**aPaaS学习与总结**



**修订**

|  |  |  |  |
| --- | --- | --- | --- |
| **修订时间** | **修订者** | **角色** | **修改内容** |
| 2022-05-06 | 于明龙、汤太泉 | 作者 | 初稿 |
|  |  |  |  |
|  |  |  |  |

**目录**

[1 aPaaS代码编写 4](#_Toc103184600)

[1.1. 工程依赖 4](#_Toc103184601)

[1.2. 代码编写 5](#_Toc103184602)

[1.3. 自动生成接口测试效果 6](#_Toc103184603)

[2 aPaaS的学习与理解 8](#_Toc103184604)

[3 aPaaS的适用场景 9](#_Toc103184605)

[4 当前基础架构提供的低代码的实现方案 10](#_Toc103184607)

[4.1. 实现原理 10](#_Toc103184608)

[4.2. 自动生成的代码 10](#_Toc103184609)

# aPaaS代码编写

1. 工程依赖

* 首先使用脚手架生成工程

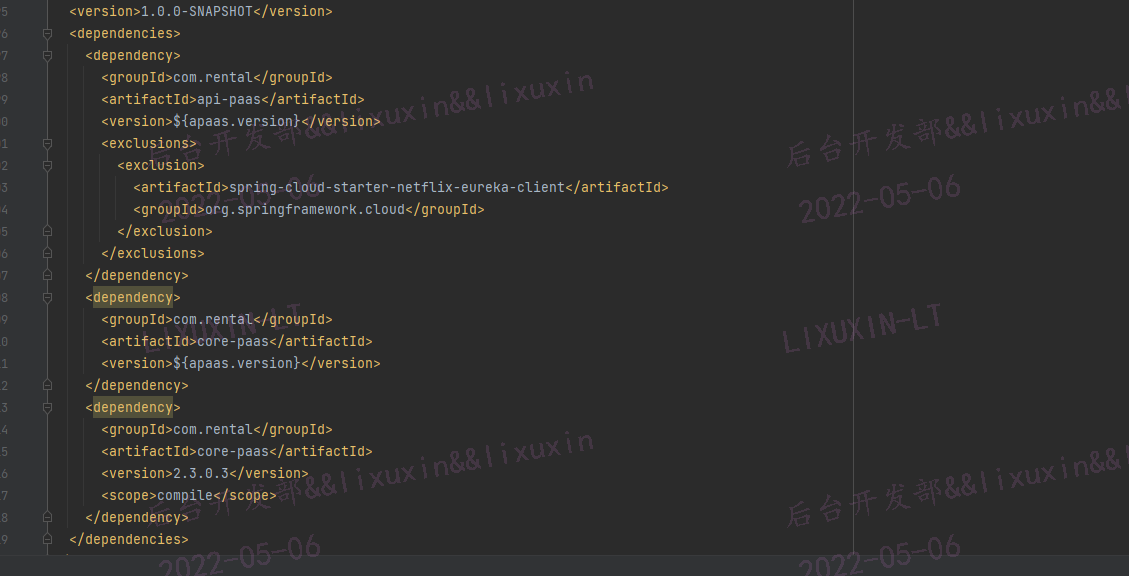
参照: [应用脚手架生成指南 - 平台化项目2.0 - Confluence (znlhzl.org)](https://wiki.znlhzl.org/pages/viewpage.action?pageId=28408083)

