

# 人人都懂设计模式：从生活中领悟设计模式：Python实现



[人人都懂设计模式：从生活中领悟设计模式：Python实现 下载链接1](#)

著者:罗伟富

出版者:电子工业出版社

出版时间:2019-4

装帧:平装

isbn:9787121361128

设计模式 (Design

Patterns)是一套被反复使用、多数人知晓、无数工程师实践的代码设计经验的总结，它是面向对象思想的高度提炼和模板化。

《人人都懂设计模式：从生活中领悟设计模式：Python实现》带你一起从生活的角度思考设计模式，以轻松有趣的小故事开始，由浅入深地讲解每一种模式，思考每一种模式，总结每一种模式！力求用更通俗的语言阐述难懂的概念，用更简单的语法实现复杂的逻辑，用更短小的代码写出强悍的程序！使枯燥乏味的概念变得更有乐趣和意义，希望能带给读者一种全新的阅读体验和思考方式。

《人人都懂设计模式：从生活中领悟设计模式：Python实现》分为3篇：“基础篇”讲解了23种经典设计模式，其中19种常用设计模式分别用单独的章节讲解，其余模式作为一个合集放在一章中讲解；“进阶篇”讲解了由基础设计模式衍生出的各种编程机制，包括过滤器模式、对象池技术、回调机制和MVC模式，它们在各大编程语言中都非常重要而且常见；“经验篇”结合工作经验和项目积累，分享了对设计模式、设计原则、项目重构的理解和看法。Python作为AI时代最重要的一种计算机语言，在各大语言中的排名逐年上升！本书所有示例代码均用Python编写，将会是国内不可多得的一本用Python来讲解设计模式的书。

《人人都懂设计模式：从生活中领悟设计模式：Python实现》适合的读者：一线互联网软件开发者、有一定编程基础的IT职场新人、对设计模式和编程思想感兴趣的人士。

作者介绍:

目录: 基础篇

第0章 启程之前，请不要错过我 2

0.1 Python精简入门 2

0.1.1 Python的特点 2

0.1.2 基本语法 3

0.1.3 一个例子让你顿悟 7

0.1.4 重要说明 11

0.2 UML精简概述 11

0.2.1 UML的定义 11

0.2.2 常见的关系 12

第1章 监听模式 16

1.1 从生活中领悟监听模式 16

1.1.1 故事剧情—幻想中的智能热水器 16

1.1.2 用程序来模拟生活 17

1.2 从剧情中思考监听模式 19

1.2.1 什么是监听模式 19

1.2.2 监听模式设计思想 19

1.3 监听模式的模型抽象 19

1.3.1 代码框架 19

1.3.2 类图 20

1.3.3 基于框架的实现 21

1.3.4 模型说明 22

1.4 实战应用 23

1.5 应用场景 26

第2章 状态模式 28

2.1 从生活中领悟状态模式 28

2.1.1 故事剧情—人有少、壮、老，水之固、液、气 28

2.1.2 用程序来模拟生活 29

2.2 从剧情中思考状态模式 32

- 2.2.1 什么是状态模式 32
- 2.2.2 状态模式设计思想 33
- 2.3 状态模式的模型抽象 33
  - 2.3.1 代码框架 33
  - 2.3.2 类图 35
  - 2.3.3 基于框架的实现 36
  - 2.3.4 模型说明 38
- 2.4 应用场景 39
- 第3章 中介模式 40
  - 3.1 从生活中领悟中介模式 40
    - 3.1.1 故事剧情—找房子问中介 40
    - 3.1.2 用程序来模拟生活 41
  - 3.2 从剧情中思考中介模式 46
    - 3.2.1 什么是中介模式 46
    - 3.2.2 中介模式设计思想 46
  - 3.3 中介模式的模型抽象 48
    - 3.3.1 代码框架 48
    - 3.3.2 类图 49
    - 3.3.3 模型说明 50
  - 3.4 实战应用 51
  - 3.5 应用场景 56
- 第4章 装饰模式 57
  - 4.1 从生活中领悟装饰模式 57
    - 4.1.1 故事剧情—你想怎么搭就怎么搭 57
    - 4.1.2 用程序来模拟生活 58
  - 4.2 从剧情中思考装饰模式 62
    - 4.2.1 什么是装饰模式 62
    - 4.2.2 装饰模式设计思想 63
  - 4.3 装饰模式的模型抽象 64
    - 4.3.1 类图 64
    - 4.3.2 Python中的装饰器 64
    - 4.3.3 模型说明 69
  - 4.4 应用场景 70
- 第5章 单例模式 71
  - 5.1 从生活中领悟单例模式 71
    - 5.1.1 故事剧情—你是我的唯一 71
    - 5.1.2 用程序来模拟生活 72
  - 5.2 从剧情中思考单例模式 73
    - 5.2.1 什么是单例模式 73
    - 5.2.2 单例模式设计思想 73
  - 5.3 单例模式的模型抽象 73
    - 5.3.1 代码框架 73
    - 5.3.2 类图 78
    - 5.3.3 基于框架的实现 78
  - 5.4 应用场景 79
- 第6章 克隆模式 80
  - 6.1 从生活中领悟克隆模式 80
    - 6.1.1 故事剧情—给你一个分身术 80
    - 6.1.2 用程序来模拟生活 80
  - 6.2 从剧情中思考克隆模式 82
    - 6.2.1 什么是克隆模式 82
    - 6.2.2 浅拷贝与深拷贝 82
  - 6.3 克隆模式的模型抽象 86
    - 6.3.1 代码框架 86

