Atitit 提升扩展性指标与方法总结

目录

[1. 扩展性常见指标 1](#_Toc30374)

[1.1. 代码简洁，应业务变动调整修改少 1](#_Toc13295)

[1.2. 免编译 1](#_Toc1827)

[1.3. 是否支持热部署 2](#_Toc26699)

[1.4. 其他。。。 2](#_Toc180)

[2. 常见的方法策略 2](#_Toc24394)

[2.1. 尽可能转移提升语言级别到4gl 2](#_Toc31176)

[2.2. 语言的代际级别关系表 2](#_Toc15995)

[2.3. 内嵌解释器 表达式解释器等 2](#_Toc20236)

[2.4. 尽可能配置化动态化 2](#_Toc28621)

[2.5. Scheme less Scheme free模式 多使用弱类型结构json等 3](#_Toc2777)

[2.6. 对于编译类型语言，多使用map等动态结构 3](#_Toc9142)

[2.7. 中间层通道化 驱动化 3](#_Toc13646)

[2.8. 无中心节点，避免庞大组件出现 3](#_Toc23918)

[2.9. Ioc 回调模式 3](#_Toc14857)

[2.10. 尽可能简洁减少代码数量，特别静态语言类 3](#_Toc8759)

[3. 其他一些扩展性方法 3](#_Toc14556)

[3.1. 适当使用写 设计模式 3](#_Toc27602)

[3.2. 适当抽象 4](#_Toc240)

[3.3. 产品化 为“普罗大众”去设计，而不是为“个例”设计 4](#_Toc16261)

[4. 提升可读性 4](#_Toc12849)

[4.1. 本地化 4](#_Toc5723)

[4.2. Dsl 自然语言化 4](#_Toc7935)

[4.3. 控制转移法 比如查表法代替if else判断 4](#_Toc26775)

[4.4. 多同步法 少异步 4](#_Toc6260)

[5. ref 4](#_Toc32098)

# 扩展性常见指标

## 代码简洁，应业务变动调整修改少

## 免编译

尽可能做到业务功能变动，代码调整，无需编译。。避免了带来的打包与热部署问题

## 是否支持热部署

调整代码无需重启服务器服务程序

## 其他。。。

# 常见的方法策略

## 尽可能转移提升语言级别到4gl

语言级别越高，可读性越高，代码越简洁，扩展性越高

语言的趋势也是趋向于免编译，动态性，可读性等

## 语言的代际级别关系表

|  |  |  |
| --- | --- | --- |
| 语言代际与级别 | 具体常见语言 | 备注 |
| 5gl | 自然语言等 | 进展很大，但还不大成熟。。 |
| 4gl | Sql h5 xml yml | 常见实现 规则引擎bpm工作流等 |
| 3.5g | Script系列 js python php等 |  |
| 3gl | Java c#等需要编译的语言 |  |
| 2gl | Asm类 |  |
| 1gl | 机器语言 |  |
|  |  |  |

## 内嵌解释器 表达式解释器等

Ongl express等 js解释器等 sql解析器（sql驱动）

## 尽可能配置化动态化

## Scheme less Scheme free模式 多使用弱类型结构json等

常见的弱类型结构还有H5 xml等

## 对于编译类型语言，多使用map等动态结构

Ui收集参数可使用request的map结构。。与后端orm工具交互参数也是map 等

## 中间层通道化 驱动化

## 无中心节点，避免庞大组件出现

## Ioc 回调模式

主要用在静态语言里面

但ioc滥用可能会影响可读性，在3.5gl 4gl 动态语言里面使用少

## 尽可能简洁减少代码数量，特别静态语言类

简洁意味着数量少，即使修改也修改范围小，工作量少

层次简化，减少不必要的类数量

# 其他一些扩展性方法

## 适当使用写 设计模式

架构解耦和测试的工具，以及各种设计模式解决方案，比如[依赖注入](http://en.wikipedia.org/wiki/Dependency_injection)（dependency injection）、[观察者模式](http://en.wikipedia.org/wiki/Observer_pattern)（observer pattern）、[中介者模式](http://en.wikipedia.org/wiki/Mediator_pattern)（mediator pattern）、[外观模式](http://en.wikipedia.org/wiki/Facade_pattern)（facade pattern）、[命令模式](http://en.wikipedia.org/wiki/Command_pattern)（command pattern）

## 适当抽象

## 产品化 为“普罗大众”去设计，而不是为“个例”设计

# 提升可读性

## 本地化

## Dsl 自然语言化

## 控制转移法 比如查表法代替if else判断

## 多同步法 少异步

# ref

Atitit 项目高扩展法.docx