[Java Platform, Standard Edition 8](https://docs.oracle.com/javase/8/docs/api/)，[The Java Language Specification | Java SE 8 Edition](https://docs.oracle.com/javase/specs/jls/se8/html/index.html)，

[The Java Virtual Machine Specification](https://docs.oracle.com/javase/specs/jvms/se8/html/index.html)，

JavaSE

Java、JDK、Unicode

Java1是一门面向对象编程语言，不仅吸收了C++语言的各种优点，还摒弃了C++里难以理解的多继承、指针等概念，因此Java语言具有功能强大和简单易用两个特征。Java语言作为静态面向对象编程语言的代表，极好地实现了面向对象理论，允许程序员以优雅的思维方式进行复杂的编程。

[JDK8新特性应用实践](https://blog.csdn.net/maitian_2008/article/details/118580392?spm=1001.2014.3001.5501)，[JDK5特性分析](https://blog.csdn.net/maitian_2008/article/details/118580219?spm=1001.2014.3001.5501)，

正确描述：JDK中提供的java、javac、jar等开发工具也是用Java编写的。（应该只有JVM不是用Java写的。）

JDK1是java开发工具包，包含JRE和java工具集。

JDK = JRE + 开发工具集

JRE = JVM + Java SE标准类库

JDK是最小开发环境，JRE是最小运行环境，JVM是实现跨平台的核心。

Java之所以能做到“一次编译，处处运行”是因为JVM隐藏了底层技术的复杂性以及机器和操作系统的差异性。

JRE根据不同操作系统和不同JRE提供商（IBM，ORACLE等）有很多版本，最常用的是Oracle公司收购SUN公司的JRE版本。

一台PC可以配置多个JDK，环境变量配置哪个，哪个生效。

cmd查看当前安装的JDK版本，Java -version。

cmd查看当前配置的JDK路径，Java -verbose。

JAVA\_HOME：配置的是JDK的根目录。

PATH：这个环境变量不需要新建，配置的是JDK的bin目录。

CLASS\_PATH：配置的也是JDK的bin目录，与Tomcat有关。

字节码文件（以byte为单位存储的文件）是描述程序要运行的虚指令的集合，这些集合与平台无关，只要配置好相对于的JRE，就可以运行JVM，从实现多个平台都能运行java文件，实现跨平台。

错误描述：Char类型在Java语言里面存储的是ASCII码。（Java中的char是Unicode编码。Unicode编码占两个字节，就是16位，足够存储一个汉字。）

Unicode1即统一码，也叫万国码、单一码，是计算机科学领域里的一项业界标准，包括字符集、编码方案等。Unicode是为了解决传统的字符编码方案的局限而产生的，它为每种语言中的每个字符设定了统一并且唯一的二进制编码，以满足跨语言、跨平台进行文本转换、处理的要求。1990年开始研发，1994年正式发布1.0版本，2021年9月14日发布14.0版本。

如果把各种文字编码形容为各地的方言，那么Unicode就是世界各国合作开发的一种语言。

在这种语言环境下，不会再有语言的编码冲突。在同屏下，可以显示任何语言的内容，这就是Unicode的最大好处。Unicode中所有的字符都用2个字节统一进行编码，这意味着2个字节就已经足够容纳世界上所有的语言的大部分文字了。

UTF-8以字节为单位对Unicode进行编码。UTF-16编码以16位无符号整数为单位。UTF-32编码以32位无符号整数为单位。

大多数计算机采用ASCII码（美国标准信息交换码），它是表示所有大小写字母、数字、标点符号和控制字符的7位编码方案。Unicode码包含ASCII码，'\u0000'到'\u007F'对应全部128个ACSII字符。

