Atitit dev eff fix v2 开发效率补充哦哦那个、、

目录

[1. 几大原则 1](#_Toc27444)

[1.1. 简单简约 1](#_Toc48)

[1.2. 可读性高 优先给人 1](#_Toc16289)

[2. Fix 2](#_Toc5652)

[3. 高效率的软件开发原则 2](#_Toc25751)

[3.1. .1. 可读性优先原则 4 2](#_Toc4838)

[3.2. 5.2. 拥抱变化原则 扩展性 4 2](#_Toc19488)

[3.3. 5.6. Keep It Simple, Stupid (KISS) 4 2](#_Toc24303)

[3.4. 5.12. 避免过早优化原则 5 2](#_Toc2618)

[3.5. 5.22. 首先为人类设计，其次为机器设计。 6 2](#_Toc16296)

[4. 管理上 政策 3](#_Toc4642)

[5. 快速部署运维 3](#_Toc14420)

[5.1. 热部署 在线修改 3](#_Toc19208)

[6. 可扩展性 3](#_Toc10365)

[7. 架构方法上选择快速开发的架构 3](#_Toc7967)

[8. 大力提升可读性 4](#_Toc24362)

[9. 开发规范上出马快速开发规范 4](#_Toc101)

[10. 编程方法上选择快速的编程范式和编程方法 5](#_Toc20752)

[11. 高效率开发流程选择 6](#_Toc30886)

[11.1. 敏捷开发 4 6](#_Toc18043)

[11.2. 持续集成 ci 6](#_Toc26036)

[11.3. 持续交付 / 部署 (DevOps) 8 6](#_Toc22354)

[12. 使用高效率简单的类库与中间件 6](#_Toc27532)

[13. 高效率设计法 设计模式 6](#_Toc11982)

[13.1. 设计上要抽象一级 参数化 6](#_Toc29605)

[13.2. 最少接口原则 6](#_Toc3577)

[13.3. Wrap包装法 6](#_Toc1637)

[14. 开发方法论上需要快速的方法 6](#_Toc279)

[15. Ref 7](#_Toc30352)

