Atitit api设计--------提升扩展性最简化设计

架构设计 扩展性 api设计提升总结

目录

[1. 简化层次与跳转 1](#_Toc13366)

[2. 接口通用化，大量简化 1](#_Toc13350)

[2.1. 查询资源也参数化 1](#_Toc2234)

[2.2. 查询接口 + 修改接口 2](#_Toc5635)

[2.3. 复合接口 2](#_Toc29838)

[2.4. 各种socket接口转rest接口 2](#_Toc3945)

[2.5. 各种二进制rpc接口转rest接口 2](#_Toc32341)

[3. 参数传递简化 动态化 2](#_Toc2637)

[3.1. 多使用动态接口 HttpServletRequest  2](#_Toc22177)

[3.2. 动态对象等 Map 2](#_Toc29088)

[3.3. 使用dsl最动态参数 2](#_Toc28322)

[4. Other 2](#_Toc25547)

[4.1. 使用stream 兼容高性能 3](#_Toc32048)

[4.2. 使用file 兼容目录与文件 3](#_Toc19990)

[4.3. 使用Consumer 兼容回调 高性能借口 3](#_Toc11538)

[4.4. 细粒度 api 3](#_Toc24641)

[4.5. 异常在方法定义 throw 3](#_Toc18112)

[4.6. 性能考虑，复用连接作为参数，，不要直接使用host u pwed 3](#_Toc462)

[4.7. Ref 3](#_Toc30959)

# 简化层次与跳转

简化跳转与文件数量

这样也方便扩展，修改文件数量地方比较少。。

# 接口通用化，大量简化

提升抽象少量接口满足大量业务需求

## 查询资源也参数化

## 查询接口 + 修改接口

## 复合接口

## 各种socket接口转rest接口

Mysql redis等 敏感接口需要脱敏，id化即可

## 各种二进制rpc接口转rest接口

# 参数传递简化 动态化

## 多使用动态接口 HttpServletRequest

Spring和Mock在单元测试中的使用

在某些方法中，为了减少代码量和提高程序的可读性，我们有时候需要直接传入 **HttpServletRequest** 或 **ServletContext** 对象，如果我们想对这种方法进行测试，可以利用**[Mock](http://mockito.org/)**来模拟相关的对象。

## 动态对象等 Map

## 使用dsl最动态参数

# Other

## 使用stream 兼容高性能

## 使用file 兼容目录与文件

## 使用Consumer 兼容回调 高性能借口

## 细粒度 api

## 异常在方法定义 throw

## 性能考虑，复用连接作为参数，，不要直接使用host u pwed

## Ref

Atitit lib api design 原则 类库设计原则