# JavaSE

## ==和equals区别？

equals默认比较的是内存地址，重写了的话比较的是重写的内容

基本数据类型和基本数据类型==比较的值，引用数据类型比较的是内存地址

## switch语句使用细节

支持的数据类型:byte,short,int,char**(没有long),**四种基本基本数据类型的包装类,以及开始java5支持的枚举,和java7开始支持的String类型.

**本质:switch仅仅只能支持int类型**(byte,short,char会自动提升为int类型)

问题:为什么switch支持枚举类型?

枚举本质是int类型.

**2.栈,堆,方法区,堆中的常量池分别存储的是什么?  
栈**:存的是对象的引用,局部变量,方法的出口等信息,每一个方法都会创建一个栈帧, 栈帧中存放着当前方法的数据信息(局部变量),当方法调用完毕,该方法的栈帧就被销毁了.

**堆:**存放的是对象,每new一个对象就在堆中开辟一块内存空间.

**方法区:** 线程共享的内存区域，存储已被虚拟机加载的类信息、常量、静态变量即时编译器编译后的代码数据等(这个区域的内存回收目标主要是针对常量池的回收和对类型的卸载)。

**堆中的常量池:** 常量池是堆内存中的一部分，专门用来存储字符串常量和基本数据类型常量；所以String a="abc"中，引用a是存储在栈内存的，指向常量池中的"abc";

**理解:**在编译时期就将一部分常量放进常量池了,被JVM加载后存放到方法区

Java6和6之前，[常量池](https://so.csdn.net/so/search?q=%E5%B8%B8%E9%87%8F%E6%B1%A0&spm=1001.2101.3001.7020" \t "_blank)是存放在方法区（永久代）中的。

Java7，将常量池是存放到了堆中。

Java8之后，取消了整个永久代区域，取而代之的是元空间。运行时常量池和静态常量池存放在元空间中，而字符串常量池依然存放在堆中。

**this关键字:**

**static不能和this一起使用;**

当字节码被加载进JVM,static成员以及存在了.

但是此时对象还没有创建,没有对象,就没有this.

## 面向对象

**封装思想:**

概念:把对象的状态和行为看成一个统一的整体,将二者存放在一个统一的模块中(类)

**封装的好处:**

1):使调用者正确,方便的使用系统功能,防止调用者随意更改属性,

2):提高组件复用性

3):达到组件之间的低耦合性

封装其实就是要让有些类看不到另外一些类里面做了什么事情.所以java提供了访问权限修饰符来规定在一个类里面能看到什么,能暴露什么.

**继承:**

**子类继承父类之后,可以拥有父类的某一些状态和行为(子类复用了父类的功能或状态).**

继承的好处:

1. 提高了代码的复用性
2. 解决了代码重复
3. 表现出一个体系

继承的缺点:

1.增加代码的耦合度,父类要改的话,考虑子类也要改

**多态:**

**同一种事物表现出的多种形态**

**多态的前提:**继承类或者实现接口

**多态的好处:**

1.提高了代码的维护性（继承保证）；

2.提高了代码的**扩展性**。

## final关键字

**列举5个Java中内置的使用final修饰的类.**

java里final修饰的类有很多，比如八大基本数据类型包装类和String等。

**final修饰类，表示该类不可以被继承**

**final修饰方法，表示该方法不可以被覆盖**

final修饰基本类型变量：表示该变量的值不能改变，即不能用“=”号重新赋值。

final修饰引用类型变量：表示该变量的引用的地址不能变，而不是引用地址里的内容不能变。

## 二 接口和抽象类的区别

1 （不能为普通方法提供方法体）接口里只能包含抽象方法，静态方法和默认方法（加default），不能为普通方法提供方法实现，抽象类则完全可以包含普通方法，接口中的普通方法默认为抽象方法。

2 .(public static final 赋值)抽象类中的成员变量可以是各种类型的，而接口中的成员变量只能是 public static final 类型的，并且必须赋值，否则通不过编译。

3 （是否有构造器）接口不能包含构造器，抽象类可以包含构造器，抽象类里的构造器并不是用于创建对象，而是让其子类调用这些构造器来完成属于抽象类的初始化操作。

1. .（不能包含初始化块）接口里不能包含初始化块，但抽象类里完全可以包含初始化块。

5.（继承一个抽象类、多个接口）一个类只能继承一个抽象类，而一个类却可以实现多个接口。

## equals和hashCode关系?

**也就是说equals相等hashcode一定相等,hashcode相等equals不一定相等**

## finally:关键字

finally的两种语法:

1):try...finally: 此时没有catch来捕获异常,因为此时根据应用场景,我们会抛出异常,自己不处理.

