Atitit 提升开发效率几大策略

目录

[1. 提升效率三原则 3](#_Toc3982)

[1.1. 更少的代码量简化 3](#_Toc20788)

[1.2. 优化配置减少等待 3](#_Toc32495)

[1.3. 提升一次性处理能力 3](#_Toc15210)

[2. 重要原则（业界归纳） 3](#_Toc7137)

[2.1. Don’t Repeat Yourself(DRY) 3](#_Toc17394)

[2.2. Keep It Simple, Stupid (KISS) 3](#_Toc3537)

[2.3. You Ain’t Gonna Need It(YAGNI) 3](#_Toc20142)

[2.4. Common Reuse Principle (CRP)–共同重用原则 3](#_Toc17044)

[3. 目前存在的情况 3](#_Toc19868)

[4. 实现目标 4](#_Toc6599)

[5. 提升语言级别到4gl （对开发效率数量级提升） 4](#_Toc18734)

[5.1. 语言的代际关系 （4gl）sql 》（3gl）script java net c# 4](#_Toc11317)

[5.2. 使用4gl dsl语言与api 4](#_Toc26778)

[5.3. 免编译 多使用脚本语言js一类 5](#_Toc5995)

[5.4. 动态化 5](#_Toc24998)

[6. 架构层次的优化 方法优化最简化框架 5](#_Toc14047)

[6.1. 后端通道化 servless模式 中间层通道化驱动化 5](#_Toc13227)

[6.2. 免部署热部署 多使用数据库提供的sp等机制 6](#_Toc16210)

[6.3. 简化设计 优先使用简单设计与实现 6](#_Toc28652)

[6.4. 适当减少层次bs cs 双层 三层 6](#_Toc5726)

[6.5. 优先使用配置化代替编码化，大力减少编码 6](#_Toc8568)

[7. 数据传输与存储层面的优化 7](#_Toc20297)

[7.1. Scheme free模式 多使用nosql json 半结构化数据 7](#_Toc14630)

[7.2. Mysql5.7以上可多多 使用json数据 7](#_Toc16275)

[7.3. 子母表设计可使用json集合模式等 7](#_Toc18031)

[7.4. 适当的反范式设计 7](#_Toc28894)

[7.5. 可以跨库调用，可以把次模式看成看成一个调用socket非文本模式接口即可（ 通过数据库驱动） 7](#_Toc19514)

[8. 前后端交互 7](#_Toc8986)

[8.1. 通用的数据操作模型 7](#_Toc14542)

[8.2. 使用json或http param 7](#_Toc26915)

[8.3. Sql（默认存在注入问题，可以通过sqlparse来解决） 8](#_Toc646)

[9. 效率提升总结 8](#_Toc31531)

[10. 存在的改进空间 9](#_Toc14078)

[10.1. 使用java js 等语言来扩展sql的问题 9](#_Toc30126)

[11. -------------------其他---------------- 9](#_Toc30930)

[12. 提升可读性 9](#_Toc12478)

[12.1. 语句通顺 9](#_Toc2083)

[12.2. Sql式命名法 dsl命名法 9](#_Toc31460)

[12.3. 汉字》拼音》英文（复杂名称不推荐使用纯英文） 9](#_Toc5250)

[12.4. 综合混杂模式各取所需 9](#_Toc7155)

[13. 工具 10](#_Toc19185)

[13.1. All in one 10](#_Toc26488)

[13.2. 内嵌web sesrver （比如springboot一类的） 10](#_Toc26896)

[13.3. 单元测试junit main运行 10](#_Toc9136)

[13.4. Ide db view 10](#_Toc9921)

[14. 其他方法提升 10](#_Toc28974)

[14.1. 分类优先级，与分区域，减少要操作或提取的内容与范围 10](#_Toc4538)

[14.2. 1.5. div分而治之 聚沙成塔 分布式 并发 2 10](#_Toc140)

[14.3. 1.6. 抓大放小 3 10](#_Toc4836)

[14.4. 简化设计 10](#_Toc3813)

[14.5. 分支化开发法，防止冲突干扰 10](#_Toc21368)

[15. 沟通与反馈 11](#_Toc9964)

[15.1. 集中式开发 11](#_Toc10295)

[15.2. 适当的全栈 11](#_Toc21460)

[15.3. 每日会议 daily report 11](#_Toc6961)

[15.4. Train and ted 11](#_Toc7275)

[16. other 11](#_Toc12535)

[17. ref 11](#_Toc4116)

