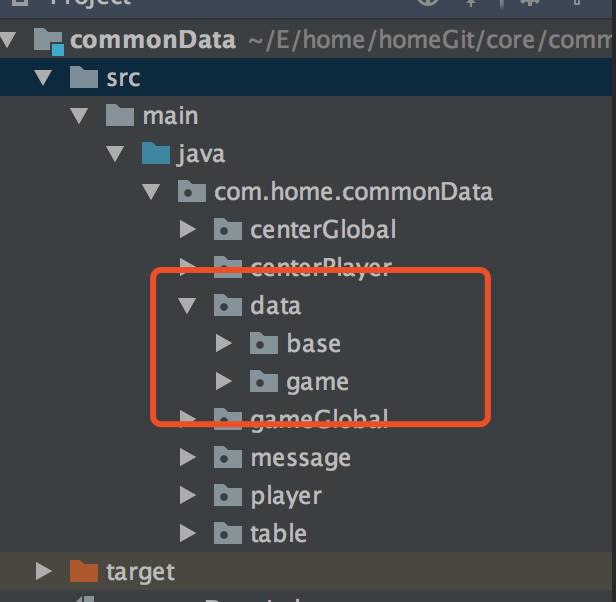
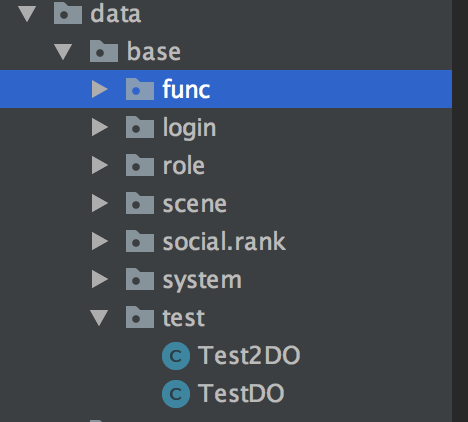
1. 项目的数据工程在develop/common/data目录下,commonData为c层工程,gameData为g层工程，用IDEA打开编辑，格式均为.java，为了使用强大的IDE作为编辑环境。C层和g层拥有相同的结构，g层可以继承c层的内容,进行拓展。所有数据工程的类，均为框架类，详情见《框架类规则》文档
2. 数据结构部分,



base目录为基类工程数据(game服,center服,login服都用的数据)(base与game的分离也是主为服务器架构而设置)

game目录为游戏工程数据(game服和客户端用的),新增数据结构的时候要区分放在哪个目录，能放game的就别放base里(最小知道原则)。

数据结构的命名后缀必须为DO(全大写),不同目录亦不可重名



每个数据结构层下都下分这样几个子目录(暂称为一级目录定义)

system:系统相关(存储,计时,共享部分)

login:登录登出相关

role:角色自身相关

item:道具相关

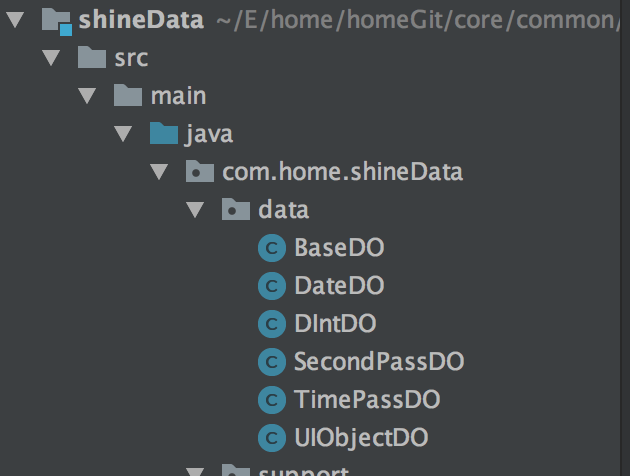
scene:场景相关

func:功能与功能插件相关

social:社交相关

activity:活动相关

1. 引擎数据结构:



BaseDO为所有数据类基类

DateDO为日期数据

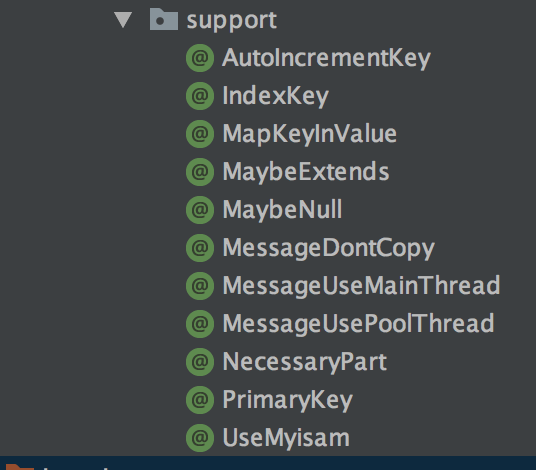
DIntDO为双整形数据

TimePassDO为时间经过数据(做计时逻辑用,单位ms)

SecondPassDO为时间经过数据(做计时逻辑用,单位s)

UIObjejctDO为引擎用UI过程数据，无视之

1. 注解功能



AutoIncrementKey:用在表对象TO的字段上，表示此字段自增

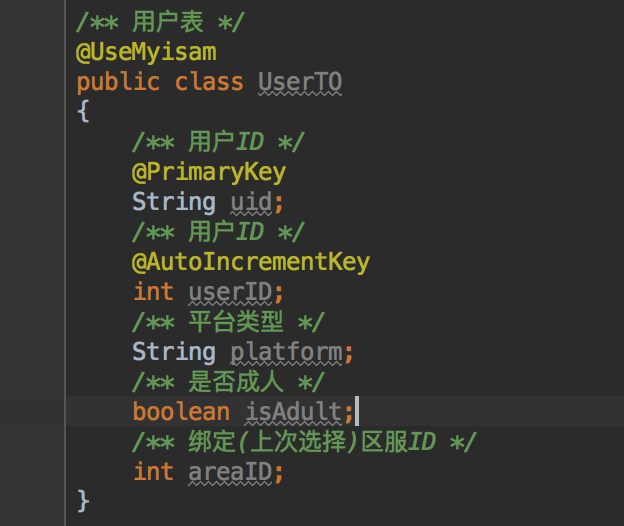
IndexKey:用在表对象TO的字段上,表示此字段为索引

PrimaryKey: 用在表对象TO的字段上,表示此字段为主键,如



UseMyisam: 用在表对象TO的类上，表示该表的存储引擎改为myisam(默认存储引擎为innoDB),如

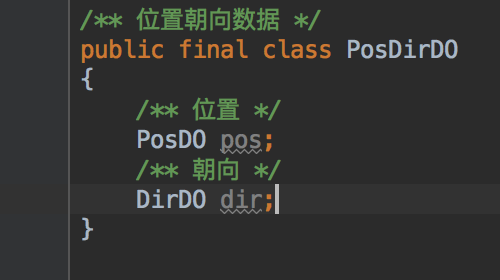
UseText:用在类型为String的TO类字段上，表示该字符串的长度较长，需要用Text存储，而不是varChar(128)。



MapKeyInValue:用在Map数据类型上,代表此map的key值序列化时从value的某字段取值(为减少序列化字节数),如



示如class前加了final，则该对象为不可继承，序列化时也会少2个字节的头长，如:



MaybeNull:表示此值可能为空值,应用在非基础类型或集合类型上,如



MessageDontCopy表示此消息在序列化时不用先进行深拷(类似proto的build模式),消息中有对象类型或集合类型时，才有必要(因为基础类型没有深拷浅拷区分)，应用在使用该消息时，参数在之后不会再被修改时,可用此方式来节省新生代内存开销(不确定时慎用)，例如:



MessageUseMainThread,表示消息使用主线程处理，默认客户端消息是池线程处理，服务器间消息是主线程处理，用此注解可以让消息直达主线程进行处理，主要应用在客户端消息上,例如



