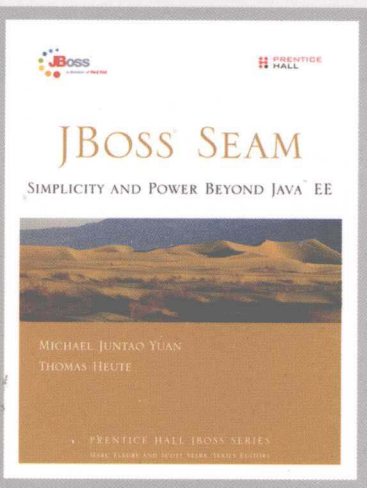


JBoss Seam Web 应用开发

[美] Michael Yuan Thomas Heute 著
王庆梅 徐杰 张辉 译



JBoss Seam: Simplicity and Power Beyond Java EE



JBoss Seam Web 应用开发

[美] Michael Yuan Thomas Heute 著
王庆梅 徐杰 张辉 译

人民邮电出版社
北京

图书在版编目 (CIP) 数据

JBoss Seam Web应用开发 / (美) 袁 (Yuan, M.),
(美) 霍伊特 (Heute, T.) 著; 王庆梅, 徐杰, 张辉译.
北京: 人民邮电出版社, 2009. 1
ISBN 978-7-115-19026-0

I. J… II. ①袁…②霍…③王…④徐…⑤张… III. JAVA
语言—程序设计 IV. TP312

中国版本图书馆CIP数据核字 (2008) 第161459号

版 权 声 明

Authorized translation from the English language edition, entitled JBoss Seam: Simplicity and Power Beyond Java EE, 9780131347960 by Michael Yuan, Thomas Heute, published by Pearson Education, Inc, publishing as Prentice Hall, Copyright © 2007 Pearson Education, Inc.

All rights reserved. No part of this book may be reproduced or transmitted in any form or by any means, electronic or mechanical, including photocopying, recording or by any information storage retrieval system, without permission from Pearson Education, Inc. CHINESE SIMPLIFIED language edition published by PEARSON EDUCATION ASIA LTD., and POSTS & TELECOMMUNICATIONS PRESS Copyright © 2008.

本书封面贴有 **Pearson Education** (培生教育出版集团) 激光防伪标签。无标签者不得销售。

JBoss Seam Web 应用开发

- ◆ 著 [美] Michael Yuan Thomas Heute
- 译 王庆梅 徐杰 张辉
- 责任编辑 李际
- 执行编辑 刘映欣
- ◆ 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>
- 三河市海波印务有限公司印刷
- ◆ 开本: 800×1000 1/16
- 印张: 19.5
- 字数: 432千字 2009年1月第1版
- 印数: 1-3000册 2009年1月河北第1次印刷

著作权合同登记号 图字: 01-2007-5293号

ISBN 978-7-115-19026-0/TP

定价: 42.00元

读者服务热线: (010)67132705 印装质量热线: (010)67129223

反盗版热线: (010)67171154

内 容 提 要

本书通过多个应用案例深入浅出地讲解了 JBoss Seam 的基本组件和各种功能，为程序员快速掌握 JBoss Seam 的开发方法提供了简单实用的途径。本书首先介绍了什么是 Seam，然后逐步深入地阐述了如何使用 Seam 进行应用程序开发、测试和产品部署以及异常处理，特别是书中关于如何使用实例程序作为模板开发自己的应用程序的内容，值得所有 JBoss Seam 开发人员借鉴，这点对于初学者尤为重要。

本书内容全面深刻，语言通俗易懂，可作为使用 JBoss Seam 开发 Web 应用程序的工具指导书，也可供初学者阅读参考。

关于本书

JBoss Seam在首次发行仅6个月后，就成为商业Java中最热门的构架之一，每个月的下载次数超过了10000次。Seam将标准的Java EE技术与几个非标准但很有趣的技术集成为一个具有统一规范的编程模型，这些技术包括JSF、EJB3.0、JPA、Hibernate、Facelets、jBPM、JBoss Rules (Drools)、iText等。Seam几乎可以运行在所有领先的Java应用程序服务器上，这些服务器有JBoss AS和Tomcat，但并不仅限于此。

本书是第一本由Seam团队的开发者所编写的综合指南，将介绍最新的Seam，解释设计背后的原理，讨论Seam中的替代技术。本书还根据作者的实践经验给出使用Seam的相关技巧和最好的实践方法。