Java语言采用Unicode编码，它为每个字符制订了一个唯一的数值，因此在任何的语言，平台，程序都可以放心的使用。

英语ASCII码128个，法语等ASCII码256个，汉字则大于255。

数据类型、变量、关键字

[java基本数据类型所占字节数 - 乔瑟琳 - 博客园 (cnblogs.com)](https://www.cnblogs.com/yuanting/p/4756632.html#:~:text=JAVA%E5%9F%BA%E6%9C%AC%E6%95%B0%E6%8D%AE%E7%B1%BB%E5%9E%8B%E6%89%80%E5%8D%A0%E5%AD%97%E8%8A%82%E6%95%B0%E6%98%AF%E5%A4%9A%E5%B0%91%EF%BC%9F%20%EF%BC%8832%E4%BD%8D%E7%B3%BB%E7%BB%9F%EF%BC%89%20JAVA%E6%98%AF%E9%87%87%E7%94%A8Unicode%E7%BC%96%E7%A0%81%E3%80%82%20%E6%AF%8F%E4%B8%80%E4%B8%AA%E5%AD%97%E8%8A%82%E5%8D%A08%E4%BD%8D%E3%80%82%20%E4%BD%A0%E7%94%B5%E8%84%91%E7%B3%BB%E7%BB%9F%E5%BA%94%E8%AF%A5%E6%98%AF%2032%E4%BD%8D%E7%B3%BB%E7%BB%9F%EF%BC%8C%E8%BF%99%E6%A0%B7%E6%AF%8F%E4%B8%AAint%E5%B0%B1%E6%98%AF%204%E4%B8%AA%E5%AD%97%E8%8A%82%20%E5%AD%97%E7%AC%A6,2%E5%AD%97%E8%8A%82%EF%BC%8816%E4%BD%8D%EF%BC%89%20u0000~uFFFF%EF%BC%88%E2%80%98%E2%80%99~%E2%80%98%EF%BC%9F%20%E2%80%99%EF%BC%89%20%E2%80%980%E2%80%99%20Character%20%E5%9C%A8%E7%A7%BB%E5%8A%A8%E5%BC%80%E5%8F%91%E4%B8%AD%E7%94%B1%E4%BA%8E%E7%A7%BB%E5%8A%A8%E8%AE%BE%E5%A4%87%E5%86%85%E5%AD%98%E7%9A%84%E5%B1%80%E9%99%90%E6%80%A7%EF%BC%8C%E5%BE%80%E5%BE%80%E9%9C%80%E8%A6%81%E8%80%83%E8%99%91%E4%BD%BF%E7%94%A8%E7%9A%84%E6%95%B0%E6%8D%AE%E7%B1%BB%E5%9E%8B%E6%89%80%E5%8D%A0%E7%94%A8%E7%9A%84%E5%AD%97%E8%8A%82%E6%95%B0%E3%80%82%20%E4%B8%8B%E9%9D%A2%E7%AE%80%E5%8D%95%E4%BB%8B%E7%BB%8D%E4%B8%8BJava%E4%B8%AD%E5%87%A0%E7%A7%8D%E5%9F%BA%E6%9C%AC%E6%95%B0%E6%8D%AE%E7%B1%BB%E5%9E%8B%EF%BC%8C%E4%BB%A5%E5%8A%A0%E6%B7%B1%E8%AE%B0%E5%BF%86%E3%80%82%20%E5%9C%A8Java%E4%B8%AD%E4%B8%80%E5%85%B1%E6%9C%898%E7%A7%8D%E5%9F%BA%E6%9C%AC%E6%95%B0%E6%8D%AE%E7%B1%BB%E5%9E%8B%EF%BC%8C%E5%85%B6%E4%B8%AD%E6%9C%894%E7%A7%8D%E6%95%B4%E5%9E%8B%EF%BC%8C2%E7%A7%8D%E6%B5%AE%E7%82%B9%E7%B1%BB%E5%9E%8B%EF%BC%8C1%E7%A7%8D%E7%94%A8%E4%BA%8E%E8%A1%A8%E7%A4%BAUnicode%E7%BC%96%E7%A0%81%E7%9A%84%E5%AD%97%E7%AC%A6%E5%8D%95%E5%85%83%E7%9A%84%E5%AD%97%E7%AC%A6%E7%B1%BB%E5%9E%8B%E5%92%8C1%E7%A7%8D%E7%94%A8%E4%BA%8E%E8%A1%A8%E7%A4%BA%E7%9C%9F%E5%80%BC%E7%9A%84boolean%E7%B1%BB%E5%9E%8B%E3%80%82)，