MessageUsePoolThread: 表示消息使用池线程处理，主要应用在服务器消息上,暂无例子

NecessaryPart:必须模块，应用在模块数据类上，表示此模块在g层必须存在,如

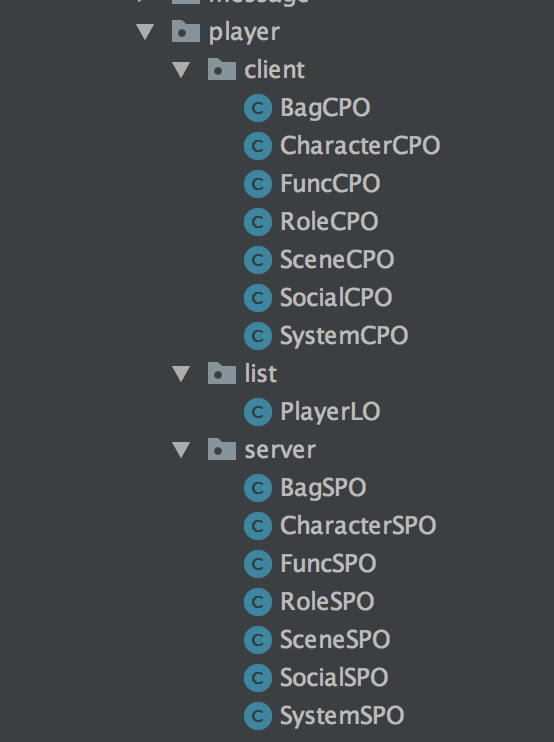


NoUpgrade:无需字段升级,应用在字段上,表示此字段无需考虑类型升级，能减少一次instanceOf调用，略微提升性能

1. 模块数据部分

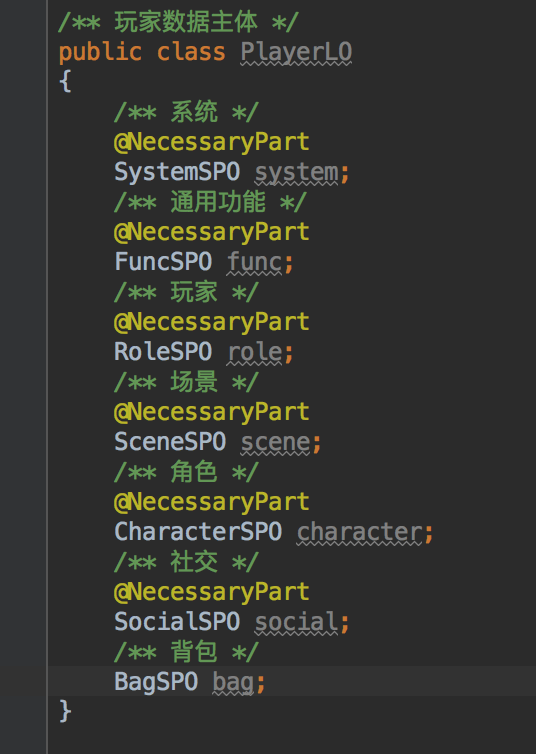
模块数据目前有角色player,中心服角色centerPlayer,游戏服全局gameGlobal,中心服全局centerGlobal，这几个，未来还有工会(union)以及项目自定义的模块化数据

下面以角色模块举例



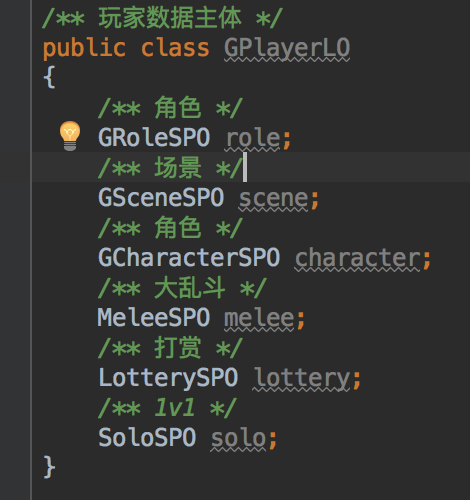
1.在C层和g层都有server目录定义好的各个模块数据(存库数据)( 结尾后缀必须为SPO),还有client目录定义好的各个模块初始化客户端数据( 结尾后缀必须为CPO),以及list目录的模块列表定义, ( 结尾后缀必须为LO)