当然，因为Seam快速更新的特性，几乎每个月就有新的Seam版本发布，本书将配合新的Seam版本。本书覆盖了Seam 1.2.0版本，在可预见的将来，Seam的新版本应该至少与1.2.0兼容，但对于那些想在该领域密切跟踪技术前沿的读者，我们在<http://www.michaelyuan.com/blog/>中开辟了有关本书的博客，给出Seam最新的更新，欢迎访问。

本书用一系列应用实例来说明如何编写 Seam 应用程序，这些例子应用程序的源代码可以从本书的网站 <http://www.michaelyuan.com/seam/>中下载。

致谢

首先，我们想感谢整个JBoss Seam团队的了不起的工作，和其他许多成功的开源项目一样，Seam也是大家共同努力的结果，同样，如果没有一个非常积极专注的用户群也不可能有这本书。我们从论坛、博客和邮件列表的Seam用户那里学到了很多。多谢了，朋友们！请再接再厉！

我们要把特别的感谢给予Gavin King——Seam的创始人和先驱开发者，如果没有他的远见、智慧和努力工作，就不可能有今天的Seam。Gavin从一开始就非常支持本书的写作，很耐心地解答我们这些新手遇到的很多问题，审查本书的内容，并在整个写作过程中一直给予我们鼓励。除了Gavin，其他Seam的开发者，包括Norman Richards、Emmanuel Bernard、Max Andersen、Shane Bryzak、James Williams、Christian Bauer以及Steve Ebersole也对本书的编写提供了很多帮助。Seam的确是本着开源精神和团队努力的结果。

我们也要非常感谢那些对审查本书初稿并给我们提出很好的反馈意见的人，他们是Ian White、Tony Herstell、Rich Rosen、Wes Boudville、Bil Lewis、Gregory Pierce、David Geary、Bruce Scharlau、Kito Mann、Daniel Brum以及Chris Mills。感谢他们所有的帮助！

编写本书的整个过程中，我们得到了Prentice Hall编辑队伍极其专业的大力支持。我们的编辑Greg Doench 和Kristy Hart容忍了我们多次的延期，并逐步引导我们走完了复杂的出版过程，没有他们的奉献就不可能有这本书。

最后也是最重要的，我们想感谢我们的家人，感谢他们的爱和支持。他们是我们所有成就背后真正的英雄。

关于作者

Michael Yuan是Red Hat公司的产品经理和技术推广者，主要工作包括Seam、JBoss应用程序服务器及其他中间件产品，还负责Seam项目中的代码，并在他的博客（<http://www.michaelyuan.com/blog/>）中编写Seam相关的文章。在加入JBoss前，Michael是移动端对端应用程序的软件顾问，在此期间，他出版了3本有关移动技术的书，包括*Enterprise J2ME* 和*Nokia Smartphone Hacks*。

Thomas Heute在完成pre-JBoss Portal项目后于2004年受雇于JBoss公司，他起初在JBoss Portal团队中做软件开发工作，接着在2005年成为JBoss Seam项目的共同负责人。他有着将EJB3.0向JSF靠近的远见。2006年年底Thomas回到JBoss Portal团队，继续履行他的职责，完成不同的任务。

