1. turnilo健康检查

<http://turnilo.fast.miui.srv/>

通过机器定时发送请求请求服务来测试服务是否成功执行

1. 在“走路赚钱”任务中，既记录了用户id兑换金币，又记录了设备id兑换金币?????

PaceController/exchange

PaceBiz / RedisKeyEnum.***UserStepExchanged*** RedisKeyEnum.***StepDeviceExchangeToday***

1. 在xbox上不要选择自己的分支进行部署，不然别人无法部署。

|  |  |
| --- | --- |
| 未登录状态 | 无userId |
| 游客状态 | 有userId,但是没有绑定微信和手机号 |
| 登陆状态 | 有userId,绑定了微信或者手机号 |

4.在想看项目中，由于线上和线下数据库隔离。因此对于同一个任务，线上与线下的taskId可能会不一样。在查询任务时，直接使用taskId查询是会产生问题的！！！！

正确的做法是：

1. 先将所有的任务进行缓存（缓存到本地）
2. 根据groupId和uuid筛选出对应的taskid

5.OTO是客户端打点程序

服务端打点数据都是通过在数据工厂通过hive指令查询打点数据

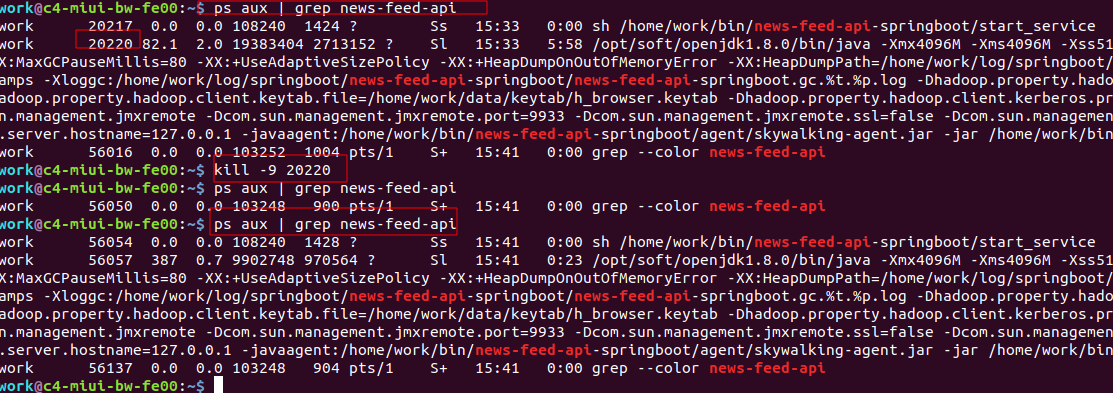
6.Mybatis数据库分表的实现

7.在xbox上编译部署的时候。编译成功后会打包将新代码拷贝到机器上进行部署。一般某台机器(如

c4-miui-bw-fe00.bj)部署失败后，直接联系dir 将该机器kill掉后重新启动即可。但是xbox平台的状态仍然是failed。

ps aux | grep news-feed-api

kill -9 20220



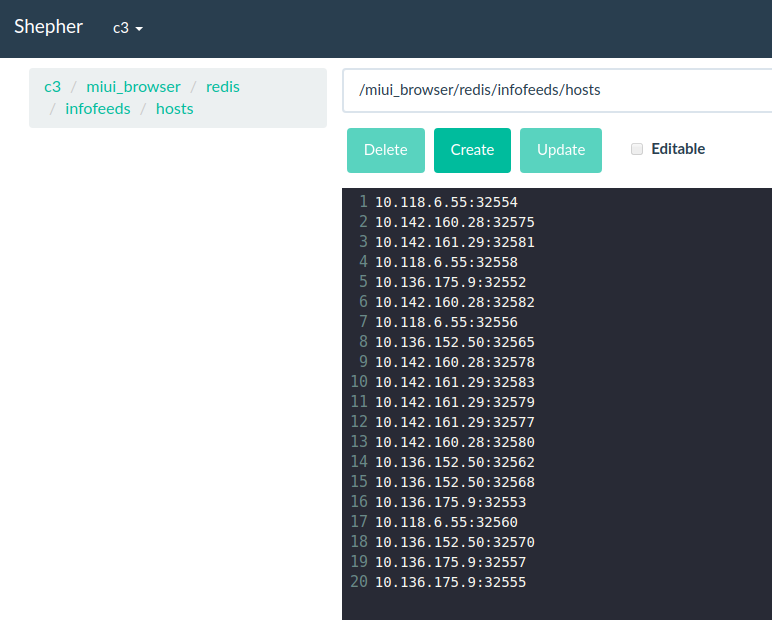
8.想看线上redis集群

