards，他让我将我所知道的Seam和盘托出，并教导我不要给读者提供不切实际的建议。下面我要列出由制作和后期制作团队的其余成员所完成的大量工作，对他们表示感谢：Katie Tennant校对原稿，消除了所有书写方面的错误。Dottie Marsico和Gordan Salinovic在短时间内将原稿从办公文档格式转变成了你眼前所见到的专业排版格式。Leslie Haimes将本书变得能在书架上引人瞩目，吸引读者。Tiffany Taylor维护文档的模板；Gabriel Dobrescu处理本书在manning.com网站上的相关事宜。Steven Hong一直在宣传本书，并且筹备了营销材料。

请与我一起感谢Gavin King，他将自己对Seam及其上下文组件模型的见解作为开源的项目与全世界共享。也感谢那些Seam的开发人员，他们将自己成熟的见解变成了今天这个健壮的集成框架。

我要感谢马里兰州的Panera Bread，当我在家找不到写作灵感的时候，他为我提供了第二办公室。我很喜欢那些茶和免费提供的无线网络，希望越来越多的公司能够像你们一样提供先进服务。

或许我漏掉了一些名字，但也同样感谢你们，以及文中提到的人们，是你们帮助我完成了人生中最大的一个目标。再次感谢我的妻子，感谢她在此期间对我的不懈支持。

① 本书第14章和第15章是以网上发布的形式公开在原书出版商网站上的，英文原文可登录www.manning.com/dallen/下载。中文版情况详见译后序。——编者注

前　言

“要想解决问题，就先要改变思维”。

——爱因斯坦

我在写这篇前言的时候，正飞越大西洋，这是我同一个月里第二次从欧洲回到美国。这次是到托斯卡纳参加会议，讨论Seam的未来；前一次是到苏黎世，当时我在Jazoon'08年度研讨会上做了关于Seam的演讲。那次旅行的意义对我来说尤其重大，因为那是我有生以来第一次到北美之外的地方旅行。我原以为这一天永远也不会到来，但是我竟然如愿以偿了，感谢Seam。（还要感谢我的哥哥Kevin，是他帮我买了机票。）

这听起来让人觉得有些不可思议，我人生中的这一重要里程碑居然要归功于Seam。毕竟，一种框架怎么可能激发一个人去做一次空前的旅行呢？你肯定认为我疯了，但请先听我解释我是如何与Seam结缘，以及它是如何影响着我，让我大开眼界的。

Seam还处在开发阶段的时候，我正被一个利用Spring和JSF构建的项目搞得焦头烂额。有一年多的时间，我一直在一成不变地努力管理应用程序的状态，和一些无关紧要的决策较劲，例如，要把业务对象命名为Manager还是Service，合理使用几个层，以及哪个层应该负责某项指定的任务。所有这些使人分心的事情阻碍了项目的进展，也阻碍了我的成长。我决心要找到一条出路。

Seam吸引我的亮点在于，它提供了通过页面描述符控制JSF请求的细粒度控制。我之所以迷上Seam（并且最终决定撰写此书），远远不只是因为它为我解决了当时的燃眉之急。

Seam有影响力，是因为它遵循一致的方法，而不强加任意的限制。它利用注解、拦截器、基于XHTML的模板以及JSF组件，尽量提高你的编码效率。当你需要对象时，或者在你需要对象的地方，Seam都提供了访问对象的权限，并替你管理对象。它还帮助你建立从一个页面请求到下一个页面请求的连续性。Seam给了你极大的自由，你可以根据自己的需要组织应用程序，选择构建应用程序的工具，比如用Java还是用Groovy，用XML还是用注解，用JavaScript还是用多功能的小部件，用内建组件还是用定制组件，等等。

但是我们往往会受到“框架”（framework）的束缚，忘记了编写软件的初衷是满足用户或者客户端用户的需要。这是开始学习这些工具时应该持有的观点。

用户并不想没日没夜地把结果集一页页地从头翻到尾，也不太关心视图中是否出现了延迟初始化异常的问题。他们只想要成熟的软件。他们要高级搜索，要用PDF或者Excel生成的报表，要图表、电子邮件、文件上传、动态图片、向导、工作空间，等等。用户要的一般都是真正难以开发的东西，至少比单纯只通过CRUD生成工具来操作数据库更难。Seam也提供了CRUD生成工具，