目录

第 1 部分 入门

第 1 章 什么是 Seam	2	3.1.2 使用 Facelets 的 Hello World 实例	25
1.1 整合和强化 Java EE 框架	2	3.1.3 Facelets 用做模板引擎	26
1.2 一个为 ORM 设计的 Web 框架	3	3.1.4 数据列表组件	30
1.3 专为有状态 Web 应用而设计	4	3.2 Seam JSF 的增强功能	30
1.4 支持 Web 2.0	4	3.2.1 Seam UI 标签	30
1.5 依赖双向注入的 POJO 服务	5	3.2.2 Seam JSF EL 的增强功能	31
1.6 异常配置	5	3.2.3 随处使用 EL	33
1.7 避免滥用 XML	6	3.2.4 Seam 过滤器	33
1.8 为测试而设计	6	3.2.5 有状态的 JSF	33
1.9 卓越的工具支持	7	3.3 Facelets 和 Seam UI 支持的补充说明	33
1.10 开始编码	7	3.4 PDF、邮件和富文本	36
第 2 章 Seam Hello World	8	3.4.1 生成 PDF 报告	36
2.1 创建数据模型	9	3.4.2 基于模板的邮件	39
2.2 映射数据模型到 Web 表单	10	3.4.3 显示富文本	41
2.3 处理 Web 事件	11	第 4 章 应用程序快速开发工具	43
2.4 有关 Seam 编程模型的更多内容	12	4.1 前提条件	43
2.4.1 Seam POJO 组件	12	4.2 快速指南	44
2.4.2 Seam 的内置组件	13	4.2.1 创建 Seam Gen	44
2.4.3 易于测试	14	4.2.2 生成一个框架应用程序	47
2.4.4 基于 Getter/Setter 的双向注入	14	4.2.3 理解框架	48
2.4.5 避免过量的双向注入	15	4.2.4 开发应用程序	49
2.4.6 JSF 中的页面导航	16	4.2.5 创建和部署	50
2.4.7 通过 EntityManager 访问数据库	16	4.2.6 运行测试案例	50
2.5 配置和打包	17	4.3 与 IDEs 一起工作	51
2.5.1 WAR 文件	19	4.3.1 NetBeans	51
2.5.2 Seam 组件 JAR 包	21	4.3.2 Eclipse	53
2.6 为何如此简单	22	4.4 由数据库生成 CRUD 应用程序	54
第 3 章 推荐的 JSF 增强功能	23	第 2 部分 轻松生成有状态的应用程序	
3.1 Facelets 简介	24	第 5 章 有状态框架简介	58
3.1.1 为什么使用 Facelets	24	5.1 正确使用 ORM	58

5.2	更好的性能	60	10.3	触发确认动作	109
5.3	更好的浏览器导航支持	61	10.4	在 Web 表单中显示错误信息	112
5.4	更少的内存泄漏	62	10.5	使用 JSF 自定义的校验器	114
5.5	细粒度 (High Granularity) 组件生命周期	62	第 11 章	提供超链接功能的数据表	115
5.6	减少程式化编码	63	11.1	实现提供超链接功能的数据表	116
第 6 章	简单的有状态的应用程序	65	11.1.1	显示数据表	116
6.1	有状态组件	66	11.1.2	把选取对象注入事件处理器	117
6.1.1	有状态实体 Bean	67	11.1.3	在数据表中使用扩展的 EL	118
6.1.2	有状态会话 Bean	68	11.2	Seam 的数据绑定 (Data-Binding) 框架	119
6.1.3	有状态组件的生命周期	69	第 12 章	支持书签的 Web 页面	121
6.1.4	工厂方法模式	70	12.1	使用页面参数	122
6.2	页面导航流	71	12.2	以 Java 为中心 (Java-Centric) 的方法	125
第 7 章	对话	74	12.2.1	在 HTTP GET 请求中获取查询参数	125
7.1	默认的对话范围	74	12.2.2	为页面加载数据	126
7.2	长对话进程	77	12.2.3	对书签页面的进一步处理	128
7.2.1	定义长对话进程组件	81	第 13 章	Seam CRUD 应用程序构架	130
7.2.2	开始一个对话	81	13.1	数据访问对象	130
7.2.3	在对话内部	82	13.2	Seam 的 CRUD DAO 即为 POJO	131
7.2.4	结束对话	84	13.3	声明式 Seam DAO 组件	132
7.2.5	链接和按钮	86	13.3.1	使用实体对象的简称 (Simpler Names)	133
7.3	新前沿	88	13.3.2	实体对象的提取 (Retrieving) 和显示 (Displaying)	134
第 8 章	工作空间和并行对话	89	13.3.3	初始化一个新的实体实例	135
8.1	什么是工作空间	89	13.3.4	成功消息	135
8.2	工作空间切换器	92	13.4	查询	136
8.3	跨工作空间对话	94	13.4.1	动态查询	137
8.4	管理对话 ID	95	13.4.2	显示多页 (Multipage) 查询结果	139
第 9 章	事务	97	第 14 章	优雅地失败	142
9.1	管理事务	98	14.1	为什么不用标准的 Servlet 错误页面	143
9.2	强制事务回滚	99	14.2	设置异常过滤器 (Exception Filter)	144
9.2.1	通过可控异常回滚事务	99	14.3	注解异常	144
9.2.2	通过返回值回滚事务	100	14.4	为系统异常使用 pages.xml	146
9.3	原子级对话 (Web 事务)	101	14.5	调试信息页面	148
9.3.1	手动清除持久上下文	101	14.5.1	Facelets 的调试页面	148
9.3.2	一个对话对应一个事务	102	14.5.2	Seam 的调试页面	149
第 3 部分 集成 Web 和数据组件					
第 10 章	验证输入数据	106			
10.1	表单验证基础	106			
10.2	实体 Bean 上的校验标注	107			

