Atitit it sftwr dev 原则准则 principle

目录

[第一章 简单原则 kiss 1](#_Toc12737)

[第一节 . You Ain’t Gonna Need It(YAGNI)避免过度设计 1](#_Toc19803)

[第二节 Convention over Configuration（CoC）–惯例优于配置原则 惯例优先原则 1](#_Toc15757)

[第三节 各种轻量化技术，追求极简模式 1](#_Toc5221)

[第四节 最小功效原则 原始数据 1](#_Toc2120)

[第五节 勉强使用即可的标准 2](#_Toc14154)

[第二章 集成原则 集成现有的 2](#_Toc28275)

[第一节 比如搜索 ctrl+f 2](#_Toc4061)

[第二节 ftp传输直接集成客户端 2](#_Toc23710)

[第三节 音视频播放等直接集成 2](#_Toc4685)

[第三章 多元化原则 开发原则 百花齐放百家争鸣 多姿多彩 2](#_Toc20632)

[第一节 多个mvc db orm等 2](#_Toc7716)

[第二节 单一化的问题 2](#_Toc26861)

[第四章 自由化理念 自由化元则 3](#_Toc18488)

[第五章 性价比原则 3](#_Toc7571)

[第六章 循序渐进原则 vs革命 3](#_Toc27968)

[第七章 保证技术能力的原则 3](#_Toc21168)

[第一节 自给自足，原则 diy 3](#_Toc6630)

# 简单原则 kiss

## . You Ain’t Gonna Need It(YAGNI)避免过度设计

过度oo ，，fp面向方面也不错

## Convention over Configuration（CoC）–惯例优于配置原则 惯例优先原则

## 各种轻量化技术，追求极简模式

## 最小功效原则 原始数据

互联网之父Tim Berners-Lee 在他的著作 《最小功效原则》中的一段话：

“在过去的40年中，计算机科学一直在开发尽可能功能强大的语言。现在我们必须领会这样的道理：要选择功能最不强的解决方案而不是最强的。语言的功效越小，对于储存在该语言中的数据你能做的事情就越多。如果你把程序写成简单的描述性形式，任何人都可以编写一个程序来分析它。比如，如果一个包含了天气数据的网页，里面使用RDF 格式来描述这些数据，用户就可以把它当做一个表格来查找，也许求它的平均值，或者用它绘图，或把它和其他信息结合在一起进行推理分析。另一个极端情况是，这些天气信息是用漂亮的Java applet描绘出来的。虽然这样可能做出很酷的用户界面，但它完全无法用于分析。找到这个页面的搜索引擎会压根不知道这些数据是什么或者是干什么用的。唯一能发现Java applet的含义的方式就是让它在一个人面前运行起来。”

## 勉强使用即可的标准

# 集成原则 集成现有的

## 比如搜索 ctrl+f

## ftp传输直接集成客户端

## 音视频播放等直接集成

# 多元化原则 开发原则 百花齐放百家争鸣 多姿多彩

## 多个mvc db orm等

## 单一化的问题

僵化 ，成本高，容易过时，不利于发挥团队能力

有些人可能更熟悉某一些技术

# 自由化理念 自由化元则

自由化参数 json map等半结构化参数，dsl等全功能参数

自由化功能接口 dsl

自由化返回数据结构json

# 性价比原则

# 循序渐进原则 vs革命

# 保证技术能力的原则

## 自给自足，原则 diy

Mvc orm等。。

问题就在生态链上。