[https://zkview.d.xiaomi.net/clusters/c4miuisrv/nodes?path=%2Fmiui\_browser%2Fredis%2Fdefault%2Fhosts](https://zkview.d.xiaomi.net/clusters/c4miuisrv/nodes?path=/miui_browser/redis/default/hosts)

/miui\_browser/redis/infofeeds/hosts c3-miui-bw-admin00.bj

cd /home/work/znb/redis-3.0.0/src

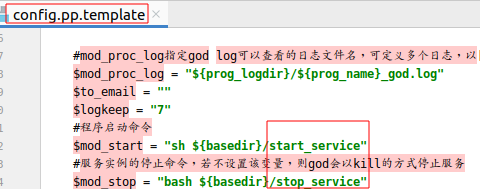
./redis-cli -c -h 10.118.6.55 -p 32554



9.项目在xbox上的编译->部署->打包流程(以news-feed-api-springboot为例)：

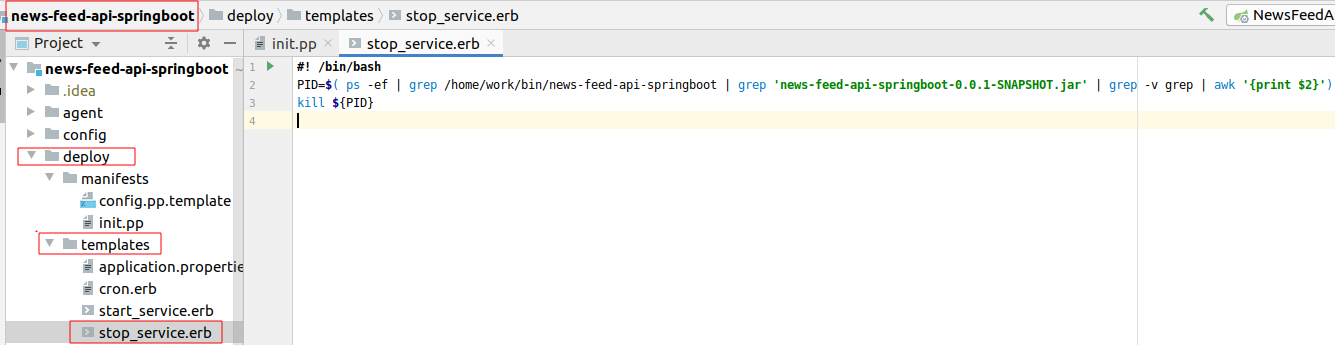
|  |  |
| --- | --- |
| 编译 | build.sh |
| 部署 | Xbox约定读取deploy/manifests/下的  config.pp.template和init.pp |
| 打包 | Screenshot from 2020-07-10 11-44-49  打包时，将部署机器上的release/agent目录下的文件打包到另一台机器的agent目录下。（部署和打包不是同一台机器！！！） |
| 启动服务 | spring boot 启动服务 |
| 服务注册 | 服务启动成功后，所有安装了eureka-client的项目向eureka-service进行注册 |
| 服务下线 | Xbox执行god stop，对应到执行项目的deploy/templates/stop\_service.erb |

