深度整合指南

黄睿 编著

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY
http://www.phei.com.cn

Spring深度整合指南

黄 睿 编著

電子工業出版社
Publishing House of Electronics Industry
北京 • BEIJING

内 容 简 介

Spring是当今J2EE世界中最受欢迎的轻量级框架，通过Spring我们可以实现原来只有重量级框架才能够实现的功能。本书针对J2EE应用程序开发中的一些经典场景，重点向读者介绍了如何实现Spring的应用程序构建。内容包含Spring的IoC容器、Spring AOP框架、Spring JDBC支持、Spring Hibernate支持、Spring iBATIS支持、Spring的事务支持、Spring的Web框架、在Spring中支持的部分J2EE服务以及如何实现测试。

通过本书的学习，读者可以快速找到利用Spring实现一般应用程序的捷径，并且能够根据Spring所倡导的思考方法，实现结构更为优良的应用程序。

本书适用于所有J2EE程序员、Spring使用者以及研究者。

未经许可，不得以任何方式复制或抄袭本书之部分或全部内容。
版权所有，侵权必究。

图书在版编目（CIP）数据

Spring深度整合指南/黄睿编著.—北京：电子工业出版社，2007.1
ISBN 7-121-03478-6

Ⅰ. S… Ⅱ. 黄… Ⅲ. Java语言—程序设计 Ⅳ. TP312

中国版本图书馆CIP数据核字（2006）第135772号

责任编辑：徐云鹏
特约编辑：卢国俊
印　　刷：北京天竺颖华印刷厂
装　　订：三河市金马印装有限公司
出版发行：电子工业出版社
　　北京市海淀区万寿路173信箱　邮编：100036
　　北京市海淀区翠微东里甲2号　邮编：100036
开　　本：787×1092　1/16　印张：18.375　字数：470千字
印　　次：2007年1月第1次印刷
定　　价：28.00元

凡所购买电子工业出版社图书有缺损问题，请向购买书店调换。若书店售缺，请与本社发行部联系，联系电话：（010）68279077。邮购电话：（010）88254888。
质量投诉请发邮件至zlts@phei.com.cn，盗版侵权举报请发邮件至dbqq@phei.com.cn。
服务热线：（010）88258888。

前　言

Spring的出现是J2EE世界中发生的大事，Spring出现的目的并不是要替代J2EE，而是让J2EE变得更加方便和容易。Spring是一个开源的框架，任何人都可以无偿地获得并使用它，Spring可以整合J2EE中使用的大多数开源框架。Spring的出现并不是偶然的，这是因为J2EE的某些部分在开发过程中确实相当费力不讨好，例如EJB，有的时候我们使用EJB仅仅是希望能够使用它的声明式事务编程的能力，但是我们要想搭建这样的一个平台，需要花费很大的精力，更不必说测试和后期维护的成本了。Spring的出现就是希望通过使用轻量级的框架实现原来重量级框架才能完成的任务。Spring不但改变了J2EE程序员的想法，也改变了J2EE本身，这一点上我们可以从EJB 3.0中看到。

Spring的组成是以它的IoC容器为核心的，与针对DAO、AOP、Web、安全以及其他J2EE服务的功能模块共同组成了Spring现在的框架。本书中Spring的版本是1.2.8，2.0版本目前已经发布，但书中的概念、案例均适用于2.0版本。

在本书中我们考虑的是如何将Spring众多的功能和J2EE程序员的编程实际结合起来，重点介绍那些我们最关心的内容，而没有介绍在Spring中使用并不广泛的框架。我们在本书中重点介绍的内容包括：Spring IoC框架、Spring AOP框架、Spring DAO支持以及Spring的Web层支持，我们特别在第11章介绍了Spring对测试的支持以及一些TDD的知识，相信这部分内容对程序员和项目管理人员都有一定的帮助。

具体而言，在本书中我们介绍了下面的内容：

在第1章中，回顾了J2EE中传统的开发方式，并介绍了Spring的原理以及Spring可以在什么地方帮助我们完成更好的程序。除此之外，我们还介绍了在进行Spring开发之前应当做的准备工作，并使用一个小例子作为结束。

