

**##Oracle 分析函数**

**实现 分组显示第一条的效果**

SELECT

--count(\*)

\*

FROM (SELECT **ROW\_NUMBER() OVER(PARTITION BY civilregionalismid,reportdate,bblx,isvalide,state ORDER BY registerdate DESC)** rn, hjcnhistory.\* FROM hjcnhistory)

WHERE rn > 1

and isvalide='01'

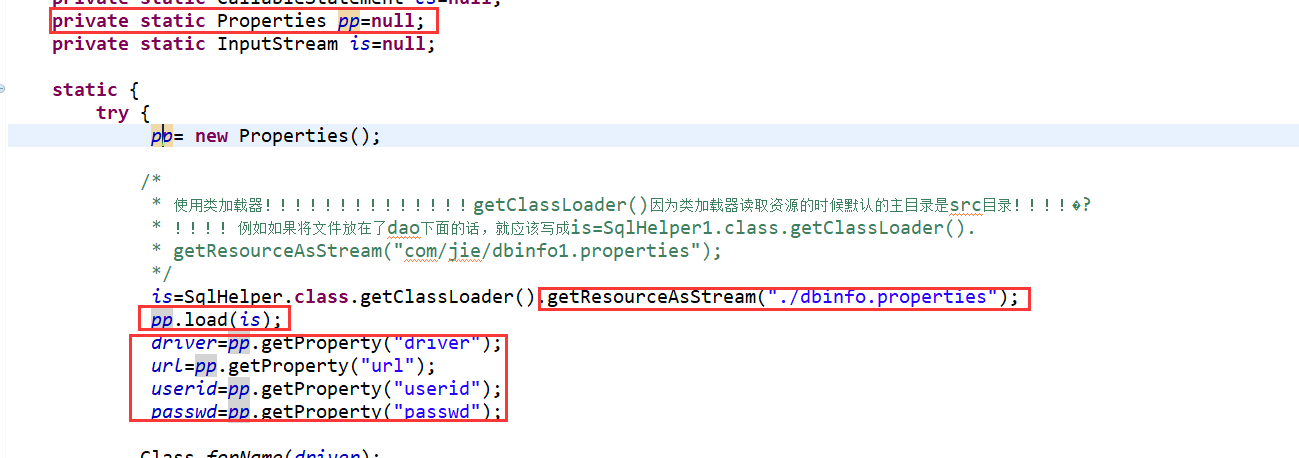
and state='4'

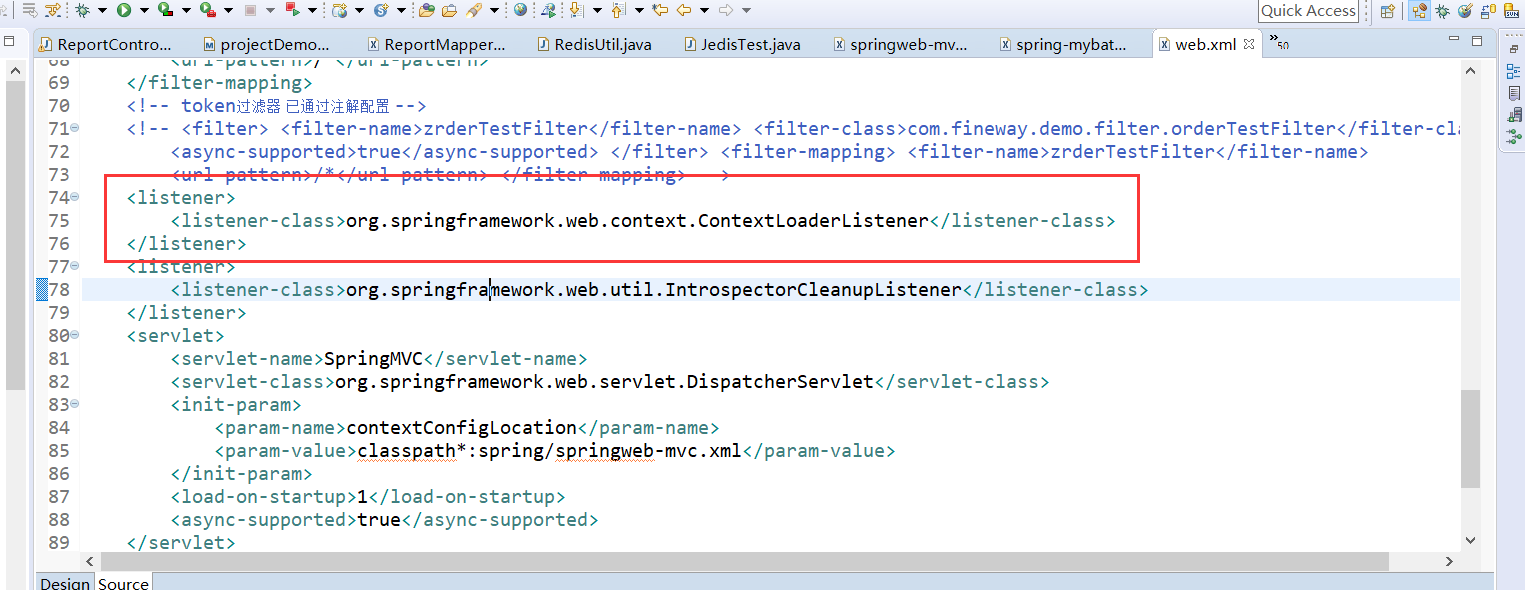
--and civilregionalismid='HJCN2017230011' and reportdate='2017-04'

