# Java笔记

## 工具方法

### 去掉字符串中重复的字符

|  |
| --- |
| /\* 去掉字符串重复的字符 \*/  Set<String> mLinkedSet = **new** LinkedHashSet<String>();  String[] strArray = userIds.split(",");  StringBuffer sb = **new** StringBuffer();  **for** (**int** j = 0; j < strArray.length; j++) {  **if** (!mLinkedSet.contains(strArray[j])) {  mLinkedSet.add(strArray[j]);  sb.append(strArray[j] + ",");  }  }  userIds = sb.toString().substring(0, sb.toString().length() - 1); |

### 取将list的对象的值，并且去掉重复的对象值

|  |
| --- |
| Set<Integer> mLinkedSet = **new** LinkedHashSet<Integer>();  **for** (User us : users) {// 先取出所有的id，生成字符串，然后去掉重复的id  **if** (!mLinkedSet.contains(us.getUserId())) {  mLinkedSet.add(us.getUserId());  }  }  得出的mlinkedset是没有重复的 |

### StringUtils

字符串工具类，需要引入jar包

|  |
| --- |
| if(StringUtils.contains(messageType, "1")){  \* StringUtils.contains(null, \*) = false  \* StringUtils.contains(\*, null) = false  \* StringUtils.contains("", "") = true  \* StringUtils.contains("abc", "") = true  \* StringUtils.contains("abc", "a") = true  \* StringUtils.contains("abc", "z") = false  \* StringUtils.contains("abc", "A") = true  \* StringUtils.contains("abc", "Z") = false |

### Trim()是去两边空格的方法

|  |
| --- |
| toString().trim()是什么意思？  [小丸子坲](http://zhidao.baidu.com/usercenter?uid=1d994069236f25705e794d1c) | 浏览 50827 次  问题未开放回答 |举报  邀请更新  发布于2013-08-25 16:39  最佳答案  获得对象的字段的值，然后转成string类型，并且去掉前后空白~~  ToString()是转化为字符串的方法 Trim()是去两边空格的方法  把StringBuffer转换成String类型时 没有用.trim()会导致意想不到的结果－－  用[if语句](https://www.baidu.com/s?wd=if%E8%AF%AD%E5%8F%A5&tn=44039180_cpr&fenlei=mv6quAkxTZn0IZRqIHckPjm4nH00T1Y3rj6dn1csnWubnyPBP19B0ZwV5Hcvrjm3rH6sPfKWUMw85HfYnjn4nH6sgvPsT6KdThsqpZwYTjCEQLGCpyw9Uz4Bmy-bIi4WUvYETgN-TLwGUv3EPH6dn1m1nH6k)判断时跟想的不一样  if(returnStr==null||returnStr==""){//没有.trim()这句通不过  //if(returnStr.length()==0){//判断长度为零没有.trim()也可以。 |

### 获取两个数组之间的交集或者不相交的集合

|  |
| --- |
| */\*get t1里面不包含t2 的数据\*/* **public static** <*T*> **List**<*T*> compare(*T*[] *t1*, *T*[] *t2*) {  **List**<*T*> list1 = **Arrays**.**asList**(*t1*);  **List**<*T*> list2 = **new** ArrayList<*T*>();  **for** (*T* t : *t2*) {  **if** (!list1.contains(t)) {  list2.add(t);  }  }  **return** list2; } |

