## Java基础

1. **简述&和&&的区别？**

&&是逻辑操作符，而&是位操作符。当&&和&作为逻辑操作符时，&表示当运算符两边的表达式的结果都为 true 时，整个运算结果才为 true，否则，只要有一方为 false，则结果为 false。&&还具有短路的功能，即如果第一个表达式为 false，则不再计算第二个表达式。&还可以用作位运算符，当&操作符两边的表达式不是boolean类型时，&表示按位与操作。

1. **==和equals的区别？**

它的作用是判断两个对象的地址是不是相等。即，判断两个对象是不是同一个对象(基本数据类型 == 比较的值，引用数据类型 == 比较的是内存地址。String 中的 equals 方法是被重写过的，因为 object 的 equals 方法是比较的对象的内存地址，而 String 的 equals 方法比较的是对象的值。

1. **i++ 和 ++i 的区别？**

i++：是先把i拿出来使用，然后再+1；

++i ：是先把i+1，然后再拿出来使用；

1. **switch语句能否作用在byte上，能否作用在long上，能否作用在String上？**

int基本类型或Integer包装类型，由于byte,short,char都可以隐含转换为int，所以这些类型以及这些类型的包装类型也是可以的。显然long和String类型都不符合switch语法的规定，并且不能隐式的转换成int类型，所以它们不能作用于switch语句中。JDK1.7之后支持Sring。

1. **String为什么是不可变类？**

String的成员变量是private final 的。不可变的String类是安全的，多线程的，高性能的

1. **String,StringBuffer与StringBuilder的区别？**

String的值是不可变的。 StringBuilder和StringBuffer代表可变字符串对象。不同的是：StringBuffer是线程安全的，而StringBuilder则没有实现线程安全功能，所以性能略高。

1. **简述你对面向对象思想的理解？**

面向对象是向现实世界模型的自然延伸，这是一种“万物皆对象”的编程思想。在现实生活中的任何物体都可以归为一类事物，而每一个个体都是一类事物的实例。面向对象的三大特征是继承，封装，多态。封装隐蔽了对象内部不需要暴露的细节,使得内部细节的变动跟外界脱离,只依靠接口进行通信.封装性降低了编程的复杂性，使得代码的复用性更高。通过继承,使得新建一个类变得容易,一个类从派生类那里获得其非私有的方法和公用属性的繁琐工作交给了编译器. 而 继承和实现接口和运行时的类型绑定机制 所产生的多态,使得不同的类所产生的对象能够对相同的消息作出不同的反应,极大地提高了代码的通用性.

1. **重载与重写的区别？**

重载（Overload）:发生在本类,方法名相同,参数列表不同,与返回值无关,只和方法名,参数列表,参数的类型有关.是位于一个类之中或者其子类中，具有相同的方法名，但是方法的参数不同，返回值类型可以相同也可以不同。方法名必须相同，方法的参数列表一定不一样。访问修饰符和返回值类型可以相同也可以不同。

重写（override）：一般都是表示子类和父类之间的关系，其主要的特征是：方法名相同，参数相同，但是具体的实现不同。

1. **Collection和Collections的区别及各自用途。**

collection是集合类的上层接口。本身是一个Interface，里面包含了一些集合的基本操作,是Set接口和List接口的父接口。Collections是一个集合框架的帮助类，里面包含一些对集合的排序，搜索以及序列化的操作。

1. **ArrayList和Vecotor的区别。**

Vector是多线程安全的，而ArrayList不是。两个都是采用的线性连续空间存储元素，但是当空间不足的时候，两个类的增加方式是不同的，Vector增加原来空间的一倍，ArrayList增加原来空间的50%。Vector可以设置增长因子，而ArrayList不可以。

1. **Throw和throws的区别。**

throw代表动作，表示抛出一个异常的动作；throws代表一种状态，代表方法可能有异常抛出。throw用在方法实现中，而throws用在方法声明中。throw只能用于抛出一种异常，而throws可以抛出多个异常。