order by civilregionalismid,bblx,registerdate

;

sh命令是调用的shell解析器，因此只要对文件的权限最低要求是可读就行了，而./则要求得更高一些，要有可读、写、执行权限。





不加此监听器会导致service注解@Autowired注入失败

东方通web服务器 后台启动+指定日志输出文件目录

nohup ./tongserver &>>/home/tongweb.log

使用ps –ef|grep java 查看TongWeb 的进程号

使用prstat –L –p 13516(注:13516为Java 进程号)查看Java 进程中每个线程占用的CPU



Hibernate指定字段查询 + 关联查询 + 自定义对象接收

grep -rn 字符串 .

查询当前目录下包含指定字符串的文件

Git上传大文件失败fatal: The remote end hung up unexpectedly

修改git config --global http.postBuffer 2428000

Git add .

Git commit -m ‘’

Git push

ls --full-time

grep -rn ‘字符串’ .

./startup.sh和catalina.sh run 区别：

startup.sh本质是执行catalina.sh start

项目根目录下 (Cd [项目名])

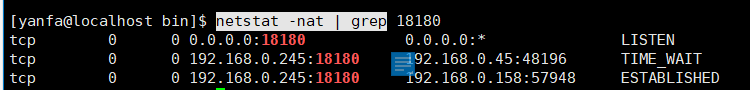
gradle clean

gradle build -i

(加 -i 输出信息 ；war生成在./build/lib下)

jvm内存排查监测 **jvisualvm jconsole**

netstat -nat | grep 【字符串】

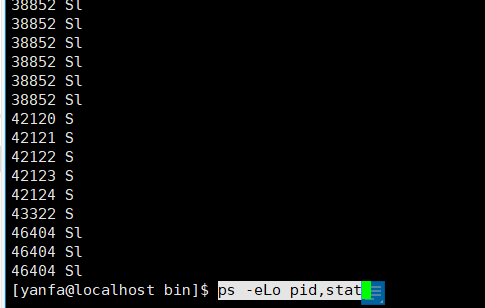


Window查看端口

netstat -aon | findstr :【关键字】

执行命令 ps -eLo pid,stat

显示



解释

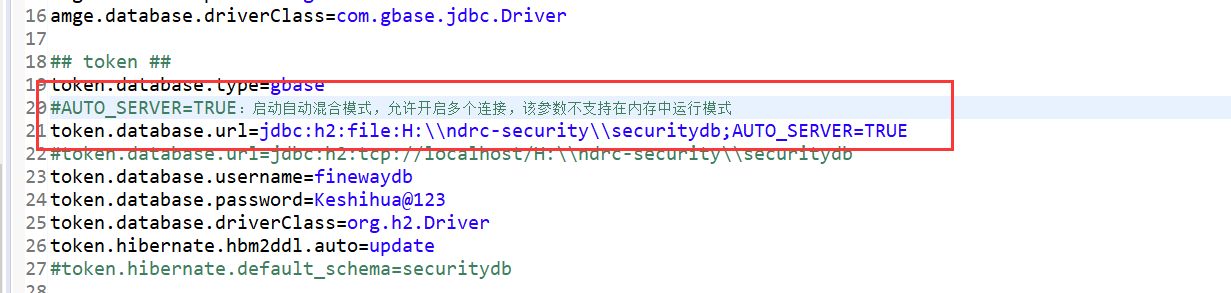
D    不可中断     Uninterruptible sleep (usually IO)  
    R    正在运行，或在队列中的进程  
    S    处于休眠状态  
    T    停止或被追踪  
    Z    僵尸进程  
    W    进入内存交换（从内核2.6开始无效）  
    X    死掉的进程  
  