服务的启动与停止





News-feed-api-springboot是直接 kill 服务进程

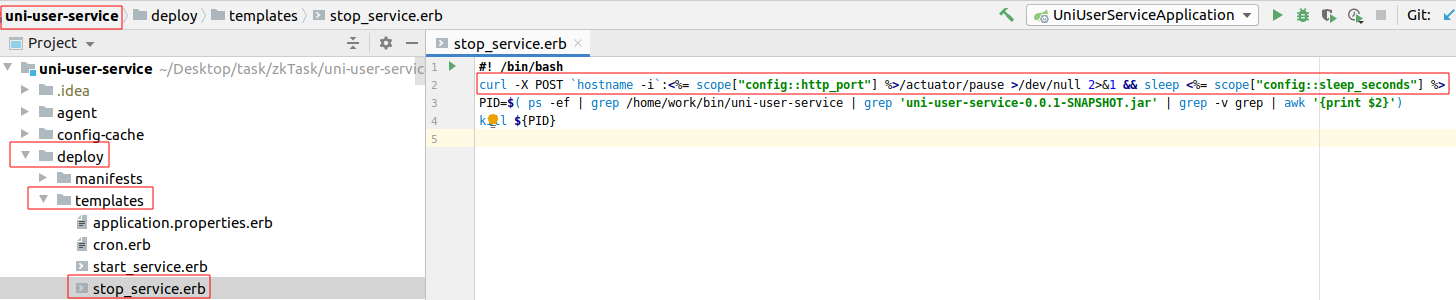


Uni-user-service对应的是

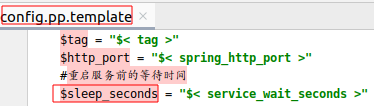
(1)给本机服务发送pause请求,eureka-server上该服务的状态更新为”DOWN”

