## 概述

MMServerEngine是一个游戏服务器引擎。

支持如下几点：

1、集群：可设置多个逻辑处理节点。

2、缓存：支持本地和远程两层缓存

3、异步数据库：对于更新，插入和删除操作支持异步更新到数据库

4、服务层事务：任何一个服务处理可以添加事务。

## 目标

MMServerEngine设计的最主要的目标有两点：

1、插件化：即具体使用的技术（如缓存，网络，数据库等）可以通过插件的形式加入到本框架。

2、使用高效：尽量去掉和具体游戏逻辑无关的枝节，进而减少bug出现的概率。

## 框架

1、系统框架

mainServer

nodeServer

nodeServer

.......

asyncServer

remoteCache

client

2、Server框架

网络入口（编码解码）

request入口

netEvent入口

request协议解码

service（业务逻辑）

dataCenter（数据访问服务）

cache服务

async（异步服务）

DB

本地

公共

Aop

Monitor

Bean容器

3、Service框架

service

（有无事务）

request

netEvent

update

event

job

其它

## 使用

1、编写客户端消息入口  
（1）要根据使用的协议定义入口，入口继承自Entrance，  
（2）对进入的消息解析出一个访问标识符（opcode）和一个对象,如果需要session，也要解析出session的id，并通过SessionService来创建获取删除等  
（3）调用RequestService.handle方法处理  
2、编写Service，  
（1）对服务添加Service注解，其变量init和destroy为初始化和销毁的方法  
（2）服务包括Request,Event,Update,Job,NetEvent  
（3）如果需要使用其它Service，可以通过BeanHelper.getServiceBean方法获得。（建议在init方法中设置好，提高效率，并且系统启动可以检查其存在）  
3、编写数据库对应实体类  
（1）需要用DBEntity来注解，如果字段名字和表的字段名不同，要添加Column注解  
（2）继承Serializable接口，因为需要进行缓存  
（3）通过DataService来对DB数据进行操作  
4、Request  
（1）对于处理入口来的某个opcode的方法添加Request注解，并设置opcode变量  
5、Event  
（1）接受事件的服务方法，要添加EventListener注解，并设置event变量  
（2）发送事件要先定义EventData对象，设置其event标识，和发送的对象  
（3）通过EventService发送事件，其中fireEvent异步发送事件，fireEventSyn同步发送事件  
6、NetEvent（网络事件在编写普通服务的时候极少用到）  
（1）接受事件的服务方法，要添加NetEventListener注解，并设置netEvent变量。注意，NetEventListener只能有一个，如果需要多个可通过Event转发  
（2）发送事件要先定义NetEventData对象，设置其netEvent标识，和发送的对象  
（3）通过NetEventService来发送事件，有多种发送方式，具体参见NetEventService  
（4）系统有默认的网络事件入口，如果需要自己的网络事件入口（如特定的协议编解码），可自己编写，并在配置文件中配置  
7、Update  
（1）对相应的方法使用Updatable注解  
（2）update可以设置同步调用和异步调用，其中同步调用将使用系统的调用周期（可在配置文件中配置），不易添加复杂的逻辑  
（3）通过runEveryServer设置其是否在所有的服务器中运行（鉴负载与集群），若否，则将由系统分配其运行服务器  
（4）调用时间点可通过cycle设置一个常数，或者通过cronExpression设置，注意，若cronExpression设置了，cycle将不起作用  
8、Job  
（1）job是指在某个时间之后做一件事情  
（2）定义一个Job对象，设置其id（要求唯一，可以通过Service名+方法名+玩家id等）、执行时间、服务类和方法、参数  
（3）通过设置db来设置该job在系统重启之后是否要加载  
（4）通过JobService来启动或者删除job  
9、tx事务  
（1）需要添加事务的方法添加Tx注解  
（2）变量tx确定是否使用事务  
（3）lock使DBEntity的提交加锁，而lockClass可以设置该事务加锁DBEntity的类型，不设置所有flush的DBEntity都将被加锁。  
（4）加锁必须在事务内，如果方法调用时已经在事务内，则共用已经有的事务。注意：由于事务提交之后才解锁事务中加的锁，所以在同一个事务中不易有太多的加锁操作  
10、全局锁（较少用到）  
（1）如果需要加全局锁，即多个服务器共用锁，用LockerService可以完成，但需要自己解锁  
11、缓存服务（较少用到）  
（1）通过CacheService可以缓存自己想要缓存的对象  
12、运行状态监控（还未实现）  
（1）通过MonitorService可以来监控自己服务的运行状态