  
    <    高优先级  
    N    低优先级  
    L    有些页被锁进内存  
    s    包含子进程  
    +    位于后台的进程组；  
    l    多线程，克隆线程  multi-threaded (using CLONE\_THREAD, like NPTL pthreads do)

Eclipse启动web项目（h2数据库，嵌入式）报错：file is locked

（打包部署到tomcat正常启动）

解决办法：

添加url参数



JAVA\_OPTS='-server -Xms1536m -Xmx8192m -XX:PermSize=256m -XX:MaxPermSize=512m'

Maven命令行打包

mvn clean package -DskipTests

（-DskipTests 跳过单元测试）

Maven ：

Build 是编译

Package 是打包但不放进本地库

Install 是打包并放进本地库不放进远程库

Maven打包指定生成的war名称

<!-- 产生的构件的文件名，默认值是${artifactId}-${version}。 -->

<finalName>ShareData</finalName>

Maven指定jdk版本打包

<!-- compiler插件 -->

<plugin>

<groupId>org.apache.maven.plugins</groupId>

<artifactId>maven-compiler-plugin</artifactId>

<version>3.2</version>

<configuration>

<source>${jdk.version}</source>

<target>${jdk.version}</target>

<!-- <source>1.8</source>

<target>1.8</target> -->

<showWarnings>true</showWarnings>

<showDeprecation>true</showDeprecation>

<verbose>false</verbose>

<compilerArguments>

<extdirs>libs</extdirs>

</compilerArguments>

</configuration>

</plugin>

nohup command>/dev/null 2>&1 &

截取文件指定行

sed -n '432437,581294p' fgw\_keshihua120190401.sql > keshihualast2019\_5.sql



Gbase数据库命令行登录

gccli -ugbase -pgbase20110531

gbase数据库重启（全部服务重启）

service gcware restart

更新嵌套查询SQL

**update** t\_res t0 **set** t0.urlPath=(**select** **replace**(t1.urlPath,'192.168.57.101:8082','192.168.0.245:18180') **from** t\_res t1 **where** t0.code=t1.code) **where** t0.urlPath **like** '%:8082%';

Mysql不支持merge语法（可能my..存储引擎支持），使用

“ON DUPLICATE KEY” 语法代替

**insert** **into** t\_user\_photo(id,user\_level,upload\_time) **values**('ee85c3ad09d94925b5bdf0d001ef3437000',1,**now**()) **ON** DUPLICATE **KEY** **UPDATE** id=id;

linux防火墙

查看 firewall-cmd --list-all

永久开放端口 firewall-cmd --zone=public --add-port=54321/tcp --permanent

重新加载 firewall-cmd --reload

参考 <https://blog.csdn.net/u013514928/article/details/80411110>

Linux进程端口号查看

1. ps -ef |grep java
2. netstat -nltp |grep 80
3. 杀死进程 ：kill -9 【pid】

Windows进程查看

1. tasklist |findstr java
2. netstat -ano|findstr 80
3. taskkill -PID 【pid】 -F (-F:强制杀死进程)

idea整合springboot+jpa+docker

参考: <https://www.cnblogs.com/hsz-csy/p/9488469.html>

重点是pom.xml和dockerfile配置

Linux netstat参数

-a (all)显示所有选项，netstat默认不显示LISTEN相关

-t (tcp)仅显示tcp相关选项

-u (udp)仅显示udp相关选项

-n 拒绝显示别名，能显示数字的全部转化成数字。(重要)

-l 仅列出有在 Listen (监听) 的服務状态

-p 显示建立相关链接的程序名(macOS中表示协议 -p protocol)

-r 显示路由信息，路由表

-e 显示扩展信息，例如uid等

-s 按各个协议进行统计 (重要)