(2)睡眠**sleep\_seconds时间**

(3)kill 服务进程

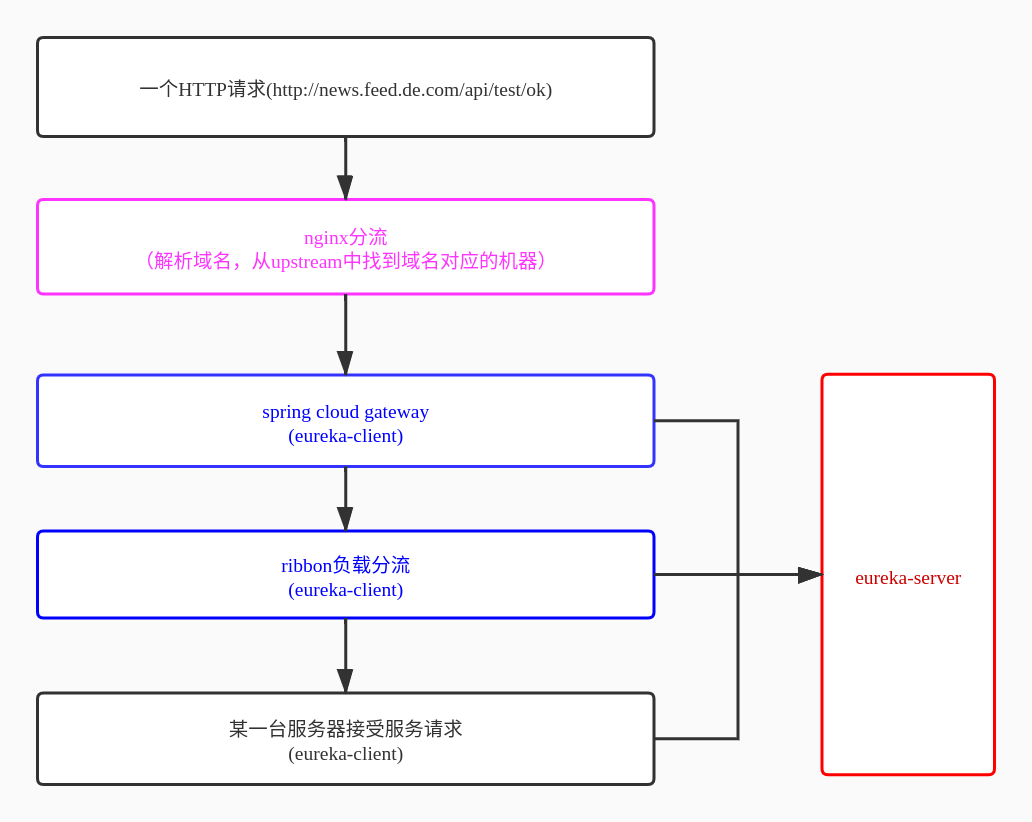


**sleep\_seconds 来源于：(这个配置不加好像也可以直接从xbox读到)**

****

**service\_wait\_seconds 是在xbox端配置的。**项目启动时读取xbox配置文件后填充。

**一个HTTP请求的完整过程，以/api/test/ok为例：**

****

|  |  |
| --- | --- |
| **1** | 客户端发起一个http请求 |
| **2** | Nginx通过域名解析，找到对应的机器（或者集群） |
| **3** | 该机器如果配置了spring cloud gateway，那么会自动集成ribbon进行分流（？？？？？？？？？）。即spring cloud gateway与ribbon是安装在同一台机器上面的。该机器如果配置了eureka-client，就会通过心跳定时从eureka-server拉取服务注册列表。  Ribbon根据该列表选择一台可供服务的机器作为请求的应答机器。 |
| **4** | 被选择的机器接受请求，并产生应答。被选择的机器需要配置eureka-client,定时向eureka-server进行服务续约。 |
| **注意** | Spring cloud gateway与ribbon是可选的。nginx是一般都会有的。 |

**服务优雅下线的原理：**

当服务部署时，首先会让旧服务下线。xbox通过god stop命令执行服务下线操作。而god stop命令对应的是服务deploy/templates/stop\_service.erb中的指令，即

|  |
| --- |
| #! /bin/bash  curl -X POST `hostname -i`:<%= scope["config::http\_port"] %>/actuator/pause >/dev/null 2>&1 && sleep <%= scope["config::sleep\_seconds"] %>  PID=$( ps -ef | grep /home/work/bin/misc-service | grep 'misc-service-0.0.1-SNAPSHOT.jar' | grep -v grep | awk '{print $2}')  kill ${PID} |

1. curl 给本机发送pause请求（借助于spring-boot-starter-actuator），修改服务状态

此时，本机仍然能正常提供服务！！！！

1. Eureka-server会立即将该服务的状态置为”DOWN”
2. 让本服务睡眠一段时间sleep\_seconds
3. 获取本服务的进程🆔id：PID
4. 杀死本服务进程。此后本机才无法正常提供服务

当eureka-server将该服务的状态更新后DOWN后，且其他eureka-client感知到该服务的DOWN状态后，就不会将请求再转发到该机器上。

Eureka Server收到Instance cancel register的请求后，立刻执行相应的操作，比如更新连接数、更新租约阈值值。

Eureka Server返回给Eureka Client的值会存在一个缓存里，默认每隔30s（eureka.responseCacheUpdateIntervalMs）更新一下缓存值。

Eureka Client本地也有缓存，默认情况下，Client每隔30s（eureka.client.refresh.interval）跟服务器端同步一下信息。

总结：由于EurekaServer和EurekaClient都有缓存，所以Client获得服务变动消息会有一个延迟。 延迟时间为0 ~ (eureka.responseCacheUpdateIntervalMs + eureka.client.refresh.interval)

因此：只要在延迟的这段时间里面，置为“DOWN”状态的机器仍然能继续提供服务，就可以实现优雅下线。即只需要保证：

sleep\_seconds 》 eureka.responseCacheUpdateIntervalMs + eureka.client.refresh.interval