让你可以立即投入开发，但它的功能远不止这些。

Seam值得你去学习，是因为它几乎触及了Java EE的每一个方面。要学习的东西固然很多，但是有了Seam，Java平台的每个层面都变得非常容易使用，并让你能够在项目开发早期就处理应用程序的高级部分。你不必再担心用户提出天马行空般的疯狂需求了，反而觉得自己有能力编写应用程序，甚至会列出期待的特性清单。

作为一个集成框架，Seam网罗了众多随手可用的技术。因此，你会发现自己正在尝试一些前所未有的技术，并相信自己的应用程序和自身能力都会很快成熟起来。你还会开始在应用程序中引入新的交互风格，如事件观察者（event-observer）模式，或者像Ajax Push这类具有革命性的东西。你会习惯于探索新的领域，同时又不必放弃熟悉的东西。Seam影响了你对生活的整体态度。

回到我一开始提到的话题。Seam是激发我最终飞越北美的动力。它还促使我开始了我的写作和顾问生涯，让我置身于这个成功的开源项目中，让我有幸能与众多才华横溢的人们相识。Seam将如何改变你的职业生涯？它又将如何改变你的人生呢？

2008年7月写于大西洋上空的某个地方

关于本书

如果你准备成为一名Seam方面的专家，我保证这本书可以让你如愿以偿。我不会为了显示我的聪明，故意用一些让你费解的术语。我不会说："相信我，它可以解决所有问题。"当你正在努力钻研眼前的内容时，我不会用下一章的概要来转移你的注意力。我不会在类上放很多@In和@Out注解，希望你能自己领悟到它们的功能。我讲述事实。我给出了步骤，揭示了逻辑，还画了流程图。我最喜欢编程的原因是，每件事情的发生都是有原因的。目前最大的挑战在于，要了解是什么原因，然后回头考虑如何将它付诸实践。我承认，Seam的有些领域比较难以理解。但是相信通过引导，你会理解的。天无绝人之路，千万不要放弃！

我不仅教你Seam是如何工作的，还会告诉你原理和原因，这样你以后才能把Seam教给别人。我已经探索了Seam的每一个角落，并且愿意将我的经验与大家分享，激励大家自己去达到这种境界。我想将Seam教给我的东西教给你，即将程序员的潜能充分发挥出来的能力。这正是有助于你准确理解Seam的最好资源。

导读

本书的目的是让你快速掌握Seam。本书共分成四部分。第一部分对Seam进行综述，让你为学习Seam做好准备。第二部分关注核心概念，直到你能看出点名堂来。第三部分学习Seam的状态管理解决方案以及Java持久化支持。最后一部分教你如何使应用程序更加安全，并在竞争中独领风骚。最重要的是你能从中得到乐趣。

第1章回答了3个问题：Seam是什么？为什么要创造Seam？Seam应用程序是什么样的？这一章解释了Seam怎样适应Java EE，并列举了它扩展Java平台的方式，来使这个平台变得更易用，更恰到好处。本章呈现了一个基础的Seam应用程序，并给出了下文概要。

第2章并没有直接一头扎进Seam的基础知识，而是先引导你建立一个Seam项目。这不仅可以让你试验书中其余篇幅所阐述的Seam概念，还给你留下了一个完整的CRUD应用程序，它支持变动的递增热部署（hot deployment）。

由于JSF是Seam的主要视图框架，因此第3章对它进行了简要介绍，指出了它的弱点，并说明了Seam对它做了哪些改进。通过学习Seam对JSF提供的面向页面的强化功能，你对Seam如何参与到JSF生命周期便会有较为深刻的理解。学完本章，你应该能体会到"用JSF进行合理开发的唯一方式就是用Seam"。

第4章探讨了Seam的核心：上下文容器。你会知道Seam组件是什么，它与组件实例有什么区别，可以在哪些范围中保存实例和其他上下文变量，并知道Seam如何管理整个生命周期。你会