|                      |     |
|----------------------|-----|
| 6.3.2 类图             | 86  |
| 6.3.3 基于框架的实现        | 87  |
| 6.3.4 模型说明           | 87  |
| 6.4 实战应用             | 88  |
| 6.5 应用场景             | 90  |
| 第7章 职责模式             | 91  |
| 7.1 从生活中领悟职责模式       | 91  |
| 7.1.1 故事剧情—我的假条去哪儿了  | 91  |
| 7.1.2 用程序来模拟生活       | 92  |
| 7.2 从剧情中思考职责模式       | 96  |
| 7.2.1 什么是职责模式        | 96  |
| 7.2.2 职责模式设计思想       | 96  |
| 7.3 职责模式的模型抽象        | 97  |
| 7.3.1 代码框架           | 97  |
| 7.3.2 类图             | 98  |
| 7.3.3 基于框架的实现        | 99  |
| 7.3.4 模型说明           | 102 |
| 7.4 应用场景             | 103 |
| 第8章 代理模式             | 104 |
| 8.1 从生活中领悟代理模式       | 104 |
| 8.1.1 故事剧情—帮我拿一下快递   | 104 |
| 8.1.2 用程序来模拟生活       | 105 |
| 8.2 从剧情中思考代理模式       | 107 |
| 8.2.1 什么是代理模式        | 107 |
| 8.2.2 代理模式设计思想       | 107 |
| 8.3 代理模式的模型抽象        | 107 |
| 8.3.1 代码框架           | 107 |
| 8.3.2 类图             | 109 |
| 8.3.3 基于框架的实现        | 110 |
| 8.3.4 模型说明           | 111 |
| 8.4 应用场景             | 111 |
| 第9章 外观模式             | 113 |
| 9.1 从生活中领悟外观模式       | 113 |
| 9.1.1 故事剧情—学妹别慌，学长帮你 | 113 |
| 9.1.2 用程序来模拟生活       | 114 |
| 9.2 从剧情中思考外观模式       | 116 |
| 9.2.1 什么是外观模式        | 116 |
| 9.2.2 外观模式设计思想       | 116 |
| 9.3 外观模式的模型抽象        | 117 |
| 9.3.1 类图             | 117 |
| 9.3.2 软件的分层结构        | 117 |
| 9.3.3 模型说明           | 119 |
| 9.4 实战应用             | 119 |
| 9.5 应用场景             | 123 |
| 第10章 迭代模式            | 124 |
| 10.1 从生活中领悟迭代模式      | 124 |
| 10.1.1 故事剧情—下一个就是你了  | 124 |
| 10.1.2 用程序来模拟生活      | 125 |
| 10.2 从剧情中思考迭代模式      | 128 |
| 10.2.1 什么是迭代模式       | 128 |
| 10.2.2 迭代模式设计思想      | 129 |
| 10.3 迭代模式的模型抽象       | 130 |
| 10.3.1 代码框架          | 130 |
| 10.3.2 Python中的迭代器   | 132 |