<https://wiki.n.miui.com/pages/viewpage.action?pageId=405046226>

10.测试（北京3武清测试）|线上（北京C3）环境使用pegasus查看某条记录

(1) ssh 登陆测试环境的机器

ssh [wuxia5@relay.xiaomi.com](mailto:wuxia5@relay.xiaomi.com)

tj1-miui-bn-stage01.kscn（**线上：c3-miui-bw-admin00**）

1. 进入到pegasus

cd /home/work/tianhui/pegasus-cli

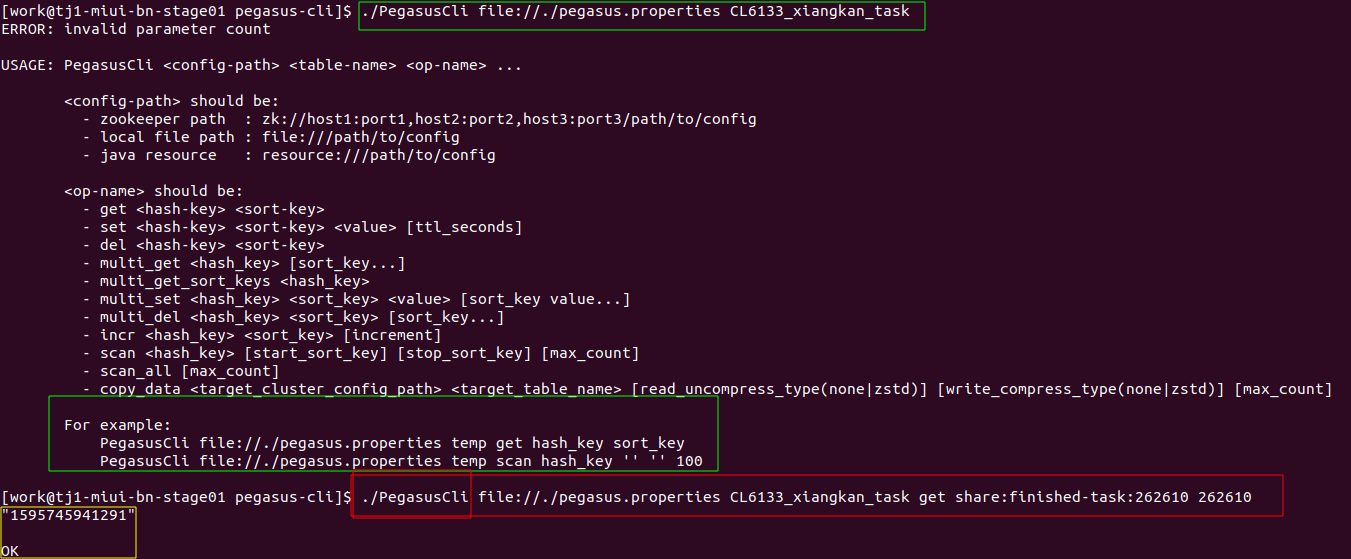
1. 执行查询

表名：CL6133\_xiangkan\_task（ **xd\_user**）

hashKey：share:finished-task:262610

sortedKey:262610

./PegasusCli file://./pegasus.properties CL6133\_xiangkan\_task get share:finished-task:262610 262610



./PegasusCli file://./pegasus.properties CL6133\_xd\_user get new\_device\_meta:521d7ad896fad414 521d7ad896fad414