觉得像是在用注解控制应用程序一样。你还会学到访问组件的方式，以及组件什么时候被实例化。

在第5章介绍Seam的中央枢纽（组件描述符）。你会学到它的两项主要功能：用XML和注解来定义组件，并为属性赋初始值，以此控制某个组件的行为，或者构建对象原型。虽然这个文件中的元数据是XML，Seam却利用命名空间使配置变成是类型安全的（type-safe）。你还将学会开发自己的命名空间。该章最后介绍了Seam简单但却强大的消息管理法。

第6章极为重要，因为它介绍了Seam最引人注目、最先进的特性：双向注入（bijection）。使用双向注入的最大好处是使处于不同作用域中的组件实例能够安全地合作，而不会存在范围受阻或者并发干扰的风险。本章的另一个主题是：Seam如何按需初始化对象。

第7章介绍了Seam的另一项重要特性：对话（conversation）。基于Java的Web应用程序一直缺少一个与用户行为相关联的作用域。你会发现Seam的对话正好符合这一需要，它克服了HTTP会话（session）的缺点，并为用户提供了一种管理并行活动的方法。对话最重要的用处是管理持久化上下文。

要体会到Seam对Java持久化的改进之处，必须先了解什么是Java持久化。第8章的介绍会让你对Java持久化有大概的了解，它还指出了关于这一主题的有用资源；同时阐明了在纯Java EE环境中是如何管理Java持久化的，并帮助你了解Hibernate和JPA的区别。

第9章介绍服务于Seam的Java持久化，并示范了Seam如何正确地持久化，这也是Java EE的不足之处。你会了解到，对话作用域的持久化上下文让你避免了延迟初始化错误和脏合并操作。你还会知道，Seam将请求放在一系列事务中，进一步确保了对某个请求中的全部操作提供保障。本章通过检验多用户Web应用程序中最重要的特性而得出结论：应用程序事务（application transaction）在原子对话中进行了持久化操作。

第10章围绕着开发CRUD应用程序的两个例子展开——只有这一次，你是自己完成所有工作。当然，并不是真的“所有”。你将学到如何利用Seam应用程序框架中的类来处理大多数样板代码，因此你所要做的就是设计和定制用户接口。读完第2章和第10章，闭着眼睛都应该能完成CRUD了。

应用程序如果不具备安全性，它也就没多大用处了。第3部分的第11章教你怎样验证用户，然后接着教你如何实现基于角色的基本验证和基于规则的上下文验证，以强有力的方式保护应用程序。

Seam还有一个优点，就是它使其他技术看起来很棒。在第12章，你会学到如何利用RichFaces或者ICEfaces组件将Ajax交互添加到应用程序中，而不必触及一行JavaScript代码。Seam管理状态可以确保这些Ajax交互不会使服务器资源变慢。你还会学到扩展JavaScript功能的方法，即让它直接访问服务器端的组件，并学会将Seam与像GWT这样有着丰富用户界面的技术整合起来。

第13章教你创建各种各样的内容类型，如PDF、电子邮件、图表、图片和二进制文档，从而摆脱单调的HTML。你还会学到如何为应用程序设定样式，并通过用户界面进行控制。

关于这最后两章，我有太多与本书无关的东西想谈。本书的配套网站为www.manning.com/SeaminAction，你可以从中查阅到第14章介绍的Seam业务流程管理方案以及第15章介绍的Seam的Spring整合。

附录A介绍了如何构建Seam和支持环境，并为你准备了与本书范例配套的源代码。

目标读者

有一位评论员说本书是“专家为专家写的”，所以如果你选择这本书是希望它的知识面足够宽，那你的选择是正确的。还有人说：“有经验的Seam开发人员也能从本书中有所收获。”还有一位说：“即便你已经是这方面技术的专家了，本书也不会让你感到失望的。”因此如果你想要掌握Seam，本书是必不可少的。