第4部分 AJAX支持工具

第15章 用户和AJAX UI 组件	152
15.1 局部表单提交示例	153
15.2 自动补全文本输入的例子	155
15.3 联合使用ICEfaces和Seam	158
15.4 其他JSF 组件库	160
第16章 为现有组件启用AJAX	162
16.1 AJAX 验证框架的例子	162
16.2 可编程的AJAX	164
16.3 AJAX 按钮	167
16.4 AJAX 容器	168
16.5 其他组件	168
16.6 配置Ajax4jsf	169
16.7 利弊分析	171
第17章 直接集成JavaScript	172
17.1 AJAX 的名字验证实例(已重新加载)	172
17.1.1 服务器端组件	173
17.1.2 触发Web 页面的JavaScript 事件	174
17.1.3 产生一个AJAX 请求	175
17.2 AJAX 进度条	177
17.2.1 Seam 组件	178
17.2.2 通过JavaScript 访问Seam 组件	179
17.3 集成Dojo 工具	181
17.3.1 视觉效果	181
17.3.2 输入控件	183

第5部分 业务进程和规则

第18章 管理业务进程	188
18.1 jBPM 基础和有关词汇	188
18.2 应用程序用户和jBPM 的参与者	191
18.3 创建业务进程	192
18.3.1 定义进程	193
18.3.2 创建业务进程实例	194
18.3.3 在进程范围中绑定数据对象	195
18.4 管理任务	197
18.4.1 为任务实施业务逻辑	197
18.4.2 指定要执行的任务	199

18.4.3 在UI 中选择一个任务	200
18.5 jBPM 库和配置	202
第19章 有状态的页面流	205
19.1 pages.xml 中的状态化导航规则	205
19.2 把业务进程与Web 页面关联起来	208
19.3 页面流和有状态的对话	211
19.4 配置	212
第20章 基于规则的安全架构	214
20.1 认证(Authentication) 和用户角色(User Roles)	214
20.2 声明性访问控制(Declarative Access Control)	216
20.2.1 Web 页面	216
20.2.2 UI 组件	217
20.2.3 方法层(Method-Level) 的访问控制(Access Control)	218
20.3 基于规则的访问控制	219
20.3.1 简单的访问规则	219
20.3.2 基于实例的访问规则	221
20.3.3 配置JBoss 规则	222

第6部分 测试Seam 应用程序

第21章 单元测试	226
21.1 一个简单的TestNG 测试案例	227
21.2 模拟依赖性双向注入	229
21.3 模拟数据库和事务	230
21.4 加载测试设施	232
第22章 集成测试	235
22.1 完整的测试脚本	236
22.1.1 模拟JSF 交互过程	236
22.1.2 使用JSF EL 表达式	237
22.2 访问不带EL 的Seam 组件	238
22.2.1 获取Seam 组件	238
22.2.2 把数据与组件捆绑	239
22.2.3 触发UI 事件处理器方法	239
22.2.4 校验响应	239

第7部分 产品部署

第23章 部署Java EE 5.0	242
23.1 JBoss AS 4.0.5	242
23.2 JBoss AS 4.2.x 和 JBoss AS 5.x	242