在第2章中介绍了IoC与Dependency Injection的原理、如何在Spring容器中对对象进行配置和绑定、“依赖性”解析原理、如何在Spring容器中对对象进行生存周期管理、对服务和资源的访问的抽象、后处理BeanFactory和ApplicationContext接口的实际应用。

在第3章中介绍了Spring AOP框架的基本内容，特别是术语、各种Advice以及ProxyBeanFactory。关于ProxyBeanFactory的各种不同形态，我们将在后续章节中有所了解。

在第4章中介绍的是Spring的JDBC支持，包括如何通过Spring的JDBC框架连接到数据库、查询数据，以及使用存储过程等方面的内容。

第5章中介绍了Spring的事务支持，以及如何使用Spring事务框架进行编程事务或声明事务。

在第6章中介绍了Hibernate支持。我们首先简单介绍了什么是O/R Mapping框架，然后介绍了如何在Spring中实现Hibenate的操作，并实现Hibernate事务。

在第7章中介绍的是iBATIS——另外一种非常有效的O/R Mapping框架，同样介绍了如何实现相同的SQL操作的方法，以及如何在iBATIS中实现事务。

在第8章中介绍了Spring的Web框架，即如何通过Spring的MVC模型来将日常的Web应用程序开发。

在第9章中介绍了如何使用日程框架来使应用程序自动运行。

在第10章中介绍了如何使用JavaMail API以及Spring所提供的mail支持在Spring的应用程序中发送电子邮件。

在第11章中介绍了测试的相关概念，并介绍了Mock对象在测试中的作用，其中包括，如何编写自己的Mock对象以及使用Mock框架——easyMock来自动化Mock的生成。最后还介绍了Spring所提供的一些重要的Mock对象。

在第12章中介绍了如何将前面的内容综合起来开发一个论坛的应用程序。我们从需求的分析开始，结合Spring的相关框架，向读者介绍了一个较为完整的Spring应用程序开发过程。

本书由黄睿编著，马然、吴振荣、马忠颖、张慧丽、王军、王冬梅、龚蕊、张慧霞和董立强等同志在本书的编写过程中提供了很大的帮助，在此表示感谢。同时，要感谢电子工业出版社和北京美迪亚电子信息有限公司老师们的辛勤工作，没有你们，就不会有本书的出版。

由于编者水平有限，错误在所难免，希望读者能够批评指正。

为方便读者阅读，本书配套资料请登录“华信教育资源网”（http://www.hxedu.com.cn），在“教学资源”频道的“综合资源下载”栏目下载。

此为试读，需要完整PDF请访问：www.ertongbook.com

反侵权盗版声明

电子工业出版社依法对本作品享有专有出版权。任何未经权利人书面许可，复制、销售或通过信息网络传播本作品的行为；歪曲、篡改、剽窃本作品的行为，均违反《中华人民共和国著作权法》，其行为人应承担相应的民事责任和行政责任，构成犯罪的，将被依法追究刑事责任。

为了维护市场秩序，保护权利人的合法权益，我社将依法查处和打击侵权盗版的单位和个人。欢迎社会各界人士积极举报侵权盗版行为，本社将奖励举报有功人员，并保证举报人的信息不被泄露。

举报电话：（010）88254396；（010）88258888
传　　真：（010）88254397
E-mail：　dbqq@phei.com.cn
通信地址：北京万寿路173信箱
　　电子工业出版社总编办公室
邮　　编：100036

目　录

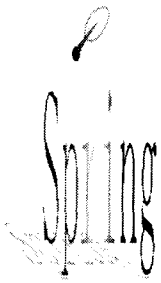
第1部分　Spring核心技术

第2部分 Spring数据层

第3部分 Web层应用

第4部分 Spring提供的其他服务

第1部分 Spring核心技术



第1章 Spring入门

本章内容概述

在本章中，我们将会了解到下面的内容：

* Spring与J2EE
* Spring Project
* 获取Spring框架
* 理解Inversion of Control（IoC）
* 实现面向方面的编程
* 其他轻量级框架
* 搭建Spring工作环境