对于Seam的初学者来说，本书也不会令你失望。如果你是一位Seam新手，或者是管理者，那么前两章就可以让你受益良多。如果你想要进一步，就得问问自己：是否愿意学习这门技术，是否愿意为此付出努力。你准备好成为这方面的专家了吗？如果答案是否定的，你最好从Seam的参考文档或者入门书学起。当你想要更加细致地了解Seam，而那些资料已经满足不了你的需求时，我相信你会回头再来阅读本书。

如果你明白我的意思，记得要在开始阅读本书之前，先具备一些这方面的经验。我能够在书中深入细节，是因为已经省略了那些随处可见的基础知识。我希望你们至少具有Java开发经验，用过Java的Servlet API，并且部署到了应用服务器或者Servlet容器中。我假设你们至少了解JSF和ORM技术，因此会将这些内容一笔带过。你还应该了解方法拦截器及其工作原理，虽然这可以从文中推断出来。最后，如果你对书中的EJB 3集成或者Spring集成部分感兴趣，事先还需要具备这些技术的经验。这样听起来似乎需要具备很多前提条件，但是如果专注于此，也可以在阅读本书的过程中，从书中推荐的书籍和资源中学到这方面的知识。

不必担心你不懂JSF，下面会简单地做个介绍。如果你认为需要更多的解释，我还可以推荐一些其他的资料。不过，基础的JSF是很简单的，Seam中的JSF要比这复杂得多。

需要了解 JSF 的哪些方面才能使用 Seam

JSF是个面向组件的用户界面（UI）框架，与Struts这种基于动作的框架相反。Struts需要你编写一个定制的动作处理器来处理请求，然后将控制权转发到JSP页面，来渲染HTML响应。相反，JSF则自动地根据请求解析一个视图模板（通常是个JSP页面），并自动地让其与请求交互。没有前端控制器好像是一种倒退，但视图模板改进了这一问题。

JSF读取视图模板，包含定制的JSP或者Facelets标签，并构造一个UI组件树，有效地推迟了渲染过程。UI组件树是个层次状的Java对象图，代表页面的结构。渲染只是个次要的关注点，只有当组件树被“编码”到客户端（浏览器）时才会发生渲染。每个组件附带的渲染器会生成标记。

UI组件树的主要关注点是充当服务器端的视图表示法，并监听UI中发生的事件。组件树中的元素和页面中的元素之间是一对一的映射关系（文字型的HTML除外）。例如，如果页面包含带有输入域和按钮的表单，UI组件树中就会有相应的表单和嵌入它的输入组件和按钮组件。由于视图模板的处理与UI组件树的编码是分开的，你可以利用其他的视图技术构建组件树，例如Facelets或者纯Java。除了HTML之外，组件树也可以生成其他标记。

JSF的设计不仅仅是通过中间对象图将视图定义和视图渲染分开，它还利用组件树捕捉事件，并允许通过编程在服务器端操作视图。这与Swing很相似，但它是在Web环境的上下文中运作。用户执行的任何事件都会产生一条HTTP请求。在这个请求期间，或者回传（postback）期间，组

件树会从它的前一种状态中“恢复”过来。事件得到了处理，组件树再一次被编码到客户端（HTML响应）。

下面介绍一个事件机制的简单示例。用户点击JSF表单中的某一个按钮（比如UICommand组件）的时候，与该按钮动作绑定的方法就会得到执行。你不必担心如何处理请求或者如何准备映射关系。如果表单有输入（如UIInput组件），那些输入值就会被赋给与它们绑定的JavaBean属性。这样，当动作方法执行时，它就可以使用这些属性。与UI组件绑定的对象被称作managed bean。你在后面会知道，JSF在负责管理它们。

那么，managed bean又是如何与UI组件绑定的呢？这种绑定是利用EL（Expression Language，表达式语言）符号来完成的，JSP中也有EL符号。既有值绑定表达式，也有方法绑定表达式，但是后者是JSF所特有的。与JSP不同的是，JSF除了用值表达式输出属性值之外，还可以用它捕捉属性值。方法表达式被用来将方法绑定到UI组件上，因此当组件被激活时，就会调用该绑定方法。

