婚礼

三、功能实现

1. 工程结构

1．**首先**

拆分装配接口，一个是装配接口，一个是调度接口。这样可以保持接口的但一直职责，避免使用方调用装配操作。装配是活动在创建或者审核的时候初始化的装配动作。

2．**本节**

会在实体对象中添加属于实体本身的功能，这样会让代码更加干净，也会体现出充血的作用。

- **等等**

  记得记得

2. 装配实现

2.1大家

装配由原本的策略全部装配，扩展为支持权重装配。

2.2等价

源码：cn.bugstack.domain.strategy.service.armory.StrategyArmoryDispatch <http://cn.bugstack.domain.strategy.service.armory.strategyarmorydispatch/>

源码：cn.bugstack.domain.strategy.service.armory.StrategyArmoryDispatch <http://cn.bugstack.domain.strategy.service.armory.strategyarmorydispatch/>

```java
@Override
public boolean assembleLotteryStrategy(Long strategyId) {
// 1. 查询策略配置
List<StrategyAwardEntity> strategyAwardEntities = repository.queryStrategyAwar
assembleLotteryStrategy(String.valueOf(strategyId), strategyAwardEntities);
// 2. 权重策略配置 - 适用于 rule_weight 权重规则配置
StrategyEntitystrategyEntity= repository.queryStrategyEntityByStrategyId(stra
StringruleWeight= strategyEntity.getRuleWeight();
if (null == ruleWeight) returntrue;
StrategyRuleEntitystrategyRuleEntity= repository.queryStrategyRule(strategyId
if (null == strategyRuleEntity) {
    thrownewAppException(ResponseCode.STRATEGY_RULE_WEIGHT_IS_NULL.getCode(),
}
Map<String, List<Integer>> ruleWeightValueMap = strategyRuleEntity.getRuleWei
Set<String> keys = ruleWeightValueMap.keySet();
for (String key : keys) {
    List<Integer> ruleWeightValues = ruleWeightValueMap.get(key);
    ArrayList<StrategyAwardEntity> strategyAwardEntitiesClone = newArrayList<
    strategyAwardEntitiesClone.removeIf(entity -> !ruleWeightValues.contains(
    assembleLotteryStrategy(String.valueOf(strategyId).concat("_").concat(key
}
returntrue;
}
```

1. 如图对于 第3节：策略概率装配处理 <https://t.zsxq.com/153HMmsaI> 本节对 StrategyArmoryDispatch#assembleLotteryStrategy 代码进行重构，支持权重的策略装配。

409 字