**groupId: com.znlh.kgl**

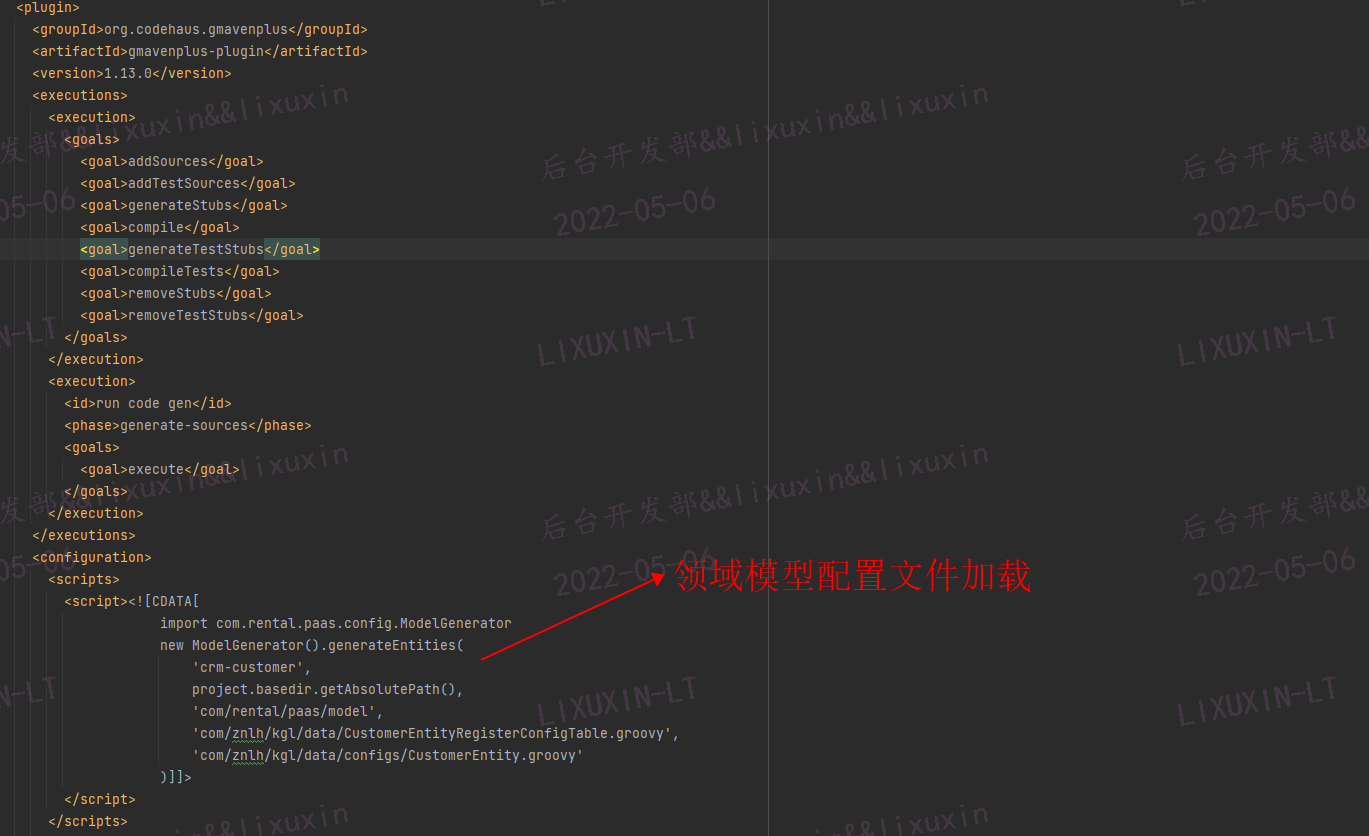
**artifactId: infrastructure-data**

**version: 1.0.0**