2):try...catch....finally:自身需要处理异常,最终还得关闭资源.

注意:finally不能单独使用.

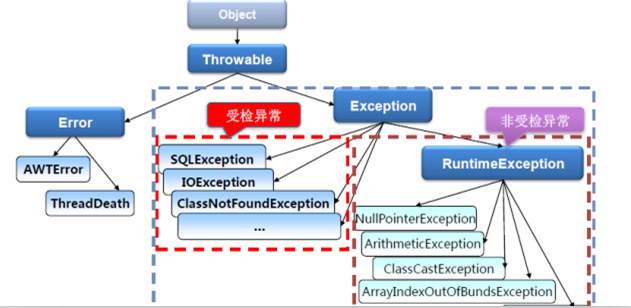
---------------------------------------------------------------------------------------

当只有在try或者catch中调用退出JVM的相关方法,此时finally才不会执行,否则finally永远会执行.

System.exit(0);//退出JVM

**如果finally有return语句,永远返回finally中的结果,避免该情况.**

## 异常:



**throw**: 运用于方法内部,用于给调用者返回一个异常对象,和return一样会结束当前方法.

**throws**: 运用于方法声明之上,用于表示当前方法不处理异常,而是提醒该方法的调用者来处理异常(**抛出异常**).

面试题:

**1):Error和Exception的区别和关系.**

**2):checked异常和runtime异常的区别.**

**3):如何保证一段代码必须执行到(finally).**

**4):finally中的代码一定会执行吗?**

不一定,代码未进入try代码块,或者system.exit退出JVM都不会执行

**5):finally和return的执行顺序?(关注finally里面的两道题)**

在try中出现return，如果有finally关键字，它需要先将返回信息暂存一下，等到finally代码执行完后再进行返回。

**6):throw和throws的区别?**

throw是在方法体中抛出一个异常,一般都抛RuntimeException,throws是在方法上抛出一个异常

**7):列举5个常见的异常类.**

**8):列举5个常见的Runtime异常类.**

1.空指针异常

2.数字格式化异常

3.类型强转异常

4.索引越界异常

5.算数异常

## 线程:

**创建线程的四种方式:**

1）继承Thread类创建线程

2）实现Runnable接口创建线程

3）使用Callable和Future创建线程

4）使用线程池例如用Executor框架

ArrayList和Vector的区别?

1.ArrayList是线程不安全的,所以执行效率高,Vector是线程安全的,执行效率低.

2.ArrayList扩容是在原来的基础上扩容0.5倍,Vector是在原来的基础上扩容1倍

**wait和sleep的区别?**

wait是object类中的,sleep是Thread类中的

最主要的区别就是,sleep不会释放锁,wait会释放锁,使得其他线程可以使用同步锁,sleep可以模拟网络延迟

**线程和进程的区别?**

不同的进程使用不同的内存空间，而所有的线程共享一片相同的内存空间。

**线程的生命周期:**

1. 新建状态:new Thread时进入新建状态
2. 可运行状态:分为就绪状态和运行状态

就绪状态:线程对象调用start方法之后,等待JVM的调度(此时该线程并没有运行).

运行状态:线程对象获得JVM调度,如果存在多个CPU,那么允许多个线程并行运行

1. 堵塞状态:当A线程试图获取同步锁时却被B线程获取,此时JVM会把A线程放进对象的锁池中,此时A线程处于堵塞状态
2. 等待状态:线程在执行过程中调用了wait方法,此时JVM会把当前线程放到等待池中
3. 计时等待状态:线程调用了wait或者sleep方法
4. 终止状态:又称为死亡状态,表示线程终止,分为正常调用完run方法而退出(正常死亡),异常死亡 结束线程调用stop方法

**堵塞状态和等待状态的区别?**

阻塞：当一个线程试图获取一个内部的对象锁（非java.util.concurrent库中的锁），而该锁被其他线程持有，则该线程进入阻塞状态。  
等待:当线程处于运行过程时,调用了wait()方法,此时JVM把当前线程存在对象等待池中

**谈谈对线程池的理解?**

线程池中存放着多个线程,每一个任务从线程池中取出线程,用完之后再把线程放进线程池中,如果线程池中的线程数不够,则会在线程队列中等待,如果线程队列中满了,就会执行拒绝策略

