软件设计的原则串讲

汪志成 (wangzhicheng@admaster.com.cn)

2014年5月19日

概观

- 架构模式
- 设计模式
- 面向对象
- 高内聚，低耦合
- 信息隐蔽

信息隐蔽原则(1972年)

Brooks为什么向Parnas认错？

什么是信息隐蔽原则？

为什么需要信息隐蔽原则？

- 便于项目管理，防止错误扩散，降低复杂度 —— 降低成本

怎样实现信息隐蔽?

抽象，抓住重点

- 面向接口编程
- 契约式设计（DbC）

专业化，把细节局部化

- 一个代码块只做一件事
- 在不同层次上达到专业化

高内聚

代码块做的事越多，内聚性越差

- 做“恰好够”的事，过多则内聚性差，过少则合作模块不得不耦合过来

它的职责能用一句话说清楚吗？

- KISS原则，单一职责原则（SRP），接口隔离原则（ISP）
- 比弄清楚“它做什么？”更重要的是弄清楚“它不做什么？”

在不同抽象层次上内聚

- 组合优于继承：Unix哲学，Decorator模式

低耦合

有哪些耦合形式？

- 回调/通知（好莱坞原则、IoC / DI）
- 保存引用（DIP依赖倒置原则）
- 调用方法
- 访问属性
- 顺序依赖
- 状态依赖

耦合与管理

- 耦合本质上是“约定和假设”，最危险的耦合是“你所不知道的约定和假设”
- 最麻烦的不是代码之间的耦合，而是项目“任务”之间的耦合
- 为什么复用（DRY）也会惹祸？低质量的“可复用模块”是定时炸弹
- 模块的质量缺陷会被耦合放大！

如何解耦？

- 保证高内聚才能低耦合，设计小而精的模块
- “知道的”越少越好（LoD迪米特法则、最少知识原则），不作死就不会死
- （SoC关注点分离），在不同的抽象层次上划分关注点

面向对象编程

面向接口编程

面向对象编程

过程式编程

设计模式

面向接口编程

- 接口描述的是“行为”，应该隐藏掉实现细节
- 接口的抽象层次比较高

组合优于继承（CoI）

- 继承是仅次于友元的强耦合
- 好的模块应该可扩展，不可修改。（OCP开闭原则）
- 继承是为了被复用，而不是为了复用（LSP里氏代换）

其他原则

- 约定优于配置（CoC）
- 共同封闭原则（CCP）
- 共同重用原则（CRP）
- 无环依赖（ADP）
- 查询与命令分离（CQS，幂等性）

分享完毕，谢谢！