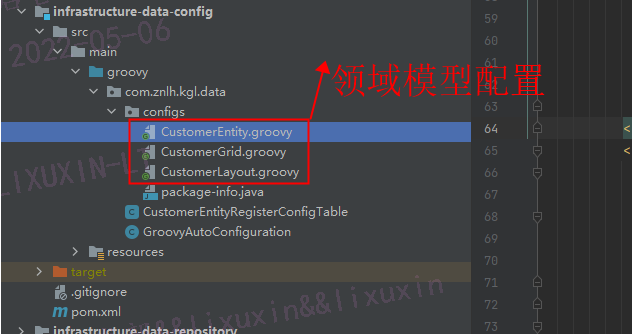
* 在pom中引入apaas相关依赖：



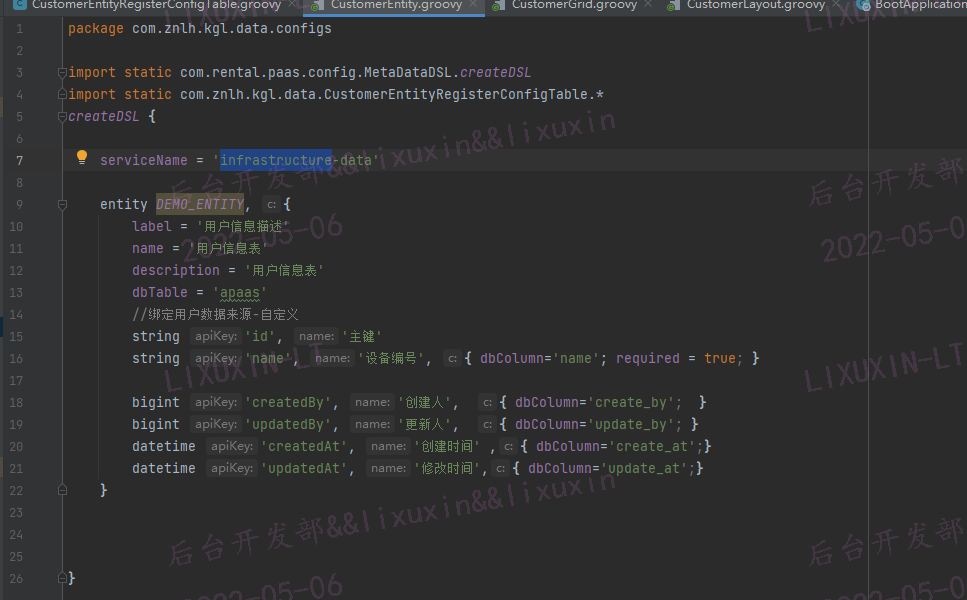
引入插件：



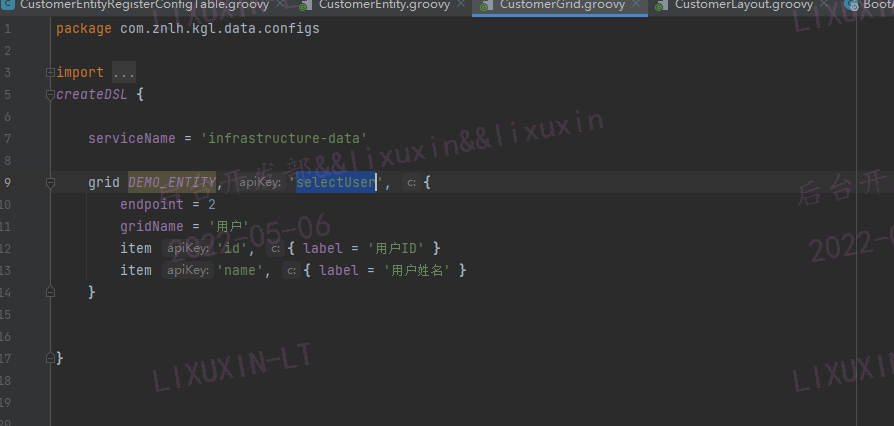
1. 代码编写