- 10.3.3 类图 136
- 10.3.4 模型说明 137
- 10.4 应用场景 138
- 第11章 组合模式 139
  - 11.1 从生活中领悟组合模式 139
    - 11.1.1 故事剧情—自己组装电脑，价格再降三成 139
    - 11.1.2 用程序来模拟生活 140
  - 11.2 从剧情中思考组合模式 146
    - 11.2.1 什么是组合模式 146
    - 11.2.2 组合模式设计思想 147
  - 11.3 组合模式的模型抽象 148
    - 11.3.1 代码框架 148
    - 11.3.2 类图 149
    - 11.3.3 模型说明 150
  - 11.4 实战应用 150
  - 11.5 应用场景 153
- 第12章 构建模式 154
  - 12.1 从生活中领悟构建模式 154
    - 12.1.1 故事剧情—你想要一辆车还是一个庄园 154
    - 12.1.2 用程序来模拟生活 155
  - 12.2 从剧情中思考构建模式 157
    - 12.2.1 什么是构建模式 157
    - 12.2.2 构建模式设计思想 157
    - 12.2.3 与工厂模式的区别 158
    - 12.2.4 与组合模式的区别 158
  - 12.3 构建模式的模型抽象 159
    - 12.3.1 类图 159
    - 12.3.2 基于框架的实现 160
    - 12.3.3 模型说明 163
  - 12.4 应用场景 164
- 第13章 适配模式 166
  - 13.1 从生活中领悟适配模式 166
    - 13.1.1 故事剧情—有个转换器就好了 166
    - 13.1.2 用程序来模拟生活 167
  - 13.2 从剧情中思考适配模式 170
    - 13.2.1 什么是适配模式 170
    - 13.2.2 适配模式设计思想 170
  - 13.3 适配模式的模型抽象 172
    - 13.3.1 代码框架 172
    - 13.3.2 类图 172
    - 13.3.3 模型说明 173
  - 13.4 实战应用 174
  - 13.5 应用场景 184
- 第14章 策略模式 185
  - 14.1 从生活中领悟策略模式 185
    - 14.1.1 故事剧情—怎么来不重要，人到就行 185
    - 14.1.2 用程序来模拟生活 186
  - 14.2 从剧情中思考策略模式 188
    - 14.2.1 什么是策略模式 188
    - 14.2.2 策略模式设计思想 188
  - 14.3 策略模式的模型抽象 189
    - 14.3.1 类图 189
    - 14.3.2 模型说明 190
  - 14.4 实战应用 190

- 14.5 应用场景 195
- 第15章 工厂模式 196
  - 15.1 从生活中领悟工厂模式 196
    - 15.1.1 故事剧情—你要拿铁还是摩卡呢 196
    - 15.1.2 用程序来模拟生活 197
  - 15.2 从剧情中思考工厂模式 199
    - 15.2.1 什么是简单工厂模式 199
    - 15.2.2 工厂模式设计思想 199
  - 15.3 工厂三姐妹 199
    - 15.3.1 简单工厂模式 200
    - 15.3.2 工厂方法模式 201
    - 15.3.3 抽象工厂模式 203
  - 15.4 进一步思考 205
  - 15.5 实战应用 205
- 第16章 命令模式 209
  - 16.1 从生活中领悟命令模式 209
    - 16.1.1 故事剧情—大闸蟹，走起 209
    - 16.1.2 用程序来模拟生活 210
  - 16.2 从剧情中思考命令模式 213
    - 16.2.1 什么是命令模式 213
    - 16.2.2 命令模式设计思想 213
  - 16.3 命令模式的模型抽象 214
    - 16.3.1 代码框架 214
    - 16.3.2 类图 215
    - 16.3.3 模型说明 216
  - 16.4 实战应用 216
  - 16.5 应用场景 224
- 第17章 备忘模式 225
  - 17.1 从生活中领悟备忘模式 225
    - 17.1.1 故事剧情—好记性不如烂笔头 225
    - 17.1.2 用程序来模拟生活 226
  - 17.2 从剧情中思考备忘模式 228
    - 17.2.1 什么是备忘模式 228
    - 17.2.2 备忘模式设计思想 229
  - 17.3 备忘模式的模型抽象 229
    - 17.3.1 类图 229
    - 17.3.2 代码框架 230
    - 17.3.3 模型说明 232
  - 17.4 实战应用 232
  - 17.5 应用场景 235
- 第18章 享元模式 236
  - 18.1 从生活中领悟享元模式 236
    - 18.1.1 故事剧情—颜料很贵，必须充分利用 236
    - 18.1.2 用程序来模拟生活 237
  - 18.2 从剧情中思考享元模式 239
    - 18.2.1 什么是享元模式 239
    - 18.2.2 享元模式设计思想 239
  - 18.3 享元模式的模型抽象 239
    - 18.3.1 类图 239
    - 18.3.2 基于框架的实现 240
    - 18.3.3 模型说明 242
  - 18.4 应用场景 243
- 第19章 访问模式 244
  - 19.1 从生活中领悟访问模式 244