Spring的一切种种其实都是关于JavaBean的讨论，在本书中我们将JavaBean简称为“Bean”。

Sun公司于1996年发布了JavaBean的1.0标准，从此Java语言踏上了模块化编程的起点。1.0标准只是约定了最简单的Java对象的定义，允许Java对象的重新使用，这样程序员就可以通过“指挥”这些组件，创建出复杂的大型应用程序。而随着Java应用不断地在企业及开发市场中攻城略地，简单的JavaBean模型已经无法满足企业级的需求，例如安全性、事务服务以及分布式计算等；这样，Enterprise JavaBean（EJB）就在这样的呼声中于1998年和我们见面了。EJB的主要任务就是提供企业级的服务，但同时由于其在诞生之初只考虑到如何解决问题，却背离了JavaBean的简单原则。熟悉EJB的程序员都应该知道，尽管EJB提供了声明式的编程模型，但大多数的EJB应用程序都需要我们编写大量的描述文件（XML Descriptor）和其他大量的辅助代码，由此造成程序难于编写并且难于测试。

EJB失败的地方恰恰是很多新的技术手段成功的地方，为了解决EJB所带来的困扰，很多技术和产品如雨后春笋般出现了。其中最为成功的就是IoC技术和AOP技术。它们都可以使用EJB中为大家所称道的声明式编程模型，同时避免了编写大量的令人头疼的EJB辅助代码。JavaBean模型的进化可以用图1-1来表示。

而Spring就是IoC容器产品中较为成功的（关于IoC和AOP技术我们将在后面的章节中详细进行叙述）。

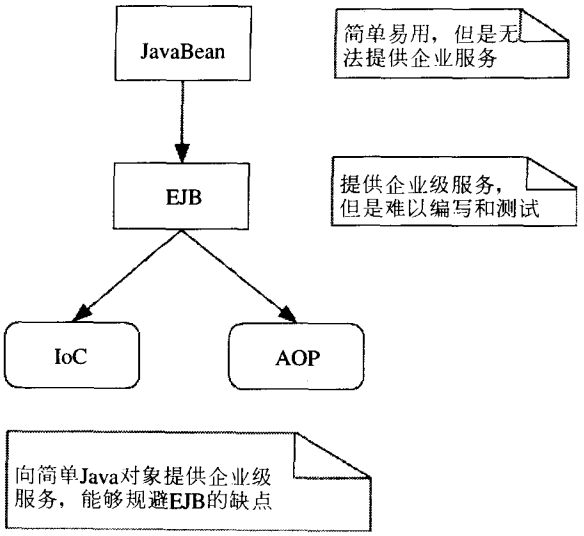


图1-1 JavaBean模型进化

1.1 Spring与J2EE

Spring是一个开源应用程序框架，其目的是简化传统的J2EE开发过程。对于一个传统的J2EE 应用程序开发人员来说，Spring带来的东西应该说是令人兴奋的。我们可以迅速开发高质量的应用程序，而其代价只是Spring的学习曲线。

需要注意的是，Spring是一个“应用程序框架”，而不是像Struct或Hibernate等解决特定问题的框架。Spring的重要作用是使用一致、高效的方法开发应用程序，并整合各种优秀的框架，如Struct或Hibernate，使之能够在统一框架下开发优秀的应用程序。

要想了解Spring的作用，我们首先看看传统的J2EE开发中都存在什么样的问题。

1.1.1 传统J2EE开发方式的问题

1999年～2000年是J2EE发展最快的时代，在此期间，J2EE得到了极大的发展，特别是在中间件的领域中产生了很多产品和标准，如事务管理等概念就是从那个时代开始兴起的。但是随着时代的发展，尽管J2EE标准和产品都在不断演化，但是J2EE产品也出现了很多问题，例如，大多数J2EE项目都变得非常复杂，难于管理而且不利于进行单元测试，在性能方面也不尽如人意。

下面我们来看看J2EE中增加应用程序复杂性的源头和其他问题。

- J2EE应用程序中充斥着大量的重复代码。这样的代码包括JNDI查找、对象传输或者是大量的try/catch/finnaly语句用于获得并释放资源，编写这些代码不但浪费资源，同时无法将我们的主要精力放在业务逻辑上，而这些无用的重复代码我们都可以用Spring提供的服务避免。
- 大多数应用程序倾向于使用分布式对象模型。这是产生冗余代码的一个主要根源，而且在大多数情况下，这并不是一种正确的思维方式，分布式对象模型不但复杂，而且在性能上完全没有优势。当然在进行应用程序设计时，我们有时必须使用分布式对象