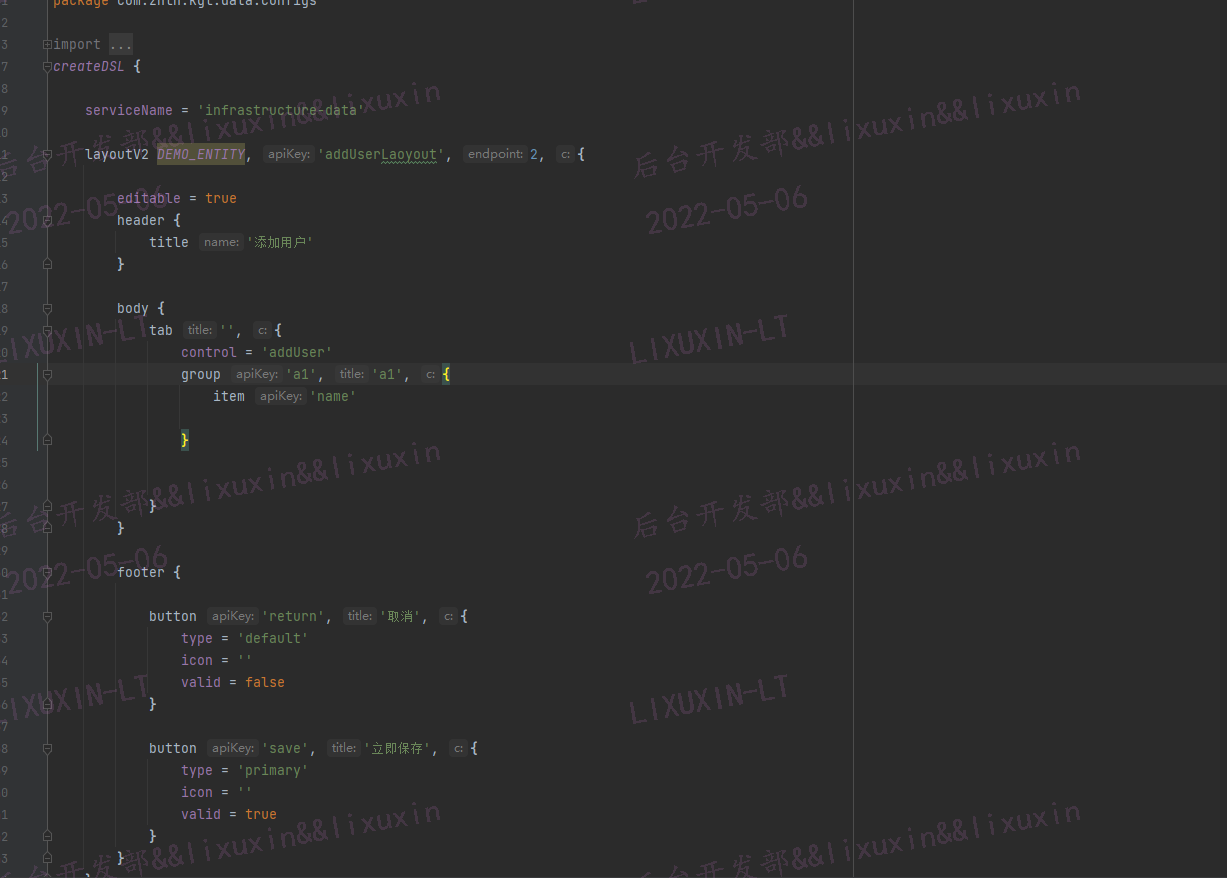
* 领域模型配置



* 列表布局配置

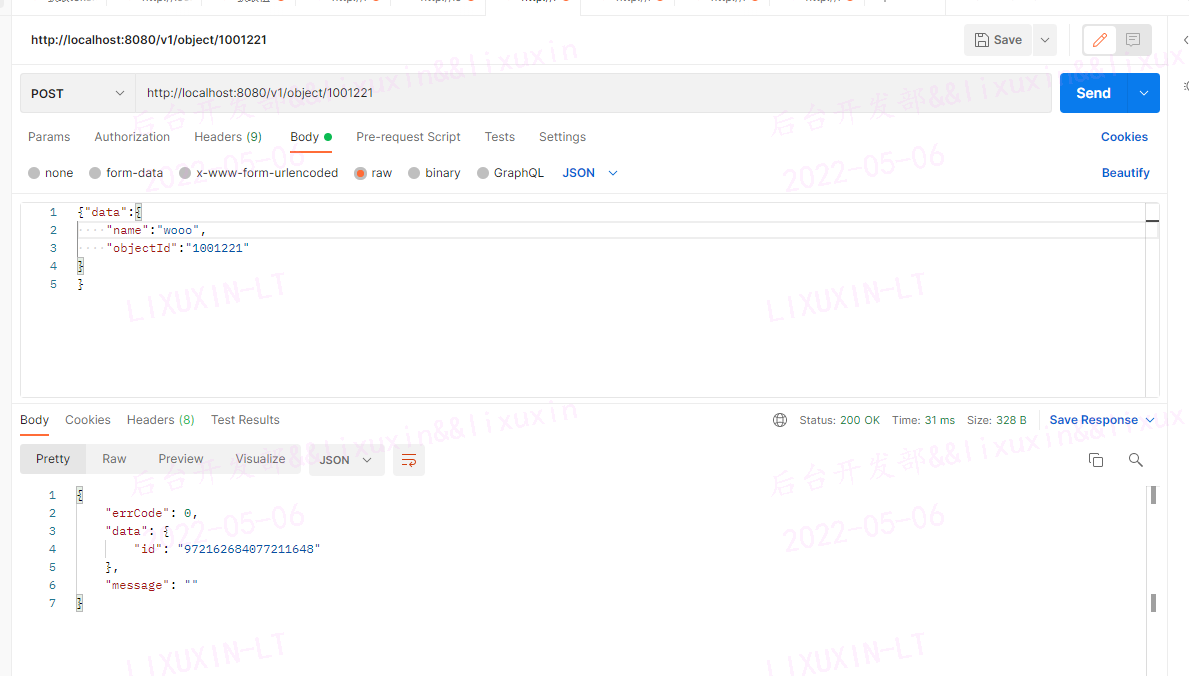


* Form表单布局配置

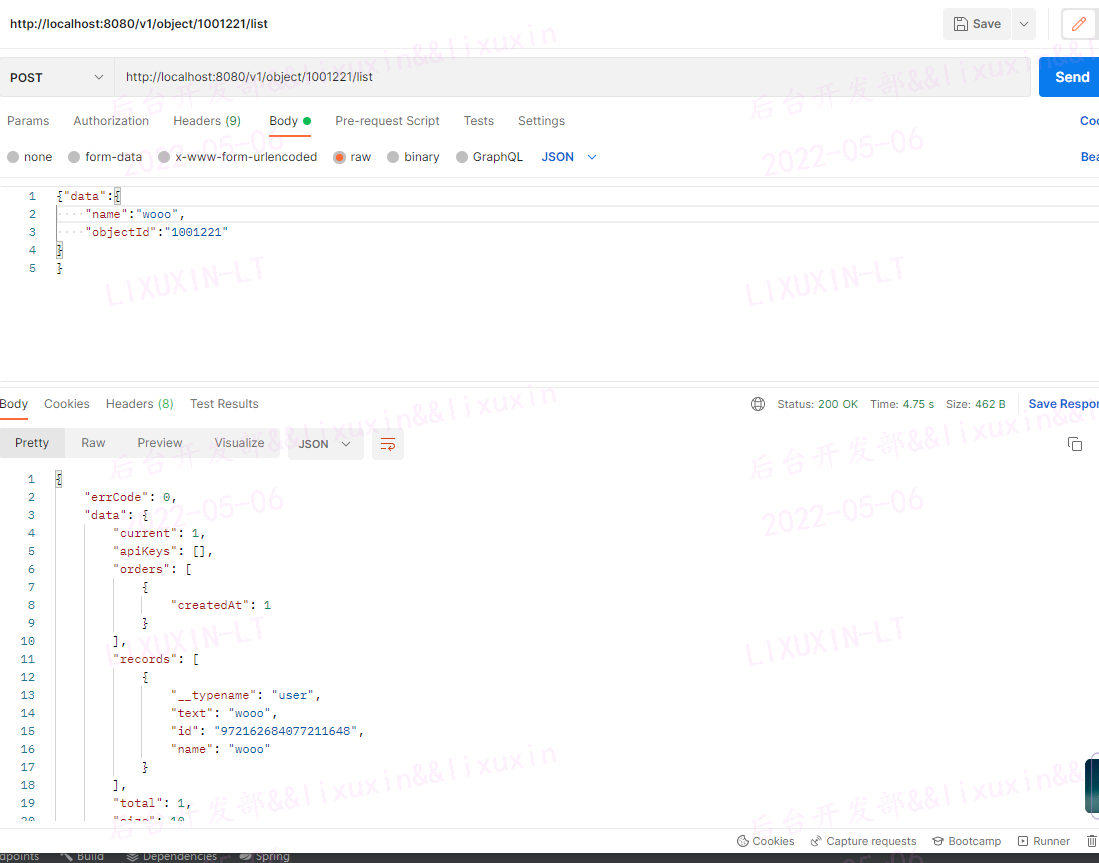