数据类型1

基本数据类型：不同于类变量和成员变量，局部变量数据一般直接存储在栈上。

引用数据类型：数据存储在堆上，栈上只存储引用地址。

8个基本数据类型：byte、short、int、long，float、double、char、boolean。

3个引用数据类型：类、数组、Interface。java.lang.String和java.lang.Enum都属于类类型。

\n算一个字符，"\n"和'\n'是等价的。

char类型的默认值'\u0000'，空字符'\0'在bytecode中表示为'\u0000'。

null代表堆内存中根本没有这个东西，而""是有内存空间的。

byte的默认值为0，但Byte的默认值为null。

凡是byte、short、char数据类型数据在运算的时候都会自动转换成int类型的数据再运算。

整数运算，结果仍是整数；有小数参与的运算，结果为小数。

只要有小数参与运算，就不会报除0错误。

float的指数范围为-2^7~2^7-1（即-128~127），而double的指数范围为-2^10~2^10-1（即

-1024~1023），并且指数位是补码的形式来划分的。

其中负指数决定了浮点数所能表达的绝对值最小的数；而正指数决定了浮点数所能表达的绝对值最大的数，也即决定了浮点数的取值范围。

错误描述：成员变量所属于对象，所以也称为实例变量。静态变量所属于类，所以也称为类变量。（正确描述：成员变量包括实例变量和类变量（静态变量）。）

错误描述：成员变量与对象共存亡，静态变量与类共存亡。（正确描述：实例变量与对象共存亡，静态变量与类共存亡。）

变量1来源于数学，是计算机语言中能储存计算结果或能表示值的抽象概念。

变量可以通过变量名访问。在指令式语言中，变量通常是可变的；但在纯函数式语言（如Haskell）中，变量可能是不可变的。

在一些语言中，变量可能被明确为是能表示可变状态、具有存储空间的抽象（如在Java中）；但另外一些语言可能使用其它概念（如C的对象）来指称这种抽象，而不严格地定义变量的准确外延。

成员变量要不要赋初值完全取决于是不是final。

类变量（以static修饰）

实例变量（不以static修饰）

成员变量什么修饰符都可以有，局部变量什么修饰符都不能有。

当本类的成员变量与局部变量同名时，被this.的那个money是本类的成员变量。

局部变量需赋值后才能使用，成员变量不用赋值，也会自动被初始化成默认值。

一般来讲局部变量存在栈中，方法执行完毕内存就被释放。

对象（new出来的东西）存在堆中，对象不再被使用时，内存才会被释放。

我们通过每个对象自己唯一的地址值来区分不同的对象。

定义常量的时候，所有字母都要大写。

关键字1是电脑语言里事先定义的，有特别意义的标识符，有时又叫保留字，还有特别意义的变量。Java一共有50个关键字，包括2个保留字（现在没用以后可能用到作为关键字），三个特殊直接量，全都是小写。

const是C和C++中的常量关键字，java中定义常量的关键字是final，const只作为保留字，暂无作用。

goto是面向过程语言中的常用关键字，用于跳转到指定位置。由于打乱了代码的先后执行顺序，易导致可读性变差，在java中仅作为保留字，暂无作用。

switch括号内支持的类型有byte，short，char，int，String和Enum，小括号中变量的类型必须与case后value的类型一致，case后不能跟变量。（Java5支持的类型引入Enum，Java7支持的类型引入String。）（switch括号里一般都是字符串和整数了。）

一个if语句可以跟随多个else if语句，但是只能有一个else语句。

break跳出当前循环，return跳出整个函数。

使用带有标号的break语句，可以跳出多层循环而不return。

public static void main(String[] args) {

ok:

for (int i = 0; i < 10; i++) {

for (int j = 0; j < 10; j++) {

System.out.println("i=" + i + ",j=" + j);

if (j == 5) {

break ok;

}

}

}

}

native关键字说明其修饰的方法是一个原生态方法，方法对应的实现不是在当前文件，而是在用其他语言（如C和C++）实现的文件中。Java语言本身不能对操作系统底层进行访问和操作，但是可以通过JNI调用其他语言来实现对底层的访问。

transient简单地说，就是让某些被修饰的成员属性变量不被序列化。当一个变量不希望被持久化的时候，比如说一些账号密码，就可以用transient关键字来表示该变量不参与序列化过程。

Serializable是一个空接口，起标记作用，具体的序列化由ObjectOutputStream和ObjectInputStream完成。

transient修饰的变量不能被序列化，static变量不管加没加transient都不可以被序列化。

变量的使用遵循就近原则，离谁近，就用谁。

从本质上讲，this是一个指向本对象的指针，super是一个指向父类对象的指针。this是一个java关键字，super也是。

this()和super()都指的是对象，均不可以在static环境中使用。

我们可以把super看作是父类的对象：Father super = new Father();

