实体类主键自增方案

直接在实现层调用雪花算法并set进去

sysUser.setId*(*idWorker.nextId*())*;

MyBatis输出sql操作日志

在application.yml文件中mybatis配置下添加代码，

参考：

mybatis:  
 mapper-locations: classpath:mapper/\*Mapper.xml  
 type-aliases-package: com.jg.pochi  
 configuration:  
 log-impl: org.apache.ibatis.logging.stdout.StdOutImpl

实体类序列化

这里以HttpSession序列化为例：

客户端访问了某个能开启会话功能的资源， web服务器就会创建一个与该客户端对应的HttpSession对象，每个HttpSession对象都要站用一定的内存空间。如果在某一时间段内访问站点的用户很多，web服务器内存中就会积累大量的HttpSession对象，消耗大量的服务器内存，即使用户已经离开或者关闭了浏览器，web服务器仍要保留与之对应的HttpSession对象，在他们超时之前，一直占用web服务器内存资源。

web服务器通常将那些暂时不活动但未超时的HttpSession对象转移到文件系统或数据库中保存，服务器要使用他们时再将他们从文件系统或数据库中装载入内存，这种技术称为Session的持久化。

将HttpSession对象保存到文件系统或数据库中，需要采用序列化的方式将HttpSession对象中的每个属性对象保存到文件系统或数据库中；将HttpSession对象从文件系统或数据库中装载进内存时，需要采用反序列化的方式，恢复HttpSession对象中的每个属性对象。所以存储在HttpSession对象中的每个属性对象必须实现Serializable接口

static

用static修饰符修饰变量或者方法即可实现在不new对象情况下用类名. 调用。

接口的静态方法不会被实现类所继承

## Map

1. 修改Map对应的键值

map.put*(*"newKey",map.remove*(*"oldKey"*))*;

Java HashMap remove() 方法

返回值

如果指定 key，返回指定键 key 关联的值，如果指定的 key 映射值为 null 或者该 key 并不存在于该 HashMap 中，此方法将返回null。

如果指定了 key 和 value，删除成功返回 true，否则返回 false。

JDK8中的defalut关键字

作用

1. 父接口方法中用default修饰，可以在实现类中调用父接口方法。
2. 比如有interface1和interface2两个接口有default修饰的hello方法，子类可同时实现这2个接口，由于hello同名，可在子类中实现hello方法解决。

3、若一个子类同实现类不仅继承了MyImplement1并且实现了Interface2,并且MyImplement1和interface2都有同方法名，则类优先级大于接口。

DTO VO DAO

**何为DTO**

DTO（Data Transfer Object），数据传输对象，顾名思义就是用于传输数据的对象，通常用于处于不同架构层次或者不同子系统之间的数据传递，或者用于外部接口参数传递，以便提供不同粒度不同信息的数据，以免造成困惑干扰，通常也是一个JavaBean。

**何为VO**

VO（Value Object），就是用于保存数据的对象；在提供给页面使用的时候，也有人解释为View Object，就是对应页面展示数据的对象，简而言之就是与前端对应,此时的VO类就不要实现序列化了(implements Serializable)。

**何为DAO**

DAO（Data Access Object），数据访问对象，与数据库做交互的对象，提供不同的接口访问数据库来实现对数据库的操作，而接口使用的数据交互通常就是PO或者DO，通过它可以使用面向对象的方式来与数据库交互。

**BeanUtils.copyProperties()**

注意是Spring的包，不要用Apache的包性能很差。

作用：

将一个对象的属性拷贝到另一个对象。

例如：  
public void save*(*SysUserVo sysUser*) {* //拷贝属性  
 SysUser user = new SysUser*()*;  
 BeanUtils.*copyProperties(*sysUser,user*)*;  
 //ID用雪花算法生成  
 sysUser.setId*(*idWorker.nextId*())*;  
 sysUserMapper.save*(*sysUser*)*;  
*}*

@Transactional(rollbackFor=Exception.class)

当作用于类上时，该类的所有 public 方法将都具有该类型的事务属性，同时，我们也可以在方法级别使用该标注来覆盖类级别的定义。

在项目中，@Transactional(rollbackFor=Exception.class)，如果类加了这个注解，那么这个类里面的方法抛出异常，就会回滚，数据库里面的数据也会回滚。

在@Transactional注解中如果不配置rollbackFor属性,那么事物只会在遇到RuntimeException的时候才会回滚,加上rollbackFor=Exception.class,可以让事物在遇到非运行时异常时也回滚，比如遇到SQLException异常, 数据库里面的数据也会回滚。