# 提升效率三原则

## 更少的代码量简化

## 优化配置减少等待

## 提升一次性处理能力

# 重要原则（业界归纳）

## Don’t Repeat Yourself(DRY)

## Keep It Simple, Stupid (KISS)

## You Ain’t Gonna Need It(YAGNI)

这个原则简而言之为——只考虑和设计必须的功能，避免过度设计。只实现目前需要的功能，在以后您需要更多功能时，可以再进行添加。

## Common Reuse Principle (CRP)–共同重用原则

# 目前存在的情况

影响开发效率的一些情况

Sql使用不足，可以在mybatis mapper xml文件中使用多条语句sql模式，此外对数据库提供的提升开发效率机制还有很大使用空间（触发器sp视图事件等机制）

Bean vo interface太多 有点滥用了，可以适当多补充使用通用动态对象map 抽象类一类或其他简单api。。一般来说少量场合确实需要它们，但多数场合无需它们也可，可以混合使用组合扩展。

rest接口有些繁多，扩展性不足，每增加一个新业务操作就得要新增加接口 。。可以进行一些统一化 归类化 抽象提升方面的工作

Mq封装较繁琐（springboot本身对rocketmq的封装有些繁琐了），mq无法简单main函数或junit单元测试，可以自己简化封装

貌似Mybatisplus框架存在很大bug，稍微复杂sql无法解析报错，可以直接使用mybatis SqlSessionTemplate即可

有些mybatis 使用地方还是繁琐复杂过度设计（有很多mapper接口类）了，mybatis的使用SqlSessionTemplate更加精简，无需接口mapper类

实际开发中我们操作数据库持久化，总是需要写大量类似的的mapper，service浪费了我们大量的时间，在这里推荐大家使用SqlSessionTemplate

Spring data jpa Jpa hb的orm模式实现有些情况可能较繁琐，可以多使用它的那native sql api，这样基本与mybatis趋同了。

# 实现目标

表结构就是实体属性 对应。。所以原则上可以无需再重复建立bean实体，当然少数场合列外

快速增加 调整业务功能，尽可能免编译免部署（少编译少部署）

通用万能接口，以不变应万变，增加业务数量无需同步增加接口数量。一般对外接口数量在少数3--5个即可应付大量业务数量

# 提升语言级别到4gl （对开发效率数量级提升）

## 语言的代际关系 （4gl）sql 》（3gl）script java net c#

## 使用4gl dsl语言与api

优先多使用mybatis 里面sql，或者使用sp存储过程也可。。比如首充赠送金额活动可以这样

案例范例.xml

<select id="首充活动流程" resultType="map" parameterType="map">

set @userid=#{userid}

-- '216654713456709632';

-- 查询 count() from 流水表；

select count(\*) into @充值次数 from platform\_finance.fs\_flow\_t where user\_id=@userid and platform\_id=#{platform\_id};

-- select @userid,@充值次数,if(@充值次数=1,'是首充','非首充') 首充判断结果;

if @充值次数=1 then

select 首充赠送金额 into @首充赠送金额 from 活动表 where 类型='首充活动';

update 账本表 set 余额=余额+@首充赠送金额 where userid=@userid

insert 流水表 () values();

insert 操作日志表()values("");

--- 其他后记操作

select 1;

end if;

</select>

## 免编译 多使用脚本语言js一类

## 动态化

动态语言开发效率更高，动态对象map list objetc json等更简便

比如spring的动态代理

# 架构层次的优化 方法优化最简化框架

## 后端通道化 servless模式 中间层通道化驱动化

一个通用接口 解决常见所有通用接口（此接口也可以解决Update，delete等更新操作）

http://xxxxxxxx?selectId=查询xxxx数据&code=1111552&para=xxxxx

Controler.java

@Test  
@RequestMapping("/query")  
public Object query(HttpServletRequest req) {  
 return R.*data*(SqlSessionTemplate1.selectList(req.getParameter("selectId"), RequestUtil.*getMap*(req)));  
}

报表.xml

<!-- 查询每天统计余额 group by platform\_id -->  
<select id="查询xxxx数据" resultType="map" parameterType="map">  
  
 select platform\_id,*COUNT*(id) Total\_number\_of\_people,*sum*(balance) total\_balance,*sum*(balance\_safe) safebox\_total\_balance from fs\_book\_t group by platform\_id  
</select>