如果想指定本类的成员变量，使用this.变量名来指定。

如果想指定父类的成员变量，使用super.变量名来指定。

使用this()调用本类的无参构造，使用this(参数)调用本类对应参数的构造方法。

构造方法与本类同名且没有返回值类型(void也是一种返回值类型)，主要功能是完成对象创建或者初始化。

使用super()调用父类的无参构造，使用super(参数)调用父类对应参数的构造方法。

super()默认存在于子类的构造方法体第一行，会先实现父类的构造方法，再实现子类的构造方法。如果父类没有无参构造，需要手动指定调用哪个含参构造。（这个super()里面还大有文章哩。）

构造方法不能继承，因为构造方法要求名字是本类的类名，我们不能在子类中出现父类名字的构造方法。

JNI（Java Native Interface）是Java SDK（Software Development Kit）的一部分。JNI允许Java代码使用以其他语言编写的代码和代码库。Invocation API（JNI的一部分）可以用来将Java虚拟机（JVM）嵌入到本机应用程序中，从而允许程序员从本机代码内部调用Java代码。

成员变量什么修饰符都可以有，局部变量什么修饰符都不能有。（static修饰成员变量就是静态变量嘛。）

我们可以使用private关键字来封装成员变量与方法。

驼峰命名规则：java是强类型语言，针对代码中大小写敏感。类名，每个单词首字母都要大写。变量名和方法名，除了首个单词首字母不大写，其余单词首字母都要大写。

为什么要封装？

封装可以提高程序的安全性，封装可以让资源按照我们预先规定的方式来操作。

如用private封装成员变量后，就不可以随意修改其值。

包括后续单例模式也主要就是通过private封装实现的。

如何访问私有资源？

私有属性可以generate getter/setter，私有方法放在公共方法里供外界调用，但方法的封装用得不多，属性的封装才是重点。

运算符、==、equals

[Java运算符及运算符的优先级\_feifuzeng的博客-CSDN博客\_java运算符优先级由高到低](https://blog.csdn.net/feifuzeng/article/details/108862001)，

运算符1指明对操作数的运算方式。组成表达式的Java操作符有很多种。运算符按照其要求的操作数数目来分，可以有单目运算符、双目运算符和三目运算符，它们分别对应于1个、2个、3个操作数。运算符按其功能来分，有算术运算符、赋值运算符、关系运算符、逻辑运算符、位运算符和其他运算符。

赋值运算符自带强转。

逻辑与(&&)、按位与(&)、逻辑或(||)、按位或(|)、按位异或(^)、逻辑非(!)、按位取反(~)。

&&有短路功能，而&具有位运算功能。

! →& →^ →| →&& →||，非→与→异或→或→短路与→短路或。

[java位运算符常用场景\_风吹千里的博客-CSDN博客](https://blog.csdn.net/maligebazi/article/details/79803438)，

注意使用位运算时，写成\* 2 << 10，就相当于乘以2的11次方。

任何一位二进制数同1异或都会变成另外一个，任何一位二进制数同0异或都保持不变。

a^a=0;

a^0=a;

a^b^c=a^c^b;

模运算%的结果的正负号只与其左操作数有关。

Java的赋值运算是有返回值的，赋了什么值，就返回什么值。

凡是byte、short、char数据类型数据在运算的时候都会自动转换成int类型的数据再运算。

运算结果的数据类型与参与运算的最大类型保持一致，int+int-->int double/int-->double。

整数运算，结果仍是整数；有小数参与的运算，结果为小数。

只要有小数参与运算，就不会报除0错误。

模运算结果的符号与被除数的符号一致。

正确描述：类A1和类A2在同一包中，类A2有个protected的方法testA2，类A1不是类A2的子类（或子类的子类），类A1可以访问类A2的方法testA2。

access-modifier好好领会！！！

编译错误：int i = -5; i = ++(i++);// 报错/variable expected

括号里面必须是一个变量，而i ++是一个字面量。

==是运算符，用于比较基础类型变量和引用类型变量。对于基础类型变量，比较变量保存的值是否相同，类型不一定要相同；对于引用类型变量，比较对象的地址值是否相同。

equals是Object类中定义的方法，通常用于比较两个对象的值是否相等。

equals未被重写等价于==，equals被重写后比较两者的内容。（一般都会重写的。）

==是不会变的，但是equals()作为方法可能被重写，比如String类就重写了该方法，比较的是字符串的内容，而不是地址值。

当有a.equals(b)==true时，则a.hashCode()==b.hashCode()必然成立；

反过来，当a.hashCode()==b.hashCode()时，a.equals(b)不一定为true。

两个数值进行二元操作时，会有如下的转换操作：

如果两个操作数其中有一个是double类型，另一个操作就会转换为double类型。

否则，如果其中一个操作数是float类型，另一个将会转换为float类型。

否则，如果其中一个操作数是long类型，另一个会转换为long类型。

否则，两个操作数都转换为int类型。

注解、枚举、单元测试、Throwable

[java注解-最通俗易懂的讲解 - Tanyboye - 博客园 (cnblogs.com)](https://www.cnblogs.com/Tanyboye/p/9138412.html#top)，