Page分页

传进mybatis sql语句中的实体类中的变量定义的时候可以写get()方法即get+变量+()形式。变量名对应相同就行。

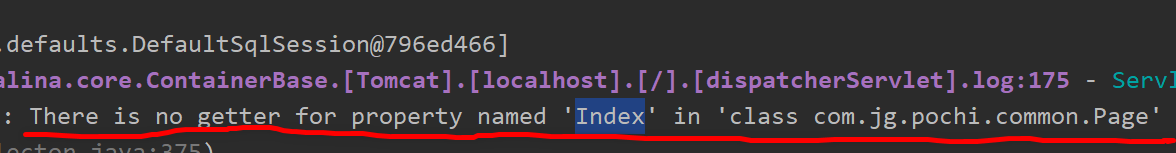
如：Page中

public Integer getindex*(){* return *(*pageNumber - 1*)* \* pageSize;  
*}*

XML中：

ORDER BY login\_time DESC  
limit #{index}, #{pageSize}

如果变量名不对应则MyBatis会抛出异常：



## java数据处理小技巧：

### Java浮点数取整

Math.floor()、Math.ceil()和Math.round()都是对浮点数取整（floor和ceil返回不带小数的double，round返回long）

floor 字面意思为地板，返回的是小于或等于该数值的最大的整数

ceil 字面意思是天花板，返回的是大于或等于该值的最小的整数

round 就是我们常见的四舍五入，不再多说

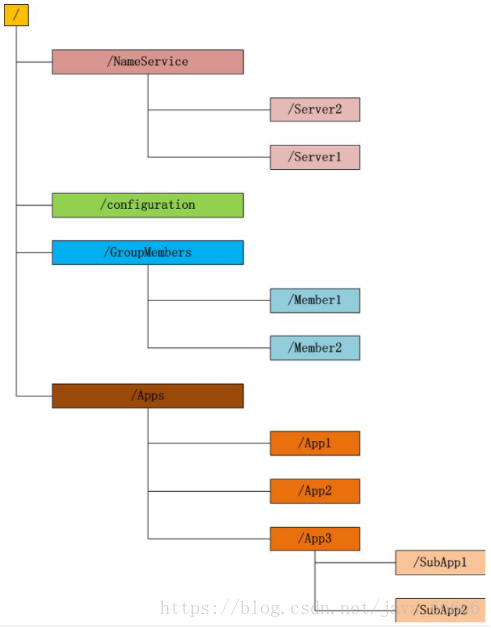
### Zookeeper

官方文档上这么解释zookeeper，它是一个分布式服务框架，是Apache Hadoop 的一个子项目，它主要是用来解决分布式应用中经常遇到的一些数据管理问题，如：**统一命名服务、状态同步服务、集群管理、分布式应用配置项的管理**等。

简单来说zookeeper=文件系统+监听通知机制。

1、 文件系统

Zookeeper维护一个类似文件系统的数据结构：



每个子目录项如 NameService 都被称作为 znode(目录节点)，和文件系统一样，我们能够自由的增加、删除znode，在一个znode下增加、删除子znode，唯一的不同在于znode是可以存储数据的。

有四种类型的znode：

PERSISTENT-持久化目录节点

客户端与zookeeper断开连接后，该节点依旧存在

PERSISTENT\_SEQUENTIAL-持久化顺序编号目录节点

客户端与zookeeper断开连接后，该节点依旧存在，只是Zookeeper给该节点名称进行顺序编号

EPHEMERAL-临时目录节点

客户端与zookeeper断开连接后，该节点被删除

EPHEMERAL\_SEQUENTIAL-临时顺序编号目录节点

客户端与zookeeper断开连接后，该节点被删除，只是Zookeeper给该节点名称进行顺序编号

2、 监听通知机制

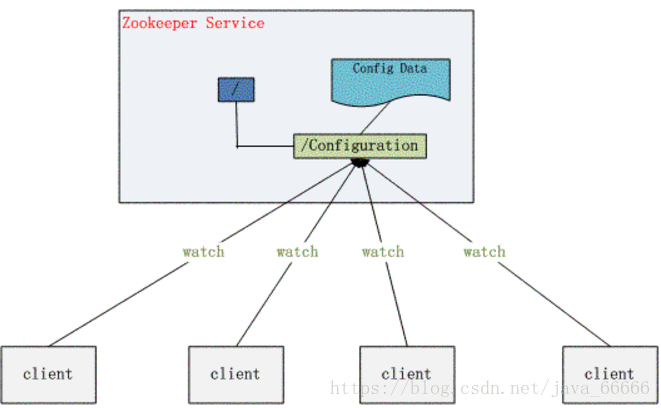
客户端注册监听它关心的目录节点，当目录节点发生变化（数据改变、被删除、子目录节点增加删除）时，zookeeper会通知客户端。

