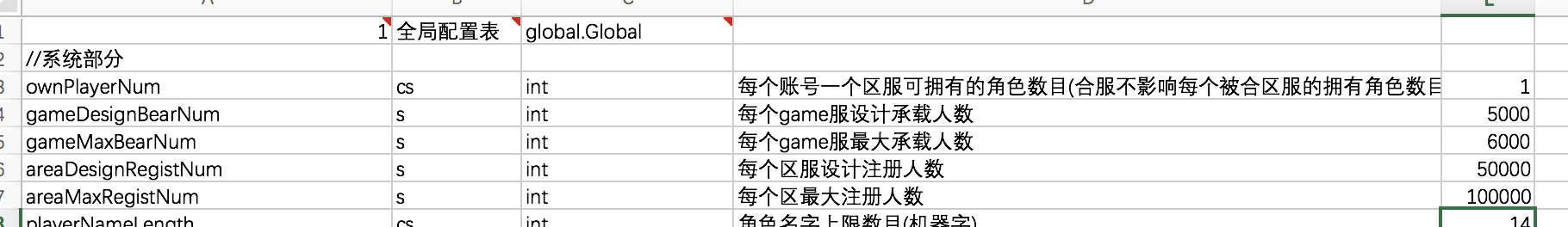
1. 首先配置表分为c层和g层两部分，c层的表主结构定义功能，g层的表才是实际的数据，c层表名的前缀都是小写c开头，并且第二个字母为大写，而g层的表为小写的驼峰。
2. g层表可以新增自己项目用的表，也可以从c层继承表结构，并可在此基础上增加字段。配置表可在config目录下任意新增目录存放，但是g层与c层有继承关系的表，必须存放在相同的目录结构中。
3. 配置表填写完后执行config目录内的 导入.bat (win10系统的c层表不可如此做),或develop/server/tools/configExport.bat来导出配置数据，工具为增量执行，如需完整执行，则执行develop/server/tools/configExportAll.bat执行
4. 配置表可分横表和纵表两种，配置表的A1格标识是横表还是纵表，0为横表1为纵表。

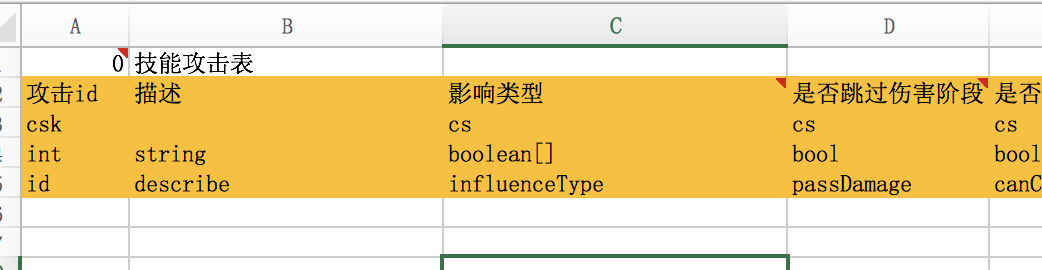
纵表目前只有一个，就是global表，里面写着各种独立的字段,如:

当为纵表时,B1为表注释,C1为表的输出类相对限定名

从第二行开始A列为字段名,B列为导出类型,C列为字段类型,D列为字段值

关于配置表类型和导出类型，下文有说。纵表中如g层有与c层同名字段时，则为覆盖c层字段内容。否则该字段从c层表中取值。所以，当g层想自定义c层字段值时，只需把该字段从c层赋值到g层，并修改即可。

表中只要B列为空，则视为无效字段，所以可以随意加注释。

1. 横表，又可分为普通数据表和枚举表,枚举表存放在enum目标下(实现上只是放到了名为enum的目录中而已)。横表形如:

在解释每个首行定义之前，先说一下支持的数据类型

字段类型支持int , boolean , string , float , double , long , dint和 []数组

1. dint为双整形类型,写法为:冒号隔开,如1:2 同样可支持dint[]这样的数组
2. 还支持byte,short这两种类型，但是只可用在主键中(为了读写加速)因为int和long本身都支持变长写入。
3. boolean字段的识别名有boolean,bool两种,值的填写方式1为true,0和空均为false
4. 数组的写法为type[],如整形组int[],数组的分割符号组为(,逗号;分号&与运算号|或运算号), 最多支持到4维数组，一维数组如1,2,3 二维数组为1,2,3;1,2,3 三维数组为1,2,3;1,2,3&1,2,3;1,2,3四维数组为1,2,3;1,2,3&1,2,3;1,2,3|1,2,3;1,2,3&1,2,3;1,2,3

再说一下字段导出类型,目标有4个标志位,分别是c:客户端字段,s:服务器字段,k:该字段为主键

r:该字段为客户端资源字段(填写资源路径,source/之后的路径)

l(小写的L):该字段为国际化文本字段(添加language表中的key)

i:该字段为客户端国际化资源字段(添加internationalResource表中的 key)

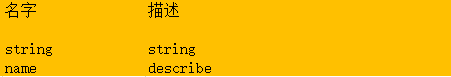
t:为时间表达式字段(目前支持两种填写方式,一种是标准cron表达式, 另一种是1:n （n为倒计时秒数）表示从指定时刻(当前)开始倒计时n 秒后的时间点)

当为横表时,B1为表注释,C1为表的key保存类型,D1为枚举类名,E1为是否从c层拷贝数据,F1为如果该表只有主键是否依旧生成Config类(占位符标记)

1. key的保存类型为1时表示尽可能使用数组存储，即当满足字段的主键类型长度(所有主键字段的类型字节和在一个short内(比如双主键byte+byte就视为在一个short内),并且该表中没有超过512的值),则该表可以开启数组存储(一般只给枚举类型用)

