|  |
| --- |
| 武汉趣游时代网络科技有限公司 |
| 服务器端技术手册 |
| 项目名暂未定 |
|  |
| **邹治武** |
| **2019Md 星期一** |

|  |
| --- |
|  |

概述：

该文档仅限于服务端开发，阅读文档必备Java1.8编程基础，熟悉多线程和Linux系统，精通Spring、Quartz、MyBaits、MySQL、Log4j、Netty4.0、POI等技术，本文讲述的重点在于规范代码和通常使用的方法的解释。

* 系统启动

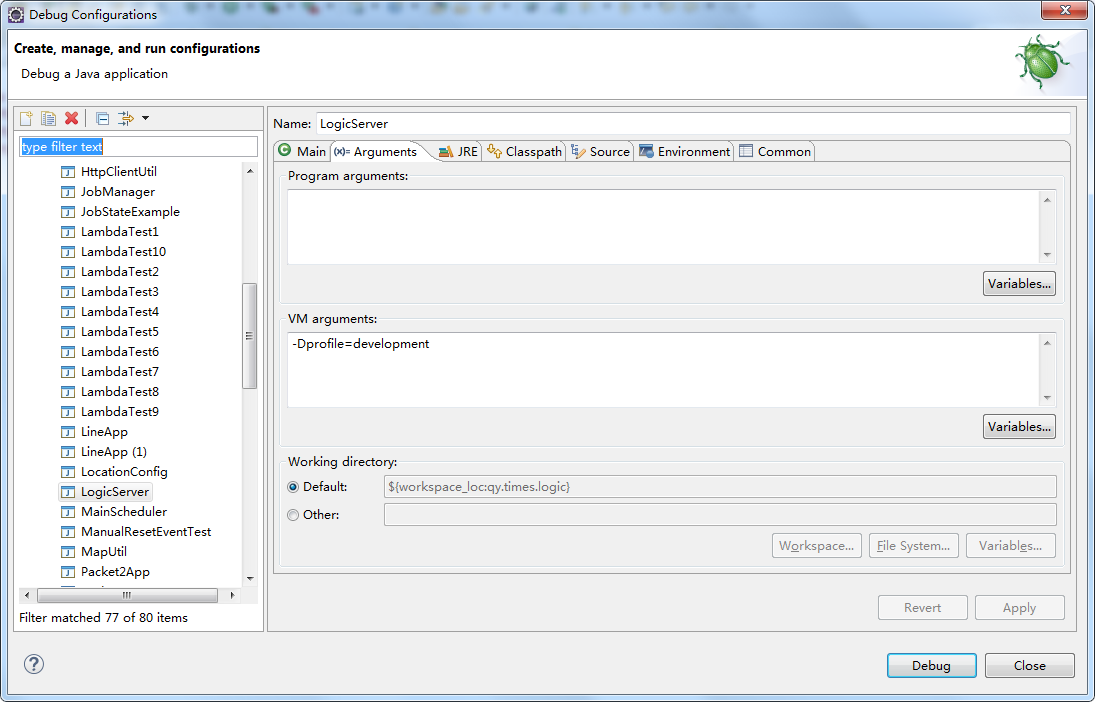
服务端启动需要配置虚拟角色，虚拟角色通过环境变量设置，本系统定义了三个虚拟角色

Development：开发环境

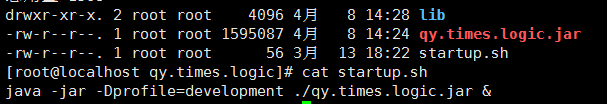
Production：生产环境

Test：测试环境

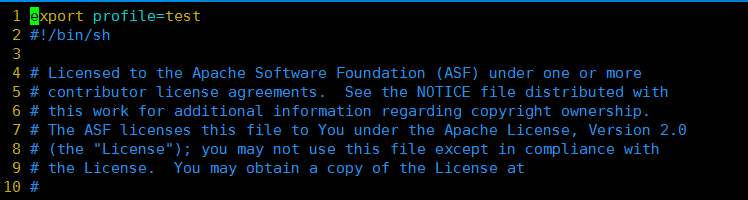
Eclipse在Debug Configurations中配置



Linux环境下的启动设置虚拟角色



Tomcat的虚拟角色在启动文件的最开始添加



* CoffeeBean

CoffeeBean是网络通信引擎，使用它客户方便快速与客户端通讯。

* GamePhysic

GamePhysic是一个模拟物理碰撞插件，战斗中技能物理碰撞不可以放在客户端计算，避免玩家容易作弊，服务器端必须模拟客户端技能碰撞

* Action

Action是服务端与客户端的接口定义，一个Action对应一个接口，同时对应一个Msgcd（Message Code的缩写），系统框架会在系统启动时扫描com.logic.action名称空间下的所有类的方法，只要方法加注解@Action即为一个接口，并且全局唯一，否则启动会报错。