就这么简单，下面我们看看Zookeeper能做点什么呢？

Zookeeper能做什么

Zookeeper功能非常强大，可以实现诸如分布式应用配置管理、统一命名服务、状态同步服务、集群管理等功能，我们这里拿比较简单的分布式应用配置管理为例来说明。

假设我们的程序是分布式部署在多台机器上，如果我们要改变程序的配置文件，需要逐台机器去修改，非常麻烦，现在把这些配置全部放到zookeeper上去，保存在 zookeeper 的某个目录节点中，然后所有相关应用程序对这个目录节点进行监听，一旦配置信息发生变化，每个应用程序就会收到 zookeeper 的通知，然后从 zookeeper 获取新的配置信息应用到系统中。



Kafka

号称大数据的杀手锏，谈到大数据领域内的消息传输，则绕不开Kafka，这款为大数据而生的消息中间件，以其百万级TPS的吞吐量名声大噪，迅速成为大数据领域的宠儿，在数据采集、传输、存储的过程中发挥着举足轻重的作用。

Apache Kafka它最初由LinkedIn公司基于独特的设计实现为一个分布式的提交日志系统( a distributed commit log)，之后成为Apache项目的一部分。

目前已经被LinkedIn，Uber, Twitter, Netflix等大公司所采纳。

优点

* 性能卓越，单机写入TPS约在百万条/秒，最大的优点，就是吞吐量高。
* 时效性：毫秒ms级
* 可用性：非常高，kafka是分布式的，一个数据多个副本，少数机器宕机，不会丢失数据，不会导致不可用
* 消费者采用Pull方式获取消息, 消息有序, 通过控制能够保证所有消息被消费且仅被消费一次;
* 有优秀的第三方Kafka Web管理界面Kafka-Manager；
* 在日志领域比较成熟，被多家公司和多个开源项目使用；
* 功能支持：功能较为简单，主要支持简单的MQ功能，在大数据领域的实时计算以及日志采集被大规模使用

缺点：

* Kafka单机超过64个队列/分区，Load会发生明显的飙高现象，队列越多，load越高，发送消息响应时间变长
* 使用短轮询方式，实时性取决于轮询间隔时间；
* 消费失败不支持重试；
* 支持消息顺序，但是一台代理宕机后，就会产生消息乱序；
* 社区更新较慢；

为什么使用CollectionUtils.isEmpty()而不是List.isEmpty()

看源码可得：

CollectionUtils.java 会判断集合是否为null

public static boolean isEmpty(@Nullable Collection<?> collection) *{* return *(*collection == null || collection.isEmpty*())*;  
*}*