保存类型为2时表示强制使用关联数据(Map)存储。

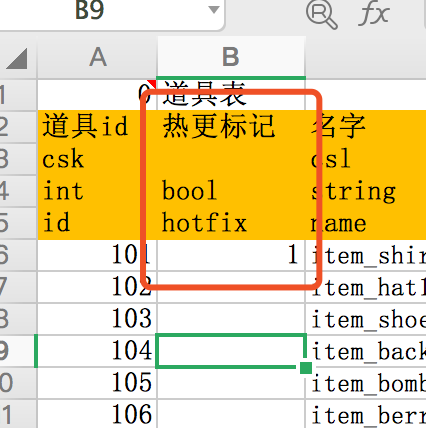
1. 关于D1枚举类名:是指该横表是否需要把所有主键字段导出到一个指定类为借鉴中(一般只枚举表会用到)，需要在表中加入下图显示的字段，name为显示枚举变量名，describe为注释文本，导出类型无需求可不添加。



1. 关于E1这个格如果值为1,并且在c层，则为必须表，就是说，即使g层没有该表，也会按c层的表数据进行导出,让g层拥有在c层预先指定的配置值。如果该表在g层，则表示是否从c层拷贝数据，就是说该表会默认存在c层表的全部行数据，(如g层定义了与c层相同主键的行，则视为使用g层的行)。主要应用在想完全继承c层数据的场合。
2. 关于F1这个格,如果值为1，则为一定生成，应对与c层表只有key，没有内容，但是g层想加内容，会遇到继承类不存的问题时使用。

横表第2行为字段注释,第3行为字段导出类型,第4行为字段类型,第5行为字段名,从第6行开始，为正式数据

1. 关于快速热更,g层的表的某行如需快速热更，在此表中，增加一列,字段类型为bool，字段名为hotfix，如:

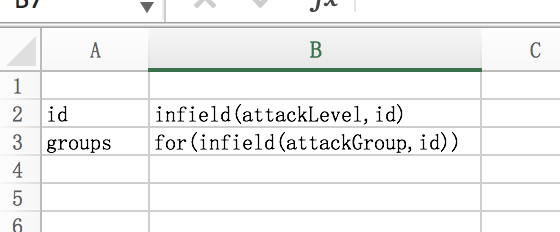


然后，该列值为1的行视为需要热更的行，所有表中，需要热更的行，会生成到configHotfix.bin里，然后在服务器端执行指令reloadConfig，执行热更新

以下是程序部分

1．横表字段约束(程序用)：

1.在横表的sheet2中，可以定义对该表的字段约束,用来提供对字段的导出时初步筛查。并且约束为g层表继承c层表的约束，并且不可覆盖c层约束。约束形如：



2.约束基本语法,伪代码形式,每一个语句均为[function](args)的形式,即方法名(参数组)的形式，语句组为f1()&f2()|f3()&f4()的形式，|为或运算法,&为与运算符，与的优先级比或高，并非从左到右执行。函数支持嵌套。

3.可支持的方法有:

(方法名全小写)

(下面用的所有表名都是g层表名,意思是c层的表也无需+c开头前缀)

基础方法:

!: 取反,与其他方法连用，如!enum(XXX)意为，不是XXX的枚举值

ref:引用其他表某字段的约束，如ref(attack,id)为引用attack表中id字段的约束内容。

refEnum: 引用枚举表，并且限定字段类型为int[][]的约束，用来做conditions,actions的通用检查。如refEnum(RoleConditionType)

for:数组循环, 限定数组类型,如for(…)

get:取数组指定位元素,限定数组类型,参数1为元素序号,如get(0,…)

getK:取dint的key元素, 限定dint类型,如getK(…)

getV:取dint的value元素, 限定dint类型,如getV(…)

switch:取数组的指定位元素进行switch判定,如switch(0,1:…,2:…)为取数组第0个元素,如值为1,则…，如值为2则…

switchField:取其他字段值进行switch判定,如switchField(type,1:…,2:…)为取该表type列的字段值,如值为1,则…，如值为2则…

compare:比较元素的值,限定数字类型,参数1为被比较常量，参数2为比较结果(-1:小于,1:大于,0:相等),如compare(0,1)为>0比较

infield:在某表的某字段值内,参数1为表的限定名(如有目录结构的组为 目录/表名),参数2为字段名，如infield(attack,id)为该值在attack表的id中出现

快捷方式:

inkey:在某表的主键中出现,限定目标表为单主键,如inkey(attack)为在attack表的主键中出现

enum:在某枚举表中出现,限定目标表为枚举表,如enum(AttackActionType)为在此枚举中出现

none:此字段为空,对于数字类型,0也算空

<=0:此值小于等于0,限定数字类型,多用于字段默认值

dkey:此dint的值恰好为某双主键表的主键,限定目标表为双主键表,如dkey(attackLevel)为此值在attackLevel的主键中存在

arrkey:此数组的某位到某位为另一个表的主键组,如arrkey(1,2,attackLevel)为此数组的arr[1]+”\_”+arr[2]为attackLevel表的主键。

2.执行导表工具会同时生成双端代码，以服务器端举例,如

程序中获取配置的方法为XXXConfig.get(…);如是g层的表，则会有大写字母G的前缀，此时用GXXXConfig.getG(…)可获取g层对象，但其实和c层取出的对象是同一个，只是类型不同。每个配置类都有如下两个方法

afterReadConfig是每个配置对象(一行数据) 读取后执行的后续处理方法

afterReadConfigAll是全部配置读取后的处理方法。