- 19.1.1 故事剧情——一千个读者一千个哈姆雷特 244
- 19.1.2 用程序来模拟生活 245
- 19.2 从剧情中思考访问模式 246
  - 19.2.1 什么是访问模式 246
  - 19.2.2 访问模式设计思想 247
- 19.3 访问模式的模型抽象 247
  - 19.3.1 代码框架 247
  - 19.3.2 类图 248
  - 19.3.3 基于框架的实现 249
  - 19.3.4 模型说明 250
- 19.4 实战应用 251
- 19.5 应用场景 255
- 第20章 其他经典设计模式 256
  - 20.1 模板模式 256
    - 20.1.1 模式定义 256
    - 20.1.2 类图结构 257
    - 20.1.3 代码框架 257
    - 20.1.4 应用案例 259
    - 20.1.5 应用场景 261
  - 20.2 桥接模式 261
    - 20.2.1 模式定义 261
    - 20.2.2 类图结构 261
    - 20.2.3 应用案例 262
    - 20.2.4 应用场景 266
  - 20.3 解释模式 266
    - 20.3.1 模式定义 266
    - 20.3.2 类图结构 266
    - 20.3.3 应用案例 267
    - 20.3.4 应用场景 271
- 进阶篇
- 第21章 深入解读过滤器模式 274
  - 21.1 从生活中领悟过滤器模式 274
    - 21.1.1 故事剧情——制作一杯鲜纯细腻的豆浆 274
    - 21.1.2 用程序来模拟生活 275
  - 21.2 从剧情中思考过滤器模式 275
    - 21.2.1 过滤器模式 276
    - 21.2.2 与职责模式的联系 276
  - 21.3 过滤器模式的模型抽象 276
    - 21.3.1 代码框架 277
    - 21.3.2 类图 278
    - 21.3.3 基于框架的实现 278
    - 21.3.4 模型说明 279
  - 21.4 实战应用 280
  - 21.5 应用场景 282
- 第22章 深入解读对象池技术 283
  - 22.1 从生活中领悟对象池技术 283
    - 22.1.1 故事剧情——共享让出行更便捷 283
    - 22.1.2 用程序来模拟生活 284
  - 22.2 从剧情中思考对象池机制 287
    - 22.2.1 什么是对象池 287
    - 22.2.2 与享元模式的联系 287
  - 22.3 对象池机制的模型抽象 288
    - 22.3.1 代码框架 288
    - 22.3.2 类图 291

