PACKT PUBLISHING

异步图书 www.epubit.com.cn

利用Python设计模式解决软件架构和设计中的实际问题

Python 设计模式

（第2版）

Learning Python Design Patterns

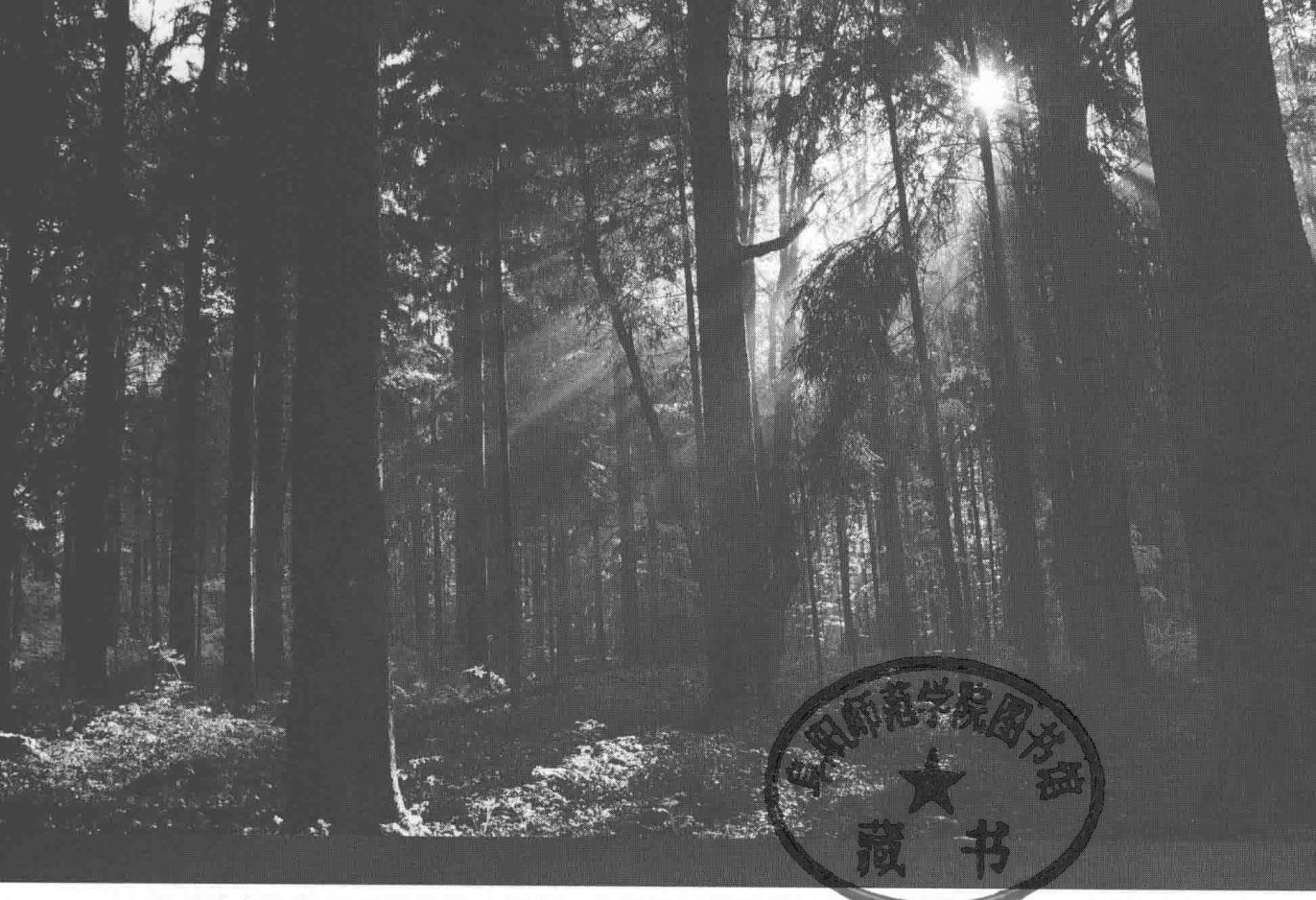
Second Edition

[印度] Chetan Giridhar 著

韩波 译

中国工信出版集团

人民邮电出版社 POSTS & TELECOM PRESS



Python设计模式

（第2版）

[印度] Chetan Giridhar 著
韩波 译

人民邮电出版社
北京

图书在版编目（CIP）数据

Python设计模式 : 第2版 / (印) 吉里德尔(Chetan Giridhar) 著 ; 韩波译. -- 北京 : 人民邮电出版社, 2017.8
ISBN 978-7-115-45880-3