-c 每隔一个固定时间，执行该netstat命令。

参考：<https://www.cnblogs.com/echo1937/p/6677325.html>

JPA手动拼接SQL

1. 注入

@PersistenceContext  
private EntityManager entityManager;

2、

Query query = entityManager.createNativeQuery(sql);  
List list = query.getResultList();

(createNativeQuert可加第二个参数指定返回实体类.class)

Windows退出telnet窗口

Ctrl+] 🡪 quit

Linux硬链接、软链接

创建：

硬： ln 真实文件绝对路径 链接绝对路径

（文件夹只能软链接）

软： ln -s 真实文件夹绝对路径 软链接地址

删除：

Rm 链接文件

（最后不能加/ ，否则为删除真实文件目录）

修改：

软：ln -snf 新文件路径 软链接地址

Mysql8修改了密码加密方式使用以前的工具无法登陆

修改加密方式

方法一：

ALTER USER 'root'@'localhost' IDENTIFIED WITH mysql\_native\_password BY 'root';

方法二：

修改mysql配置文件 /etc/my.cnf

default\_authentication\_plugin=mysql\_native\_password

Docker启动mysql镜像

docker run --name mysql -p 3306:3306 -e MYSQL\_ROOT\_PASSWORD=123456 -d mysql

docker exec -it mysql /bin/bash

Git切换分支

git checkout 【分支名】

Git取消add 根据head匹配

git reset HEAD .

Springboot使用定时任务

第一步：在启动类添加注解@SpringBootApplication

第二步：注入类的方法上添加注解@Scheduled

使用cron表达式定义执行时间

注意：上述方法，如果有多个定时任务将在一个线程中执行，会有任务阻塞现象

Curl发送post请求实例（windows、linux通用）：

curl -H Content-Type:application/json;charset=UTF-8 -XPOST http://localhost:19002/jm/word/test -d "{\"id\":\"qwe\",\"name\":\"ls\",\"url\":\"XXX\",\"overview\":\"as\"}"

注意：1、-H 后面不能加引号；2、-d 后面必须为双引号;3、cmd中由于windows默认GBK编码所以不能携带中文

Tomcat部署两个相同的系统，只能启动一个

解决：

向其中一个系统Web.xml添加配置：

<context-param>

<param-name>webAppRootKey</param-name>

<param-value>project.root1</param-value>

</context-param>

（即重命名webAppRootKey）

Springboot启动修改jvm参数

java -jar -XX:MetaspaceSize=128m -XX:MaxMetaspaceSize=128m -Xms1024m -Xmx1024m -Xmn256m -Xss256k -XX:SurvivorRatio=8 -XX:+UseConcMarkSweepGC XXX.jar

参数说明：

-XX:MetaspaceSize=128m （元空间默认大小）

-XX:MaxMetaspaceSize=128m （元空间最大大小）

-Xms1024m （堆最大大小）

-Xmx1024m （堆默认大小）

-Xmn256m （新生代大小）

-Xss256k （棧最大深度大小）

-XX:SurvivorRatio=8 （新生代分区比例 8:2）

-XX:+UseConcMarkSweepGC （指定使用的垃圾收集器，这里使用CMS收集器）

-XX:+PrintGCDetails （打印详细的GC日志）

注意：

JDK8之后把-XX:PermSize 和 -XX:MaxPermGen移除了，取而代之的是

-XX:MetaspaceSize=128m （元空间默认大小）

-XX:MaxMetaspaceSize=128m （元空间最大大小）

JDK 8开始把类的元数据放到本地化的堆内存(native heap)中，这一块区域就叫Metaspace，中文名叫元空间。

使用本地化的内存有什么好处呢？最直接的表现就是java.lang.OutOfMemoryError: PermGen 空间问题将不复存在，因为默认的类的元数据分配只受本地内存大小的限制，也就是说本地内存剩余多少，理论上Metaspace就可以有多大（貌似容量还与操作系统的虚拟内存有关？这里不太清楚），这解决了空间不足的问题。不过，让Metaspace变得无限大显然是不现实的，因此我们也要限制Metaspace的大小：使用-XX:MaxMetaspaceSize参数来指定Metaspace区域的大小。JVM默认在运行时根据需要动态地设置MaxMetaspaceSize的大小。