注解1 同类和接口一样，注解也属于一种Java类型，始于SE5.0版本。

注解通过元注解来定义。元注解是一种基本注解，它可以注解到包括自身在内的所有注解上。

@CrossOrigin允许跨域访问。

同源策略：

1. 浏览器访问地址
2. Ajax请求地址

对于①和②，协议://域名:端口都相同则称之为同源策略或同域访问，反之则为跨域访问。

1)

浏览器地址: http://www.jd.com/xx/xx

Ajax地址: https://www.jd.com/yy/yy 跨域/协议不同

2)

浏览器地址: http://www.jd.com/xx/xx

Ajax地址: http://www.jd.com/yy/yy 同域

3)

浏览器地址: http://www.jd.com:80/xx/xx

Ajax地址: http://www.jd.com/yy/yy 同域/http默认80

4)

已知域名与IP对应

浏览器地址: http://www.jd.com/xx/xx

Ajax地址: http://8.8.8.8/yy/yy 跨域/通过DNS对应不等于域名相同

5)

浏览器地址: https://www.jd.com/xx/xx

Ajax地址: https://www.jd.com:443/yy/yy 同域/https默认443

6)

浏览器地址: http://www.jd.com/xx/yy/xx

Ajax地址: http://miaoshi.jd.com/yy/yy 跨域/添加了前缀子域名（典型的不同服务器。）

枚举1是一个特殊的类，一般表示一组常量，比如一年的4个季节，一个年的12个月份，一个星期的7天，方向有东南西北等。

枚举类使用enum关键字来定义，各个常量使用逗号，来分割。

[从JUnit 4迁移到JUnit 5：重要的区别和好处](https://zhuanlan.zhihu.com/p/144763642)，

单元测试1是指对软件中的最小可测试单元进行检查和验证。对于单元测试中单元的含义，一般来说，要根据实际情况去判定其具体含义，如C语言中单元指一个函数，Java里单元指一个类，图形化的软件中可以指一个窗口或一个菜单等。总的来说，单元就是人为规定的最小的被测功能模块。单元测试是在软件开发过程中要进行的最低级别的测试活动，软件的独立单元将在与程序的其他部分相隔离的情况下进行测试。

[异常分类.png](file:///D:\Pkg\图片\Java\异常分类.png)，[错误与异常.png](file:///D:\Pkg\图片\Java\错误与异常.png)，

[以下说法哪个是正确的（）\_\_牛客网 (nowcoder.com)](https://www.nowcoder.com/questionTerminal/bcb966db4d2c43768bc077296a8a1017)，

Exception1还包括编译异常，编译器能够实时报错提醒。Exception是所有异常类的父类，但如果Throwable也在讨论范围内，就认为是前者吧。

Error是系统级的错误，而Exception是可以避免的程序的错误。

throw是语句抛出一个异常，throws是方法可能抛出异常的声明。

finally里面的return会覆盖try里面的return。

SQLException和自定义异常都属于检查异常，所以在DBHelper里面经常会看到try-catch语句捕获异常。（可能MyBatis源码也涉及。）

错误描述：自定义异常必须继承Exception。（可继承自Throwable。）

错误描述：对于非运行时异常，程序中一般可不做处理，由JVM自动进行处理。（运行异常，可以让JVM自行处理；非运行异常，我们应该捕获或者抛出。）

面向对象、类、引用、按值传递

面向对象1程序中将具有相同属性的方法的事物归为一类，而对象就是类的一个实例。面向对象能够避免重复造轮子，具有封装、继承、多态三大特征。面向对象是相对于面向过程而言的，面向对象把相关的数据和方法组织为一个整体来看待，从更高的层次来进行系统建模，更贴近事物的自然运行模式。