Ⅰ. ①P… Ⅱ. ①吉… ②韩… Ⅲ. ①软件工具－程序设计 Ⅳ. ①TP311.561

中国版本图书馆CIP数据核字(2017)第142331号

版权声明

◆ 著　[印度] Chetan Giridhar
译　韩　波
责任编辑　胡俊英
责任印制　焦志炜

◆ 人民邮电出版社出版发行　北京市丰台区成寿寺路 11 号
邮编　100164　电子邮件　315@ptpress.com.cn
网址　http://www.ptpress.com.cn
北京市艺辉印刷有限公司印刷

◆ 开本：800×1000　1/16
印张：8.75
字数：186 千字　2017 年 8 月第 1 版
印数：1－2 500 册　2017 年 8 月北京第 1 次印刷

著作权合同登记号　图字：01-2016-9212 号

定价：39.00 元

读者服务热线：(010)81055410　印装质量热线：(010)81055316
反盗版热线：(010)81055315
广告经营许可证：京东工商广登字 20170147 号

内容提要

设计模式是构建大型软件系统最强大的方法之一，优化软件架构和设计已经逐渐成为软件开发和维护过程中的一个重要课题。

本书通过 11 章内容，全面揭示有关设计模式的内容，并结合 Python 语言进行示例化的解析。全书囊括单例设计模式、工厂模式、门面模式、代理模式、观察者模式、命令模式、模板方法模式、复合模式、状态设计模式以及反模式等多种设计模式。

本书适合那些关注软件设计原则，并想将优秀的设计模式应用到 Python 编程当中的读者阅读，也适合普通的软件工程师、架构师参考。

序

“控制复杂度是计算机编程的本质”。

——Brian Kernighan

“计算机科学中的所有问题都可以通过抽象来解决”。

——David Wheeler

上面引自两位著名计算机科学家的名言，深入阐释了现代软件设计人员所面临的问题，即迫切需要为软件设计提供一个优良、稳定、可重用、灵活的解决方案。

实际上，设计模式能够以最优雅的方式来解决上述问题。设计模式抽象并存在于整洁、精心设计的组件和接口中，是众多软件设计师和架构师在解决类似问题方面长年积累的宝贵经验。在可重用性、灵活性、可扩展性和可维护性方面，这些解决方案都历经了长时间的考验。

目前，除了堪称设计模式奠基之作的 Gang of Four（GoF）的作品之外，该领域也涌现出了大量图书。

然而，在这个 Web 和移动计算的时代，人们越来越倾向于使用诸如 Python、Ruby 和 Clojure 之类的高级语言来编写程序，这就需要有更多的图书来将设计模式中那些深奥的语言翻译成人们更熟悉的术语，从而帮助人们使用这些更加新颖、更动态化的编程语言来编写可重用的代码。对于程序员新手来说尤其如此，因为他们通常会在设计与实现的复杂性中迷失方向，从而更需要专家的帮助。

书中不仅沿用了 GoF 书中的设计模式模板，同时为兼顾完整性，还添加了一些其他模

式。但在介绍各种模式之前，首先为年轻和缺乏经验的读者提供了软件设计原则方面的基础知识，它们是这些设计模式产生和发展的思想基础。本书没有将读者一把推进模式世界的迷宫中，而是在打开这扇门之前首先介绍相应的基础知识，然后带领读者沿着这条道路循序渐进地学习。

在本书中，模式的示例代码都是用 Python 语言编程实现的，这一点是非常有意义的。作为一个为这种精彩的编程语言社团效力了 12 年多的人，我可以证明，它不仅优美而简洁，而且能够有效地解决各种从常规到最复杂的问题。Python 非常适合新手和年轻的程序员，因为它不仅易于学习，而且用它编写代码简直妙趣横生。年轻的程序员将会发现，他们花在 Python 社区和本书上的时间将是非常有益且卓有成效的。最后，作者 Chetan Giridhar 在 Python 方面经验丰富，因为他已经跟它打了 7 年多的交道了。

本书由他来编写是再合适不过了，因为他多次参与软件的设计和实现，对这些阶段中的复杂性有着切身的体会，并在这个过程学到了很多。同时，他还是 Python 各种论坛的著名演讲者，并多次在 Python 大会（如 PyCon India）上发表演讲。此外，他曾应邀在美国、亚太地区和新西兰的会议上发表演讲。

我相信本书将是对 Packt 系列图书的一个很好的补充，并且将为年轻的 Python 程序员提供一组相应的技能，使他们能够轻而易举通过 Python 进行模块设计和高效编程。