1. 自动生成接口测试效果

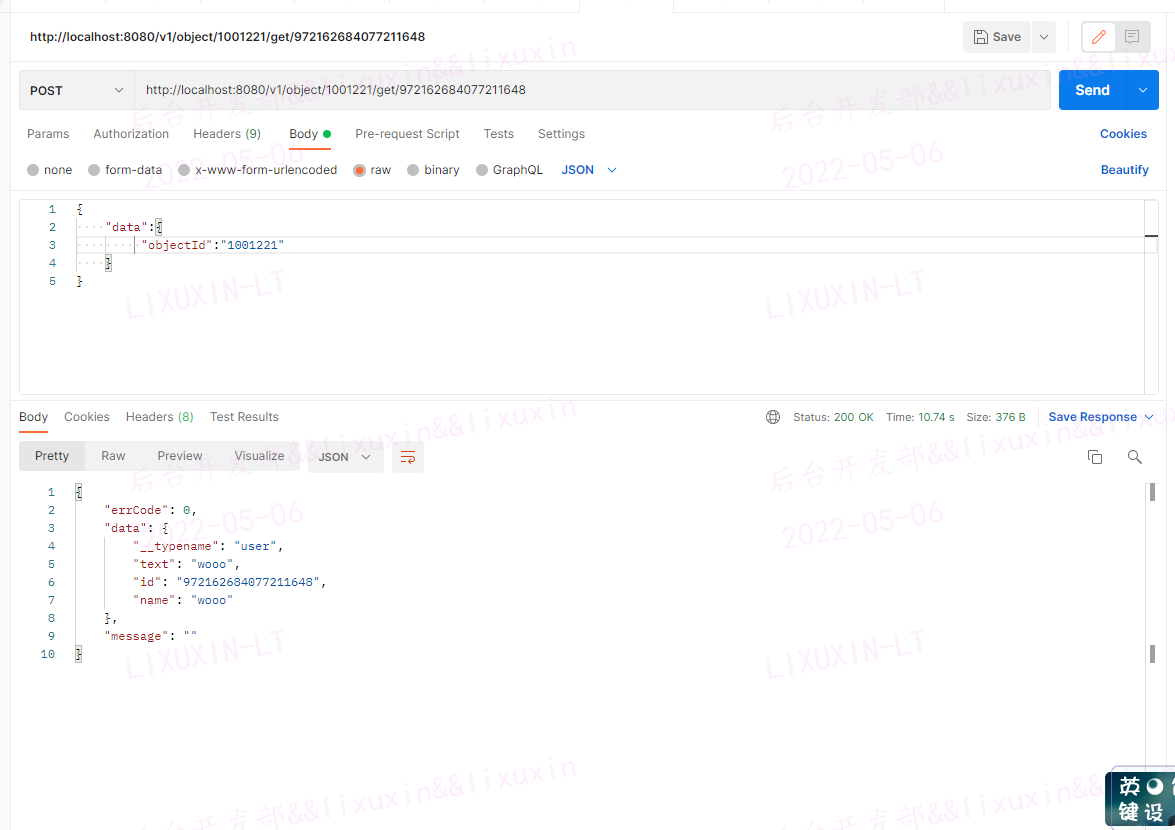
* 新增/修改



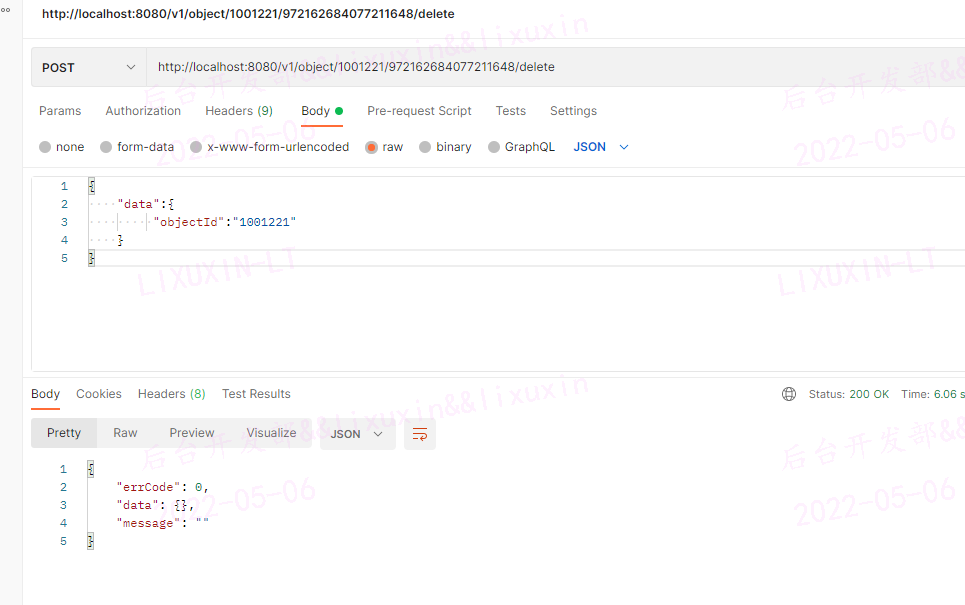
* 列表查询



* 详情查询



* 删除



# aPaaS的学习与理解

aPaaS是一种低代码开发平台，使用者可以通过脚本、少量代码甚至无需代码即可快速搭建出针对不同客户的个性化MIS应用服务。aPaaS一般有如下优点：支持简单的数据或界面展示的个性化订制，缩短开发周期，降低开发与运维成本；

咱们的aPaaS的内核机制是：以数据表为操作单元，即一个数据表分配唯一一个实体Key，aPaaS根据前端传入实体Key查到对应的数据表，并对此数据表执行简单的增、删、改、查，以实现只需要通过Layout、Grid、Entity等Groovy脚本的配置，即可实现从前端展示到后面逻辑的低代码处理。