1. **Final，finalize,finally有什么区别？**

final 表示不可修改的，可以用来修饰类，方法，变量。final修饰class表示该class不可以被继承。final修饰方法表示方法不可以被overrride（重写）。final修饰变量表示变量是不可以修改。finally是Java的异常处理机制中的一部分。finally块的作用就是为了保证无论出现什么情况，finally块里的代码一定会被执行。finalize是Object类的一个方法，是GC进行垃圾回收前要调用的一个方法。

1. **Java中有几种类型的流？**

字节流 InputStream/OutputStream。

FileInputStream/FileOutputStream：文件字节流，用于文件的读写操作。BufferedInputStream/BufferedOutputStream：加缓冲区的字节流，用于提高效率。

字符流 Reader/Writer

FileReader/FileWriter：文件字符流，用于文本文件的读写操作BufferedReader/BufferedWriter：加缓冲区的字符流，用于提高效率。

转换流 InputStreamReader/OutputStreamWriter

1. **字节流与字符流的区别？各自应用场景。**

字符流处理的单元为 2 个字节的 Unicode 字符，分别操作字符、字符数组或字符串，而字节流处理单元为 1 个字节，操作字节和字节数组。Java 内用 Unicode 编码存储字符，字符流处理类负责将外部的其他编码的字符流和 java 内 Unicode 字符流之间的转换。而类 InputStreamReader 和 OutputStreamWriter 处理字符流和字节流的转换。字符流（一次可以处理一个缓冲区）一次操作比字节流（一次一个字节）效率高。

1. **Java序列化是什么？什么情况下使用序列化？如何实现序列化？**

Java序列化是指将 Java 对象转换成字节流的过程。当 Java 对象需要在网络上传输 或者 持久化存储到文件中时，就需要对 Java 对象进行序列化处理。序列化的实现：类实现 Serializable 接口，这个接口没有需要实现的方法。实现 Serializable 接口是为了告诉 jvm 这个类的对象可以被序列化。

1. **事务的四大特性。**

事务是指是程序中一系列严密的逻辑操作，而且所有操作必须全部成功完成，否则在每个操作中所作的所有更改都会被撤消。原子性（Atomicity）一致性（Consistency）隔离性（Isolation）持久性（Durability）

1. **线程和进程的区别。**

进程是操作系统资源分配的基本单位，而线程是任务调度和执行的基本单位。进程有自己的独立地址空间，而线程是共享进程中的数据的，使用相同的地址空间。线程之间的通信更方便，进程之间需要以IPC通信方式进行。

1. **Wait和sleep有什么区别？**

wait()方法则是指当前线程让自己暂时退让出同步资源锁，以便其他正在等待该资源的线程得到该资源进而运行，只有调用了notify()方法，之前调用wait()的线程才会解除wait状态，可以去参与竞争同步资源锁，进而得到执行。sleep()方法可以在任何地方使用；wait()方法则只能在同步方法或同步块中使用；sleep()是线程类（Thread）的方法，调用会暂停此线程指定的时间，但监控依然保持，不会释放对象锁，到时间自动恢复；wait()是Object的方法，调用会放弃对象锁，进入等待队列，待调用notify()/notifyAll()唤醒指定的线程或者所有线程，才会进入锁池，不再次获得对象锁才会进入运行状态；

1. **阐述java垃圾回收机制及工作流程。**

JVM实现了java垃圾的自动回收机制。自动垃圾回收是一种在堆内存中找出哪些对象在被使用，还有哪些对象没被使用，并且将后者删掉的机制。在JVM中，new出来的对象会被放在堆区。将对象实例运行数据放在栈区。堆区分为“年轻代”，“年老代”，“持久代”3个区域。“年轻代”又分为伊甸区（刚出生的地方）和2个缓冲区。刚new出来的对象一般先放到伊甸区。持久代中存放的是java类import的类信息。垃圾回收主要涉及年轻代和年老代。年轻代中的回收属于轻量级回收，年老代和持久代存满会触发重量级（FULL GC）。通过判断对象上有没有强引用来确定是否回收，JDK早期使用‘引用计数法’，后期使用‘根搜索算法’。程序员也可以手动调用System.gc() 来启动FULL GC。