**主线程怎么给子线程传输数据?**

可以定义一个类实现Runnable接口,然后加一个属性,也就是要共享的数据,然后加一个有参构造器,然后run方法就可以用这个属性了

主线程new对象开启线程的时候,就把这个属性通过构造器传入进去

**什么情况下会造成死锁？**

**当线程通信的时候很容易造成死锁，死索无法解决，只能避免**

**当B线程等待A持有的锁，当A线程等待B线程持有的锁时，发生死锁现象，JVM不检测也不避免这种情况**

**比如：A线程和B线程，都操作o1，o2对象，A线程操作顺序o1，o2，B线程操作顺序o2,o1,A操作o1对象的时候上锁了，要操作o2的时候，此时B线程把o2上锁了，然后此时就会导致死锁。**

## 集合框架:

List:记录元素先后添加的顺序,允许重复

Set:不记录元素先后添加的顺序,不允许重复

treeSet:专门用来做排序的类,有自然排序和定制排序

自然排序实体类实现comparable接口,定制排序是在treeSet对象中传入Comparator接口的实现类,并重写方法compare,编写排序规则

Map:从严格上来说Map不属于集合,因为他没有继承Collection,因为能存储数据,所以我们把他称为集合,他是由多个entry组成,每个entry的key就相当于是set集合,value就是相当于是list集合,可以这么理解,这也就说明了为什么map和set的很多实现类都类似

hash表算法,在java8之前底层是数组加链表,java8之后底层是数组、链表和红黑树.

**内存泄漏和内存溢出区别?**

内存溢出就是你要求分配的内存超出了系统能给你的，系统不能满足需求，于是产生溢出。

内存泄漏是指你向系统申请分配内存进行使用(new)，可是使用完了以后却不归还(delete)，结果你申请到的那块内存你自己也不能再访问（也许你把它的地址给弄丢了），而系统也不能再次将它分配给需要的程序

**序列化**

在分布式系统中,需要共享的数据的JavaBean对象,都得做序列化,此时需要把对象再网络上传输,此时就得把对象数据转换为二进制形式.

## 反射的使用场合和作用

**反射是在运行状态中，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一个方法和属性;这种动态获取的信息**

**以及动态调用对象的方法的功能称为 Java 语言的反射机制。**

使用场合：在编译时根本无法知道该对象或类可能属于哪些类，程序只依靠运行时信息来发现该对象和类的真实信息。

主要作用：通过反射可以使程序代码访问装载到JVM 中的类的内部信息

a) 获取已装载类的属性信息

b) 获取已装载类的方法

c) 获取已装载类的构造方法信息

**什么时候用断言?**

简单的说，如果希望在不满足某些条件时阻止代码的执行，就可以考虑用断言来阻止它。

String类常用的操作方法:  
equals()方法:比较是否相等

length()方法:字符串的长度

trim()方法:去除字符串的前后空格

tocharArray()方法:字符串转数组

split()方法:把字符串按照规则分割,返回String类型数组

# JavaWeb

## Servlet

servlet接口的实现类genericServlet抽象类和HttpServlet抽象类,如果实现servlet接口必须复写所有的方法,有很多方法用不到所以会有代码冗余,对此出现了genericServlet类,但是这个类处理一般的请求响应的,因为我们是浏览器和服务器交互所以我们一般继承与httpServlet,

在servlet中就干了三件事:

1.servlet是接收请求参数,封装成对象,

2.调用业务方法,来处理请求

3.控制页面跳转

**生命周期** : 当输入url第一次访问servlet,如果设置了load-on-startup的话,servlet会直接初始化,调用init方法,init方法调用servletConfig来加载web.xml中的参数,然后创建对象,然后调用service方法,service方法调用doGet或者doPost方法执行操作,销毁的时候会先执行destory方法,destory方法执行完之后,servlet就被清空

正常关闭才会执行destory方法,强制关闭服务器不会执行destory方法

## http协议

**什么是http协议？**

http协议是基于tcp/ip的应用层协议,简单来说就是基于应用层的通信规范,双方要通信必须遵循这个规范,就是http协议 **Http协议工作原理?**用户输入url,浏览器会先解析ip和端口号,然后与ip三次握手建立连接,服务器接受请求做处理进行响应,响应期间会四次挥手断开连接

**长连接,短连接,无连接**

**长连接:**一直建立连接,效率高,但是占用连接资源

**短连接:**是在无连接和长连接之间的,在一次连接内多次请求和多次响应,交互完数据之后断开连接,效率较高

**无连接:**建立连接后,请求一次相应一次就断开连接,效率很低

**HTTP协议的特点是什么?**

http协议是无状态的, 服务器端在处理相应请求后不会保留任何客户端的信息，每次请求都是独立的.