此为试读，需要完整PDF请访问：www.ertongbook.com

模型，尽管这时我们可以使用Spring提供的服务有效地进行分布式对象的管理，但是我们还是应当慎重抉择。

- EJB组件模型过于复杂。EJB的初衷是减少J2EE应用程序中实现逻辑层的复杂性。但是EJB在这方面并没有成功。
- EJB被过度使用。EJB的最初设计目标是实现分布型、事务型的应用程序，尽管所有大型的应用程序都需要事务，分布性却不应当作为对象模型的基础部分进行设计。这样就导致了不管是否需要分布性，我们都必须考虑到EJB所内置的分布式组件模型。
- 大多数J2EE设计模式并非真正的设计模式，而是为了解决某些技术限制的变通之法。过分使用分布式应用程序，以及使用复杂的API都是这些设计模式之源。我们应当不断寻找替代的方案。
- J2EE应用程序不易进行单元测试。J2EE API特别是EJB组件模型是在敏捷开发模式出现之前定义的，所以在设计时并没有充分考虑到组件的单元测试问题。在实际操作中，非常难以在应用程序服务器之外测试使用J2EE API和EJB API的应用程序，而独立于应用程序服务器对应用程序进行单元测试可以提高测试覆盖面，同时还可以模拟许多异常情况，如连接中断等。同时保证测试的时间尽可能地短也是非常重要的。
- 某些J2EE技术根本就是不成功的，如实体Bean技术。实体Bean在生产性上效率非常低，同时它还在实现面向对象编程方面有很大的限制。

这些问题并不是今天才发现的，从问题出现的第一天起，开发者们就开始寻找各种解决方案，针对不同的问题出现了相当多的工具以解决特定的问题，如我们可以使用XDoclet来进行代码的自动生成，通过使用这些工具，我们可能避免了某些问题的产生，同时又会带来新的问题。

这就是工具的局限性，它们针对某个问题并解决问题，大多数都会伴随其他问题。经过了众多开发者的努力，我们发现使用框架是一种更好的解决问题的方法。传统的代码生成的方法过于僵硬，而框架则可以在运行时动态地处理。框架的出现是以开发者的视角为出发点，让开发者可以在充分利用平台能量的同时能够给开发者一个简单的、高生产率并且高度抽象的平台。而框架的作用不仅仅体现在开发中，同时也体现在维护和排错时。想像一下通过自动生成得到的代码的复杂性，以及我们在调试和重写此类代码时的工作量，一切就会一目了然。

让我们进一步来明确框架的作用。J2EE给我们提供底层基础结构的标准访问方式，但是它并没有给我们提供简单的访问这些功能的方式，而J2EE留下的空白就是如Spring等框架大展拳脚的地方。

当大多数开发者和公司认识到框架的重要性之后，他们都开始尝试编写自己的框架产品。这些产品只有小部分取得了一定的成功，解决了一些问题，但是维护这些框架的成本也就立即显现出来了，而大多数的框架产品都可以认为是失败的。因为框架所提供的解决方案是针对一般性问题的，所以我们应当采用主流的框架产品，最好是采用开源的框架产品，这样我们才能够在后期根据自己的需要对框架进行扩展。

Spring就是主流开源框架中的一种，根据设计者的初衷，Spring的其重要精髓在于能够使POJO参与到企业服务当中，而不是必须使用EJB。这样我们在设计应用程序的时候就能够更加灵活，同时使应用程序能够更加符合OOP的要求。

Struct是一种专门用于Web端的框架，而Hibernate则是用于持久层的框架，它们的特点是解决了经典3层开发结构中（表示层、业务逻辑层和数据层）特点层面的问题，而Spring则是一种应用程序框架，其作用贯穿于整个应用程序的各个层面当中。

1.1.2 轻量级框架

我们已经提到过，Spring是一种轻量级框架，那么所对应的“重量级”框架又是什么呢？为了弄清轻量级框架的优点，我们首先来看一下所谓“重量级”框架——EJB在J2EE开发过程当中的实现。