封装可以提高代码安全性。（私有化属性，公有化方法。）

继承是面向对象最显著的一个特征。在实际应用中继承关系是一种强耦合关系，继承关系层次最好不要多于三层，否则可能难于维护。

继承+重写构成了多态的前提。

重载（同类同名不同参）、重写（两同两小一大）、多态（父类引用指向子类对象；编译看左边，运行看右边）。

成员类一般没有main方法，是一个零件/工具。有main方法的叫做执行类。

定义成员类就是一个抽类的过程，提高代码复用率/重用率。

面向对象五大基本原则：

1)单一职责原则SRP(Single Responsibility Principle)

类的功能要单一，不能包罗万象，跟杂货铺似的。

2)开放封闭原则OCP(Open-Close Principle)

一个模块对于拓展是开放的，对于修改是封闭的。

3)里式替换原则LSP(the Liskov Substitution Principle LSP)

子类可以替换父类出现在父类能够出现的任何地方。

4)依赖倒置原则DIP(the Dependency Inversion Principle DIP)

高层次的模块不应该依赖于低层次的模块，他们都应该依赖于抽象。抽象不应该依赖于具体实现，具体实现应该依赖于抽象。

5)接口分离原则ISP(the Interface Segregation Principle ISP)

设计时采用多个与特定客户类有关的接口比采用一个通用的接口要好。（类似于SRP。）

类1是相同种类对象的抽象，是它们的公共属性。对象是类的实例。类在现实世界并不存在，它只是一种对象的引用数据类型。

类是抽象的，对象是具体的，我们通过每个对象自己唯一的地址值来区分不同的对象。

如果子类对从父类继承来的成员变量进行重新定义，即出现了子类变量对父类变量的隐藏。

构造函数直接不能互相调用，否则会死循环。

super()默认存在于子类的构造方法体第一行，会先实现父类的构造方法，再实现子类的构造方法。如果父类没有无参构造，需要手动指定调用哪个含参构造。

[Java基础篇 - 强引用、弱引用、软引用和虚引用\_零壹技术栈的博客-CSDN博客\_java 弱引用 软引用](https://blog.csdn.net/baidu_22254181/article/details/81979663?ops_request_misc=%257B%2522request%255Fid%2522%253A%2522163722254416780271566112%2522%252C%2522scm%2522%253A%252220140713.130102334..%2522%257D&request_id=163722254416780271566112&biz_id=0&utm_medium=distribute.pc_search_result.none-task-blog-2~all~top_positive~default-1-81979663.first_rank_v2_pc_rank_v29&utm_term=%E5%BC%BA%E5%BC%95%E7%94%A8%E5%92%8C%E5%BC%B1%E5%BC%95%E7%94%A8&spm=1018.2226.3001.4187)，

引用1

[这一次，让你彻底明白Java的值传递和引用传递！\_Java后端技术的博客-CSDN博客](https://blog.csdn.net/bntx2jsqfehy7/article/details/83508006)，

按值传递1

Java里面只有一种参数传递方式就是按值传递（对于对象参数，值的内容是对象的引用），即将实际参数值的副本传入方法内，而参数本身不受影响。

值传递和引用传递的区别并不是传递的内容，而是实参到底有没有被复制一份给形参。

Object、包装类、Math、Date

[彻底搞懂为什么重写equals还要重写hashcode？\_初心JAVA的博客-CSDN博客\_为什么重写equals还要重写hashcode](https://blog.csdn.net/xl_1803/article/details/111941059?ops_request_misc=%257B%2522request%255Fid%2522%253A%2522163678607416780269851533%2522%252C%2522scm%2522%253A%252220140713.130102334..%2522%257D&request_id=163678607416780269851533&biz_id=0&utm_medium=distribute.pc_search_result.none-task-blog-2~blog~top_click~default-1-111941059.pc_v2_rank_blog_default&utm_term=%E4%B8%BA%E4%BB%80%E4%B9%88%E9%87%8D%E5%86%99equals%E8%BF%98%E8%A6%81&spm=1018.2226.3001.4450)，

Object1是所有类的顶级父类。需要牢记Object中三个方法hashCode()、equals()、toString()的默认实现方式，后续只要遇到与Object中默认的实现方式不同，说明发生了重写，重写效果视具体情况各有不同。

[Thread.State (Java SE 11 & JDK 11 ) (runoob.com)](https://www.runoob.com/manual/jdk11api/java.base/java/lang/Thread.State.html)，

Thread.sleep()，Object.wait()。wait后进入等待锁定池，当针对此对象发出notify/notifyAll方法后获得对象锁进入Runnable状态。