23.3 GlassFish	243	26.4 配置持久化引擎	270
第 24 章 没有 EJB 3.0 的 Seam	247	第 27 章 性能优化与集群	272
24.1 带有 JPA 的 Seam POJO	248	27.1 单机服务器的性能优化	272
24.1.1 一个有关 Seam POJO 的例子	248	27.1.1 避免值调用	272
24.1.2 配置	250	27.1.2 JVM 选项	273
24.1.3 打包	252	27.1.3 减少日志记录	274
24.2 使用 Hibernate 的 POJO 和 API	254	27.1.4 优化 HTTP 线程池	275
24.2.1 使用 Hibernate API	254	27.1.5 在客户端和服务端状态存储 中的选择	276
24.2.2 配置	256	27.1.6 使用生产数据源	277
第 25 章 Tomcat 的部署	258	27.1.7 使用一个二级数据库高速 缓存	277
25.1 为 Tomcat 打包 POJO 应用程序	259	27.1.8 小心使用数据库事务	280
25.1.1 支持绑定 JAR	259	27.2 集群的可扩展性和故障转移	280
25.1.2 配置事务性数据源	261	27.2.1 粘 session 的负载均衡	281
25.1.3 引导 JBoss MicroContainer	263	27.2.2 状态复制	281
25.2 为 Tomcat 打包一个 EJB 3.0 应用 程序	263	27.2.3 Failover 架构	282
25.2.1 在 WAR 文件中捆绑必需的 JAR	264	附录 A 安装和部署 JBoss AS	283
25.2.2 捆绑 Embeddable EJB 3.0 配置 文件	265	A.1 要求 JDK 5.0 以上版本	283
25.2.3 引导 JBoss MicroContainer	265	A.2 安装 JBoss AS	283
25.2.4 应用其他数据源	266	A.3 部署和运行应用程序	286
第 26 章 使用生产数据库	268	附录 B 使用例子程序作为模板	287
26.1 安装并搭建数据库	268	B.1 基于 EJB 3.0 的简单 Web 应用程序	287
26.2 安装数据库驱动程序	269	B.2 基于 POJO 的 Web 应用程序	293
26.3 定义一个数据源	270	B.3 Tomcat 集群应用程序	299
		B.4 更多复杂的应用程序	300



第 1 部分

入 门

在这一部分，我们将介绍 **JBoss Seam** 的概况以及它的主要特点和优势，并通过一个简单的例子“**Hello World**”展示 **Seam** 如何将数据库、网络 UI 和事务性业务逻辑捆绑在一起，进而形成一个应用程序。还要讨论 **Seam** 和 **Facelets** 提供的 **JSF** 增强功能，该功能使得 **JSF** 成为最好的网络应用程序的构架之一，而且这点对 **Seam** 应用程序是至关重要的。对于那些不想把时间浪费在搭建公共 **Seam/Java EE** 配置文件的读者，我们介绍了一个叫做 **Seam Gen** 的工具，这种工具可以生成带有完整的 **Eclipse** 和 **NetBeans IDE** 支持的软件项目，这是快速开始 **Seam** 应用程序的最好方法。



第 1 章

什么是 Seam

根据 JBoss 的官方网站的介绍, JBoss Seam 是一个“为 Java EE 5.0 量身定制的轻量级的框架”。这是什么意思呢? 难道 Java EE (企业版) 5.0 本身不是一套“框架”吗? 为什么在官方规范之外还需要另外一个框架呢? 简单地说, 我们把 Seam 看成本应该包括在 Java EE 5.0 中的一个“遗漏的框架”。它位于 Java EE 5.0 框架的上层, 为所有在企业 Web 应用中的组件提供一个一致的、易于理解的编程模型。它同样使得基于状态的应用程序和业务流程驱动的应用程序的开发易如反掌。换句话说, Seam 的一切都是围绕开发者的开发效率和应用扩展性而存在的。

本书还将介绍 Seam 如何使开发变得容易, 并通过几个 Web 应用程序的例子进行阐明。但是在接触具体的代码例子之前, 首先解释 Seam 到底是做什么的, 其关键的设计原则有哪些。这样可有助于更好地通过贯穿本书的应用程序实例理解 Seam 是如何工作的。

1.1 整合和强化 Java EE 框架

Java EE 5.0 的核心框架是 EJB (Enterprise JavaBeans) 3.0 和 JSF (JavaServer Faces) 1.2。EJB 3.0 是在 Plain Old Java Objects (POJO) 的基础上为业务服务和数据库的持久化而开发的轻量级框架; JSF 则是为 Web 应用而开发的基于 Model-View-Controller (MVC) 组件的框架。大多数 Java EE 5.0 Web 应用都包含业务逻辑的 EJB 3.0 模块和 Web 应用前端显示的 JSF 模块。不过, 尽管 EJB 3.0 和 JSF 互为补充, 但它们还是根据各自的理念设计或独立的框架。例如, EJB 3.0 使用注解配置服务, 而 JSF 则用 XML 文件进行配置, 并且 EJB 3.0 和 JSF 组件在框架层面是互不敏感的, 要想使 EJB 3.0 和 JSF 一起工作, 就需要开发者手动构造 facade 对象 (如: JSF 支持 bean), 将业务组件与 Web 页面和样板代码 (又名管道代码, plumbing code) 进行绑定, 以便实现跨框架调用方法, 有效地把这些技术集成在一起是 Seam 的职责之一。