|                                           |     |
|-------------------------------------------|-----|
| 22.3.3 基于框架的实现                            | 292 |
| 22.3.4 模型说明                               | 294 |
| 22.4 应用场景                                 | 295 |
| 第23章 深入解读回调机制                             | 296 |
| 23.1 从生活中领悟回调机制                           | 296 |
| 23.1.1 故事剧情—把你的技能亮出来                      | 296 |
| 23.1.2 用程序来模拟生活                           | 296 |
| 23.2 从剧情中思考回调机制                           | 298 |
| 23.2.1 回调机制                               | 298 |
| 23.2.2 设计思想                               | 299 |
| 23.3 回调机制的模型抽象                            | 299 |
| 23.3.1 面向过程的实现方式                          | 299 |
| 23.3.2 面向对象的实现方式                          | 300 |
| 23.3.3 模型说明                               | 301 |
| 23.4 实战应用                                 | 302 |
| 23.4.1 基于回调函数的实现                          | 302 |
| 23.4.2 基于策略模式的实现                          | 303 |
| 23.4.3 回调在异步中的应用                          | 307 |
| 23.5 应用场景                                 | 310 |
| 第24章 深入解读MVC模式                            | 311 |
| 24.1 从生活中领悟MVC模式                          | 311 |
| 24.1.1 故事剧情—定格最美的一瞬间                      | 311 |
| 24.1.2 用程序来模拟生活                           | 312 |
| 24.2 从剧情中思考MVC模式                          | 316 |
| 24.2.1 MVC模式                              | 317 |
| 24.2.2 与中介模式的联系                           | 317 |
| 24.2.3 与外观模式的联系                           | 317 |
| 24.3 MVC模式的模型抽象                           | 318 |
| 24.3.1 MVC                                | 318 |
| 24.3.2 MVP                                | 318 |
| 24.3.3 MVVM                               | 319 |
| 24.3.4 模型说明                               | 320 |
| 24.4 应用场景                                 | 320 |
| 经验篇                                       |     |
| 第25章 关于设计模式的理解                            | 324 |
| 25.1 众多书籍之下为何还要写此书                        | 324 |
| 25.2 设计模式玄吗                               | 324 |
| 25.3 如何区分不同的模式                            | 325 |
| 25.4 编程思想的三重境界                            | 325 |
| 第26章 关于设计原则的思考                            | 327 |
| 26.1 SOLID原则                              | 327 |
| 26.1.1 单一职责原则                             | 327 |
| 26.1.2 开放封闭原则                             | 331 |
| 26.1.3 里氏替换原则                             | 334 |
| 26.1.4 依赖倒置原则                             | 337 |
| 26.1.5 接口隔离原则                             | 341 |
| 26.2 是否一定要遵循这些设计原则                        | 348 |
| 26.2.1 软件设计是一个逐步优化的过程                     | 348 |
| 26.2.2 不是一定要遵循这些设计原则                      | 349 |
| 26.3 更为实用的设计原则                            | 349 |
| 26.3.1 LoD原则 (Law of Demeter)             | 349 |
| 26.3.2 KISS原则 (Keep It Simple and Stupid) | 350 |
| 26.3.3 DRY原则 (Don't Repeat Yourself)      | 351 |
| 26.3.4 YAGNI原则 (You Aren't Gonna Need It) | 353 |

|                                         |                      |
|-----------------------------------------|----------------------|
| 26.3.5 Rule Of Three原则                  | 353                  |
| 26.3.6 CQS原则 (Command-Query Separation) | 354                  |
| 第27章 关于项目重构的思考                          | 355                  |
| 27.1 什么叫重构                              | 355                  |
| 27.2 为何要重构                              | 355                  |
| 27.3 什么时机进行重构                           | 356                  |
| 27.4 如何重构代码                             | 357                  |
| 27.4.1 重命名                              | 357                  |
| 27.4.2 函数重构                             | 358                  |
| 27.4.3 重新组织数据                           | 359                  |
| 27.4.4 用设计模式改善代码设计                      | 360                  |
| 27.5 代码整洁之道                             | 360                  |
| 27.5.1 命名的学问                            | 360                  |
| 27.5.2 整洁代码的案例                          | 362                  |
| 附录A 23种经典设计模式的索引对照表                     | 368                  |
| 附录B Python中__new__、__init__和__call__的用法 | 370                  |
| 附录C Python中metaclass的原理                 | 377                  |
| • • • • •                               | <a href="#">(收起)</a> |

[人人都懂设计模式：从生活中领悟设计模式：Python实现\\_下载链接1](#)

## 标签

Python

设计模式

编程

计算机

思维

讲故事

工具书

入门

## 评论

好书！简单、形象！看完收获不小，但是也发现平时我也用了不少里面的设计模式。很强的程序员应该会觉得简单

---

此书看似无人问津，但基本都是作者的心得体会，时而温故知新

---

花了4天时间看完了，这个90后小伙儿写设计模式还是有点想当然了，虽然类图没什么问题（桥接模式的示例图是错的），但是书中的错误实在是有点多，举的例子也不都是很恰当，示例代码写的完全不优雅，为了设计模式而设计模式，让人看了很不舒服。比如对象池案例，借充电宝时怎么可能以充电宝序列号作为输入参数呢？虽然在实际中通常应该是多种设计模式一起使用，但是这样的示例代码和讲解出现在一本讲设计模式的书中，实在是让人很气愤。

---

[人人都懂设计模式：从生活中领悟设计模式：Python实现\\_下载链接1](#)

## 书评

---

[人人都懂设计模式：从生活中领悟设计模式：Python实现\\_下载链接1](#)