EJB提供了一种被普遍认可的访问企业服务的方法，其方式是要求开发者遵从其代码结构。那么要实现EJB，我们最少要编写3～4个Java类；为每一个EJB Jar文件提供2个详细的部署描述文件；而且我们还需要编写大量访问EJB的客户端代码以及EJB访问环境的代码。显而易见，EJB的编写非常烦琐而且容易出错，更重要的，现在我们提倡使用TDD的开发方式，但是EJB组件模型非常不利于单元测试，而且在很多应用中，EJB都没有能够实现其设计目标。到EJB 2.1的时候情况还是没有改善，EJB专家组已经意识到了这个问题，并准备在EJB 3.0中对其进行大修，但是时间不等人，我们不可能被动地等待EJB 3.0的到来，使用Spring就完全可以避免EJB的失败。

在J2EE的早期，还有其他框架出现，它们都在某种程度上取得了成功，但是无一例外地都要求应用程序代码使用框架所提供的API，而且要求开发者改变其原有的编程方式。事实上，一个好的框架应当最大限度地做到提供一个良好的编程模型而不改变开发者的思考方式，也就是说，能够让开发者将主要精力放在对需求的思考上而不是对所采用的编程模型的思考上。用下面这句话可以概括一个好的框架所应该体现的价值：It should make the right thing easy to do。也就是说，框架应当鼓励好的编程习惯和正确的思考方式，如OOP和AOP的思想。

重量级框架所表现出来的与J2EE开发中的种种不适应被众多的开发者在开发中发现并改进，也有相当多的轻量级框架应运而生。这些框架产品的目标是鼓励用户实现良好的编程思路，让开发者关注业务逻辑，而不是框架本身。这些轻量级框架的首要目标是让用户使用POJO作为工作的中心，而不是例如EJB的特殊对象。

就像其名称所暗示的那样，轻量级框架可以降低应用程序的复杂性。而且轻量级框架的启动时间、依赖性以及可测试性都可以满足开发者的要求。

轻量级框架在很大程度上受到了XP开发方法的启发，但是在编程实践中并没有强迫用户实现XP编程方法。但是，我们需要了解XP编程方法的核心内容，即：

- Communication
- Simplicity
- Feedback
- Courage

特别是其中的Simplicity，使事情变得简单，其核心内容是将复杂的问题抽象为可以独立管理的小块，并且逐步地去实现，在每一个实现步骤中都要保证严格的测试。关于XP编程方法的详细内容可能需要一整本书的篇幅去讲解，我们在这里只需要了解XP编程方法的大概思

想就可以了。需要说明的是，XP方法并不是一种严格的约束，它只是一些基本的理念，现在有很多工具和产品都可以作为实现XP编程方法的手段。

1.2 Spring Project

Spring起源于Rod Johnson所编著的*Expert One-to-One J2EE Design and Development*一书，在该书中，Rod Johnson第一次提出了Spring框架的基础——Interface 21框架。在此基础上，Spring于2004年3月24日推出了1.0版本，到今天为止，一共有三个主要版本，即1.0、1.1和1.2。

尽管Spring是完全由Java编写并且面向Java世界的，但Spring并没有否定.NET世界，因此，Spring还包含一个专门为.NET定制的框架产品，这两个Spring产品在本质上差别不大，读者可以访问www.springframework.net获得更多的信息。

1.2.1 Spring Rich Client平台

Spring Rich Client平台是Spring项目中的一个子项目，该项目通过使用基础的Spring概念，向丰富客户端开发的应用程序提供基础的类库。该项目现在还处于早期开发阶段，但是已经具备了相当多的成熟的功能。有关信息参见Spring官方网站中的信息。

1.2.2 Spring IDE

Spring IDE产品主要是以主流IDE插件的形式出现的。其主要目的是帮助用户设置Spring配置文件，在开发早期指出可能存在的问题，并加快开发的周期。

1.2.3 Acegi安全性系统

Acegi安全性系统也是Spring的子项目之一，其主要功能是给J2EE应用程序提供全面的安全性解决方案，其中包括多个验证后端、单点登录支持以及缓存。在本书的后续章节中，我们将对此进行详细说明。