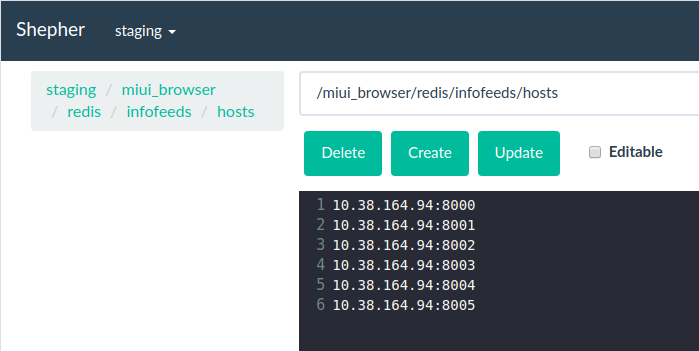
**./PegasusCli file://./pegasus.properties xd\_user get new\_device\_meta:521d7ad896fad414 521d7ad896fad414**

1. Api项目EventTaskBiz(1281)的方法理解：

|  |
| --- |
| */\*\*前4bit记录的是当前数据是本周第几天的，后面60bit，每20bit一个单元，记录当天的次数  \* 紧邻4bit的第一个20bit 代表本周几分享的次数\*/* **private void** increaseShareCount(String docId, LocalDate now) **throws** PException {  **final int** dayOfOneWeek = 7;  **final int** totalDay = 3;  **final int** dayUnit = 20;  **final int** todayShiftUnit = totalDay - 1; *// 2* **final int** todayShift = todayShiftUnit \* dayUnit; *// 40* **final int** dayOfWeekShift = totalDay \* dayUnit; *// 60* **final long** maxCountEveryDay = 0xFFFFFL;  **final long** todayCountMask = maxCountEveryDay << todayShift; *// 0x0FFF FF00 0000 0000L* **final long** maxRecordDay = 0xFL;  **final long** dayMask = maxRecordDay << dayOfWeekShift; *// 0xF000 0000 0000 0000L  //获取今天是周几，比如 4* **long** dayOfWeek = now.getDayOfWeek().getValue();  String countHashKey = getDocShareCountHashKey(docId);  String countStr = BrowserPegasusClient.*get*(**recordTable**, countHashKey, docId);  *//获取当前文章到上次为止的分享次数, 比如 0X 2 00001 00002 00003 (代表周二分享1次，周一分享2次，周日分享3次)* **long** recordCount = NumberUtils.*toLong*(countStr);  *//获取上次分享是周几，比如 2* **long** recordDayOfWeek = (recordCount & dayMask) >> dayOfWeekShift;  *// 去掉周几信息，比如0x 0 00001 00002 00003* **long** newCount = recordCount & (~dayMask); *// 去除记录的天信息  //如果今天分享的周几数与上次分享的周几数不同（比如上次是周一，这次是周一，周几数相同，否则不同）* **if** (recordDayOfWeek != dayOfWeek) {  *//顺时针循环计算当前周与上次周相差几，比如 (4-2+7)%7 = 2,每相差一位就循环右移20位* **long** shiftUnit = (dayOfWeek - recordDayOfWeek + dayOfOneWeek) % dayOfOneWeek;  newCount = newCount >>> (shiftUnit \* dayUnit); *// newCount是去除记录的天信息后的数据* }  *//获取第一个20bit的值，该值代表周几（如周4）上次分享的次数* **long** todayCount = (newCount & todayCountMask) >>> todayShift;  *//没有超过次数上线时就自加1* **if** (todayCount < maxCountEveryDay) {  ++todayCount;  }  *//去掉今天的分享次数，比如 0 00000 00000 00001(第一个20bit的00000代表周四上次分享0次，第二个20bit的00000代表周三上次分享0次，第三个00001代表周二上次分享1次)* newCount &= ~todayCountMask; *// 去除记录的今天的数据  //数据组合今天的分享次数 4 00001 00000 00001 （代表截至到本次为止，最后一次分享是周4，分享了1次，周三上次分享了0次，周二上次分享了1次）* newCount |= (todayCount << todayShift) | (dayOfWeek << dayOfWeekShift);  BrowserPegasusClient.*set*(**recordTable**, countHashKey, docId, String.*valueOf*(newCount), totalDay \* ***ONE\_DAY\_SECONDS***); } |