在这个按钮范例中，managed bean中的方法通过表达式#{beanName.methodName}与按钮动作绑定起来。这个表达式解析成名为beanName的JSF managed bean的实例上的methodName()方法。managed bean在JSF描述符faces-config.xml中利用<managed-bean>元素进行定义。JSF会按需自动创建这些managed bean的实例。

虽然它们的作用截然不同，但值表达式与方法表达式类似。输入组件的值可以利用表达式#{beanName.propertyName}绑定到managed bean的一个属性上。点击按钮之后，当页面被渲染并把从输入组件中捕捉到的新值写到设值方法setPropertyName()中时，JSF从JavaBean取值方法getPropertyName()中读取该值。你仍然不必担心如何从HttpServletRequest对象中读取请求值。这项工作会自动完成，你只需去关心业务逻辑。

EL是JSF和Seam的一个重要部分，你一定要弄明白它。我推荐两个参考资料，一个是发表在java.net[①]上的文章*Unified Expression Language for JSP and JSF*，另一个是seamframework.org[②]上有关EL的常见问题解答（FAQ）。

刚才介绍的范例看起来简单到不能再简单了，但在每个JSF请求期间，尤其是在回传期间所发生的事情，会更复杂一些。每个请求都激活了JSF生命周期，这个生命周期包含了6个阶段：

(1) Restore View（恢复视图）；

(2) Apply Request Values（应用请求值）；

(3) Process Validations（and conversions）[处理验证（和对话）]；

(4) Update Model Values（更新模型值）；

(5) Invoke Application（调用应用程序）；

(6) Render Response（渲染响应）。

如果该请求是回传，UI组件树就会在恢复视图（Restore View）阶段恢复。如果这是一个初始请求，即URL是在浏览器的地址栏或者常规链接请求中得来的，生命周期就会直接跳到渲染响应（Render Response）阶段。

① http://today.java.net/pub/a/today/2006/03/07/unified-jsp-jsf-expression-language.html

② http://seamframework.org/Documentation/WhatIsAnExpressionLanguageEL

回传则贯穿于整个生命周期。在恢复视图之后的3个阶段中，表单值会被捕捉、转换和验证，并赋给managed bean上与这些值绑定的JavaBean属性。作为嵌套标签，或者与JSF描述符中的属性类型有关的输入组件会得到验证和对话。

调用应用程序（Invoke Application）阶段是执行动作方法的地方。最多可以有一个主动作和任意数量的次动作监听器。这两种类型的区别在于，只有主动作可以触发导航规则。导航规则也是在JSF描述符中定义，它规定了要渲染的下一个视图，并且调用应用程序阶段一完成，就会引用这些规则。

最后，在渲染响应（Render Response）阶段，UI组件树是用视图模板构建而成的，随后被顺序地编码到HTML（或其他输出），并且被发送到浏览器（或者客户端）。

JSF就是这么回事。如果你对这个框架还不熟悉，这个简单的解释也许还不足以满足你的好奇心。我推荐几个关于JSF的优秀资源，可以帮助你快速掌握它。如果你没有读过其他书籍，可以看看IBM developerWorks上的*JSF for nonbelievers*系列[①]。同时，建议你也看看*Facelets fits JSF like a glove*[②]这篇文章，了解一下Facelets，这是Seam应用程序使用的另一种视图技术。如果你愿意在JSF学习上做点投入，那就买本*JavaServer Faces in Action*（Manning，2004）或者*Pro JSF and Ajax*（Apress，2006）。在读这些资料的时候，记住你是通过学习JSF来了解如何使用Seam，因此不必深陷于JSF本身。在第3章，你会学到Seam为JSF带来的许多强化功能，那一定不会让你失望。

由于本书不少地方都引用了高尔夫球方面的知识，因此为了帮助你更好地理解相关内容，下面先介绍一些高尔夫球的背景知识。

高尔夫球游戏

高尔夫球的目标很简单，用最少的杆数将你的球入洞。一个标准的高尔夫球场，从发球区（称作tee box或简称tee）开始，会有18个这样的洞。每个洞都有一个标准杆数（par），规定了应用几杆让球进洞，这个数字在计算得分时非常重要。

