Atitit 架构之提升扩展性问题

目录

[1. 扩展性大原则 2](#_Toc22910)

[1.1. 适当向上抽象 更加通用 2](#_Toc24896)

[1.2. 产品化 为“普罗大众”去设计，而不是为“个例”设计 2](#_Toc13288)

[2. 扩展性指标 3](#_Toc8475)

[2.1. 前瞻性设计，接口不只是为了已知的业务，还为未来的未知业务而服务 3](#_Toc302)

[2.2. 方便修改，容易添加某个业务功能尽可能无需改动程序代码 或少量改动 3](#_Toc9313)

[2.3. 修改某个部分避免跨越太多多文件 3](#_Toc307)

[2.4. 动态化 实现少编译免编译 热部署 或免部署 3](#_Toc9580)

[2.5. 松耦合(ioc) 业务无关性 3](#_Toc7907)

[2.6. 库表结构自动化扩展，随时添加字段 修改字段名 而无需调整程序代码 3](#_Toc32140)

[2.7. 通用化接口，少量接口实现大量的业务功能，接口业务无关性 3](#_Toc18455)

[2.8. DSL声明化（不具体指明细节），结果导向模型 4](#_Toc7635)

[3. Top3 是三件套dsl业务脚本 通用接口 半结构化数据 4](#_Toc433)

[3.1. 无中心节点组件设计 避免庞大组件出现， 4](#_Toc22356)

[3.2. 分段建设，避免大节点 4](#_Toc358)

[3.3. 单一用途&模块化，小粒度化 4](#_Toc13870)

[3.4. 组合（Composition），而不是继承（inheritance） 4](#_Toc8979)

[3.5. 适当使用写 设计模式 4](#_Toc3608)

[3.6. 利用db和java语言 os操作系统等提供的扩展机制api 5](#_Toc32153)

[4. dsl业务脚本 工作流BPM 规则引擎 5](#_Toc3160)

[4.1. 提升语言级别与抽象DSL 5](#_Toc15235)

[4.2. 各种QL查询语言 各种EL表达式语言 5](#_Toc573)

[4.3. 内嵌解释器 表达式解释器等 5](#_Toc15885)

[4.4. 脚本。脚本是扩展复杂功能的利器php python js等 5](#_Toc29845)

[4.5. 表映射表处理扩展if else 5](#_Toc1352)

[4.6. Sql脚本 sp view等 5](#_Toc8123)

[4.7. 频繁变动的业务配置化 脚本化 配置业务脚本 5](#_Toc28268)

[5. 通用接口参数化（查询和修改） 6](#_Toc23856)

[5.1. 类似于sql一样的通用接口 6](#_Toc8592)

[5.2. Socket接口 转rest 6](#_Toc32475)

[5.3. 类似usb接口那样通用 6](#_Toc4712)

[5.4. 接口转换器 adapter模式 6](#_Toc5474)

[6. 半结构化数据 与动态类型 6](#_Toc21438)

[6.1. 数据扩展 schema free , schema less模式 半结构化数据表示 6](#_Toc575)

[6.2. 对于编译类型语言，多使用map等动态结构 6](#_Toc15280)

[6.3. 数据扩展性 数据库适当Json字段提升扩展性 7](#_Toc31606)

[7. 数据库系统提供的扩展性机制 7](#_Toc1797)

[7.1. 扩展Sql（带流程控制） xxsql Tsql Plsql等 7](#_Toc5666)

[7.2. View 于重写sql模式 7](#_Toc23580)

[7.3. Sp udf trigger（拦截器机制） 7](#_Toc22059)

[7.4. Db timer（mysql 里面叫event事件机制） 7](#_Toc21046)

[7.5. Json数据类型方便可扩展 7](#_Toc29921)

[7.6. 调用外部代码交互（mysql不直接支持） 7](#_Toc2091)

[7.7. Shell api （mysql不支持） 7](#_Toc23898)

[7.8. Other 7](#_Toc28118)

[8. 编程语言一般提供的扩展性机制 7](#_Toc5306)

[8.1. 脚本解释器 api 解释器(或脚本引擎scripting engine 7](#_Toc15792)

[8.2. Cli接口 Shell解释器 api 8](#_Toc13784)

[8.3. Sql解释器（通过sql驱动） 8](#_Toc20596)

[8.4. 正则与其他QL EL解释器 8](#_Toc8916)

[8.5. 通过其他dsl类库解释器模式等 8](#_Toc712)

[8.6. 配置与元数据解释器api 8](#_Toc25758)

[8.7. 流程解释器 规则解释器 lib 8](#_Toc25913)

[8.8. Timer定时机制 事件机制 8](#_Toc12160)

[8.9. Vm拦截器机制过滤器机制 插件机制等（例如Java Agent ） 8](#_Toc28943)

[9. 利用os操作系统提供的扩展机制 9](#_Toc32039)

[9.1. Shell api 9](#_Toc4289)

[9.2. Timer机制（linux crontab，win 定时器等） 9](#_Toc28607)

[9.3. 拦截器机制 9](#_Toc276)

[9.4. 其他（系统托盘通知等 9](#_Toc3866)

[10. 其他 10](#_Toc30761)

[10.1. 表格化 excel 10](#_Toc14340)

[10.2. 观察者模式 事件驱动 mq 10](#_Toc10762)

[11. 功能扩展 vs 性能扩展 10](#_Toc5929)

[12. Ref 10](#_Toc15978)