C层每增加一个模块都必须要在对应的LO里把SPO写进去



如果是要求g层必须存在的模块，则打上@NecessaryPart标记，这样即使g层没有注册该模块，该模块也会强制注册。

角色实际存在的模块以g层LO的为依据，

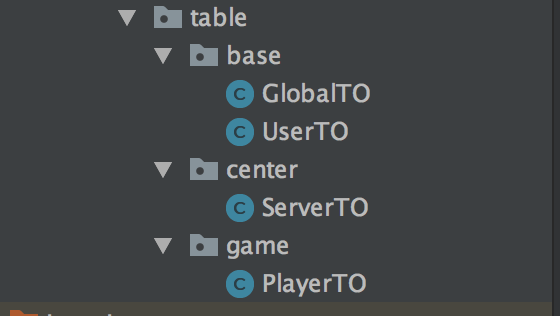


只写了SPO而没往LO里注册的，则不会导出模块类(XXXPart)

2. SPO与CPO 默认应一一对应，如不写对应的CPO时，认为此CPO从SPO完全拷贝数据，当CPO中的字段名和类型与SPO中的类型完全一致时，创建客户端初始化数据时，在Part中不必考虑字段赋值问题，引擎会帮你做好赋值工作

3. g层可以继承c层的SPO，同时的CPO也要继承

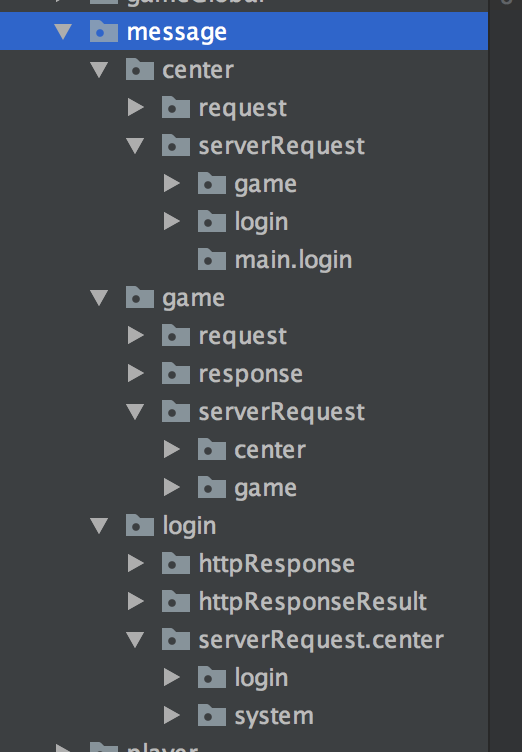
1. 数据库表部分



项目的数据库目前是单表blob模式，所以所用的表并不多，center服和game服均有global表, center服和login服都会用User表,player表只有game服用.

定义了表结构后执行工具，会生成服务器代码类，并将表结构生成到serverConfig.xml指定的数据库中，并且会在develop/server/save下生成sql文件。

1. 网络消息部分



center,game,login目录分别代码此服务下的消息内容，

center/request为center->client的直达消息

center/response为client->center的直达消息

center/serverRequest/game 为 center->game的服务器消息

center/serverRequest/login 为 center->login的服务器消息

game/request为game->client

game/response为client->game

game/serverRequest/center 为 game->center的服务器消息

game /serverRequest/ game为 game->game的服务器直达消息

login/httpResponse为client->login 的http短链消息

login/httpResponsResult为login->client 的http短链结果返回

game/serverRequest/center 为 login->center的服务器消息

每个MO都会在发送端生成Request类，在接收端生成Response类

使用Request时，只需要xxx.send(new XXXRequest(…));即可

处理消息接收时，在每个Response的execute方法里处理响应就好。

G层消息可以继承c层的消息,同一目录的消息也可以继承。