“洞”（hole）既指地上的洞，又指发球区中配的洞。每个洞有固定数量的发球区，每个发球区用一种颜色标识。从洞到发球区之间设定了不同的距离，表示不同的经验等级，使得这种游戏对于那些高手来说更具挑战性。你选择一种颜色，并从每个洞中该颜色的指定区域开始。这些起始点就是你的tee set。在一轮高尔夫球比赛中，要按顺序击打你tee set中的每一个洞。

要让球前进，得使用一套高尔夫球杆。每支高尔夫球杆由一个球杆和一个杆头组成。杆头的角度决定了你击球时球的杆面角。一般来说，杆面角越小，球走得越远（注意这是需要技巧的）。为了击中球，你得将球杆像挥棒球棒一样挥起，千万别告诉职业高尔夫球手这是我说的。你用一种称作推杆（putter）的特殊球杆推动球穴区（洞周围的区域）的球。在使用推杆的时候，得轻轻地敲球，而不是朝着它挥杆。每次接触球时，无论你使用了哪种球杆，都计为一次击打。

在击球前，允许用高尔夫球座将球垫高，但是，只有在一个洞的第一次击球时才允许使用这

① http://www.ibm.com/developerworks/views/java/libraryview.jsp?sort_order=asc&sort_by=Date&search_by=nonbelievers%3A&search_flag=true

② http://www-128.ibm.com/developerworks/java/library/j-facelets/

种帮助。球座用于调节木杆的挥打角度，木杆是球袋中杆面角最小的球杆。一旦你在指定的洞上进行了第一次击打，就要用球杆让球前进，直到球入洞为止。然后捡起球走（或骑车）到下一个发球区。在一轮结束时，你把所有击球次数相加，计算出基本得分（此处不深入讲解“差点”（handicap）的概念，你只要知道它是用来计算得分的即可）。这个数字越小，表示打得越好。

我之所以在此选择高尔夫球作为范例，是因为它就像编程，极具挑战。在高尔夫球游戏中，只有更好。这听起来很像编程，不是吗？一旦我们掌握了某一种技术，其背后一定会有一些规则要学。幸运的是，有很多书籍可以助我们成为行业之中的佼佼者。

代码约定

本书提供了丰富的范例，包括Seam应用程序中需要创建的所有东西：Java代码、基于XML的描述符、Facelets模板和Java属性文件。代码清单或者文中的源代码会与普通的文字分开，以代码体显示。我会用粗体来强调范例中的重点。此外，Java的方法名称、Java类名、Seam组件名和上下文变量名、事件名、请求参数名、Java关键字、对象属性、EL表达式、Java 5注解和枚举常量、XML元素和属性以及文中的命令，也都将以固定宽度的字体显示。当注解出现在文中的时候，默认用@符号。

Java、XHTML和XML都可能很冗长。许多时候，初始源代码（可在线下载）都重定了格式。我添加了换行符，并重新进行了缩排，以适应书中可用的页面空间。在有些情况下，这么做还不够，如代码清单中还包括了“接上行”（➥）的符号。

我还应用了几种其他的空间优化方法。代码清单中已经省略了源代码中的注解，代码是在文中进行说明的。Java类中的类导入往往也要占用许多空间，因此当代码编辑器可以很容易地替你解析时，我就会将它们省略。源代码中可以看到完整的导入语句。当方法的实现不太重要，或者与之前的代码清单相比没有变动时，你就会看到{ ... }，这是个代码折叠。通常，为了节省空间，我会将Java 5注解与它们所应用到的属性或者方法放在同一行上。我个人更喜欢在自己的代码中在每个Java 5注释之后换行。

一些源代码清单会提供代码说明，强调一些重要的概念。有时候，有序项目符号会链接到代码清单后面的一些说明。

每个应用程序的位置会用一个变量符号在全书中进行引用。例如，JBoss应用服务器的路径就用${jboss.home}表示。

源代码下载

Seam是在LGPL许可下发布的一个开源项目。Seam发行版本的下载说明（包括源代码和二进制代码），可以在Seam的社区网站上找到：http://seamframework.org/Download/SeamDownloads。

