atitit 软件框架类库设计的艺术.docx

目录

[1. index 1](#_Toc6046)

[2. 第2章 设计api的动力之源 14 2](#_Toc11538)

[2.1. .1 分布式开发 14 2](#_Toc8174)

[2.2. 2.2 模块化应用程序 16 2](#_Toc2149)

[3. 第4章 兼容性 42 2](#_Toc15309)

[3.1. 4.2.1 源代码兼容 43 2](#_Toc2906)

[3.2. 4.2.2 二进制兼容 44 2](#_Toc3796)

[3.3. 4.2.3 功能兼容——阿米巴变形虫效应 50 2](#_Toc24500)

[3.4. 反射解决兼容性 2](#_Toc1563)

# index

第1章 软件开发的艺术 4

第2章 设计api的动力之源 14

第3章 评价api好坏的标准 26

第4章 不断变化的目标 42

第5章 只公开你要公开的内容 67

第6章 面向接口而非实现进行编程 85

第7章 模块化架构 98

第8章 设计api时要区分其目标用户群 129

第9章 牢记可测试性 147

第10章 与其他api协作 158

第11章 api具体运行时的一些内容 184

第12章 声明式编程 223

第13章 极端的意见有害无益 236

第14章 api设计中的矛盾之处 247

第15章 改进api 258

第16章 团队协作 286

第17章 利用竞赛游戏来提升api设计技巧 300

第18章 可扩展visitor模式的案例 328

第19章 消亡的过程 348

第20章 未来 356

第1章 软件开发的艺术 4

# 第2章 设计api的动力之源 14

## .1 分布式开发 14

## 2.2 模块化应用程序 16

第3章 评价api好坏的标准 26

# 第4章 兼容性 42

4.2 向后兼容 43

## 4.2.1 源代码兼容 43

## 4.2.2 二进制兼容 44

## 4.2.3 功能兼容——阿米巴变形虫效应 50

#### **API演进不同于SPI**

API中添加内容是容许的，移除是不容许的  
SPI中添加内容是有损的，移除是可以的

## 反射解决兼容性

第5章 只公开你要公开的内容 67

第6章 面向接口而非实现进行编程 85

第7章 模块化架构 98

.1 模块化设计的类型 100

7.2 组件定位和交互 103

7.3 编写扩展点 116

7.4 循环依赖的必要性 117

7.5 满城尽是lookup 121

7.6 lookup的滥用 126

第8章 设计api时要区分其目标用户群 129

第9章 牢记可测试性 147

第10章 与其他api协作 158

第11章 api具体运行时的一些内容 184

第12章 声明式编程 223

第13章 极端的意见有害无益 236

第14章 api设计中的矛盾之处 247

第15章 改进api 258

第16章 团队协作 286

第17章 利用竞赛游戏来提升api设计技巧 300

第18章 可扩展visitor模式的案例 328

第19章 消亡的过程 348

9.1 明确版本的重要性 349

19.2 模块依赖的重要性 349

19.3 被移除的部分需要永久保留吗 352

19.4 分解庞大的api 352

第20章 未来 356

20.1 原则性内容 357

20.2 无绪长存 358

20.3 api设计方法论 360

20.4 编程语言的演变 361

软件框架设计的艺术\_百度百科.html