1. **单例模式的实现。**

public class Singleton {

private final static Singleton INSTANCE = null;

private Singleton() {}

public static Singleton getInstance() {

return new Singleton();

}

}

1. **map底层实现原理。**

底层使用数组实现，数组中每一项是个单向链表，即数组和链表的结合表。当链表长度大于一定阈值（默认8）时，链表转换为红黑树，这样减少链表查询时间。HashMap在底层将key-value当成一个整体进行处理，这个整体就是一个Node对象。HashMap底层采用一个Node[]数组来保存所有的key-value对，当需要存储一个Node对象时，会根据key的hash算法来决定其在数组中的存储位置，在根据equals方法决定其在该数组位置上的链表中的存储位置**；**

1. **事务级别。**

**READ UNCOMMITTED（读未提交数据）：** 允许事务读取未被其他事务提交的变更数据，会出现脏读、不可重复读和幻读问题。

**READ COMMITTED（读已提交数据）：** 只允许事务读取已经被其他事务提交的变更数据，可避免脏读，仍会出现不可重复读和幻读问题。（oracle数据库默认）

**REPEATABLE READ（可重复读）：** 确保事务可以多次从一个字段中读取相同的值，在此事务持续期间，禁止其他事务对此字段的更新，可以避免脏读和不可重复读，仍会出现幻读问题。（Mysql InnoDB引擎默认）

**SERIALIZABLE（序列化）：** 确保事务可以从一个表中读取相同的行，在这个事务持续期间，禁止其他事务对该表执行插入、更新和删除操作，可避免所有并发问题，但性能非常低。

1. **垂直分表和水平分表的区别？**

垂直分表：适用于一个表有很多个字段的情况，将原先一个表中的某些字段拆出来，单独放到一个或者多个表。

水平分表：根据具体的切分规则，将数据划分到不同的表上面。

1. **arraylist和linkedlist的底层实现**

ArrayList 默认初始容量为10,基于1.5倍动态扩大容量。

ArrrayList底层的数据结构是数组，支持随机访问，而 LinkedList 的底层数据结构是双向循环链表。

1. **HashMap与Hashtable的区别？**

Hashtable是java一开始发布时就提供的键值映射的数据结构，而HashMap产生于JDK1.2，两个作者不同。

继承的父类不同。HashMap是继承自AbstractMap类，而HashTable是继承自Dictionary类。不过它们都实现了同时实现了map、Cloneable（可复制）、Serializable（可序列化）这三个接口。

Hashtable既不支持Null key也不支持Null value。Hashtable的put()方法的注释中有说明。HashMap中，null可以作为键，这样的键只有一个；可以有一个或多个键所对应的值为null。当get()方法返回null值时，可能是 HashMap中没有该键，也可能使该键所对应的值为null。

Hashtable是线程安全的，它的每个方法中都加入了Synchronize方法。HashMap不是线程安全的，在多线程并发的环境下，可能会产生死锁等问题。虽然HashMap不是线程安全的，但是它的效率会比Hashtable要好很多。

初始容量大小和每次扩充容量大小的不同。Hashtable默认的初始大小为11，之后每次扩充，容量变为原来的2n+1。HashMap默认的初始化大小为16。之后每次扩充，容量变为原来的2倍。

计算hash值的方法不同。Hashtable比HashMap多提供了elments() 和contains() 两个方法。

1. **ConcurrentHashMap的并发度是什么?**

ConcurrentHashMap的并发度就是segment的大小，默认为16，这意味着最多同时可以有16条线程操作ConcurrentHashMap，这也是ConcurrentHashMap对Hashtable的最大优势。

1. **String,StringBuffer,StringBuilder**

String:（不可变字符对象）因为String的值是不可变的（final修饰的），这就导致每次对String的操作都会生成新的String对象，这样不仅效率低下，而且大量浪费有限的内存空间。

StringBuffer：（可变字符对象、效率低、线程安全）每个StringBuffer对象都有一定的缓冲区容量，当字符串大小没有超过容量时，不会分配新的容量，当字符串大小超过容量时，会自动增加容量。