每一次交互都是请求和响应.

http协议应包括一个请求行,若干请求头,一个空行,请求实体

## Cooke

**Cookie的domain:**

Cookie的path是在同一主机中指定共享Cookie,如果主机不同,那么就一定不能共享Cookie,无论path是什么.

如果希望不同的二级域名中可以共享Cookie,那么就要设置Cookie的domain和path了

**Cookie的缺点:**

Cookie存储在浏览器不安全

Cookie存储中文比较麻烦,需要先编码后解码

Cookie的value是String类型,一个Cookie只能存储一个数据

销毁cookie:cookie对象.setMaxAge(0)

setMaxAge>0表示持久化cookie

setMaxAge<0表示会话cookie

## Session

session存储在服务器

session本质上就是一个会话Cookie

session对象.removeAttribute(“currentName”)删除指定属性名的值

session对象.invalidate()销毁session对象(session中所有的属性都不存在)

## jsp

jsp三大指令

page、include、taglib

page:配置当前jsp页面的属性

include:分为静态包含和动态包含

静态包含:在翻译阶段就已经把多个jsp合并到一起

动态包含:在运行阶段把多个jsp合并到一起

taglib:导入标签库

jsp四大作用域

pageContext、request、session、application

jsp九大内置对象

request、response、page、pageContext、session、application、exception、out、config

**page和pageContext的区别**

page是设置一些当前页面的属性,pageContext是当前页面的上下文,可以存放数据

## URL和URI

URI是一个比较广泛的概念,URL是URI的一个子集,是URI的一种

## 对过滤器的理解

filter是过滤器,用来对请求前或者请求后的数据进行处理

filter的生命周期:实例化->初始化->进行过滤->销毁->释放资源

一个过滤器必须实现filter接口

一般我们用filter来编码的处理和安全验证

常见错误码:400:请求方法错误

403:跨域问题

404:找不到资源

415:不支持的媒体类型

500:后台服务器错误

## 常见端口号

**tomcat**:8080

**vue**：8080

**nginx**：80

**https协议**：443

**http协议**：80

**redis**:6379

**zookeeper**:2181

**rabbitMq**:5672

# Spring

## 什么是Spring?

spring是一个轻量级的java开发框架,他的目的是解决企业开发中的业务逻辑层和其它层的耦合度问题,他是一个开源框架

spring的是使命就是简化开发,使调用者只关注业务需求

## Spring的优缺点是什么?

**优点:**

1. **方便解耦，简化开发**
2. **AOP编程的支持**
3. **声明式事务的支持**
4. **方便集成各种框架**

**缺点:依赖反射,影响性能,使用门槛高,入门需要较长的时间**

## Spring框架中用了哪些设计模式?

单例设计模式

每一个bean对象的作用域默认都是单例的

工厂设计模式

在创建bean对象的时候用到了工厂设计模式

代理设计模式

Aop的就是代理设计模式

## 对IoC的理解?

IoC:控制反转,不是什么技术,是一种设计思想,好比MVC

就是将原本在程序中手动创建对象的控制权，交由 IoC 容器来管理。

**什么是依赖注入?**

**给实体类中的属性注入值**

1. **静态代理和动态代理**

Java中的静态代理要求代理类(ProxySubject)和委托类(RealSubject)都实现同一个接口(Subject),代理类在编译阶段字节码就已经生成了

