2018年8月30日星期四

1、把集合转换成字符串

StringUtils.join(user.getFav(),",");

2、字符串转换成集合

//字符串

String str="篮球,足球,排球";

//用逗号将字符串分开，得到字符串数组

String[] strs=str.split(",");

//将字符串数组转换成集合list

List list=Arrays.asList(strs);

3、防止mybatis查询mysql字段为null

ifnull(PREPAYMENT\_FEE,0)

4、mybatis中取参数值的两种方式 #{param}（解析出来的值带单引号） ${param}

5、MyBatis的if、when里面的test表达式对参数进行判断时，可以调用java的java.lang.String中定义的方法

6、mybatis 的foreach collection

1. 如果传入的是单参数且参数类型是一个List的时候，collection属性值为list .
2. 如果传入的是单参数且参数类型是一个array数组的时候，collection的属性值为array .
3. 如果传入的参数是多个的时候，我们就需要把它们封装成一个Map了，当然单参数也可以封装成map，实际上如果你在传入参数的时候，在MyBatis里面也是会把它封装成一个Map的，map的key就是参数名，所以这个时候collection属性值就是传入的List或array对象在自己封装的map里面的key.

8、flatmap(stream) 返回stream

9、小数据点保留位数

|  |
| --- |
| double f = 111231.5585;  BigDecimal b = new BigDecimal(f);  double f1 = b.setScale(2, BigDecimal.ROUND\_HALF\_UP).doubleValue(); |

|  |
| --- |
| double d = 3.1415926; String f = String.*format*("%2.3f", d); |

%m.n格式，m表示这个数的宽度，包括[小数点](https://www.baidu.com/s?wd=%E5%B0%8F%E6%95%B0%E7%82%B9&tn=SE_PcZhidaonwhc_ngpagmjz&rsv_dl=gh_pc_zhidao)，如果你实际的数值宽度大于m，那么以实际为准，如果数值实际宽度小于m，那么默认右对齐，前面补空格。n表示[小数点](https://www.baidu.com/s?wd=%E5%B0%8F%E6%95%B0%E7%82%B9&tn=SE_PcZhidaonwhc_ngpagmjz&rsv_dl=gh_pc_zhidao)的位数！

|  |
| --- |
| DecimalFormat df=new DecimalFormat(".##");  double d=1252.2563;  String st=df.format(d); |

|  |
| --- |
| double x=23.5455;  NumberFormat ddf1=NumberFormat.getNumberInstance() ;  ddf1.setMaximumFractionDigits(2);  String s= ddf1.format(x) ; |

下面这种方法有坑，下面这个小数不能正确保留2位小数

Double dd = Math.*round*(0.34525652589491257 \* 100) \* 0.01d;  
System.*out*.println(dd);

因为list.size用#{}是取不出来的，需要用${}，把#换成$就可以了。

#{}能够有效防止SQL注入，但是也有它的缺点，它会把传入的数据自动加上一个双引号，所以如果要的是数字的话，就会比较尴尬。

而${}可以直接解析出原本的数据，所以需要数值比较的话，还是要加${}

10、mybatis

单个参数 可以直接#{参数名}获取

多个参数，会封装成map，key为param1，param2，...paramN，或者参数的索引

通过#{param1}，或者#{0}来获取值

也可用@Param("name")注解指定单数名称 然后就可以#{name}

如果是对象，#{属性名}来获取值

如果是map， #{key}来获取值

##特别注意：如果是Collection（List、Set）类型或者是数组，  
也会特殊处理。也是把传入的list或者数组封装在map中。  
key：Collection（collection）,

如果是List还可以使用这个key(list)数组(array)  
public Employee getEmpById(List<Integer> ids);  
取值：取出第一个id的值：   #{list[0]}

11、主线程等待线程池任务全部结束。

1、ExecutorService 线程池

可以用shutdown()方法，再用isTerminated()判断。

|  |
| --- |
| public static void main(String args[]) throws InterruptedException {           ExecutorService exe = Executors.newFixedThreadPool(50);           for (int i = 1; i <= 5; i++) {               exe.execute(new SubThread(i));           }           exe.shutdown();           while (true) {               if (exe.isTerminated()) {                   System.out.println("结束了！");                   break;               }               Thread.sleep(200);           }       } |

