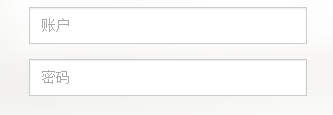
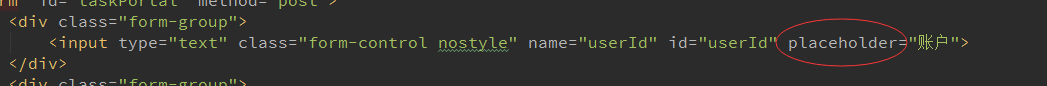
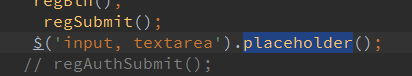
1. 

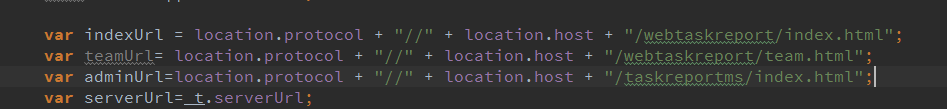
设置



在调用



2，



Location.protocol = “http:”

//即协议

Location.hostname = “” 127.0.0.1:8080

//即域名

3，

url 中含有#，+，%，&…等时会有问题，先转码

inputComment = inputComment.replace(/\%/g,"%25");  
inputComment = inputComment.replace(/\#/g,"%23");  
inputComment = inputComment.replace(/\&/g,"%26");  
inputComment = inputComment.replace(/\+/g,"%2B");

4， ../ 表示上级目录 ./ 表示当前目录 /表示根目录

项目目录一般在跟目录下：

比如： /project

5

 一般情况下，Select Count (\*)和Select Count(1)两着返回结果是一样的

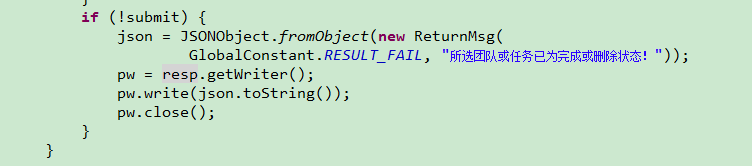
    假如表沒有主键(Primary key), 那么count(1)比count(\*)快，