Java中的动态代理依靠反射来实现.动态代理分为jdk动态代理和CGLIB动态代理,代理类的字节码在JVM运行时动态生成，而不是编译期就能确定。

## Aop

全称叫面向接口编程，底层是动态代理

## jdk动态代理和CGLIB动态代理的区别?

JDK动态代理只能对实现了接口的类生成代理，而不能针对类

CGLIB是针对类实现代理，主要是对指定的类生成一个子类，覆盖其中的方法（继承）

## Spring AOP相关术语

**Joinpoint**：连接点，一般指需要被增强的方法。where：去哪里做增强。

**Pointcut**：切入点，哪些包中的哪些类中的哪些方法，可认为是连接点的集合。where：去哪些地

方做增强。

**Advice**：增强，当拦截到 Joinpoint 之后，在方法执行的什么时机（when）做什么样（what）的

增强。根据时机分为：**前置增强、后置增强、异常增强、最终增强、环绕增强。**

**Aspect**：切面，Pointcut + Advice，去哪些地方 + 在什么时候 + 做什么增强。

**Target**：被代理的目标对象。

**Weaving**：织入，把 Advice 加到 Target 上之后，创建出 Proxy 对象的过程。

**Proxy**：一个类被 AOP 织入增强后，产生的代理类。

## Bean相关的

### Bean的生命周期

1.启动Spring容器

2.创建Bean对象----->实则在调用Bean对象的构造器

3.给Bean注入属性

4.调用Bean对象的初始化方法--init-method

5.我们获取Bean对象,调用bean对象的某一个方法

6.调用bean对象的销毁方法destroy-method

7.Spring容器销毁

### BeanFactory和FactoryBean的区别?

BeanFactory是Spring中最底层的接口,BeanFactory是创建和管理Bean对象的,而FactoryBean本身就是一个Bean对象

### Bean注入容器有哪些方式？

1. 使用@Controller、@Service、@Repository、@Component 注解标注该类，然后启用@ComponentScan自动扫描
2. 使用@Import 方法。
3. @Configuration与@Bean注解

### Bean的作用域有哪些?

**singleton:单例的,prototype:多例的,request,session:Http session,** global-session:全局session

### @Bean和@Component有什么区别？

都是使用注解定义 Bean。@Bean 是使用 Java 代码装配 Bean，@Component 是自动装配 Bean。

## Spring自动装配的方式有哪些？

Spring的自动装配有三种模式：byType(根据类型)，byName(根据名称)、constructor(根据构造函数)。

## @Autowired和@Resource的区别？

**@Autowired会先按照类型来进行装配,如果找不到，就报错，找到一个就注入，如果有多个,在按照名称来装配，找不到就报错**

**@Resource会先按照名称来进行装配,如果找到就注入，如果找不到bean在按照类型来装配，如果找到多个或者找不到就报错，如果找到一个就注入**

## @Qualifier 注解有什么作用?

**如果用@Autowired注解找到多个同类型的bean，再按照名称来找，我们也可以使用@Qualifier指定名称依赖注入**

## Spring事务实现方式有哪些?

Spring事务机制主要包括声明式事务和编程式事务。

**编程式事务**:可以通过编程的方式管理事务,这种方式很灵活,但是难维护

**声明式事务:**可以将事务管理和业务代码分离,并且自动帮我们管理事务,@Transactional注解或者是配置文件方式

**当我们使用transactional的时候，我们必须开启注解，在配置文件中配置一个bean，名称为TransactionManager来开启这个注解。**

**transactional默认运行异常才回滚，我们可以在他的注解的属性设置，rollbackFor等于throwable.class**

## 事务的传播行为

情况一:遵从当前事务

**REQUIRED:**必须存在事务,如果当前存在一个事务,就加入该事务,否则将创建一个事务(缺省) **required**

**SUPPORTS:**支持当前事务,指如果当前存在事务,就加入到该事务,如果当前没有事务,就以非事务方式执行 **supports**

**MANDATORY:**必须有事务,使用当前事务执行,如果当前没有事务,则抛出异常IlleaglTransactionStateException **mandatory**

情况二:不遵从当前事务

**REQUIRES\_NEW:**不管当前是否存在事务,每次都创建新的事务 **requires\_new**

**NOT\_SUPPORTED:**以非事务方式执行,如果当前存在事务,就把当前事务暂停,以非事务方式执行. **not\_supported**

**NEVER:**不支持事务,如果当前存在事务,则抛出异常,IllegalTransactionStateException

情况三:寄生事务(外部事物和嵌套事务) **never**

**NESTED:**寄生事务,如果当前存在事务,则在内部事务内执行,如果当前不存在事务,则创建一个新的事务, **nested**

嵌套事务使用数据库中的保存点来实现,即嵌套事务回滚不影响外部事务,但外部事务回滚将导致嵌套事务回滚.

寄生事务:DataSourceTransactionManager默认支持,而HibernateTransactionManager默认不支持,需要手动开启

## ApplicationContext和BeanFactory的区别

1、 BeanFactory需要等到获取某一个bean的时候才会创建该bean—延迟初始化。

2、 ApplicationContext在启动Spring容器的时候就会创建所有的bean（web应用建议）

## Spring在ssm中有什么作用?

Bean工厂,用来管理Bean对象的生命周期和框架集成  
ssm中的Spring的配置文件内容?  
开启IoC和Di注解

配置数据库

事务管理器

开启事务

配置mybatis

配置sql的映射文件

# Spring mvc

## 简单介绍下你对springMVC的理解？

Spring MVC是一种基于java，使用了MVC架构模式的思想，将Web层进行职责解耦，简化我们日常Web开发。Spring MVC核心类是· DispatcherServlet，它是一个 Servlet，顶层是实现的Servlet接口。