2、实现方案可以用CountDownLatch

在主线程中new一个CountDownLatch对象，参数为在主线程中提交的任务数量

然后在任务中，执行完任务后，使用CountDownLatch.countDown()方法。

CountDownLatch用到了 volatile来保证多线程操作的安全性。

@Data  
@Service  
@Scope("prototype")  
public class InsertDataTask implements Runnable {  
  
 private List<TCookiePushDO> dataList;  
  
 private CountDownLatch latch;  
  
 @Autowired  
 private MysqlService mysqlService;  
  
 @Override  
 public void run() {  
 mysqlService.insertData(dataList);  
 latch.countDown();  
 }  
}

@Test  
public void insertDataTest() {  
 CountDownLatch latch = new CountDownLatch(100);  
 for (int m = 0; m < 10; m ++){  
 long categoryId = Math.*round*(1000000\*Math.*random*());  
 for (int j = 0; j < 10; j ++){  
 List<TCookiePushDO> dataList = new ArrayList<>();  
 for (int i = 0; i < 50000; i ++){  
 TCookiePushDO tCookiePushDO = new TCookiePushDO();  
 tCookiePushDO.setCategoryId(Integer.*valueOf*(String.*valueOf*(categoryId)));  
 String cookieId = UUID.*randomUUID*().toString();  
 tCookiePushDO.setCookieId(cookieId);  
 dataList.add(tCookiePushDO);  
 }  
 InsertDataTask insertDataTask = context.getBean("insertDataTask", InsertDataTask.class);  
 insertDataTask.setDataList(dataList);  
 insertDataTask.setLatch(latch);  
 threadPoolTaskExecutor.submit(insertDataTask);  
 }  
 }

12、String做字符传连接最慢，StringBuilder速度最快，StringBuffer线程安全，速度较慢

总结一下  
**String：适用于少量的字符串操作的情况**

**StringBuilder：适用于单线程下在字符缓冲区进行大量操作的情况**

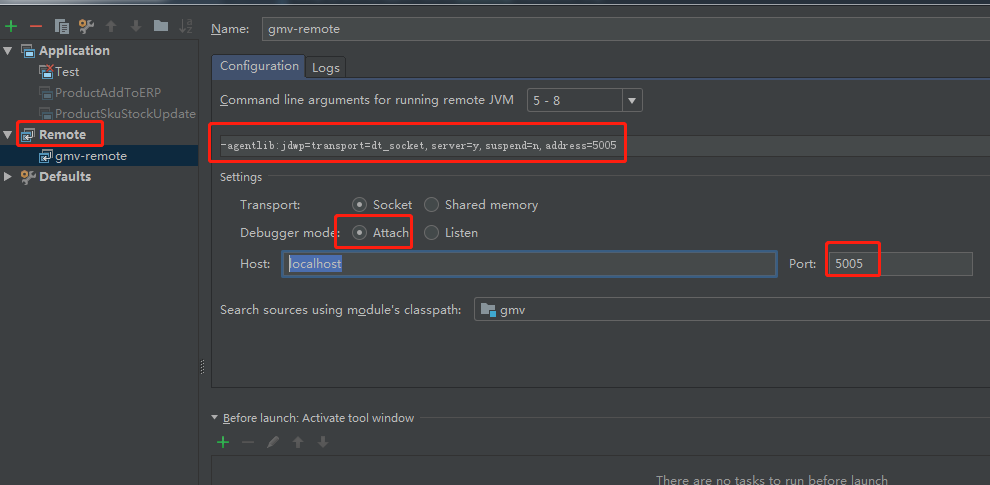
**StringBuffer：适用多线程下在字符缓冲区进行大量操作的情况**

13、java中int类型只能保存32位2机制，long可以保存64位2进制，如果保存的数据很大要用浮点类型float、double类型、但是小数不能做与操作&

14、远程调试jar，jar中是main方法

|  |
| --- |
| java -Xdebug -Xrunjdwp:transport=dt\_socket,address=5005,server=y,suspend=y -jar -Xms128m -Xmx512m gmv-1.0-SNAPSHOT-jar-with-dependencies.jar test yinni sss sss |

在idea中



参考文档：https://blog.csdn.net/wanglei9876/article/details/82145682?utm\_source=copy



这个报错数据太大，是因为0开头的数字是8进制，只有0-7