Seam 打破了 EJB 3.0 和 JSF 间的人为屏障, 为整合 EJB 3.0 和 JSF 提供了一个统一的、基于注解的解决方法。通过几个简单的注解, Seam 中的 EJB 3.0 业务组件能够直接用来支持 JSF Web 表单或直接处理 Web UI 事件。有了 Seam, 开发者就可以为所有的应用组件使用带有注解的 POJO。与基于其他 Web 框架开发的应用相比, Seam 应用在概念上更简洁,

为实现同样功能却需要少得多的代码（包括在 Java 和 XML 中）。如果你此时已经没有足够的耐心，或者急于弄明白一个 Seam 应用程序到底有多简单，可以看第 2 章中所描述的 Hello World 实例。

Seam 同样可使在 JSF 上“难以实现”的任务变得容易完成。例如，JSF 最头疼的一个问题就是过分依赖 HTTP POST，这使得为一个 JSF 网页添加书签后，通过 HTTP GET 访问该页面是相当困难的。但是有了 Seam，生成一个带标签的 RESTful 网页却非常简单（参见第 12 章）。Seam 提供一系列 JSF 组件标签和注解，增加了“Web 的友好性”，并提高了 JSF 应用程序中网页的效率。第 3 章将介绍不同的 Seam JSF 增强功能和为 JSF 而开发的 Facelets 可视化框架。接着在第 3 部分将讨论一些专题，如：端对端认证（参见第 10 章）和自定义的异常页面（参见第 14 章），在第 19 章中还将讨论如何通过 jBPM 业务进程改善 JSF 页面流。

同时，Seam 拓展了 EJB 3.0 到 POJO 的组件模式（参见第 2.4.1 节），并从 Web 层到业务层都有了状态上下文（参见第 5 章）。进一步说，Seam 把一系列其他主要的开源框架如 jBPM、JBoss Rules（又名 Drools）、iText 和 Spring 等整合在一起，这种整合并不是简单地把它们“有机结合在一起”，而是类似于整合 JSF 和 EJB 3.0 那样从框架层面进行了强化。

尽管 Seam 位于 Java EE 5.0 的底层，但其应用却并不局限于 Java EE 5.0 服务器，事实上，在本书中将展示如何在 J2EE 1.4 应用服务器（参见第 24 章）和普通 Tomcat 服务器（参见第 25 章）上部署 Seam 应用程序，这意味着现在能在 Seam 应用中得到产品化支持。

1+1>2

或许有这样一种误解，认为 Seam 只不过是把各种不同框架串在一起而得到的另外一个集成框架。Seam 提供其自身管理的状态上下文，这种功能允许 Seam 框架能够通过注解、Expression Language (EL) 表达式与其他框架进行深层次的整合，整合的程序来源于 Seam 开发者对第三方框架的认知，第 1.2 节是有关 Web 框架的实例。

1.2 一个为 ORM 设计的 Web 框架

对象关系映射（Object Relational Mapping, ORM）解决方案在当今企业应用中广泛使用。但是，大多数当前的业务和 Web 框架并不是为 ORM 设计的，它们并不在整个 Web 交互生命周期（从请求来临到响应完成）管理持久上下文。这必定导致包括令人望而生畏的 LazyInitializationException 异常在内的各种类型的 ORM 异常，也带来了如“数据传输对象（Data Transfer Object, DTO）”等丑陋的伎俩。

Gavin King 发明了 Seam，同时他也发明了目前在世界上广为使用的 ORM 解决方案——Hibernate。为了继承和发扬 ORM 的最佳实践，Seam 进行了重新设计，有了 Seam，就不再写 DTO，所要做的只是延迟加载。由于扩展后的持久上下文就如同一个自然的高速缓存，可以减少与数据库的交互，ORM 的性能就会被极大地改进。