每个Class作为一个模块，模块下的所有接口监听9002端口下的消息，收到消息框架会将消息解析消息头，并且分发给每个Action，每个Action负责处理消息。

* Msgcd

Msgcd按照每100个接口作为一个模块区分，也就是说每个模块最多有100个接口，经过实践每个模块接口不可能超过100个，所以不必担心接口数量会超过上限。

* Message

Message是CoffeeBean的类，用来定义客户端与服务端的通信类，该类分消息头和消息体

消息头：msgcd + memberId + token + deviceType + bodyLen + errorcd + errorInfo

消息头比较简，由以上几个字段组成，服务端收到客户端消息后，这些字段都会有值，每次收到消息时，服务端框架都会验证是否合法通讯，验证token是否正确，memberId就是玩家Id。

消息体：buffer

消息体其实就由一个队列加一个指针position，putxxx和getxxx方是相对的两个方法，分别向队列追加数据和获取数据，追加的顺序即为获取的顺序，如图1

例如：



图1

第60行和第61行分别从消息体获取到了两个float的数据，从接口文档可以知道这两个数据分别是出生地的坐标x和z，这样获取的顺序是因为客户端传输数据时按照这个顺序put数据，而服务端应该根据该顺序（协议）获取数据，如图2。

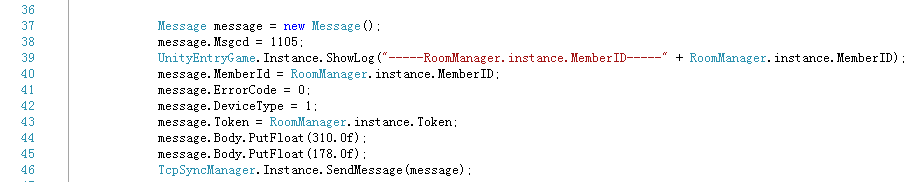
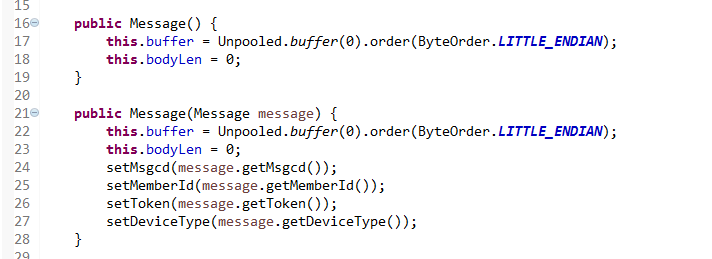
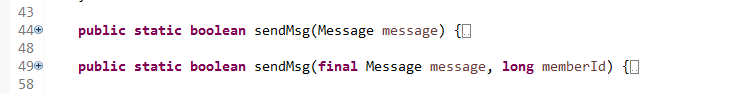


图2

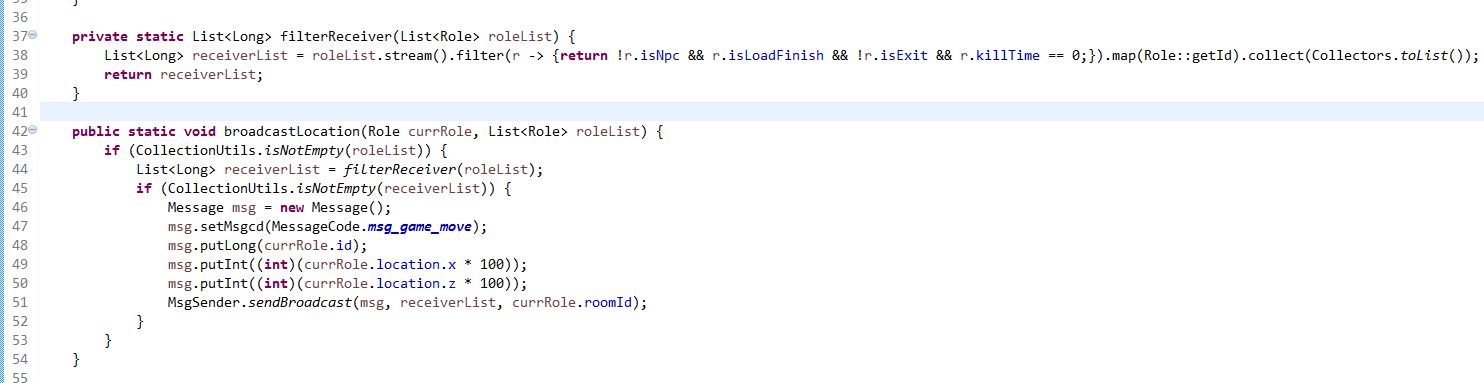
另外图1第65行Message有两个构造方法，如图3



第2个构造方法从参数中获取一个message，根据msgcd、memberId、token、deviceType四个字段生成新的message，这样做的目的是为了简化代码，直接根据收到的消息生成响应的消息，免去了设置这几个字段的麻烦，MsgSender为发送一条消息给发送的客户端，这条消息默认返回给发送消息的玩家，sendMsg还重载了另一个方法，可以指定特定的玩家