——Anand B Pillai

CTO-Skoov.com

Python 软件基金会董事会成员

班加罗尔 Python 用户组创始人

作者简介

Chetan Giridhar是一位技术领导者、开源爱好者和Python开发人员。他曾在*LinuxForYou*和*Agile Record*等杂志上发表多篇技术和开发实践方面的文章，并在*Python Papers*杂志上发表过技术论文。他曾在PyCon India、Asia-Pacifc和New ZealandPyCon等会议上发表演讲，并热衷于实时通信、分布式系统和云应用等领域。Chetan已经是Packt出版社的技术评审，并为*IPython Visualizations*和*Core Python*等图书撰稿。

我要在此向Packt团队致谢，特别是Merint Thomas Mathew，以及技术评审Maurice HT Ling，感谢他们为本书做出了积极的贡献。特别感谢我的导师Anand B Pillai欣然接受本书的审阅工作，并为本书作序。如果没有我的父母Jyotsana和Jayant Giridhar的祝福，以及我的妻子Deepti和女儿Pihu不断的支持和鼓励，本书是不可能完成的！

技术审稿人简介

Maurice HT Ling 自 2003 年以来一直从事 Python 编程。在墨尔本大学取得生物信息学博士学位以及分子和细胞生物学的理学士（大学荣誉学位）学位后，他目前担任新加坡南洋理工大学的研究员，以及澳大利亚墨尔本大学的名誉研究员。Maurice 是计算和数学生物学的主编，同时也是 *Python Papers* 的共同编辑。最近，Maurice 共同创立了新加坡首屈一指的合成生物学创业公司 AdvanceSyn Pte，并担任公司董事和首席技术官。此外，他还是新加坡 Colossus Technologies LLP 的主要合伙人。他的研究兴趣主要在生命科学领域（生物生命、人造生命和人工智能），使用计算机科学和统计学作为工具来了解生命及其各个方面。空闲时，Maurice 喜欢阅读、品咖啡、写个人日记，或对生命的各个方面进行哲学探讨。你可以在他的个人网站和 LinkedIn 上访问他的个人主页，地址分别为 http://maurice.vodien.com 和 http://www.linkedin.com/in/mauriceling。

前言

设计模式是构建大型软件系统最强大的方法之一。随着人们对优化软件架构和设计的日益关注，对于软件架构师而言，在架构或设计层面上考虑对象创建、代码结构和对象之间的交互等方面的优化也显得日益重要。因为这样不仅可以让软件的维护成本变低，使得代码可以轻松重用，同时还能使得代码可以适应变化。此外，具有可重用性和独立性的框架是当今软件开发的关键所在。

本书组织架构

第 1 章“设计模式简介”介绍了面向对象编程的基础知识，同时详细讨论了面向对象编程的设计原则。本章简要介绍了设计模式的概念，以便帮读者了解软件开发中设计模式的相关背景和应用。第 2 章“单例设计模式”讲解了应用程序开发中使用的最简单和最著名的创建型设计模式之一——单例设计模式。同时，还介绍了利用 Python 创建单例模式的不同方式。此外，本章还介绍了 Monostate（或 Borg）设计模式，它是单例设计模式的一个变体。第 3 章“工厂模式”讨论了另一种创建型模式，即工厂模式。同时，本章还借助 UML 图、现实场景和 Python 3.5 来帮助读者了解工厂方法模式和抽象工厂模式。

第 4 章“门面模式”向读者展现了另一种设计模式，即结构型设计模式。我们不仅介绍了门面的概念，并借助门面设计模式介绍了如何将其用于软件设计。同时，还通过实际场景中的 Python 示例应用程序来介绍其实现过程。

第 5 章“代理模式”讲解了一种结构型设计模式——代理模式。我们首先介绍了代理的概念，并讨论了相应的设计模式，然后介绍了该模式在软件应用程序开发中如何应用。

此外，本章还讲解了代理模式的各种变体——虚拟代理、智能代理、远程代理和保护代理。

第 6 章“观察者模式”探讨了第三种设计模式——行为型设计模式。在本章中，我们将以实例的形式介绍观察者设计模式。同时，本章还详细展示了如何实现观察者模式的推模型和拉模型以及松耦合原则。在云应用程序和分布式系统方面，这种模式是至关重要的。第 7 章“命令模式”将为读者介绍命令设计模式。我们不仅介绍了命令设计模式，还借助现实世界场景和相应的 Python 实现介绍了如何将其应用于软件应用程序开发当中。除此之外，本章还考察了命令模式的两个主要方面，即重做或回滚操作以及异步任务执行的实现。第 8 章“模板方法模式”讨论了模板设计模式。跟命令模式类似，模板模式也属于行为型模式。在这一章中，不仅讨论了模板方法模式，还通过其实现来介绍了钩子技术。此外，本章还通过好莱坞原则来帮助读者加深对这种模式的理解。