# aPaaS的适用场景



aPaaS 的低代码能力，给使用者带来了明显的优势之外，同时也决定了它一定以“不支撑复杂的业务逻辑”为代价的，其适用场景具体分析如下：

|  |  |  |  |
| --- | --- | --- | --- |
| **序号** | **技术能力** | **aPaaS是否**  **提供** | **影响场景** |
| 1 | 通过Groovy脚本配置，可以实现简单的增、删、改、查，包括前端的展示与后端的数据处理能力 | 提供 | 适用于：业务处理逻辑简单、展示逻辑简单的场景 |
| 2 | 通过Groovy脚本配置，可以实现针对同一个数据表，为不同的客户提供不同字段的展示信息 | 提供 | 适用于：同源数据，根据客户的要求有不同字段的展示 |
| 3 | 多表关联操作 | 提供 | 通过分批多次查询获取的方式支撑，但在大数量操作场景下，可能存在性能的问题，以及功能场景限制 |
| 4 | 事务操作 | 未提供 | 例如客户下单、商品发布、平台商品聚合等，保持数据一致性；事务控制、事务传播、数据隔离级别控制。涉及到多个表操作需要业务控制，针对重复校验需要调整数据隔离级别。 |
| 5 | 多线程操作 | 未提供 | 例如订单信息获取以及商品详细信息获取；性能优化针对RPC多次查询通过多线程并行获取处理检查单次服务的RT，提高用户体验。 |
| 6 | 锁互斥操作 | 未提供 | 商品聚合需要分布式锁保证同一个商品串行处理 |
| 7 | 缓存 | 未提供 | 例如商品列表以及商品详情查询等场景的性能优化 |
| 8 | 消息发送 | 未提供 | 例如商品修改、发布、审批等状态变更通知 |
| 9 | 定时任务 | 未提供 | 例如商品状态变更通知消息发送失败异常重试 |
| 10 | 统一配置中心 | 未提供 | 应用配置项更新配置推送多台服务器生效 |
| 11 | 自定义复杂的SQL操作 | 未提供 | 例如BOSS端标准指导价管理：查询搜索商品标准指导价、多表关联分页查询等 |
| 12 | 函数 | 未提供 | 例如BOSS端商品和订单搜索：模糊匹配LIKE,需要函数CONCAT，日期转换搜索等， group by, having等常用函数不支持 |
| 13 | 批量处理数据 | 未提供 | 例如客户下单订单行插入，性能优化场景较多应用 |
| 14 | 分库分表 | 未提供 | 例如计费以及支付业务数据量较大进行分表。 |
| 15 | 自定义服务接口 | 未提供 | 例如下单、商品审批、发布等接口服务需要按照业务操作扩展，服务接口能够语义化，便于维护以及YAPI注册 |
| 16 | 跨服务间方法调用 | 未提供 | 订单调用调度等其他系统服务 |
| 17 | 针对特定服务的某些接口的限流、降级、熔断 | 未提供 | 线上服务的保护以及流量监控，无法针对具体业务精确控制。 |
| 18 | 针对特定服务的某些接口不同版本的流控 | 未提供 | 新上线接口进行灰度发布 |
| 19 | 针对特定服务的某些接口的调用链路监控 | 未提供 | Hubble监控，业务接口使用现有Restful无法分类监控以及统计告警 |
| 20 | 针对特定服务的某些接口的慢SQL监控 | 未提供 | Hubble监控，业务接口使用现有数据操作无法精确分析优化以及慢SQL统计告警 |
| 21 | 针对特定服务的某些接口的健康度监控 | 未提供 | Hubble监控，业务接口使用现有Restful无法分类监控以及统计告警 |
| 22 | 服务权限控制 | 未提供 | 用户权限以及角色权限控制 |
| 23 | 防重提交 | 未提供 | 下单以及商品提交等存在防重控制 |
| 24 | 业务幂等控制 | 未提供 | 现有商品新增以及下单等场景涉及到业务幂等控制，主要针对重要敏感数据的幂等处理 |
|  |  |  |  |

# 当前基础架构提供的低代码的实现方案

针对表的简单的增删改查的功能，基础架构也提供了一套代码能力，具体如下：

1. 实现原理

脚手架集成了Mybatis-Plus，使用了内置代码生成器，通过Maven 插件连接具体的数据库表进行快速生成 Mapper 、 Model 、 Service 、 Controller 层代码，即针对表的新生成源码，即可做到低代码支撑简单的数据操作，如需要支撑复杂的业务逻辑，开发人员可以直接基于已生成的代码进行二次开发，且与当前脚手架的技术体系完全匹配。

1. 自动生成的代码