进一步讲，因为 Seam 整合了 ORM 层、业务层和表示层，开发者就能够在表示层直接展

示 ORM 对象（参见第 11 章），也能把数据库验证注解用于输入表单（参见第 10 章），以及重新定向 ORM 例外到定制的错误页面（见第 14 章）。

1.3 专为有状态 Web 应用而设计

Seam 是专为有状态 Web 应用而设计的。Web 应用程序是天生的多用户应用程序，电子商务应用向来也是有状态的和有事务的。但是，大多数已有的 Web 应用框架是面向无状态应用的，开发者必须操作 HTTP 会话（session）对象对用户状态进行管理，与核心业务逻辑无关的代码不仅会混乱你的应用，而且引发一系列的性能问题。

Seam 中所有的基础应用组件本来就是有状态的，它们使用起来要比 HTTP session 容易，这是因为它们的状态是由 Seam 公开管理的，从而就没有必要在 Seam 应用程序中编写会引起麻烦的状态管理代码，而只需在其组件上注解其作用域、生命周期方法以及其他状态属性，Seam 就会掌管其他方面（译者注：指这些组件的生命周期）。Seam 状态组件要比 HTTP 会话（session）能更好地管理用户状态，例如，可以有多个会话同时进行，每个会话由一个 HTTP 会话（session）中的一系列 Web 请求和业务方法调用组成。有关 Seam 的有状态组件的更多知识请参考第 5 章。

进一步说，Seam 中的数据库缓存和事务能自动与应用程序的状态相关联，Seam 在内存中自动保存数据库的更新，只有等到对话结束后才提交到数据库，内存中的缓存能大大减轻复杂状态应用程序中数据库的负载。

除了以上这些，Seam 支持整合开源 JBoss jBPM 业务程序引擎，大大提升了 Web 应用中的状态管理。你现在可以为一个机构中的不同工作人员（诸如客户、经理、技术支持人员等）指定工作流程，利用工作流程来驱动应用程序，而不是依赖用户界面事件处理和数据库进行驱动。

□ 声明性上下文组件

Seam 中的每个状态组件都有一个作用域或者相关的上下文，例如购物车组件在购物会话之初创建，当所有事项检测完毕并结束会话时组件被注销。所以这个组件存活于一个会话上下文。应用的上下文是通过对组件注解进行简单声明的，Seam 可由此自动管理组件的创建、状态和删除等操作。

Seam 提供几个层面的状态上下文，范围涵盖了从单个 Web 请求到多页面对话，到 HTTP 会话，甚至长业务流程。

1.4 支持 Web 2.0

Seam 为 Web 2.0 应用进行了充分的优化，它为 AJAX（异步 JavaScript 和 XML，增加页面交互的一种技术）提供多种支持——从内置“零 JavaScript”的 AJAX 组件（参见第 15 章）到有 AJAX 支持的 JSF 组件（见第 16 章），再到定制的 JavaScript 库（见第 17 章），Seam 为

浏览器端的 Javascript 对象提供了直接访问 Seam 服务器组件的途径，实质上提供了一个先进的并发模型，有效地管理来自同一用户的多个 AJAX 请求。

对于 AJAX 应用程序，不断增长的数据库负载是一个巨大的挑战。与非 AJAX 应用程序相比，一个 AJAX 应用程序要向服务器发送更频繁的请求，一旦数据库必须对所有这些 AJAX 请求做出响应，数据库就会不堪重负。Seam 中的有状态持久上下文正如一个内存中的缓存，它能在整个长会话过程中保存信息，最终帮助减少数据库的交互。

Web 2.0 应用程序往往为其数据使用复杂的关系模型（例如，一个社会性网络站点所做的就是处理和显示“用户”之间的关系），对于这些站点，延迟加载对 ORM 层是至关重要的，否则，一个简单的查询就能级联地加载整个数据库。正如前面所讨论过的，Seam 是现今唯一能正确支持 Web 应用程序延时加载的 Web 框架。

1.5 依赖双向注入的 POJO 服务

Seam 是一个“轻量级”框架，这是因为它使用了 POJO（Plain Old Java Objects）作为服务组件。在应用程序中，POJO 没有使用框架接口或抽象类来“钩住”组件，理所当然的问题就是，这些 POJO 间如何进行交互以形成一个应用？它们与容器服务（例如，数据库持久化服务）又是如何交互的？

Seam 通过使用一个广泛应用的、被称作依赖注入（DI）的设计模式联结所有 POJO 组件。在这种模式下，Seam 框架管理着所有组件的生命周期，当一个组件需要使用另外一个组件时，它通过注解（annotation）向 Seam 声明此依赖，Seam 依据应用程序的当前状态得到这个依赖组件，并将其注入提出需求的组件中。