在Java中，hashCode代表对象的特征。

散列表存储的是键值对，拿着key能快速的检索到对应的value。因此利用到散列码可以快速找到所需要的对象。

哈希码并不是完全唯一的，它是一种算法，让同一个类的对象按照自己不同的特征尽量的有不同的哈希码，但不表示不同的对象哈希码完全不同。也有相同的情况，看程序员如何写哈希码的算法。（通过对象唯一的地址值求对应的哈希值即哈希算法。）

当你把对象加入HashSet时，HashSet会先计算对象的hashcode值来判断对象加入的位置，同时也会与其他已经加入的对象的hashcode值作比较，如果没有相符的hashcode，HashSet会假设对象没有重复出现。但是如果发现有相同hashcode值的对象，这时会调用equals()方法来检查hashcode相等的对象是否真的相同。如果两者相同，HashSet就不会让其加入操作成功。如果不同的话，就会重新散列到其他位置。这样我们就大大减少了equals的次数，相应就大大提高了执行速度。

==是运算符，用于比较基础类型变量和引用类型变量。对于基础类型变量，比较变量保存的值是否相同，类型不一定要相同；对于引用类型变量，比较对象的地址值是否相同。

String重写了equals，equals底层使用!=比较字符串的内容，而不是地址值。

equals是Object类中定义的方法，通常用于比较两个对象的值是否相等。

equals未被重写等价于==，equals被重写后比较两者的内容。（一般都会重写的。）

==是不会变的，但是equals()作为方法可能被重写，比如String类就重写了该方法，比较的是字符串的内容，而不是地址值。

当有a.equals(b)==true时，则a.hashCode()==b.hashCode()必然成立；

反过来，当a.hashCode()==b.hashCode()时，a.equals(b)不一定为true。

equals()、hashCode()为什么要同时重写？

如果我们重写了equals()，那么必须重写hashCode()，使其满足以下规定：

1)equals相等，hashcode一定相等。（假设都未重写，都与地址值相关。）

2)equals不等，hashcode不一定不等。

3)hashcode相等，equals不一定相等。

4)hashcode不等，equals一定不等。

之所以有规定，是为了使诸如HashMap这样的哈希表正常使用。

包装类1共有8种，Byte、Short、Integer、Long、Float、Double、Character、Boolean。分别对应8种基本数据类型，整型（byte、short、int、long）、浮点型（float、double）、字符型char、布尔型boolean，包装类包装类创建对象的方式就跟其他类一样。

只要Integer类第一次被使用到，Integer的静态内部类就被加载，加载的时候会创建-128到127的Integer对象，同时创建一个数组cache来缓存这些对象。当使用valueOf()方法创建对象时，就直接返回已经缓存的对象，也就是说不会再新建对象；当使用new关键字或使用valueOf()方法创建小于-128大于127的值对象时，就会创建新对象。

Integer重写了equals()，比较的是对象的值，而不是像==一样比较的是对象是否是同一个对象。所以，当需要比较两个Integer对象的值是否相等时，尽量使用equals()。用==比较的话由于缓存机制的存在，可能产生一些让人困扰的结果。（以后一遇到==就要提高警惕了。）

此外，在8种包装类型中，有缓存区的有Character、Byte、Short、Integer、Long，而且它们的实现方式基本一样，都是-128到127的缓存范围。Boolean虽然没有缓存区，但是因为只有两个值true、false，所以Boolean在成员变量中就创建了两个相应的对象。没有缓存区的只有Float、Double，之所以没有原因很简单，即便是0到1这么小的范围，浮点数也有无数个，使用缓存区缓存它们不具备可能性和实用性。

JVM中一个字节以下的整型数据会在JVM启动的时候加载进内存，除非用newInteger()显式的创建对象，否则都是同一个对象。

只要涉及到基本数据类型，比较的都是值。

[数学相关类.png](file:///D:\Pkg\图片\java\数学相关类.png)，

[RoundingMode (Java SE 11 & JDK 11 ) (runoob.com)](https://www.runoob.com/manual/jdk11api/java.base/java/math/RoundingMode.html)，