## MVC思想

它是一种软件设计思想，进行分层设计，mvc分别是

m：model：模型层：dao、service

v:view:视图层:前端的jsp、vue等

c:controller:控制层

好处就是降低模块之间的耦合性，提高了可维护性和可拓展性

## 如何解决 POST 请求和 GET 请求中文乱码问题？

解决 POST 请求乱码问题：在 web.xml 中配置一个 CharacterEncodingFilter 过滤器，设置成 utf-8；

（2）get请求中文参数出现乱码解决方法有两个：

①修改tomcat配置文件添加编码与工程编码一致，如下：

<ConnectorURIEncoding=“utf-8” connectionTimeout=“20000” port=“8080” protocol=“HTTP/1.1” redirectPort=“8443”/>

②另外一种方法对参数进行重新编码：

String userName = new String(request.getParamter(“userName”).getBytes(“ISO8859-1”),“utf-8”)

ISO8859-1是tomcat默认编码，需要将tomcat编码后的内容按utf-8编码。

# mybatis

## 什么是mybatis？

MyBatis是一个半自动的ORM持久层框架，它内部封装了jdbc，使开发者只需要关注sql语句本身，而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。

## mybatis的优缺点？

**优点：**

1. 易于上手和掌握
2. sql写在xml里，便于统一管理和优化
3. 解除sql语句和程序代码的耦合
4. 解决了数据库字段和实体类字段不一致问题
5. 提供xml标签，支持编写动态sql

