Atitit 提升可读性的艺术

目录

[1. 几大原则 2](#_Toc7856)

[1.1. 直接原则，无脑原则。。。 2](#_Toc29161)

[1.2. 避免过度设计 2](#_Toc8773)

[2. 本地化命名法 2](#_Toc29128)

[2.1. 可以使用管理命名法 多个api 比如old api，new api 2](#_Toc1383)

[2.2. 相关方法使用前缀发命名空间 2](#_Toc6213)

[2.3. 提高抽象级别，what与how相分离，优先指明what 2](#_Toc14828)

[2.4. 命名规范 参考知名api 2](#_Toc15331)

[2.5. 使用命名空间，不支持命名空间的事业类似前缀 2](#_Toc24619)

[3. 管理代码存储区域 优先文件，函数 oo模式 3](#_Toc6135)

[3.1. 静态模块最好了 3](#_Toc22811)

[3.2. 其次静态方法 ，最后oo动态方法 3](#_Toc6344)

[4. 减少嵌套与跳转 3](#_Toc23139)

[4.1. 减少简化层次结构，单层其实可以满足大部分架构，其次双层，多层 3](#_Toc13786)

[4.2. 一个流程代码不要跨文件，跳转过多 3](#_Toc7182)

[4.3. 减少语法噪音嵌套使用lambda 与脚本 3](#_Toc13143)

[4.4. 减少跳转 内部类也不错 3](#_Toc13503)

[4.5. 匿名方法，匿名构造函数 初始化块，动态map模拟匿名类匿名方法 输出 3](#_Toc11248)

[5. 减少语法噪音 4](#_Toc8339)

[5.1. Lambda 4](#_Toc3766)

[5.2. 多使用static import导入机制。。注意函数名不要重复了。。 4](#_Toc4123)

[6. 适当分类 命名空间模式 4](#_Toc22705)

[6.1. 数据库使用分库提升可读性 4](#_Toc18446)

[6.2. 命名空间前缀发 4](#_Toc20550)

[7. 参数 使用通用接口 大力减少代码 5](#_Toc24486)

[8. Dsl 5](#_Toc30883)

[8.1. 表格映射表代替选择 决策树模式等 5](#_Toc23233)

[9. 同步 法 优先于异步法 5](#_Toc26069)

[10. Other 5](#_Toc17987)

[10.1. 适当的事件驱动法 5](#_Toc7723)

[10.2. 异常模式代替返回 5](#_Toc28746)

[10.3. 方法链 5](#_Toc23573)

[10.4. 字符串模板技术 5](#_Toc25391)

[10.5. 不要使用try catch包裹，直接方法上throw 出去，提升可读性，减少语法噪音 5](#_Toc23459)

[10.6. 数据结构参照标准规范，方便文档对照 各种微格式等rss 5](#_Toc16749)

[10.7. 递归代替循环 6](#_Toc31052)

[10.8. 中缀表达式 优先于 前后缀表达式 6](#_Toc20964)

[11. 各种dsl 6](#_Toc21420)

[11.1. Ognl nodejs python 6](#_Toc18196)

[11.2. Guava等lib 6](#_Toc28750)

# 几大原则

## 直接原则，无脑原则。。。

## 避免过度设计

# 本地化命名法

## 可以使用管理命名法 多个api 比如old api，new api

## 相关方法使用前缀发命名空间

## 提高抽象级别，what与how相分离，优先指明what

## 命名规范 参考知名api

参考知名api 参考知名sdk 游戏cocos2d、等..

Sql style api

这样可以大大减少资料文档的编撰。。互联网上已经有了

## 使用命名空间，不支持命名空间的事业类似前缀

缺点就是命名长度变长了，单是可读性优先，会提升可读性，名字长度有ide自动补全缓解。。

# 管理代码存储区域 优先文件，函数 oo模式

## 静态模块最好了

## 其次静态方法 ，最后oo动态方法

# 减少嵌套与跳转

## 减少简化层次结构，单层其实可以满足大部分架构，其次双层，多层

## 一个流程代码不要跨文件，跳转过多

特别是脚本会很麻烦。。带来ide和大脑跳转过多。。

## 减少语法噪音嵌套使用lambda 与脚本

## 减少跳转 内部类也不错

## 匿名方法，匿名构造函数 初始化块，动态map模拟匿名类匿名方法 输出

匿名函数虽然没有名字，但也是可以有构造函数的，它用构造函数块来代替，那上面的3个输出就很清楚了：虽然父类相同，但是类还是不同的。

# 减少语法噪音

## Lambda

## 多使用static import导入机制。。注意函数名不要重复了。。

# 适当分类 命名空间模式

## 数据库使用分库提升可读性

## 命名空间前缀发

# 参数 使用通用接口 大力减少代码

# Dsl

## 表格映射表代替选择 决策树模式等

# 同步 法 优先于异步法

# Other

## 适当的事件驱动法

## 异常模式代替返回

## 方法链

## 字符串模板技术

## 不要使用try catch包裹，直接方法上throw 出去，提升可读性，减少语法噪音

结构可读性

分层

## 数据结构参照标准规范，方便文档对照 各种微格式等rss

数据库的schedu\_info

日程的ics

邮件eml

文章 meatawebblog

通讯录 vcf

## 递归代替循环

## 中缀表达式 优先于 前后缀表达式

# 各种dsl

## Ognl nodejs python

## Guava等lib

Atitit readablity enhance art 可读性的艺术v4 t99.docx

Atitit. 提升可读性推荐标准规范解决方案 关于编程语言的v5 docx

Atitit. 提升可读性推荐标准规范解决方案 关于编程语言的v5 docx

1. 提升可读性的意义 1

2. 提升可读性大原则： 2

2.1. 分解 分类 层次结构 2

2.2. 命名规范推荐标准 2

2.3. 关注点分离原则 2

2.4. 面向人类编程，优先于面向机器，可读性优先于性能原则 2

3. 具体措施 2

3.1. （推荐）Dsl \*\*重要 2

3.2. \*\*（推荐）使用汉字命名，获取更大的可读性，适合于绝大多数项目利大于弊（推荐） 2

3.3. （推荐）使用命名空间，不支持命名空间的事业类似前缀 3

3.4. （推荐）有时候异常处理也会提升可读性 3

3.5. （推荐）限制使用spring等框架范围，防止滥用 3

3.6. （推荐）提高抽象级别，what与how相分离，优先指明what 3

3.7. （推荐）减少架构层次，双层比三层四层架构更加简单易读 3

3.8. （推荐）注意学院派与工程派完全不同 3

3.9. 命名规范 参考知名api 3

3.10. （推荐）Sql style api 4

3.11. 适当分层、DI和AOP是继OO之后的分解方法 4

3.12. 函数式样 流程控制全部函数化 4

3.13. 递归代替循环 4

3.14. 中缀表达式 优先于 前后缀表达式 5

3.15. 防止出现大量接口，，接口过多过滥 5

3.16. 减少 嵌套级别 5

3.17. （推荐）使用模板法，关注点分离。。字符串拼接太难读怎么办？？ 5

3.18. 减少“语法噪音” 5

3.19. 参考 5

4. Refactor 5

4.1. 方法链 5

5. 参考资料 5

Atitit 可读性技术与实践范例 艾提拉著