12.新岛测试环境的redis集群连接

[https://zkview.d.xiaomi.net/clusters/staging/nodes?path=%2Fmiui\_browser%2Fredis%2Finfofeeds/hosts](https://zkview.d.xiaomi.net/clusters/staging/nodes?path=/miui_browser/redis/infofeeds/hosts)



SSH 登陆跳板机

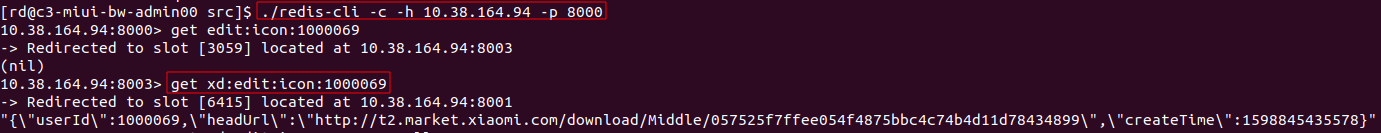
输入机器名：c3-miui-bw-admin00.bj

cd /home/work/znb/redis-3.0.0/src

Screenshot from 2020-08-31 17-07-39

./redis-cli -c -h 10.38.164.94 -p 8000

get xd:edit:icon:1000059:cnt



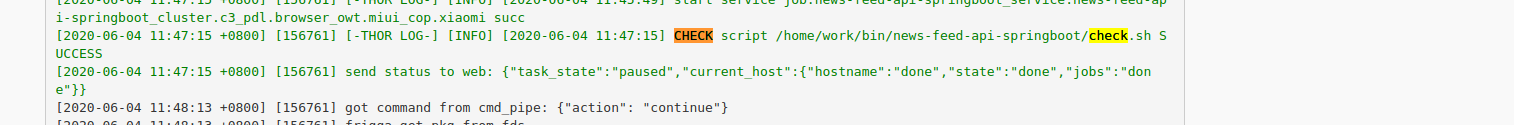
1. check.sh对服务的自动化检测
2. 在项目的build.sh文件中的deploy\_module()中执行

mkdir -p release //在部署机器上创建文件release  
cp -r target/\* deploy check.sh release

将编译机器上目录target/\*、deploy和check.sh 的内容拷贝到release下

1. **在xbox上进行编译部署时，如果存在check.sh文件，那么会自动执行该文件的内容**

<https://wiki.n.miui.com/pages/viewpage.action?pageId=151416857>



1. 查看接口请求的响应速度

比如查看新岛线上接口/api/tracking/act的响应速度

[http://turnilo.fast.miui.srv/#fast\_browser\_nginx\_](http://turnilo.fast.miui.srv/" \l "fast_browser_nginx_)



1. Mybatis 使用逆向工程生成Example

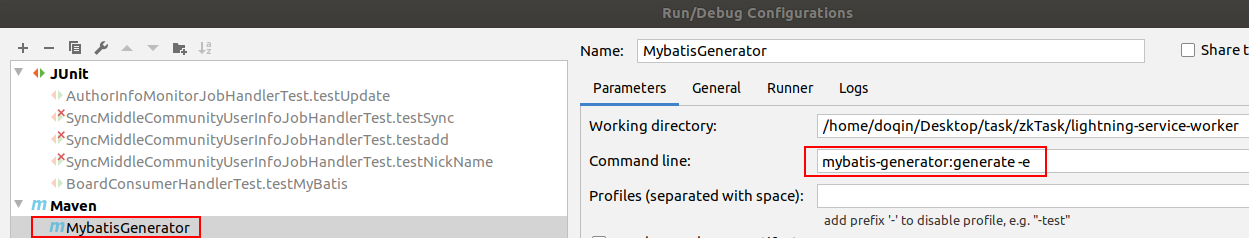
(1)在Pom.xml文件的build中添加

|  |
| --- |
| <**plugin**>  <**groupId**>org.mybatis.generator</**groupId**>  <**artifactId**>mybatis-generator-maven-plugin</**artifactId**>  <**version**>1.3.2</**version**>  <**configuration**>  <**verbose**>true</**verbose**>  <**overwrite**>true</**overwrite**>  </**configuration**> </**plugin**> |