**缺点：**

SQL语句工作量很大，尤其是字段多、关联表多时。更是如此

## #{}和${}的区别是什么？

1.#{} 是预编译处理，是占位符，${} 是字符串替换、是拼接符；

2.MyBatis 在处理 #{} 时，会将 sql 中的 #{} 替换为 ？，调用 PreparedStatement 来赋值；

3.MyBatis 在处理 ${} 时，是把 ${} 替换成变量的值，调用 Statement 来赋值；

4.使用 #{} 可以有效地防止 SQL 注入，提高系统安全性。

## Mybatis的Xml映射文件中，不同的Xml映射文件，id是否可以重复？

不同的 Xml 映射文件，如果配置了 namespace，那么 id 可以重复；如果没有配

置 namespace，那么 id 不能重复；

原因就是 namespace+id 是作为 Map <String, MapperStatement> 的 key

使用的，如果没有 namespace，就剩下 id，那么，id 重复会导致数据互相覆盖。

有了 namespace，自然 id 就可以重复，namespace 不同，namespace+id 自然

也就不同

## Mybatis是如何将sql执行结果封装为目标对象并返回的？都有哪些映射形式？

通过resultType标签写上实体类全限定名或者别名，或者配置resultMap根据resultMap的id映射上

## 使用MyBatis的mapper接口调用时有哪些要求？

1、Mapper 接口方法名和 mapper.xml 中定义的每个 sql 的 id 相同；

2、Mapper 接口方法的返回值类型和 mapper.xml 中定义的每个 sql 的

resultType 的类型相同；

3、Mapper.xml 文件中的 namespace 即是 mapper 接口的类路径。

## JDBC编程有哪些不足之处，MyBatis是如何解决这些问题的？

1.JDBC解决不了或者是很难解决数据库字段和实体类的属性不相同，mybatis提供了resultMap来解决这个问题

2. Sql语句写在代码中造成代码不易维护，实际应用sql变化的可能较大，sql变动需要改变java代码。

解决：将Sql语句配置在XXXXmapper.xml文件中与java代码分离。

3. 向sql语句传参数麻烦，因为sql语句的where条件不一定，可能多也可能少，占位符需要和参数一一对应。

解决： Mybatis自动将java对象映射至sql语句。

4. 对结果集解析麻烦，sql变化导致解析代码变化，且解析前需要遍历，如果能将数据库记录封装成pojo对象解析比

较方便。

解决：Mybatis自动将sql执行结果映射至java对象。

## mybatis的一级缓存和二级缓存

mybatis的一级缓存是默认开启的，sqlsession级别的缓存，实现在同一会话中数据的共享

一级缓存的生命周期和sqlsession的生命周期一致

二级缓存

二级缓存默认不开启，需要手动开启，sqlsessionFactory级别的缓存，实现不同会话中数据的共享，是一个全局变量

开启缓存后数据的查询流程：二级缓存>一级缓存>数据库

## mybatis的编程步骤

创建sqlsessionFactory

通过sqlsessionFactory创建sqlsession

通过sqlsession执行数据库操作

调用commit方法提交事务

调用close方法关闭事务

# Springboot

## 什么是springboot？

springboot是一个框架，一种全新的编程规范，他的产生简化了框架的使用，所谓简化是指简化了spring

众多框架中所需要的大量且繁琐的配置文件，所以SpringBoot是一个服务于框架的框架，服务的范围就是简化配置文件。

## Springboot的优点？

快速构建项目

内置tomcat等服务器，不需要手动配置

对主流框架无缝集成

避免了大量的Maven导入和各种版本冲突

## Springboot几种启动方式

主启动类main方法，也就是加了@SpringBootApplication注解的类的main方法

打jar包在服务器命令行java -jar

通过spring-boot-plugin的方式

## yml配置文件优先级

1. 项目中的config文件夹

2. 项目中

3. resources文件夹里面的config文件夹

4. resources文件夹

## 常用注解

**@SpringBootApplication**

**@Repository**

**@Service**

**@RestController**

**@ResponseBody**

**@Component**

**@ComponentScan**

**@Configuration**

**@Bean**

**@AutoWired**

**@Qualifier**

**@Resource(name="name",type="type")**

**@RequestMapping**

**@Mapper**

**@ConfigurationProperties(prefix=”前缀”)注解和@Value注解一样**

**都是从配置文件中注入属性的**

从配置文件中取出数据,给类上的所有字段赋值,需要和@Component注解一起使用,prefix里面的字母不支持驼峰命名,只能小写.

**@Value注解**

从配置文件yml中取值,贴到字段上给字段赋值,例如:@Value(“${user.username}”)

## 静态资源的放置和执行顺序

静态资源放在resources底下,有三个文件夹,分别是resource、public和static三种

一般我们方法在static文件夹下,public存放大家都要访问的资源

执行顺序:resource>static>public

## 我们常说SpringBoot约定大于配置什么意思?

* 1. 开发人员仅需规定应用中不符合约定的部分

② 在没有规定配置的地方，采用默认配置，以力求最简配置为核心思想

**那么SpringBoot中有哪些约定呢？**

1. Maven的目录结构。默认有resources文件夹,存放资源配置文件。src-main-resources,src-main-java。默认的编译生成的类都在targe文件夹下面
2. spring boot默认的配置文件必须是，也只能是application.命名的yml文件或者properties文件，且唯一
3. application.yml中默认属性。[数据库](https://cloud.tencent.com/solution/database?from=10680" \t "_blank)连接信息必须是以spring: datasource: 为前缀；多环境配置。该属性可以根据运行环境自动读取不同的配置文件；端口号、请求路径等

## springboot关于springmvc的怎么配置拦截器

1.新建类实现HandlerInterceptor,并重写preHandle方法,在里面取出cookie并做判断cookie是否为空,如果为空就重定向到登录界面,然后

return false

2.创建一个类,贴上Configuration注解,实现接口WebMvcConfigurer,覆盖addInterceptors方法,例如:

@Override

public void addInterceptors(InterceptorRegistry registry) {

registry.addInterceptor(new CheckLoginInterceptor())

.addPathPatterns("/\*\*")

.excludePathPatterns("/","/user/\*","/jquery/\*");

}

# git

区别：“**git clone”是在本地没有版本库时，将远程库整个下载到本地**；而“git pull”是在本地有版本库时，从远程库中获取最新commit数据，并合并到本地。 “git pull”相当于“git fetch”加“git merge”。

# sql优化

1、SELECT子句中避免使用 \*， 尽量应该根据业务需求按字段进行查询

2、在写sql语句的话，尽量保持每次查询的sql语句字段用大写，因为oracle总是先解析 sql语句，把小写的字母转换成大写的再执行

3、在做连表查询时，小的表在Join字段的左边

4、创建并使用正确的索引

5 多用limit，效率高

6 in中值太多，要尽可能少

## union和union all的区别？

一、区别1：

1、union: 对两个结果集进行并集操作, 不包括重复行,相当于distinct;

2、union all: 对两个结果集进行并集操作,, 不管是不是重复;

二、区别2：

1、union: 会对获取的结果进行排序操作

2、union all: 不会对获取的结果进行排序操作

## 数据库中char和varchar谁的效率高？

两相比较,char 的效率高,没有碎片,尤其更新比较频繁的时候,方便数据文件指针的操作。但不够灵活,在实际使用时,应根据实际需求来选用合适的数据类型。

## 索引的利

减少IO操作次数,提供查询效率

降低CUP使用率(在排序操作中尤为明显)