StringBuilder:（可变字符对象、效率高、线程不安全）

1. **java反射机制原理**

反射就是把java类中的各种成分映射成一个个的Java对象 例如：一个类有：成员变量、方法、构造方法、包等等信息，利用反射技术可以对一个类进行解剖，把各个组成部分映射成一个个对象。运行状态中，对于任意一个类，都能够知道这个类的所有属性和方法；对于任意一个对象，都能够调用它的任意方法和属性；

1. **autowired 与resource区别**

autowired 默认根据类型装配 bytype 与Qualifier联合只用可按名称装配

resource 默认根据名称匹配 byname 指定type时按类型装配

@Autowired能够用在：构造器、方法、参数、成员变量和注解上，而@Resource能用在：类、成员变量和方法上。

@Autowired是spring定义的注解，而@Resource是JSR-250（JDK底层）定义的注解。

1. **java8新特性**

Lambda表达式 Stream API Date Time API Option封装类

1. **线程run和start区别**

start方法来启动线程，真正实现了多线程运行，这时无需等待run方法体代码执行完毕而直接继续执行下面的代码。通过调用Thread类的start()方法来启动一个线程，这时此线程处于就绪（可运行）状态，并没有运行，一旦得到cpu时间片，就开始执行run()方法，这里方法run()称为线程体，它包含了要执行的这个线程的内容，run方法运行结束，此线程随即终止。

run()方法只是类的一个普通方法而已，如果直接调用Run方法，程序中依然只有主线程这一个线程，其程序执行路径还是只有一条，还是要顺序执行，还是要等待run方法体执行完毕后才可继续执行下面的代码，这样就没有达到写线程的目的。

1. **http状态码**

200 - 请求成功

400 -客户端请求的语法错误，服务器无法理解

401 -Unauthorized 请求要求用户的身份认证

403 -服务器理解请求客户端的请求，但是拒绝执行此请求

404 - 请求的资源（网页等）不存在

500 - 内部服务器错

## 数据库相关

1. **为什么使用数据索引能提高效率?**

数据索引的存储是有序的。

在有序的情况下，通过索引查询一个数据是无需遍历索引记录的。

极端情况下，数据索引的查询效率为二分法查询效率，趋近于 log2。

1. **MySQL 索引使用的注意事项？**

更新频繁的列不要加索引。

数据量小的表不要重复

数据多的字段不要加索引,比如性别字段。

首先应该考虑对where 和 order by 涉及的列上建立索引。

1. **索引的优缺点**

优点：

 大大加快数据的检索速度；

 创建唯一性索引，保证数据库表中每一行数据的唯一性；

 加速表和表之间的连接；

 在使用分组和排序子句进行数据检索时，可以显著减少查询中分组和排序的时间。

缺点：

索引需要占物理空间；

当对表中的数据进行增加、删除和修改的时候，索引也要动态的维护，降低数据的维护速度。

1. **数据库索引失效**

1) 没有查询条件，或者查询条件没有建立索引

2) 在查询条件上没有使用引导列（没用到对应的列）

3) 查询的数量是大表的大部分，应该是30％以上。

4) 索引本身失效 （名称不匹配或没有启用）

5)隐式转换导致索引失效.这一点应当引起重视.也是开发中经常会犯的错误. 由于表的字段tu\_mdn定义为varchar2(20),

6)使用内部函数导致索引失效

1. **数据库分页语句**

MYSQL select \*from table limit start,pageNum

oracle select \*from (select rownum from table where rownum <=endIndex ) where rownum> startIndex

## Hibernate相关

1. **hibernate运行原理。**

hibernate里面提供了3个核心接口

Configuration、SessionFactory、Session

1、hibernate启动的时候利用Configuration读取xml配置文件

2、通过配置文件创建SessionFactory对象，初始化hibernate基本信息

3、获取session然后调用CRUD方法进行数据操作，hibernate会把我们的数据进行三种状态的划分，然后根据状态进行管理我们的数据，对应的发送SQL进行数据操作