(2)在src/main/resource中添加配置文件generatorConfig.xml

|  |
| --- |
| *<?***xml version="1.0" encoding="UTF-8"***?>* **<!DOCTYPE generatorConfiguration  PUBLIC "-//mybatis.org//DTD MyBatis Generator Configuration 1.0//EN"  "http://mybatis.org/dtd/mybatis-generator-config\_1\_0.dtd"*>*** <**generatorConfiguration**>  *<!-- &lt;!&ndash;指定特定数据库的jdbc驱动jar包的位置&ndash;&gt;-->* <**classPathEntry location="/home/doqin/.m2/repository/mysql/mysql-connector-java/5.1.45/mysql-connector-java-5.1.45.jar"**/>   <**context id="default" targetRuntime="MyBatis3"**>   *<!-- optional，旨在创建class时，对注释进行控制 -->* <**commentGenerator**>  <**property name="suppressDate" value="true"**/>  <**property name="suppressAllComments" value="true"**/>  </**commentGenerator**>   *<!--jdbc的数据库连接 -->* <**jdbcConnection  driverClass="com.mysql.jdbc.Driver"  connectionURL="jdbc:mysql://10.38.164.94:3307/browser?characterEncoding=UTF-8"  userId="alchemy\_x"  password="miuiservermysql"**>  </**jdbcConnection**>    *<!-- 非必需，类型处理器，在数据库类型和java类型之间的转换控制-->* <**javaTypeResolver**>  <**property name="forceBigDecimals" value="false"**/>  </**javaTypeResolver**>  *<!-- Model模型生成器,用来生成含有主键key的类，记录类 以及查询Example类  targetPackage 指定生成的model生成所在的包名  targetProject 指定在该项目下所在的路径  -->* <**javaModelGenerator targetPackage="com.xiaomi.mig3.feeds.lightning.worker.dao.model" targetProject="./src/main/java"**>  *<!-- 是否允许子包，即targetPackage.schemaName.tableName -->* <**property name="enableSubPackages" value="false"**/>  *<!-- 是否对model添加 构造函数 -->* <**property name="constructorBased" value="true"**/>  *<!-- 是否对类CHAR类型的列的数据进行trim操作 -->* <**property name="trimStrings" value="true"**/>  *<!-- 建立的Model对象是否 不可改变 即生成的Model对象不会有 setter方法，只有构造方法 -->* <**property name="immutable" value="false"**/>  </**javaModelGenerator**>   *<!--mapper映射文件生成所在的目录 为每一个数据库的表生成对应的SqlMap文件 -->* <**sqlMapGenerator targetPackage="mappers" targetProject="./src/main/resources"**>  <**property name="enableSubPackages" value="false"**/>  </**sqlMapGenerator**>   <**javaClientGenerator type="XMLMAPPER" targetPackage="com.xiaomi.mig3.feeds.lightning.worker.dao.model" targetProject="./src/main/java"**>  *<!-- enableSubPackages:是否让schema作为包的后缀 -->* <**property name="enableSubPackages" value="false"** />  </**javaClientGenerator**>   <**table tableName="xindao\_island\_config" domainObjectName="XindaoIslandConfig" enableCountByExample="true" enableUpdateByExample="true" enableDeleteByExample="true" enableSelectByExample="true" selectByExampleQueryId="true"**></**table**>   </**context**> </**generatorConfiguration**> |

(3)在Intellij IDEA添加一个“Run运行”选项，使用maven运行mybatis-generator-maven-plugin插件点击 菜单run中Edit Configurations,会出现



在command line中输入：mybatis-generator:generate -e

(4)执行

Screenshot from 2020-12-15 14-33-20