第 9 章“模型—视图—控制器——复合模式”不仅为读者介绍了该模式本身，还讨论了如何将其应用于软件应用程序开发。MVC 已经成为最常用的设计模式之一，其实很多 Python 框架都是基于这个原理的。读者还可以通过使用 Python Tornado（Facebook 使用的框架）编写的示例应用程序来了解 MVC 实现的详细信息。

第 10 章“状态设计模式”向读者介绍了状态设计模式，就像命令或模板设计模式一样，它们都属于行为型模式。同时，本章还讨论了如何在软件应用程序开发中使用该模式。第 11 章“反模式”为读者介绍了反模式，即作为架构师或软件工程师，我们不应该采取的那些行为。

本书需要的资源

对于阅读本书来说，只需安装 Python 3.5 即可，你可以从 https://www.python.org/downloads/下载并安装该软件。

目标读者

本书的目标读者是需要关注软件设计原则和 Python 应用程序开发方面细节的 Python 开发人员和软件架构师。这要求读者对编程概念有基本了解，同时要具备初级的 Python 开发经验。此外，对于学生和老师来说，现场学习环境也是颇为有益的。

排版约定

在本书中，不同类型的信息会采用不同的排版样式，以示区别。下面针对各种排版样式及其含义进行举例说明。

文本、数据库表名、文件夹名、文件名、文件扩展名和路径名、伪 URL、用户输入和推特句柄中出现的代码文字，会显示：“对象 `Car` 具有诸如 `fuel level`、`isSedan`、`speed`、`steering wheel` 和 `coordinates` 等属性。”

代码段会显示：

```
class Person(object):
    def __init__(self, name, age): #constructor
        self.name = name    #data members/ attributes
        self.age = age
    def get_person(self,):    # member function
        return "<Person (%s, %s)>" % (self.name, self.age)

p = Person("John", 32)    # p is an object of type Person
print("Type of Object:", type(p), "Memory Address:", id(p))
```

新术语及重要词汇使用**粗体字**表示。对于在屏幕中看到的文字，如菜单或者对话框中的文字，排版形式为“对于 Python 语言来说，封装（数据和方法的隐藏）的概念不是隐式的，因为它没有提供支持封装所需的关键字，如 **public**、**private** 和 **protected**（而 C ++或 Java 语言则提供了相应的关键字）”。

提示：

- 警告或者重要的注释在此显示。

小技巧：

- 提示和小技巧在此显示。

读者反馈

我们欢迎读者对本书进行反馈，希望了解你对本书的看法：你喜欢哪些方面或不喜欢哪些方面。在帮助本社推出真正符合读者需要的图书方面，读者的反馈信息至关重要。

如果想为我们提供一般性的反馈，请向 feedback@packtpub.com 邮箱发送电子邮件，并在邮件的标题中指出相应的书名即可。

如果某些主题是你擅长的领域，并且有意著书或撰稿，请进入 www.packtpub.com/authors，进一步阅读作者指南。

客户支持

你已经是 Packt 出版社的尊贵用户，为了让你的订购物超所值，我们将为你提供一些增值服务。

下载示例代码

访问 http://www.packtpub.com 网站并登录账户后，读者便可以下载所有已购 Packt 出版社图书的示例代码。如果是在其他地方购买的本书，可以访问 http://www.packtpub.com/support 并注册，通过电子邮件获取相应的代码。

勘误

虽然我们会全力确保书中内容的准确性，但错误仍在所难免。如果你在本书中发现了错误（文字错误或代码错误），而且愿意向我们提交这些错误，我们感激不尽。这样不仅可以消除其他读者的疑虑，也有助于改进后续版本。若想提交你发现的错误，请访问 http://www.packtpub.com/submit-errata，在“Errata Submission Form”（提交勘误表单）中选择相应图书，输入勘误详情。勘误通过验证之后将上传到 Packt 网站，或添加到现有的勘误列表中。若想查看某本书的现有勘误信息，请访问 http://www. packtpub.com/support，选择相应的书名即可。

关于盗版行为

对各种媒体而言，互联网上受版权保护的各种材料都长期面临非法复制的问题。Packt出版社非常重视版权保护和版权许可，如果你在网上看到本社图书任何形式的非法复制，请立刻向我们提供网址或网站名称，以便我们及时采取补救措施。

请通过 copyright@packtpub.com 联系我们，并提供疑似盗版材料的链接信息。

感谢你帮助我们保护作者的权益，从而使我们能够提供更有价值的内容。

疑问解答

如果你对本书有任何疑问，可以通过 questions@packtpub.com 联系我们，我们将尽力为你解答。

目录