而List.isEmpty()会判断是否存在元素,如果List本身为null,则直接抛出异常。

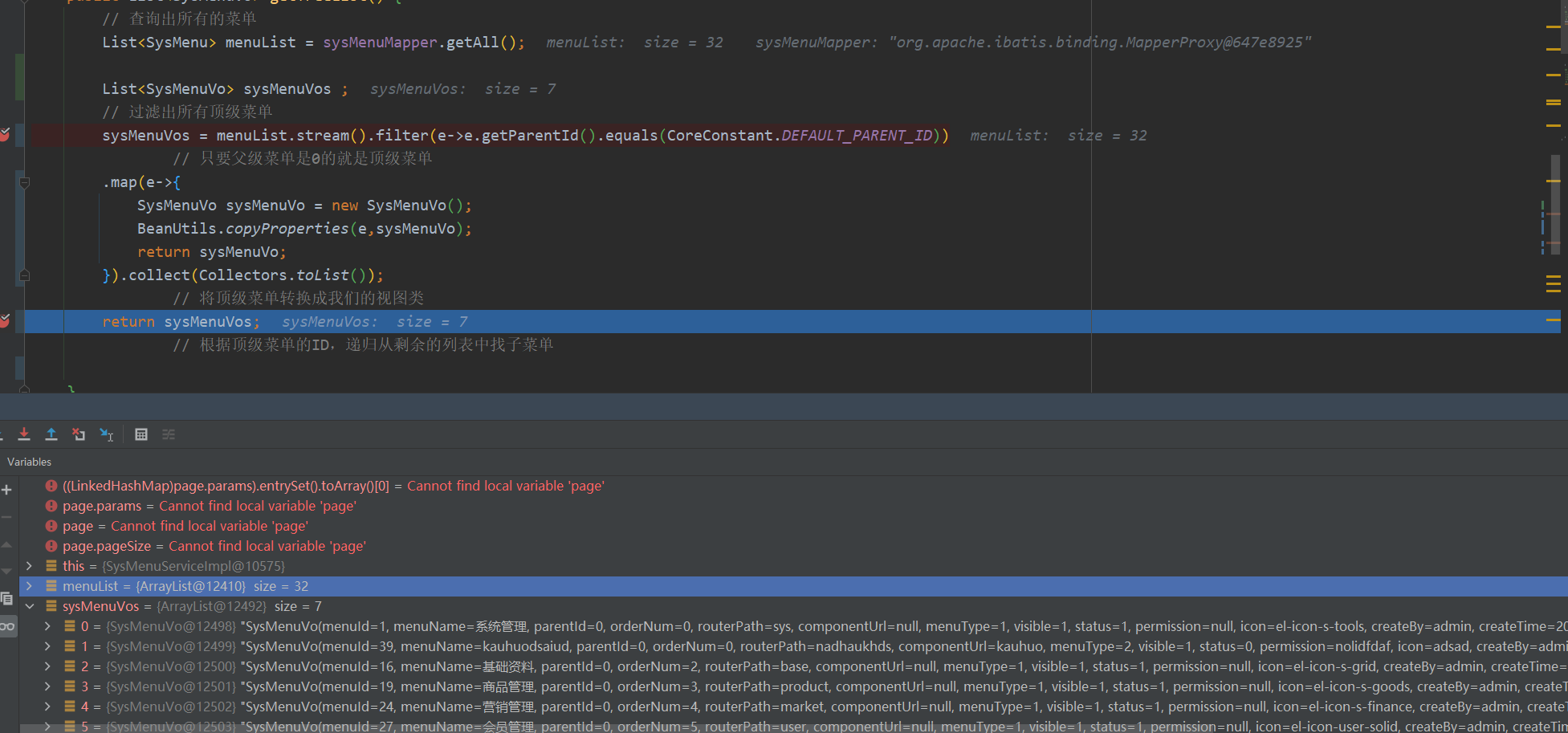
类似推荐用

String.valuesOf()而不是to.String。

## JDK1.8

List<实体类>集合通过map映射中将map中的参数e赋给新建的实体VO类，再将这个实体类return，返回结果就成了List<实体类Vo>，map方法里面记得加return

例如，在构造树形结构中可以这么使用：



collect方法三个作用 <https://blog.csdn.net/liujun03/article/details/80701999>

1、将数组组成字符串，注：Collectors.joining()有三个重载方法

2、将数组转为集合Lis

3、将list中的数据分组 并统计数量

为什么要重写equals方法？

### JAVA 8,双引号::

Java 8 中我们可以通过 `::` 关键字来访问类的构造方法，对象方法，静态方法。

## MyBatis

### Mybatis自增id

useGeneratedKeys="true"

keyColumn="role\_id" ，对应数据库字段

keyProperty="roleId",对应ID属性值，如实体列字段

## MySql

### HAVING用法

where、聚合函数、having 在from后面的执行顺序：

where>聚合函数(sum,min,max,avg,count)>having

若须引入聚合函数来对group by 结果进行过滤 则只能用having。（ 是先执行聚合函数还是先过滤 然后比对我上面列出的执行顺序 一看便知）

HAVING语句通常与GROUP BY语句联合使用，用来过滤由GROUP BY语句返回的记录集。

HAVING语句的存在弥补了WHERE关键字不能与聚合函数联合使用的不足。

语法：

SELECT column1, column2, ... column\_n, aggregate\_function (expression)

FROM tables

WHERE predicates

GROUP BY column1, column2, ... column\_n

HAVING condition1 ... condition\_n;

如果想查询平均分高于80分的学生记录可以这样写：

SELECT id, COUNT(course) as numcourse, AVG(score) as avgscore

FROM student

GROUP BY id

HAVING AVG(score)>=80;