4、关闭session，如果有事务的情况下，需要手动获取事务并开启，然后事务结束后提交事务。

5、在提交事务的时候，去验证我们的快照里面的数据和缓存数据是否一致，如果不一致，发送SQL进行修改。

## Mybatis相关

1. **什么是Mybatis及其优点？**

Mybatis是一个半ORM（对象关系映射）框架，它内部封装了JDBC，开发时只需要关注SQL语句本身，不需要花费精力去处理加载驱动、创建连接、创建statement等繁杂的过程。程序员直接编写原生态sql，可以严格控制sql执行性能，灵活度高。

基于SQL语句编程，相当灵活，不会对应用程序或者数据库的现有设计造成任何影响，SQL写在XML里，解除sql与程序代码的耦合，便于统一管理；提供XML标签，支持编写动态SQL语句，并可重用，能够与Spring很好的集成；提供映射标签，支持对象与数据库的ORM字段关系映射；提供对象关系映射标签，支持对象关系组件维护。与JDBC相比，减少了50%以上的代码量，消除了JDBC大量冗余的代码，不需要手动开关连接；能很好的与各种数据库兼容。

1. **MyBatis与Hibernate有哪些不同？**

Mybatis直接编写原生态sql，可以严格控制sql执行性能，灵活度高，非常适合对关系数据模型要求不高的软件开发，因为这类软件需求变化频繁，一但需求变化要求迅速输出成果。但是灵活的前提是mybatis无法做到数据库无关性，如果需要实现支持多种数据库的软件，则需要自定义多套sql映射文件，工作量大。Hibernate对象/关系映射能力强，数据库无关性好，对于关系模型要求高的软件，如果用hibernate开发可以节省很多代码，提高效率。

1. **#{}和${}的区别是什么？**

#{}是预编译处理，${}是字符串替换。Mybatis在处理#{}时，会将sql中的#{}替换为?号，调用PreparedStatement的set方法来赋值；Mybatis在处理${}时，就是把${}替换成变量的值。使用#{}可以有效的防止SQL注入，提高系统安全性。

## Spring框架系列

1. **Spring是什么?**

Spring是一个轻量级的IoC和AOP容器框架。是为Java应用程序提供基础性服务的一套框架， 目的是用于简化企业应用程序的开发，它使得开发者只需要关心业务需求。常见的配置方式有三 种：基于XML的配置、基于注解的配置、基于Java的配置。

IOC就是控制反转，是指创建对象的控制权的转移，以前创建对象的主动权和时机是由自己把控 的，而现在这种权力转移到Spring容器中，并由容器根据配置文件去创建实例和管理各个实例 之间的依赖关系，对象与对象之间松散耦合，也利于功能的复用。DI依赖注入，和控制反转是同 一个概念的不同角度的描述，应用程序在运行时依赖IoC容器来动态注入对象需要的外部资源。

1. **Spring注入bean的几种方式？**

Set方法注入。构造器注入：①通过index设置参数的位置，②通过type设置参数类型。

静态工厂注入。

1. **SpringMVC的流程？**

用户发送请求至前端控制器DispatcherServlet；

DispatcherServlet收到请求后，调用HandlerMapping处理器映射器，请求获取Handle；

处理器映射器根据请求url找到具体的处理器，生成处理器对象及处理器拦截器(如果有则生成)一并返回给DispatcherServlet；

DispatcherServlet 调用 HandlerAdapter处理器适配器；

HandlerAdapter 经过适配调用 具体处理器(Handler，也叫后端控制器)；

Handler执行完成返回ModelAndView；

HandlerAdapter将Handler执行结果ModelAndView返回给DispatcherServlet；

DispatcherServlet将ModelAndView传给ViewResolver视图解析器进行解析；

ViewResolver解析后返回具体View；

DispatcherServlet对View进行渲染视图（即将模型数据填充至视图中）

DispatcherServlet响应用户。

1. **spring循环依赖是怎么解决的？**

第一级缓存（也叫单例池）：Map<String, Object> singletonObjects，存放已经经历了完整生命周期的 Bean 对象。

第二级缓存：Map<String, Object> earlySingletonObjects，存放早期暴露出来的 Bean 对象，Bean 的生命周期未结束（属性还未填充完）。