### StringUtils

|  |
| --- |
| 倒commons-lang-2.3.jar包  例1: 判断是否为空(blank): StringUtils.isBlank(null): true StringUtils.isBlank(""): true StringUtils.isBlank(" "): true  例2: 判断是否不为空(blank): StringUtils.isNotBlank(null): false StringUtils.isNotBlank(""): false StringUtils.isNotBlank(" "): false  例3: 判断是否为空(empty): StringUtils.isEmpty(null): true StringUtils.isEmpty(""): true StringUtils.isEmpty(" "): false  例4: 判断是否不为空(empty): StringUtils.isNotEmpty(null): false StringUtils.isNotEmpty(""): false StringUtils.isNotEmpty(" "): true  例5: 判断是否是字母: StringUtils.isAlpha("ab"): true StringUtils.isAlpha("CD"): true StringUtils.isAlpha("efGH"): true StringUtils.isAlpha("12"): false StringUtils.isAlpha("12ab"): false  例6: 判断是否是数字: StringUtils.isNumeric("ab"): false StringUtils.isNumeric("CD"): false StringUtils.isNumeric("efGH"): false StringUtils.isNumeric("12"): true StringUtils.isNumeric("12ab"): false  例7: 判断是否是字母或数字: StringUtils.isAlphanumeric("ab"): true StringUtils.isAlphanumeric("CD"): true StringUtils.isAlphanumeric("efGH"): true StringUtils.isAlphanumeric("12"): true StringUtils.isAlphanumeric("12ab"): true  字符串trim及默认处理： 例1: 去除字符串中的空白(包括全角空格、TAB). StringUtils.deleteWhitespace("aa bb　cc"): aabbcc StringUtils.deleteWhitespace(" aa  bb    cc "): aabbcc  例2: 去除字符串头尾空白(包括全角空格、TAB). StringUtils.trim(" abc  "): abc StringUtils.trim("　abc"): 　abc StringUtils.trim("    abc"): abc  例3: 去除字符串头尾空白(包括TAB,但不包括全角空格), 结果为空串时返回空串. StringUtils.trimToEmpty("   "): StringUtils.trimToEmpty("　"): 　 StringUtils.trimToEmpty("    "):  例4: 去除字符串头尾空白(包括TAB,但不包括全角空格), 结果为空串时返回null. StringUtils.trimToNull("   "): null StringUtils.trimToNull("　"): 　 StringUtils.trimToNull("    "): null  例5: 返回可以处理null的toString(). StringUtils.defaultString("aaaa"): aaaa    字符串包含检查： 例1: 检查字符串(参数1)是否包含字符串(参数2). StringUtils.contains("defg", "ef"): true StringUtils.contains("defg", "EF"): false StringUtils.contains("defg", "fe"): false StringUtils.contains("ef", "defg"): false StringUtils.contains("aad", "aadd"): false StringUtils.contains("defg", "aadd"): false  例2: 检查字符串(参数1)是否包含字符串(参数2), 忽略大小写. StringUtils.containsIgnoreCase("defg", "EF"): true  例3: 检查字符串(参数1)中的字符是否全为字符串(参数2)中的字符的子集. StringUtils.containsOnly("defg", "ef"): false StringUtils.containsOnly("ef", "defg"): true StringUtils.containsOnly("fe", "defg"): true StringUtils.containsOnly("aad", "aadd"): true StringUtils.containsOnly("defg", "aadd"): false  例4: 检查字符串(参数1)和字符串(参数2)中的字符是否不存在交集. StringUtils.containsNone("defg", "ef"): false StringUtils.containsNone("defg", "fe"): false StringUtils.containsNone("ef", "defg"): false StringUtils.containsNone("aad", "aadd"): false StringUtils.containsNone("defg", "aadd"): false StringUtils.containsNone("defg", "xvy"): true  例5: 检查字符串(参数1)和字符串(参数2)中的字符是否存在交集. StringUtils.containsAny("defg", "ef"): true StringUtils.containsAny("defg", "fe"): true StringUtils.containsAny("ef", "defg"): true StringUtils.containsAny("aad", "aadd"): true StringUtils.containsAny("defg", "aadd"): true StringUtils.containsAny("defg", "xvy"): false    字符串处理： 例1: 将字符串重复n次，将文字按某宽度居中，将字符串数组用某字符串连接. //代码片段: String[] header = new String[3]; header[0] = StringUtils.repeat("\*", 50); header[1] = StringUtils.center("  StringUtilsDemo  ", 50, "^O^"); header[2] = header[0]; String head = StringUtils.join(header, "\n"); System.out.println(head); //运行结果: \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* ^O^^O^^O^^O^^O^  StringUtilsDemo  ^O^^O^^O^^O^^O^^ \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  例2: 缩短到某长度,用...结尾. StringUtils.abbreviate("The quick brown fox jumps over the lazy dog.", 10): The qui... StringUtils.abbreviate("The quick brown fox jumps over the lazy dog.", 15, 10): ... fox... |

## 后端

### 前端传Json字符串过来中文乱码

|  |
| --- |
| **String** strw = request.getParameter(**"content"**); |

### Calendar

|  |
| --- |
| **不能写成这样 ,否则在输出的时候,显示的日期有误**  **SimpleDateFormat format = new SimpleDateFormat("yyyy-MM-dd");**  s  SimpleDateFormat format = new SimpleDateFormat("yyyy-MM-dd **HH:mm:ss**");          Calendar c = Calendar.getInstance();            //过去七天          c.setTime(new Date());          c.add(Calendar.DATE, - 7);          Date d = c.getTime();          String day = format.format(d);          System.out.println("过去七天："+day);            //过去一月          c.setTime(new Date());          c.add(Calendar.MONTH, -1);          Date m = c.getTime();          String mon = format.format(m);          System.out.println("过去一个月："+mon);            //过去三个月          c.setTime(new Date());          c.add(Calendar.MONTH, -3);          Date m3 = c.getTime();          String mon3 = format.format(m3);          System.out.println("过去三个月："+mon3);            //过去一年          c.setTime(new Date());          c.add(Calendar.YEAR, -1);          Date y = c.getTime();          String year = format.format(y);          System.out.println("过去一年："+year); |

### 获取两个日期相差的天数

|  |
| --- |
| //获取两个日期相差天数 **public int** getDiscrepantDays(**Date** *dateStart*, **Date** *dateEnd*) {  //先将小时,分,秒设置为零,防止转换造成一天的误差  **Calendar** calendar=**Calendar**.**getInstance**();  calendar.setTime(*dateStart*);  calendar.set(calendar.get(**Calendar**.***YEAR***),calendar.get(**Calendar**.***MONTH***),calendar.get(**Calendar**.***DAY\_OF\_MONTH***),**0**,**0**,**0**);  **Date** date1 = calendar.getTime();  calendar.setTime(*dateEnd*);  calendar.set(calendar.get(**Calendar**.***YEAR***),calendar.get(**Calendar**.***MONTH***),calendar.get(**Calendar**.***DAY\_OF\_MONTH***),**0**,**0**,**0**);  **Date** date2 = calendar.getTime();  **long** day = (date2.getTime() - date1.getTime()) / (**24**\***3600**\***1000**);  **return** (**int**)day+**1**; } |