1.3 获取Spring框架

如何获得Spring是我们本节中要关注的重点。

（1）首先，在浏览器中输入www.springframework.org，打开如图1-2所示的网站。

（2）在左侧的Navigation栏中，打开download链接，接下来，在图1-3中箭头所指的地方单击，表示我们需要下载Spring的最新版本。

（3）在下面的页面中，我们需要选择希望下载的文件。其中，上面的spring-framework-1.2.7-with-dependencies.zip代表的是包含一些非Spring类库的完全版本，而下面的spring-framework-1.2.7.zip则代表纯粹的Spring发布版本，但是我们仍然建议大家下载前面的版本，如图1-4所示。

（4）在接下来的页面中，选择相应的镜像（推荐使用欧洲的镜像，速度较快），然后单击download开始下载，如图1-5所示。

图1-2 Spring官方网站

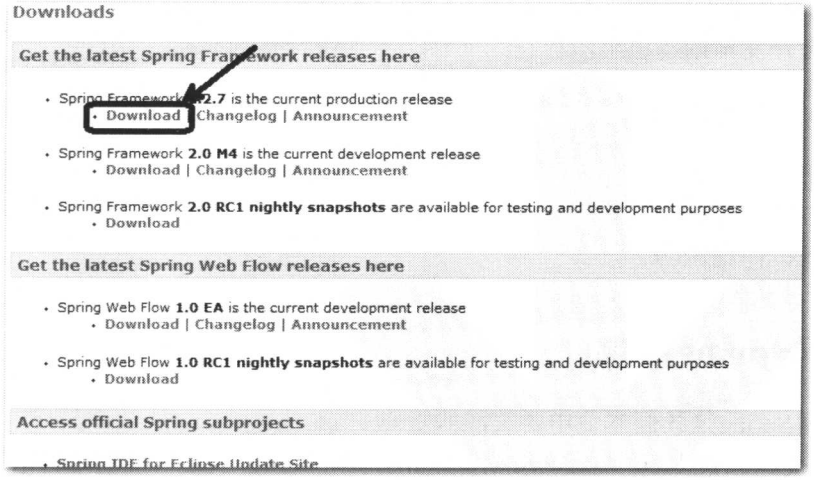


图1-3 选择最新版本下载

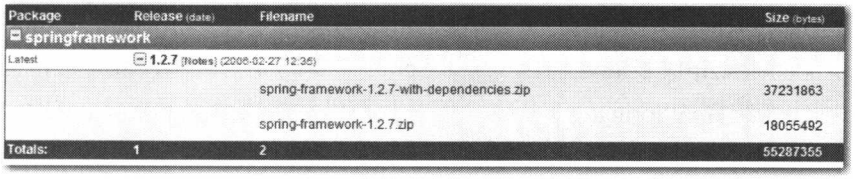


图1-4 选择Spring版本

Host	Location	Continent	Download
INTERNAP	Atlanta, GA	North America	Download
OVH.com	Paris, France	Europe	Download
JAIST 北陸先端科学技術大学院大学	Ishikawa, Japan	Asia	Download
easynews logo	Phoenix, AZ	North America	Download
HEAnet	Dublin, Ireland	Europe	Download
mesh solutions ag	Duesseldorf, Germany	Europe	Download
UNIVERSITY OF KENT UKMIRROR service	Kent, UK	Europe	Download
SWITCH	Lausanne, Switzerland	Europe	Download

图1-5 选择镜像

当下载完成之后，我们将下载的Zip文件解压缩。我们注意到其中的dist文件夹，在此文件夹中是我们编程中用到的全部的Jar文件和其他一些辅助文件，如spring.ftl就是Freemarker的模板文件。大多数的jar文件都已经包含在spring.jar文件中了，所以我们只需要将spring.jar文件部署到编程环境的classpath中即可。