* 广播消息



广播消息即发给多个玩家，通常用List集合指定接收的玩家，

被过滤的玩家有四类：

1. 真实玩家，给AI发送消息是没有意义的，所以需要过滤；
2. 同时客户端没有加载完成的玩家也不发送，否则客户端容易报错；
3. 已经离线的玩家也是要过滤的；
4. 阵亡的玩家也要过滤；

过滤方法是使用了JDK1.8新特性lambda表达式

* 断言

断言的目的是为了代码更优雅简洁



图5

服务端程序不可避免要验证客户端数据是否合法，CoffeeBean封装了断言asWarnTrue和asErrorTrue

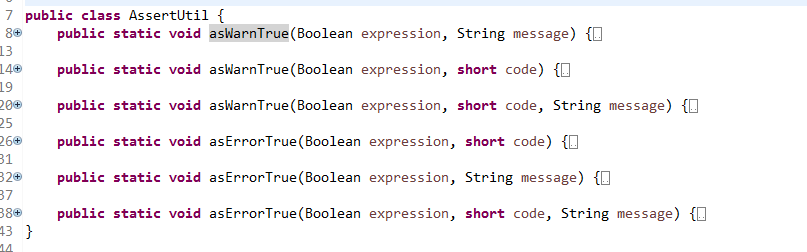


图6

asWarnTrue用来断言警告错误，asErrorTrue用来断言异常错误，区别是日志输出不同，并且返回客户端的errorId不同。如果第1个参数表达式为true，则程序继续运行下去，否则抛出警告错误或异常错误，程序中断运行，客户端会收到第二个参数的字符串内容。

Errorcd的规则为：大于0表示警告错误，等于0表示正常，小于0表示异常错误

* Service层

和Spring一样，主要业务逻辑都定义在Service

* Dao层

和Spring一样，定义Mybaits接口

* 定时任务

系统在制定定时任务时使用第三方控件Quartz，前期使用的是ThreadPool，相比较而言Quartz管理Job更加方便，目前系统使用了Quartz，ThreadPool将被之取代

* 函数式接口与Quartz

Quartz的job任务执行有时候需要调用Service类，但是job执行时是被实例化的，没有使用到Spring的Bean，因此找不到Spring注解的类，可以通过两种方法解决：

1. 通过Context.getBean(Class clazz)可以获得Spring的Service实例Bean。
2. 通过JDK1.8的函数是接口调用Service的方法。

* Quartz注解

@PersistJobDataAfterExecution

Quartz在执行execute()方法成功完成（未抛出异常）后，对JobDetail的JobDataMap进行更新，下次JobDetail执行时，接收到的就是更新后的值而不是最初存储的值。当job参数需要更新时需要添加该注解。即使有了jobDataMap.put方法就要添加该注解，否则jobDataMap.put方法put的数据不生效

@DisallowConcurrentExecution

不添加注解允许Quartz并发执行，加注解后不允许并发执行

例如：定义job1，每隔100毫秒执行一次，每次执行时间是200毫秒。

1. 如果没有添加该注解

那么执行时间是：第1个100毫秒，第2个100毫秒，第3个100毫秒...

1. 如果添加该注解

那么执行时间是：第1个100毫秒，第4个100毫秒，第7个100毫秒...

我们系统中基本都是串行方式执行，也就是都需要添加该注解

* 视野内对象集合算法

视野内对象类型包括：玩家，宝箱，技能书

计算玩家视野内对象的算法基本一样，只是计算的对象不同，没有必要对三种不同的对象各写个方法，所以把三种对象用泛型抽象出来，返回的集合主要是List和Map，所以有两个泛型类VisionList<T>和VisionMap<T>，再用适配器VisionAdapter应用于每种方法类型集合

注意，视野内对象变更用两个集合来反映，相比上一次视野范围内对象列表，分为离开视野的对象列表outList<T>，进入视野的对象列表inList<T>。

* 模板文件

策划通过模板文件控制系统数值，例如技能模板文件skill.xlsx，该文件是开发时指定的，系统启动时需要读取模板文件，生成SkillConfig.map对象集合到内存，集合元素为SkillTemplate对象，玩家攻击时需要读取相关模板文件。

地图模板map.bmp比较特殊，不能通过普通Excel文件反应实际情况，本系统使用位图，通过位图像素点标识单位分米区域是否可行走