## 数据库

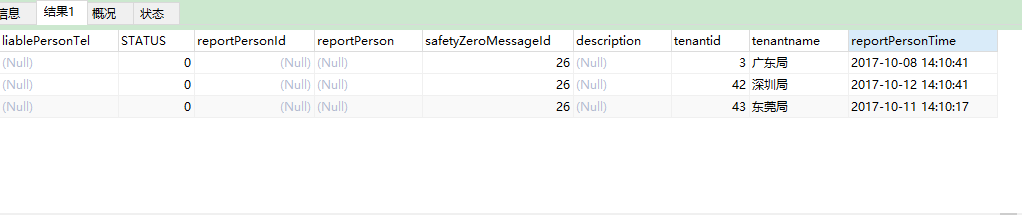
### 查询某一天的数据

|  |
| --- |
| SELECT  DATE\_FORMAT(receiveTime, "%Y-%m-%d") as receiveTime  FROM  ebs\_safetyzeromessage\_details as d  WHERE  d.safetyZeroMessageId =#{safetyZeroMessageId} AND d.tenantId=#{tenantId} AND d.status=1  GROUP BY  DATE\_FORMAT(receiveTime, "%Y-%m-%d") |

**DATE\_FORMAT(receiveTime, "%Y-%m-%d") ：去掉时分秒**

### 先排序后分组

需求：tenantId分组，并且取每一组中反馈时间最大的记录



|  |
| --- |
| SELECT \*  FROM  (  SELECT  d.safetyZeroMessageDetailId,  d.speaker,  d.liablePerson,  d.liablePersonTel,  d. STATUS,  d.reportPersonId,  d.reportPerson,  d.safetyZeroMessageId,  d.description,  t.tenantid,  t.tenantname,  d.reportPersonTime  FROM  ebs\_safetyzeromessage\_details d,  tenant t  WHERE  d.safetyZeroMessageId=#{safetyZeroMessageId}  AND d.tenantId = t.tenantId  GROUP BY  t.tenantid,d.reportPersonTime  ORDER BY  d.reportPersonTime DESC    ) m  GROUP BY m.tenantid |

### Sql字符串参数批量操作--- foreach

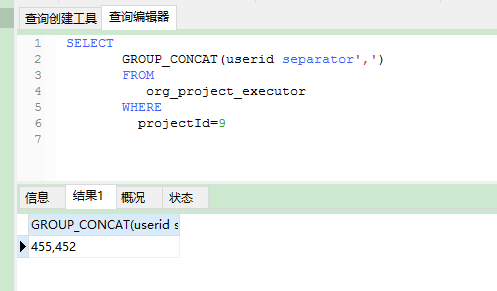
**批量删除tenantId数组**

**String**[] tenantIds = *ids*.split(**","**);  
tenantService.deleteTenant(tenantIds);

<**delete** *id***="deleteTenant"** *parameterType***="java.lang.String"**>  
 delete from tenant WHERE tenantId in  
 <**foreach** *collection***="tenantIds"** *index***="index"** *item***="tenantId"** *open***="("** *separator***=","** *close***=")"**>  
 *#{tenantId}* </**foreach**>  
</**delete**>

### 查询--返回字符串数组

|  |
| --- |
| <**select** *id***="getUserIdByProjectId"** *resultType***="java.lang.String"**>  SELECT  GROUP\_CONCAT(userid separator',')  FROM  org\_project\_executor  WHERE  projectId=*#{projectId}* </**select**> |



### 插入数据库有Id返回

|  |
| --- |
| <**insert** *id***="add"** *parameterType***="OrgProject" *keyProperty*="projectId" *useGeneratedKeys*="true"**>  insert into org\_project (projectCode, projectName,  )  values (*#{projectCode,jdbcType=VARCHAR}*, *#{projectName}*, ) </**insert**> |

## Tomcat

### 解决tomcat的时间与本地的时间不一致

[**http://ichair-126-com.iteye.com/blog/1554667**](http://ichair-126-com.iteye.com/blog/1554667)

方法一，在catalina.bat中

配置如下：

|  |
| --- |
| set JAVA\_OPTS=%JAVA\_OPTS% -Duser.timezone=GMT+08 -Xms256m -Xmx800m   -Djava.util.logging.manager=org.apache.juli.ClassLoaderLogManager   -Djava.util.logging.config.file="%CATALINA\_BASE%\conf\logging.properties"   -Xms256m -Xmx800m（初始化内存大小为256m，可以使用的最大内存为800m），   -Duser.timezone=GMT+08 //设置为北京时间 |