1.4 Inversion of Control（IoC）

IoC（Inversion of Control）是Spring的核心部分，但是需要注意的是，Spring并不是IoC的独家专利，这一技术已经在J2EE中发展了数年，如Pico的IoC容器，但是Spring的IoC容器可以认为是Spring的支柱，Spring所仰仗的轻量级容器的核心来自其IoC容器，而Dependency Injection则是Spring的IoC的实现方式，关于IoC以及Dependency Injection的更多内容我们将在后面的章节中有具体论述，在这里只是简单向读者介绍 IoC和Dependency Injection的概念。Rod Johnson在*J2EE Without EJB*中引入IoC概念的时候使用了Hollywood原则：Don't call me, I'll call you。对这句话我们需要用一些简单的代码来解说：

例如我们有下面的代码：

```
public class Test {
   public Connection getConn() {
          JDBCService service = new JDBCService();
          Connection conn = service.getConection();
   }
}
```

很明显，要获得一个JDBC连接，我们要使用JDBCService所提供的服务，初始化JDBCService的实例service，然后使用其getConnection功能来获得连接。这个例子几乎是我们每天都在编写的代码。在这个例子中，我们主动调用系统API来获得服务的实例，然后通过服务获得我们需要的功能，在这个过程中，调用系统API的“责任”在程序员编写的代码这一端。这样做有很多缺点，也就是我们所说的“紧耦合”：service的获得完全依赖于API JDBCService()这个类来提供。

下面我们将上面的这段代码改为Spring模式，如下所示：

```
public class Test {
```

```
    private ConnService service;
    public Connection getConn() {
        JDBCService service = new JDBCService();
        Connection conn = service.getConection();
    }
    public void set ConnService (ConnService service) {
        this.service = service;
    }
}
```

当这段代码经过修改之后，我们可以注意到下面的一些变化：

（1）我们在这里使用了接口ConnService，而不是实际使用的JDBCService。

（2）我们去掉了通过JDBCService直接初始化service的代码。

（3）增加了私有变量service的JavaBean Setter。

在上面的代码中，最重要的变化莫过于，我们从上面的代码中无法得知我们是如何得到Service的实例的，也就是说 Service的Dependency（依赖）在哪里我们无从得知，但是我们可以知道service完全可以满足我们的要求，因为service是ConnService接口类型的。那么谁应该具体负责提供service的Dependency呢？这就是IoC容器的责任了。IoC容器的责任就是负责向应用程序提供Dependency的。经过上面的改变，获取服务的依赖性的责任就由用户代码转移到容器中了，这就是责任的迁移。结合上面“Don't call me, I'll call you”的原则，我们可以理解为——不要来找我，我会去找你。

这样做的优点非常多，最核心的一条是，大大降低了代码的耦合度。这对应用程序开发的影响是不可估量的，而Spring本身可以说就是基于这样一条原则开发而成的，IoC容器是Spring最核心的功能。IoC的更多细节我们将在后面的章节中进行介绍。

1.5 实现面向方面的编程

相对于最初的面向过程式编程，面向对象编程（OOP）倾向于将对象和概念进行建模并作为编程的主体，面向对象编程主要是通过封装和继承等方式减少在面向过程方式中产生的大量的代码重复。但是OOP在不断的发展过程中也遇到了问题：在对操作进行日志记录、错误处理等情况下，代码重复似乎是不可避免的，而这些层面的问题似乎与OOP也存在不同程度的冲突，因为这些问题似乎并不属于OOP的解决范围。通过长期的摸索，很多类似的问题都可以使用特定的设计模式解决，但是这些设计模式并没有真正解决问题，而且制造了一些新的问题。

AOP就是类似问题的终极解决方案，它并不是要替代OOP，而是OOP概念的最好的延伸与补充，AOP核心理念就是将OOP无法解决的问题看做Concern或Aspect，即方面，或称之为考量。每一个考量都代表是某一个在同一个应用程序内重复出现的而且无法用OOP方法进行处理的需求，例如logging以及涉及多个对象的事务处理，这些需求使用传统的OOP方式都无法干净利索地处理，要么要求我们使用一些特定的设计模式，如Command设计模式或Decorator设计模式；要么需要我们引入复杂的API，而这样做就等于是给对象引入了其他的复杂性，这也是我们所不愿见到的。用一句话来概括，AOP可以给对象提供服务。

此为试读，需要完整PDF请访问：www.ertongbook.com