在这里，如果用WHERE代替HAVING就会出错。

注意事项 ：

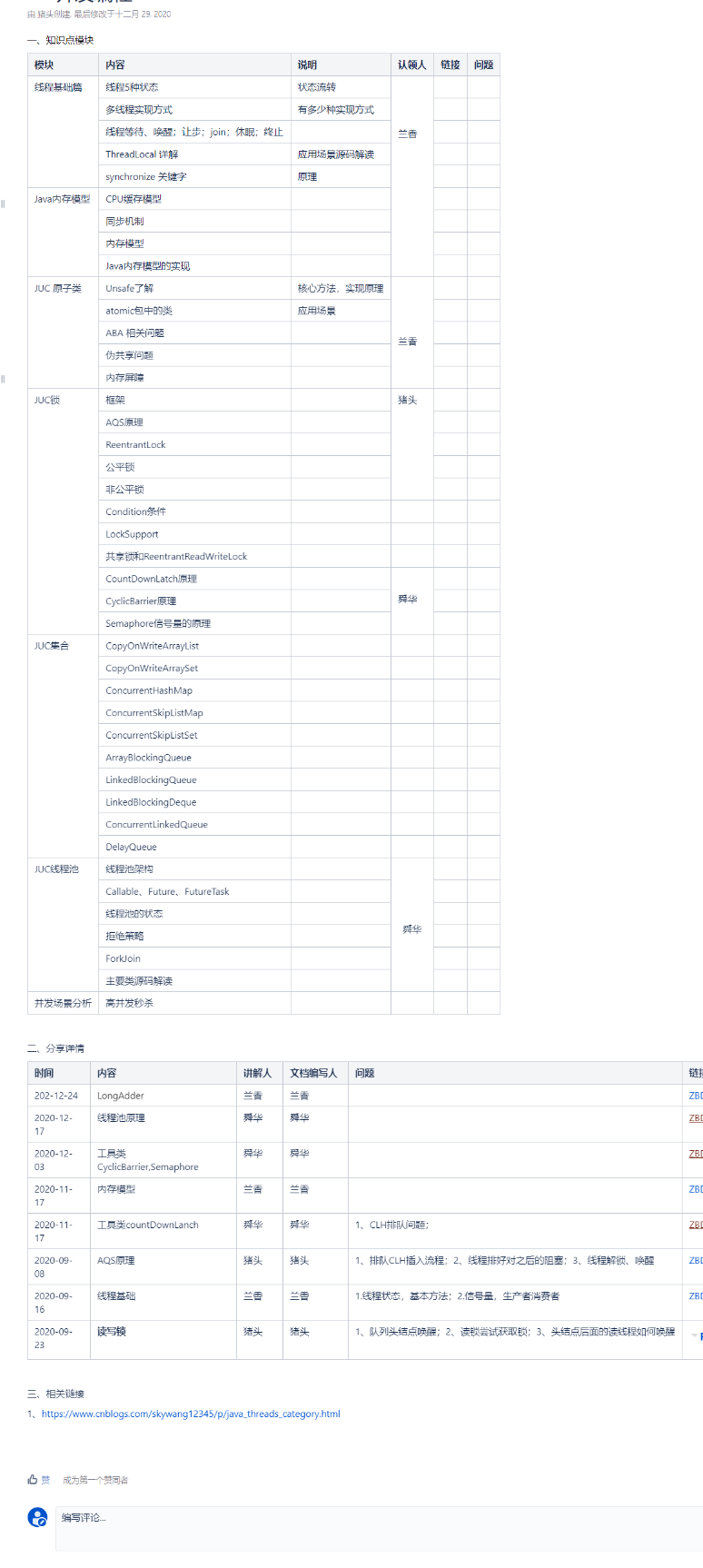
1、where 后不能跟聚合函数，因为where执行顺序大于聚合函数。

2、where 子句的作用是在对查询结果进行分组前，将不符合where条件的行去掉，即在分组之前过滤数据，条件中不能包含聚组函数，使用where条件显示特定的行。

3、having 子句的作用是筛选满足条件的组，即在分组之后过滤数据，条件中经常包含聚组函数，使用having 条件显示特定的组，也可以使用多个分组标准进行分组。

# Interview：

并发编程技术 JUC源码(放后一点)



线程池。

HashMap源码： 扩容机制如何扩容的、

红黑树太难

特点。

扩容机制。

1.7 和1.8区别。面试题延伸 之 哈希冲突及四种解决方法

mysql索引。

列计划。