<select>标签里面也可以写update ,delete等更新语句，只要确保最后一个返回sql是查询即可

## 免部署热部署 多使用数据库提供的sp等机制

## 简化设计 优先使用简单设计与实现

Dbutil mybatis 》hb

## 适当减少层次bs cs 双层 三层

层次双层优先

## 优先使用配置化代替编码化，大力减少编码

比如使用数据库提供的unique索引代替插入判断重复检查等场合（比如可以应用在在注册检查用户名等，定时统计数据判重等场合），

使用数据库默认已经提供的约束等NOTnull等

使用mybatis等框架提供的约束等

注解配置等

# 数据传输与存储层面的优化

## Scheme free模式 多使用nosql json 半结构化数据

## Mysql5.7以上可多多 使用json数据

数据库 nosql mongodb 直接json 对用redis

## 子母表设计可使用json集合模式等

## 适当的反范式设计

范式设计主要是为了减少数据冗余，对存储空间简约有帮助，时过境迁，今天存储已经很便宜。。适当的反范式设计同时也可提升性能与开发效率

## 可以跨库调用，可以把次模式看成看成一个调用socket非文本模式接口即可（ 通过数据库驱动）

这样减少自己开发接口的工作量，相当于直接使用现有的通用数据访问接口。

或者开发一个通用数据访问接口也可（传入数据库url，用户名，密码 ,sql语句来连接，得到返回数据列表 或update 操作等）

# 前后端交互

目前情况，后端依然需要写较多的查询sql语句通过mybatis selectid。。这块依然可以简化。。大大减少了查询方面的，以及简单更新删除类的操作。。只有10%的复杂操作需要写了。。

## 通用的数据操作模型

## 使用json或http param

指明要操作的对象，字段。。。。

比如

|  |  |  |
| --- | --- | --- |
|  |  | 其他选项 |
| 操作 | 查询 |  |
| 对象 | Xxx | （其实基本对应表名，或view，或sp） |
| 参数 | xxx |  |
| 条件 | 字段a=111 AND 字段b=222 |  |
| 翻页参数 | Pagesize ,page等参数 | 仅限查询操作使用 |
| 其他 |  |  |

## Sql（默认存在注入问题，可以通过sqlparse来解决）

# 效率提升总结

|  |  |  |
| --- | --- | --- |
| 情况 | 解决 | 重要性 |
| 对象vo过多 | 通用对象 动态对象解决 | 高 |
| 业务扩展性不足 | 通用化，通用操作，通用接口，通用对象 | 高 |
| rest接口过多 | 通用接口 | 高 |
| 中间层代码过多 | 简化流程，直接打通ui到mybatis通道化 | 高 |
| Java代码过多 | 提升语言层级  业务逻辑迁移到mybatis xml sp等数据库机制 | 高 |
| 普通简单sql过多 | 使用查询语言解决 | 中 |
| 复杂逻辑 | 适当简化（） | 低 |
| 通用逻辑 | 尽可能使用数据库unique merge约束trigger触发器等现有功能来简化，配置化 | 高 |
| Java扩展sp | 自定义一个扩展解释器 | 高 |
| 减少编译与部署 | 多使用sql，尽可能少使用java，可以适当使用些脚本js 等 | 高 |
| 动态存储json | Mysql json字段机制 | 中 |
| 名称命名 | 多使用中文（包括java代码与数据库中代码与表结构等），直接免注释免注解 | 中 |

# 存在的改进空间

## 使用java js 等语言来扩展sql的问题

Oracle这类数据库直接支持java来写sp存储过程，可以进行数据库api之外的所有操作。

但是mysql暂时还没有这个机制。。

需要搞个这么个机制，使得数据库sql可以直接调用java类的业务功能。。

比如操作mq ，操作http接口,es等，还是需要使用java封装的的

可以写个通用sp解释器，专门实现这几类的操作扩充即可

# -------------------其他----------------

# 提升可读性

## 语句通顺

编程其实就是使用pl编程语言写文章，如果读起来不通顺，那就调整可以

## Sql式命名法 dsl命名法

## 汉字》拼音》英文（复杂名称不推荐使用纯英文）

大力使用中文，包括java代码，数据库sql代码，表结构命名等所有一切尽可能场合

可以与英文混用，附加在后面增强可读性，英文方便ide首键提示，综合起来

凡是英文不好理解或有歧义的，只要在一看之下想不起来对应的中文，那么久最好附加中文，当然简单英文可以单独使用不影响。。

如果一个英文需要查字典翻译才能理解，那么就最好附加中文在后面提高可读性

## 综合混杂模式各取所需

# 工具

## All in one

## 内嵌web sesrver （比如springboot一类的）

## 单元测试junit main运行

## Ide db view

# 其他方法提升

## 分类优先级，与分区域，减少要操作或提取的内容与范围

## 1.5. div分而治之 聚沙成塔 分布式 并发 2

## 1.6. 抓大放小 3

## 简化设计

## 分支化开发法，防止冲突干扰

# 沟通与反馈

## 集中式开发

## 适当的全栈

## 每日会议 daily report

## Train and ted

# other

# ref

Atitit 原生应用提升效率方法大总结 attilax总结

Atitit mybatis 简化开发与提升开发效率法.docx

Atitit 提升开发效率的方法 提升语言级别

Atitit.软件开发的几大规则，法则，与原则p821.doc

MyBatis+SpringBoot整合 注入SqlSessionTemplate - qq\_33579992的博客 - CSDN博客.html