1. 固定元素
2. 通讯协议用mq。
3. Console是一对多。
4. 设计思路
5. 从console界面获取机器、容器及相关的命令。
6. 后台将容器id注入到message属性contain中，同时产生一个随机数放入id属性中（为了确定是同一条命令），并向这个队列发送命令，同时监听这个队列属性（contain=console）的消息。
7. 后台监听这个队列，获取到消息头contain值为自己容器id的消息，从而获取到这个命令。
8. 通过这个命令来找到相应的处理方法。
9. 处理完响应命令后将结果放入这个队列，并设置这个消息的属性（contain= console）和接受消息中的随机数传回去。
10. 2步骤中的监听线程获取返回的消息，用随机数匹配到那条消息的结果，再将结果返回到相应展示界面。
11. 相关表设计
12. contain (机器容器表)

|  |  |  |
| --- | --- | --- |
| 字段 | 类型 | 描述 |
| ContainId | Varchar2(20) | 容器id |
| containName | Varchar2(20) | 容器名称 |
| ContainType | Varchar2(10) | 容器类型（IN,OUT,DISCENTER,CONSOLE等） |
| Ip | Varchar2(20) | 容器地址 |
| Valid | Char(1) | 有效标示 |

1. 相关配置
2. 做一个命令对应执行命令类的xml配置。
3. 具体实现步骤（以discenter上线 23机器为例）
4. 各个启动时开启消费者监听队列，找出contain为自己容器id的消息。
5. Console上点击url为23的上线按钮。



1. 后台获取到向discenter发送对23机器的上线命令。通过容器类型找到discenter的containid(discenter只有一台)。由生产者向message将cantainid注入到contain的中，同时产生的随机数并注入到id中，最后将机器ip和上线命令放入报文，由console的生产者发出。这个线程等待（设置超时机制）。
2. Discenter获取到监听到发送的这条消息。首先将这个消息中的随机数保存，然后通过这个命令找到相应的处理类，并将ip传入处理类。
3. 处理类根据IP到内存中将这条机器的状态置为上线，并将结果返回到discenter中。
4. Discenter将结果和随机数重新包装到message中，并注入contain=console的属性，发送到队列。
5. Console中消费者监听到这消息，获取到随机数后找到等待的线程并唤醒，将处理结果通过此线程告知操作者。
6. mq队列创建及各个容器链接的设计
7. 在app\_address表中建一条serverid为console\_other的信息。PROVIDER\_URL为mq服务器地址。
8. 在各个容器启动时，将这条信息取出并建立消费者监听这个队列。