## 索引的弊

占据大量的硬盘存储空间

创建索引和维护索引要耗费时间，这种时间随着数据量的增加而增加。

**假如abc添加了索引，那么ac走不走索引？**

走索引

对于复合索引,在查询使用时,最好将条件顺序按照**索引的顺序**,这样效率最高;

select \* from table1 where col1=A AND col2=B AND col3=D

如果使用 where col2=B AND col1=A 或者 where col2=B 将不会使用索引

## 聚簇索引和非聚簇索引

**聚簇索引:把数据存储和索引放到一块,就是聚簇索引**

**非聚簇索引:把数据存储和索引分开就是非聚簇索引.**

**非聚簇索引会先按内存地址查找出主键,再根据主键去聚簇索引查找数据,查找两次**

**Innodb上的索引是聚簇索引，叶子上存储的是数据**

**MyISAM上的索引是非聚簇索引，叶子上存储的是磁盘空间地址**

## Innodb和MyISAM的区别

1、innodb支持事务，而myisam不支持事务。

2、innodb支持外键，而myisam不支持外键。

3、innodb是行锁，而myisam是表锁（每次更新增加删除都会锁住表）。

4、innodb和myisam的索引都是基于b+树，但他们具体实现不一样，innodb的b+树的叶子节点是存放数据的，myisam的b+树的叶子节点是存放指针的。

5、innodb是聚簇索引，必须要有主键，一定会基于主键查询，但是辅助索引就会查询两次，myisam是非聚簇索引，索引和数据是分离的，索引里保存的是数据地址的指针，主键索引和辅助索引是分开的。

6、innodb不存储表的行数，所以select count( \* )的时候会全表查询，而myisam会存放表的行数，select count(\*）的时候会查的很快。

## 索引失效的情况

1. where语句中索引列参与算术计算,该索引失效
2. where语句中索引列参与函数运算,该索引失效
3. where语句中做不等于( != , <> )运行,该索引失效
4. where语句中发生类型转发,该索引失效
5. where语句中模糊查询时以%开头,该索引失效
6. 在复合索引的使用时跟声明时顺序不一致或者中间有列的缺失,该索引失效

[https://blog.csdn.net/CXZ\_BGB/article/details/125477914?spm=1001.2014.3001.5502](https://blog.csdn.net/CXZ\_BGB/article/details/125477914?spm=1001.2014.3001.5502)

# Linux

**rz**:打开文件夹,选择文件上传linux服务器

**touch**：创建文件

**mkdir**：创建文件夹

**cd 路径**：进入目录

**cd ..** ：返回上一级目录

**rm**：删除文件

**rm -f** ：强制删除

**rm -r**：递归删除

**ls**：查看目录中所有文件

**./startup.sh**：启动tomcat

**tail -f catalina.out** ：查看tomcat日志

**netstat -tunlp | grep 8080**：查看8080端口的占用情况

**kill -9 PID**：杀死进程

**su 用户名**：切换用户

# Vue

## vue组件化

组件化就是将实现页面某个功能的结构、样式和逻辑,封装成一个整体,使其低内聚、高耦合,达到可复用的目的

## 对MVVM思想的理解?

MVVM 是Model-View-ViewModel 的缩写，它是一种基于前端开发的架构模式，其核心是提供对View 和 Model 的双向数据绑定，这使得Model 的状态改变可以自动传递给 View，即所谓的数据双向定。

Model(数据模型): 数据层(后端传递的数据)。它仅仅关注数据本身,不关心任何行为。

View(视图): 用户操作界面, 用户在屏幕上看到的结构、布局和外观（UI）。当ViewModel对数据进行更新时,会通过数据绑定更新到View。

ViewModel: 视图模型, 业务逻辑层。它是连接view和model的中间桥梁,它有两个方向:

一是把Model(模型)转化为View(视图),把数据转化成所看见的页面,实现方式是: 数据绑定.

二是把View(视图)转化为Model(模型),即把所看到的页面转化成数据,实现方式是: DOM事件监听

这两个方向的实现,我们称之为数据的双向绑定

## 子传父和父传子

用方法

this.$emit(“事件名”,参数,参数)

父传子用props

## 生命周期:

创建前: beforeCreate **vue实例初始化之前调用**

创建后:created **vue实例初始化之后调用**

载入前:beforeMount **挂载到DOM树之前调用**

载入后:mounted **挂载到DOM树之后调用**

更新前:beforeUpdate **数据更新之前调用**

更新后:updated **数据更新之后调用**

销毁前:beforeDestroy **vue实例销毁之前调用**

销毁后:destroyed **vue实例销毁之后调用**

## 路由传参

this.$router.push(“路径?参数”)