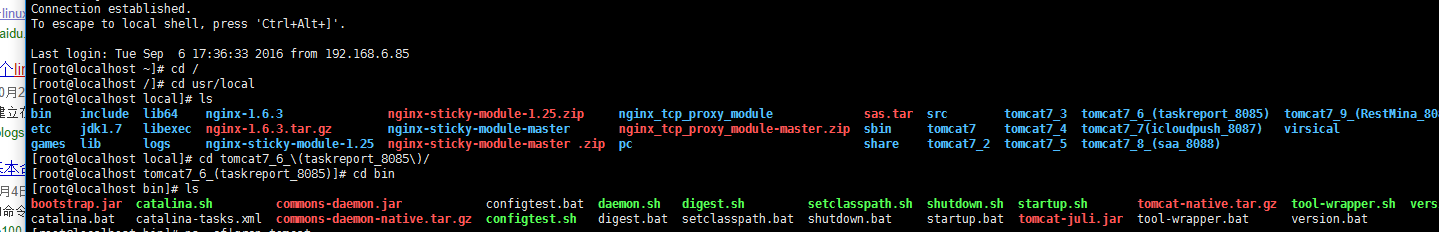
    如果有主键的話，那主键作为count的条件时候count(主键)最快

    如果你的表只有一个字段的话那count(\*)就是最快的

   count(\*) 跟 count(1) 的结果一样，都包括对NULL的统计，而count(column) 是不包括NULL的统计

HttpServletResponse response





Linux 指令： cd / ----gen2路径

cd xx/xxx ------进入具体目录

S命令：

-ls 列出 文件的详细信息

Cd xxxx\(xxx/)/ 转义

Cd bin

linux ps -ef |grep tomcat 查看当前tomcat进程

#kill -9 6211 杀掉进程号

#clear

#pwd

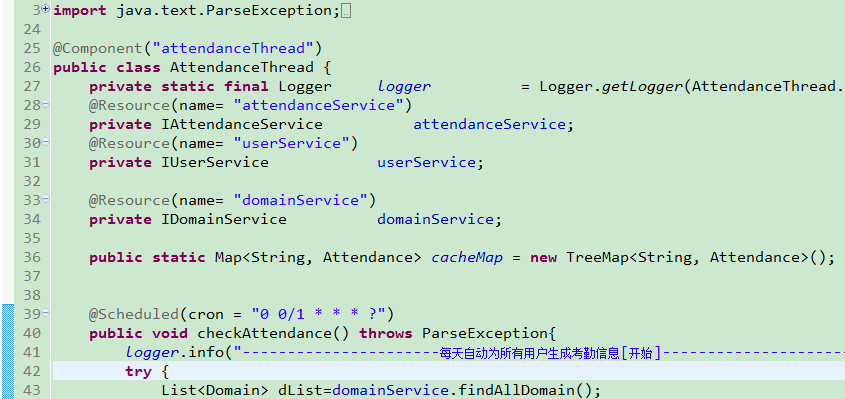
./filename 执行这个文件

Tail –n 行数 文件名

Tail –f 行数 文件名 动态刷新最后行数

MultipartFile Spring 文件上传

6quartz 注解定时触发



需要注意的几点：

1、spring的@Scheduled注解  需要写在实现上、

2、 定时器的任务方法不能有返回值（如果有返回值，spring初始化的时候会告诉你有个错误、需要设定一个proxytargetclass的某个值为true、具体就去百度google吧）

3、实现类上要有组件的注解@Component

AES 加密：



Hibernate 中 getsessin saveorupdate 方法如果updat失效，可以用 flush（）方法提交事务1

$("#groupId").children('option:selected').val()



Ecxcel

response.setContentType("Application/Octet-Stream");

response.setHeader("Content-Disposition", "attachment; filename=\"Month\_" + user.getAccount() + "\_"

+ today + ".xls\"");

response.setHeader("Pragma", "public");

response.setHeader("Cache-Control", "public");

OutputStream os = response.getOutputStream();

excel.getContent().write(os);

os.flush();

os.close();

日期：

**if**(typeFlag == 0)

{

String startDateStr = DateUtils.*convertToDateStr*(DateUtils.*getNowDate*());

startDateStr = startDateStr.substring(0, startDateStr.length()-2)+"01";

startDate = DateUtils.*convertToDate*(startDateStr);

lastDate = DateUtils.*getNowDateAndTime*();

}**else** **if**(typeFlag == 1){

//第一天

Calendar calendarFirst = Calendar.*getInstance*();

calendarFirst.add(Calendar.*MONTH*, -1);

calendarFirst.set(Calendar.*DAY\_OF\_MONTH*,1);

startDate = DateUtils.*convertToDate*(DateUtils.*convertToDateStr*(calendarFirst.getTime()));

//最后一天

Calendar calendarLast = Calendar.*getInstance*();

calendarLast.add(Calendar.*MONTH*, -1);

calendarLast.set(Calendar.*DAY\_OF\_MONTH*,calendarFirst.getActualMaximum(Calendar.*DAY\_OF\_MONTH*));

lastDate = DateUtils.*convertToDateAndTime*(DateUtils.*convertToDateStr*(calendarLast.getTime())+" "+"23:59:59");

}**else** **if**(typeFlag == 2){

//本周第一天 （周一）

Calendar calendarFirst = Calendar.*getInstance*();

calendarFirst.setFirstDayOfWeek(Calendar.*MONDAY*);

calendarFirst.set(Calendar.*DAY\_OF\_WEEK*, calendarFirst.getFirstDayOfWeek());

startDate = DateUtils.*convertToDate*(DateUtils.*convertToDateStr*(calendarFirst.getTime()));

//本周当前日期

lastDate = DateUtils.*getNowDateAndTime*();

}**else**{

Calendar calendarFirst = Calendar.*getInstance*();

calendarFirst.add(Calendar.*DAY\_OF\_WEEK*, -1\*7);

calendarFirst.setFirstDayOfWeek(Calendar.*MONDAY*);

calendarFirst.set(Calendar.*DAY\_OF\_WEEK*, calendarFirst.getFirstDayOfWeek());

startDate = DateUtils.*convertToDate*(DateUtils.*convertToDateStr*(calendarFirst.getTime()));

Calendar calendarLast = Calendar.*getInstance*();

calendarLast.add(Calendar.*DAY\_OF\_WEEK*, -1\*7);

calendarLast.setFirstDayOfWeek(Calendar.*MONDAY*);

calendarLast.set(Calendar.*DAY\_OF\_WEEK*, Calendar.*SUNDAY*);

lastDate = DateUtils.*convertToDateAndTime*(DateUtils.*convertToDateStr*(calendarLast.getTime())+" "+"23:59:59");

}