本书中Open 18范例的源代码可以从这个网站（http://code.google.com/p/seaminaction）下载到，并且是在LGPL许可下发布的。由于Seam正在不断发展之中，我决定将这些源代码做成一个开源项目，以便根据需要实时更新代码。你也可以从出版社的网站（http://www.manning.com/SeaminAction）上下载书中范例的代码。关于如何使用源代码的详情请见源代码根目录下的

README.txt文件。

软件的组织结构

为了帮助你让软件保持有序，以便可以与源代码范例同步，我建议大家采用本书的目录结构。这只是我的个人建议，你有权决定最终要把文件放在哪里，这些惯例绝不是使用Seam的前提条件。

主目录

主目录（home directory）是放置个人文件的地方。该目录结尾部分的路径通常与你的用户名相同。本书假设主目录的用户名为twoputt，无论什么时候都必须引用绝对路径。表1展示了twoputt在几个不同操作系统中的主目录。每当你看到书中使用了twoputt的主目录时，用你自己的主目录将其替换掉即可。

表1 在几个不同操作系统上的主目录

操作系统	主 目 录
Linux	/home/twoputt
Mac OSX	/Users/twoputt
Windows	C:\Documents and Settings\twoputt

列表中包含的终端输出是在Linux系统中产生的，但你可以看得更远一点，因为你用哪种操作系统来开发Seam应用程序并没有什么差别。

主目录的结构

表2列出了我在开发时会建立的几个文件夹及其用途。你会在本书的源代码中见到这些路径。

表2 开发目录下的文件夹

文 件 夹	包含哪些内容
databases	基于文件的数据库和数据库Schema
lib	Seam中没有包含的JAR文件，如H2驱动程序
opt	Java应用程序，如JBoss AS和Seam
projects	开发项目

附录A介绍了如何安装在使用Seam及本书的范例时需要用到的软件，并且引用了这一结构。

作者在线

凡购买本书的读者，均可访问由Manning出版社维护的内部网上论坛，发表对本书的评论，询问技术问题，还可以得到作者及其他读者的帮助。要访问并订阅该论坛，请登录http://www.manning.com/SeaminAction。这个页面会告诉你注册后如何登录论坛、能够获得哪些帮助，以及论坛的相关行为准则。

Manning向读者保证，这个是读者之间、读者与作者之间进行有意义交流的平台。但并不承诺作者的参与程度，作者对“作者在线”的贡献是完全出于自愿（且不计报酬）的。我们建议你多向作者提一些具有挑战性的问题，以免作者对论坛失去兴趣！因为技术领域中的人都很忙碌，因此你的问题或许不能很快得到解答。建议你在Seam社区网站http://seamframework.org上发问，那里会有大量的人在阅读和回答与Seam相关的帖子。

只要本书仍在发行，读者就可以访问出版社网站上的作者在线论坛，也可以了解以前的讨论内容。

关于封面插图

本书封面上的插图名为La Béarnaise，她是一位来自法国西南部一个山区（即前Béarne省）的妇女。该图摘自1805年版各地服饰习俗的Sylvain Maréchal四卷大纲。每幅画都由手工精心绘制、着色而成。

Maréchal专辑的丰富色彩使我们想起了200年前世界上的各城镇与地区之间存在着多大的文化差异。人们彼此隔绝，操着不同的方言和语言。相隔几十公里的两个地区的人们，通过服装风格就能辨认出对方来自哪个地区。

从那以后，服饰风格已经发生了变化，那个时期富有浓郁地方特色的服饰文化也逐渐消逝了。现在已经很难通过服饰来区分不同地区的居民了。也许我们用文化多样性换来了更加多样化的个人生活——当然是更加多样化和更快节奏的科技生活。

当所有的计算机书籍都变得千篇一律的时候，Manning出版社则通过体现两个世纪以前丰富多彩的各地生活的Maréchal的图片作为封面插图，以这样的方式来赞美计算机行业中的独创性和主动性。