第三级缓存：Map<String, ObjectFactory<?>> singletonFactories，存放可以生成 Bean 的工厂。

Spring 管理的 Bean 其实默认都是单例的，也就是说 Spring 将最终可以使用的 Bean 统一放入第一级缓存中，也就是 singletonObjects（单例池）里，以后凡是用到某个 Bean 了都从这里获取就行了

1. **spring 自动配置是如何实现的**

Spring Boot启动的时候会通过@EnableAutoConfiguration注解找到META-INF/spring.factories配置文件中的所有自动配置类，并对其进行加载，而这些自动配置类都是以AutoConfiguration结尾来命名的，它实际上就是一个JavaConfig形式的Spring容器配置类，它能通过以Properties结尾命名的类中取得在全局配置文件中配置的属性如：server.port，而XxxxProperties类是通过@ConfigurationProperties注解与全局配置文件中对应的属性进行绑定的。

1. **spring bean 作用域**

* singleton 在 Spring 容器中仅存在一个 Bean 实例， Bean 以单例的形式存在。
* prototype 每次从容器中调用 Bean 时，都会返回一个新的实例，即相当于执行 new XxxBean() 的实例化操作。
* request 每次 http 请求都会创建一个新的 Bean ， 仅用于 WebApplicationContext 环境。
* session 同一个 http Session 共享一个 Bean ，不同的 http Session 使用不同的 Bean，仅用于 WebApplicationContext 环境。
* globalSession 同一个全局 Session 共享一个 bean, 用于 Porlet, 仅用于 WebApplication 环境。

1. **Spring Boot的优点是什么?**

减少开发、测试的时间和工作量。

使用JavaConfig有助于避免使用XML。

避免大量maven导入和各种版本冲突。

提供可选的开发方法。

通过提供默认开发方式进行快速开发。

不需要单独的Web服务器。

由于没有web.xml文件，所以需要更少的配置。

1. **什么是springcloud及其优点？**

Spring cloud流应用程序启动器是基于Spring Boot的Spring集成应用程序，提供与外部系统的集成。

每个服务直接足够内聚，代码容易理解

开发效率高，一个服务只做一件事，适合小团队开发

松耦合，有功能意义的服务。

可以用不同语言开发，面向接口编程。

易于第三方集成

微服务只是业务逻辑的代码，不会和HTML,CSS或其他界面结合.

可以灵活搭配，连接公共库/连接独立库

## Redis 相关

1. **Redis有那些数据类型？**

Redis主要有5种数据类型，包括String，List，Set，Zset，Hash。

1. **redis持久化的两种方式？**

Redis 提供两种持久化机制 RDB（默认） 和 AOF 机制。RDB：是Redis DataBase缩写。

RDB是Redis默认的持久化方式。按照一定的时间将内存的数据以快照的形式保存到硬盘中，对应产生的数据文件为dump.rdb。通过配置文件中的save参数来定义快照的周期。

AOF持久化(即Append Only File持久化)，则是将Redis执行的每次写命令记录到单独的日志文件中，当重启Redis会重新将持久化的日志中文件恢复数据。

1. **Linux相关命令**

df 命令

显示文件系统的磁盘使用情况，默认情况下 df -k 将以字节为单位输出磁盘的使用量。

使用 df -h 选项可以以更符合阅读习惯的方式显示磁盘使用量。使用 df -T 选项显示文件系统类型。

pwd 命令 输出当前工作目录

mount 命令 挂载一个文件目录

grep 命令 内容输出

sed 命令 Sed主要用来自动编辑一个或多个文件；简化对文件的反复操作

ps 命令 ps 命令用于显示正在运行中的进程的信息。

top命令 top 命令包含了前面好几个命令的检查的内容。比如系统负载情况（uptime）、系统内存使用情况（free）、系统 CPU 使用情况（vmstat）等