通过拓展依赖注入的概念，一个 Seam 组件 A 不但可以构造另外一个组件 B，而且还可以把此组件 B“抛还”给 Seam 以备其他组件（例如组件 C）调用。

这种类型的双向依赖管理甚至都被广泛应用于简单的 Seam Web 应用程序中（例如第 2 章的 Hello World 一例），在 Seam 术语中，我们称之为“依赖双向注入”。

1.6 异常配置

使 Seam 如此易用的关键设计原则是“异常配置”，其思路是为组件提供一系列的默认行为，开发者只需要在预期行为是非默认行为时明确地配置组件。例如，当 Seam 将组件 A 作为属性注入组件 B 时，组件 A 就会默认地以组件 B 被注入的属性名称进行命名。Seam 中还有很多类似的细节，总的结论是在 Seam 中配置元数据要比在其他 Java 框架中简单得多。因此，大多数 Seam 应用程序通过一些简单的 Java 注解进行配置就足够了。配置的复杂性大大减少，使开发者受益匪浅。最后，与其他 Java 框架相比，Seam 应用程序能够用更少的代码实现同样的功能。

1.7 避免滥用 XML

或许你已经注意到，Java 注解在表述和管理 Seam 配置元数据时扮演着重要的角色，这样的设计使框架更易于操作。

在 J2EE 发展初期，XML 曾经被看作配置管理的“圣杯”。框架设计者把包括 Java 类和方法名称在内的所有配置信息都统统丢进 XML 文档，而不考虑为开发者带来的影响。反省之后，发现这是一个严重的错误，XML 配置文档太过重复，开发者必须重复代码中已有的信息，从而将配置和代码联结起来，这些重复使应用程序易于出错（例如，一个拼写错误的类名可能在运行时显示为一个难以调试的错误）。缺少合理的默认配置使这一问题进一步复杂化，事实上，在一些框架中，相当数量的样板代码被伪装成 XML，这可能相当于甚至超过实际应用程序中 Java 代码的数量。对于 Java 开发者而言，XML 的滥用通常被称为 J2EE 的“XML 地狱”。

企业版的 Java 社区意识到了 XML 滥用的问题，并且已经非常成功地用 Java 代码中的注解取代了 XML。EJB 3.0 是官方的 Java 标准化机构为促进企业版 Java 组件使用注解而形成的一项成果。EJB 3.0 使 XML 文档完全成为可选择的，从而向正确方向迈出了积极的一步，Seam 增加了 EJB 3.0 的注解，把基于注解的编程模型拓展到整个 Web 应用程序。

当然，XML 对于配置数据并非完全不利，Seam 的设计者意识到 XML 非常适用于特定的页面流或者定义业务工作流程的 Web 应用程序。XML 文档使开发者集中地管理整个 Web 应用程序的工作流程成为可能，同时也反对将配置信息分散于 Java 源文件中。工作流很少能与源代码耦合，因此 XML 文档中并不需要重复键入已存在于代码中的信息。有关这一主题的更多信息请参考第 19 章。

1.8 为测试而设计

Seam 是为简易测试而重新设计的，因为所有的 Seam 组件都是注解过的 POJO，所以便于进行单元测试。开发者只需使用常规的 Java new 关键词创建 POJO 实例，然后在测试框架（如 JUnit 或 TestNG）中运行任何方法即可。如果需要测试多种 Seam 组件间的交互，可以先分别实例化这些组件，然后手动设置它们之间的相互关系（也就是显示地使用 setter 方法，而不是依靠 Seam 的依赖注入功能）。在第 21 章的单元测试中，我们将解释如何为 Seam 应用程序建立单元测试，以及如何为测试实例建立数据库服务模型。

Seam 中的集成测试甚至比单元测试还要简单，在 Seam 测试框架下，可以通过编写简单的脚本模拟实现 Web 用户的交互，并对输出结果进行测试。在编写测试脚本时只需使用 JSF Expression Language (EL) 表达式命令 Seam 组件，正如在 JSF Web 页面中所做的那样。与单元测试类似，集成测试也直接从 Java SE 环境下的命令行运行，而不必为测试特意启动应用服务器。有关集成测试的更多细节请参考第 22 章。