**发出世界公告——完成 itemd.py giveItem()**

content = ‘想要发出的消息’

from daemons import chat\_clientd

content = MESSAGE

chat\_clientd.send\_world\_chat(content, is\_boardcast, save\_time, continue\_time, repeat)

**拾取物品的时候发出世界公告——完成 Combat.py pickItem()**

根据物品品质（quality）发出世界公告——完成 引入constd，遍历比较constd.QUALITY\_ARR

添加物品名称——成功 **item.query(‘name’)**

添加拾取角色名称——未

**击杀怪物获得物品的世界公告——完成 rewardd.py reward()**

添加怪物名称——未 没有monster的class\_id  
添加拾取道具名称——搞定 引入globs，**globs.PROP(class\_id)** （reward中class\_id为物品的class\_id）

根据物品品质（quality）发出世界公告——完成 同上

添加拾取角色名称——未 都不知道存在哪里...

**两个接口：击杀世界BOSS发出公告、获取物品时候发出公告**

**击杀世界BOSS发出公告**

并不能在怪物的OnDead函数中实现，缺少reward信息（参数的reward实际上是reward\_type，还需要进行randomSelectByWeight才能确定奖励物品信息）

倘若BOSS掉落物有多个会发送多条公告——未解决

加入了数量的显示

**获取物品发布公告**

可以通过item\_type以及quality等等来决定获得什么物品后发出世界公告，尝试过判断dbase中有无某个标签值来决定是否发出公告，判断dbase中有没有这个属性，若没有就false，若有的话满足条件便发出世界公告

dbase中 存在boardcast属性且boardcast不为0则进行全服公告

倘若该物品是击杀世界BOSS后获得的BOSS掉落物，则会同时发送击杀BOSS的世界公告和获取物品的世界公告

——已在reason中加入判断是否是击杀世界BOSS所得

因为获得珍惜物品是一个一个获取的，所以不用加入数量

**全服广播功能模块 (broadcastd.py)**

* 基础广播接口
* 根据文本id进行广播

现有：

* 击杀世界BOSS后，对击杀者、BOSS名称、拾取道具名进行广播（rewardd.py）
* 获得珍惜物品后，对获得者、获得物品名进行广播（itemd.py）
* 锻造出珍惜品质装备后，对锻造者，锻造物品名进行广播（forged.py）
* 炼出高品质丹药后，对炼药者、丹药名进行广播（drugd.py）

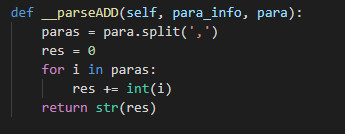
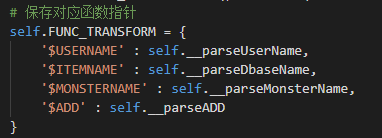
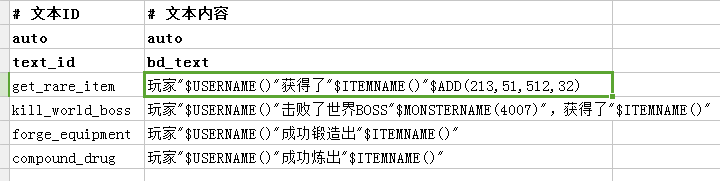
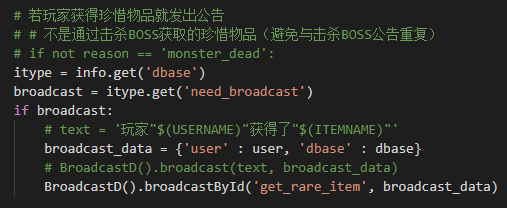
增加广播类型：

* 在broadcast\_text 配置表中按格式增加广播标准化文本  
  现有：  
  $USERNAME( [para] ) : 玩家名 $ITEMNAME( [para] ) : 物品名称   
  $MONSTERNAME( [para] ) : 怪物名称  
  ...  
  新增通配符规则：  
  （1） 开头必须为 ‘$’  
  （2） 结尾必须为 ‘)’  
  （3） ‘（）’内为参数，可以为空

**若新增通配符则需要进行以下操作：**

* def newMethod( [parameter] )
* 在FUNC\_TRANSFORM加入映射关系：  
  key：通配符 ， value：函数指针
* 在broadcast\_text 增加广播：
* 在需要广播的地方调用广播方法，参数中有广播信息字典  
  user : 玩家 dbase : 物品信息 monster : 怪物名称

**例子：**

* def newMethod( [parameter] )  
  
* 在FUNC\_TRANSFORM加入映射关系：  
  
* 在broadcast\_text 增加广播：  
  
* 在需要广播的地方调用广播方法，参数中有广播信息字典：  
  

├── Prefabs

│ └── UserReportingPrefab.prefab //预设好的用户报告组件

└──Scripts

├── Client //共23个文件，其中大多数为保存的数据结构类文件

├── UserReporting.cs

├── UserReportingScript.cs //入口

└── UserReportingClient.cs

└── Plugin