# 扩展性大原则

## 适当向上抽象 更加通用

类似于分类法

## 产品化 为“普罗大众”去设计，而不是为“个例”设计

# 扩展性指标

## 前瞻性设计，接口不只是为了已知的业务，还为未来的未知业务而服务

## 方便修改，容易添加某个业务功能尽可能无需改动程序代码 或少量改动

改动范围尽可能的小 ，

## 修改某个部分避免跨越太多多文件

## 动态化 实现少编译免编译 热部署 或免部署

## [松耦合](http://en.wikipedia.org/wiki/Loose_coupling)(ioc) 业务无关性

soma.js中提供了一系列用于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），[面向对象](http://en.wikipedia.org/wiki/Object-oriented_programming)（OOP）工具集，并提供了一个DOM操作模板引擎作为可选插件。

## 库表结构自动化扩展，随时添加字段 修改字段名 而无需调整程序代码

## 通用化接口，少量接口实现大量的业务功能，接口业务无关性

参数尽可能配置化 指明要执行的业务脚本（或函数）名称以及对应参数

## DSL声明化（不具体指明细节），结果导向模型

# Top3 是三件套dsl业务脚本 通用接口 半结构化数据

## 无中心节点组件设计 避免庞大组件出现，

类似于主板一样，接口众多，难依然难以升级

因为一旦升级，就要牵扯到众多接口同时升级。。

架构扩展性

## 分段建设，避免大节点

## 单一用途&模块化，小粒度化

粒度更小，更容易扩展

## 组合（Composition），而不是继承（inheritance）

## 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）

## 利用db和java语言 os操作系统等提供的扩展机制api

# dsl业务脚本 工作流BPM 规则引擎

## 提升语言级别与抽象DSL

## 各种QL查询语言 各种EL表达式语言

完整脚本语言node.js python php

不完整语言xml h5 +vue系列

图形化语言bpm

## 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）

Python php 等业务脚本解释器

## 脚本。脚本是扩展复杂功能的利器php python js等

## 表映射表处理扩展if else

(数据与数据处理相互分离）

## Sql脚本 sp view等

## 频繁变动的业务配置化 脚本化 配置业务脚本

# 通用接口参数化（查询和修改）

## 类似于sql一样的通用接口

## Socket接口 转rest

## 类似usb接口那样通用

## 接口转换器 adapter模式

# 半结构化数据 与动态类型

## 数据扩展 schema free , schema less模式 半结构化数据表示

## 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

## 数据扩展性 数据库适当Json字段提升扩展性

# 数据库系统提供的扩展性机制

## 扩展Sql（带流程控制） xxsql Tsql Plsql等

## View 于重写sql模式

## Sp udf trigger（拦截器机制）

## Db timer（mysql 里面叫event事件机制）

## Json数据类型方便可扩展

## 调用外部代码交互（mysql不直接支持）

## Shell api （mysql不支持）

可以自定义实现，需要通过外部代码 类似mq模式调用实现

## Other

动态 SQL

# 编程语言一般提供的扩展性机制

## 脚本解释器 api 解释器(或脚本引擎scripting engine

 javax.script 包使集成动态语言更加容易。通过使用一小组接口和具体类，这个包使我们能够简单地调用多种脚本语言。但是，Java 脚本 API 的功能不只是在应用程序中编写脚本；这个脚本包使我们能够在运行时读取和调用外部脚本，这意味着我们可以动态地修改这些脚本从而更改运行应用程序的行为。

## Cli接口 Shell解释器 api

## Sql解释器（通过sql驱动）

## 正则与其他QL EL解释器

## 通过其他dsl类库解释器模式等

## 配置与元数据解释器api

## 流程解释器 规则解释器 lib

挂接工作流 规则引擎lib api。。

## Timer定时机制 事件机制

## Vm拦截器机制过滤器机制 插件机制等（例如Java Agent ）

Java Agent ？

笼统地来讲，Java Agent 是一个统称，该功能是 Java 虚拟机提供的一整套后门。通过这套后门可以对虚拟机方方面面进行监控与分析。甚至干预虚拟机的运行。

Java Agent 又叫做 Java 探针，Java Agent 是在 JDK1.5 引入的，是一种可以动态修改 Java 字节码的技术。Java 类编译之后形成字节码被 JVM 执行，在 JVM 在执行这些字节码之前获取这些字节码信息，并且通过字节码转换器对这些字节码进行修改，来完成一些额外的功能，这种就是 Java Agent 技术。

从用户使用层面来看，Java Agent 一般通过在应用启动参数中添加 -javaagent 参数添加 ClassFileTransformer 字节码转换器。 在 Java 虚拟机启动时，执 行main() 函数之前，Java 虚拟机会先找到 -javaagent 命令指定 jar 包，然后执行 premain-class 中的 premain() 方法。用一句概括其功能的话就是：main() 函数之前的一个拦截器。

# 利用os操作系统提供的扩展机制

## Shell api

## Timer机制（linux crontab，win 定时器等）

## 拦截器机制

## 其他（系统托盘通知等

操作日志 api

脚本解释器 host宿主

# 其他

## 表格化 excel

## 观察者模式 事件驱动 mq

# 应用解耦

## Sprigoot ioc 工厂模式

## 观察者模式 Mq

# 功能扩展 vs 性能扩展

udf机制 sp 等插件机制

1.5. 脚本。脚本是扩展复杂功能的利器 2

1.6. 自定义语言 工作流 规则引擎 2

交互扩展

提供细粒度api

提供交互接口比如cli

# Ref

atitit 架构结构改造规划文档v5

atitit 高扩展性解决方案 功能扩展法