# 几大原则

## 简单简约

## 可读性高 优先给人

# Fix

Atitit enhance readable naming req 命名规范 双名法

# 高效率的软件开发原则

## .1. 可读性优先原则 4

## 5.2. 拥抱变化原则 扩展性 4

5.3. ：迪米特法则最少知识原则”（Principle of Least Knowledge） 4

5.4. 迪米特法则（Law ofDemeter, LOD） 4

## 5.6. Keep It Simple, Stupid (KISS) 4

5.8. You Ain’t Gonna Need It(YAGNI) 4

5.9. 最小惊讶原则 遵守标准约定注释说什么代码就做什么 5

5.10. 最小化耦合原则与最大内聚 5

## 5.12. 避免过早优化原则 5

5.13. 得墨忒耳定律 5

5.17. High Cohesion & Low/Loose coupling & –高内聚，低耦合 6

5.18. Convention over Configuration（CoC）–惯例优于配置原则 惯例优先原则 6

5.21. 软件，就像所有技术一样，具有天然的政治性。代码必然会反映作者的选择、偏见和期望。 6

## 5.22. 首先为人类设计，其次为机器设计。 6

5.23. 抽象原则 6

6. 《编程法则》二十一条 6

7. Recy 7

7.1. Command-Query Separation (CQS)–命令-查询分离原则 7

7.3. Design by Contract (DbC)–契约式设计

# 管理上 政策

考勤 弹性工作制

Ted沟通交流

目标导向为主 绩效考核

技术决策放手

推荐快速规范法

应该避免的慢速方法

语言（java php js python ）名录法，解决不规范问题，但是适当放宽范围，不要完全限定死

类库 名录

# 快速部署运维

## 热部署 在线修改

免编译

# 可扩展性

内嵌dsl法

抽象化 参数化提升灵活性

Url param 转换法dsl法

# 架构方法上选择快速开发的架构

Atitit 提升开发效率 cs bs架构的选择

目录

1. 集成模式 1

1.1. 集成公共现有软件，比如ftp flashftp，播放器potplayer 1

1.2. Hyberide模式 1

2. 内网app 还是外网app 1

3. 某些时候cs模式更加快速 1

4. Bs三个重要的技术标准 1

4.1. H5 ui标准 通讯 http 数据存储标准sql 1

Atitit arch design context软件架构设计的内容

目录

1. 考虑到架构设计原则 1

1.1. 开发效率 稳定性 可靠些等 性能 1

1.2. 简单原则则 1

1.3. 配置化 vs 开发 2

2. 云平台vs自建 2

2.1. Rds mysql 2

3. 开发语言选项 4gl优先 2

4. 免编译免部署 配置化 2

4.1. Db oritd模式优先 2

4.2. Table oritd vs java oritd 2

5. 架构模式 单体 vs 分布式 2

5.1. 模块化 井字模式 2

5.2. 通用化模块 vs 业务相关模块 2

6. 分层层次数 双层 》 三层 》多层 3

7. 类库选择 简单优先 3

8. 数据库选择sql erver vs mysql 3

9. Ati的架构演化路线 3

9.1. 淘宝的架构演化路劲 3

# 大力提升可读性

Atitit enhance dev eff read req提升开发效率 可读性规范

# 开发规范上出马快速开发规范

Atitit 高效率开发规范

# 编程方法上选择快速的编程范式和编程方法

Atitit 编程范式之道 attilax著 v2 u66.docx

Atitit 编程范式之道 attilax著 著

1. 编程范式与编程语言的关系是什么？ 2

2. 高效率的编程范式 2

2.1. DP(Declarative Programming)描述性范式 2

2.1.1. 俩种实现模式 LP逻辑编程 FP 函数式编程 2

2.2. LOP 面向语言编程（LOP, Language Oriented Programming） 2

2.3. AOP 3

2.4. 泛型式、元编程、切面式和事件驱动式。 3

2.5. 1.2.5. MP(Meta Programming) 6 2. Table-oriented Programming 7 3

3. OOP 3

4. OOP 3

5. Fp 函数式编程 3

6. 命令式 Imperative  3

7. 其他 3

7.1. 泛型式、 3

7.2. 产生式编程（Generative Programming） 自动生成源代码的编程也属于另一种编程范式 3

7.3. 过程化（命令式）编程  4

7.4. 事件驱动编程  ]响应式编程范式-- 4

7.5. 4个常用的编程范式：泛型式、元编程、切面式和事件驱动式。 4

8. 元编程 原来元编程就是编写能自动生成源代码的程序。" 4

8.1. 通过向导、拖放控件等方式自动生成源码 4

8.2. 产生式编程（Generative Programming 区别 4

8.3. 许多脚本语言都提供eval函数，可以在运行时将字符串作为表达式来运算[4]。 5

9. Other 5

9.1. Atitit 编程范式总结 5

10. 参考资料 6

10.1. 编程范式，程序员的编程世界观 -- 简明现代魔法.html 6

10.2. 3.2 超级范式--提升语言的级别（2） - 51CTO.COM.mhtml 6

10.3. 编程范式思考问题 - huangshanchun的专栏 - CSDN博客.mhtml 6

10.4. Atitit 编程范式总结 v2 taf.docx 6

# 高效率开发流程选择

## 敏捷开发 4

## 持续集成 ci

## 持续交付 / 部署 (DevOps) 8

IPD(Integrated Product Development）作为完善的产品开发模式、理念和方法

# 使用高效率简单的类库与中间件

Webso cket workman

Db jpa sql

# 高效率设计法 设计模式

## 设计上要抽象一级 参数化

## 最少接口原则

库表查询 操作，web api读写 文件读写的等

## Wrap包装法

# 开发方法论上需要快速的方法

Atitit各种驱动的xdd tdd bdd设计 ATDD ddd v3 u66.docx

Atitit各种驱动的xdd tdd bdd设计 ATDD ddd v2 s66

开发方法论与效率提升

Atitit各种驱动的xdd tdd bdd设计 ATDD ddd

Tdd bdd 行为驱动

目录

1. 2. 软件开发过程中最常见的两个问题 1

1.1. 需求和开发脱节： 1

1.2. 开发和测试脱节： 2

2. 高效率的开发范式 开发方法论 2

2.1. Xp 2

2.2. 完美的组合是TDD，DDD和BDD 2

2.3.  FDD (Feature-driven Development)  2

2.4. BDD 指的是Behavior Drive Development 2

3. Xdd "X" Driven-Development Methodologies 2

3.1.  DDD (Defect-Driven Development) – 3

3.2. RDD (Responsibility-Driven Design) – 3

3.3. UGDD (User Guide-Driven Development) 3

3.4.  MDD (Model-driven Development)  3

3.5. DDD (Documentation-Driven Development) 3

3.6. Edd 事件驱动 3

3.7. TFD -- Test First Development) 3

3.8. 2. BDD 指的是Behavior Drive Development，也就是行为驱动开发 3

3.9.  DDD DDD指的是Domain Drive Design，也就是领域驱动开发 3

3.10. CBD（核心Core+行为Behavior+驱动Driver）架构模式 4

3.11. TDD（测试驱动开发(Test-Driven Development)） 4

3.12. ATDD：验收测试驱动开发（Acceptance Test Driven Development） 4

4. 区别 5

4.1. 简而言之，完美的组合是TDD，DDD和BDD 5

5. ref 7

# Otehr

## View简化开发提升代码效率

## 重构兼容老的表结构。 view技术

## 适当的view 快速接口技术

3）安全性需要，主要是早期遗留系统集成需要，如：需要xx系统数据，又没接口，怎么办？直接操作数据库，这时就涉及数据安全性，合理利用视图则可以减少很多授权工作和保证数据安全性

# Ref

软件交付的演进历程 - InfoQ