Math1

Math.round()就是最接近数学常识的四舍五入方式，等价于BigDecimal.ROUND\_HALF\_UP。

Math.round(11.5)返回12，Math.round(-11.5)返回-11。四舍五入的原理是在参数上加 0.5然后进行下取整。

Math.floor()就是负无穷方向离参数最近的那一个。

如何创建一个指定的日期对象？

[SimpleDateFormat模式字母.png](file:///D:\Pkg\图片\java\SimpleDateFormat模式字母.png)，

Date1

String、正则表达式

String类常用方法？

String1类继承并重写了Object的equals()、hashCode()、toString()，全部都围绕字符串的具体内容。

hashCode()返回对象对应的哈希码值。默认根据对象的地址值生成对应的哈希码值。

equals()比较当前对象与参数对象是否相等。默认根据地址值比较当前对象与参数对象是否相等。

toString()返回对象对应的字符串。默认返回对象的名字(包名.类名)@十六进制的哈希码值。

String是一个final修饰的类，底层维护的是一个final修饰的数组，被创建后不能修改，任何对String的修改都会引发新的String对象的生成。

StringBuffer绝大部分方法使用了synchronized加同步锁，所以是线程安全的。

StringBuilder并没有对方法进行加同步锁，所以是非线程安全的。

操作少量的数据用String。

单线程操作字符串缓冲区、操作大量数据用StringBuilder。

多线程操作字符串缓冲区、操作大量数据用StringBuffer。

字符串字面量拼接操作在编译期间就执行了。

[最全常用正则表达式大全\_ZhaoYingChao88的博客-CSDN博客\_正则表达式](https://blog.csdn.net/ZYC88888/article/details/98479629?ops_request_misc=%257B%2522request%255Fid%2522%253A%2522164056529216780357270404%2522%252C%2522scm%2522%253A%252220140713.130102334..%2522%257D&request_id=164056529216780357270404&biz_id=0&utm_medium=distribute.pc_search_result.none-task-blog-2~blog~top_positive~default-2-98479629.nonecase&utm_term=%E6%AD%A3%E5%88%99%E8%A1%A8%E8%BE%BE%E5%BC%8F&spm=1018.2226.3001.4450)，

[正则表达式全解析+常用示例\_墨遥的博客-CSDN博客\_正则表达式解析](https://blog.csdn.net/xuemoyao/article/details/8033138?ops_request_misc=%257B%2522request%255Fid%2522%253A%2522164056529216780357270404%2522%252C%2522scm%2522%253A%252220140713.130102334..%2522%257D&request_id=164056529216780357270404&biz_id=0&utm_medium=distribute.pc_search_result.none-task-blog-2~blog~top_positive~default-1-8033138.nonecase&utm_term=%E6%AD%A3%E5%88%99%E8%A1%A8%E8%BE%BE%E5%BC%8F&spm=1018.2226.3001.4450)，

正则表达式1是对字符串操作的一种逻辑公式，就是用事先定义好的一些特定字符、及这些特定字符的组合，组成一个“规则字符串”，这个“规则字符串”用来表达对字符串的一种过滤逻辑。

许多程序设计语言都支持利用正则表达式进行字符串操作。例如，在Perl中就内建了一个功能强大的正则表达式引擎。

^放在中括号外面，以…开头；^放在中括号里面是排除的意思。

$以…结尾，\*任意多次，+至少1次，（）组，[]集合。

删除当前类所有的注释，(/\\*([^\*]|[\r\n]|(\\*+([^\*/]|[\r\n])))\*\\*+/|[ \t]\*//.\*)。

判断图片格式，^.+\\.(jpg|png|gif)$。

关于身份证号，以下正确的正则表达式为？AC

A)isIDCard=/^[1-9]\d{7}((0\d)|(1[0-2]))(([0|1|2]\d)|3[0-1])\d{3}$/;

B)isIDCard=/^[1-9]\d{7}((9\d)|(1[0-2]))(([0|1|2]\d)|3[9-1])\d{3}$/;

C)isIDCard=/^[1-9]\d{5}[1-9]\d{3}((0\d)|(1[0-2]))(([0|1|2]\d)|3[0-1])\d{4}$/;

D)isIDCard=/^[1-9]\d{5}[1-9]\d{3}((9\d)|(1[9-2]))(([0|1|2]\d)|3[9-1])\d{4}$/;

只能输入零和非零开头的数字，正确的正则表达式是？A

A)^(0|[1-9][0-9]\*)$

B)^(0|[1-9][1-9]\*)$

C)^(0|[1-9][0-9])$

D)^\+[1-9][0-9]\*$

接口、抽象类、内部类

[新手小白学JAVA\_接口元素更新\_程序媛 泡泡的博客-CSDN博客](https://blog.csdn.net/weixin_43884234/article/details/120985404?ops_request_misc=%257B%2522request%255Fid%2522%253A%2522163742048116780269864945%2522%252C%2522scm%2522%253A%252220140713.130102334.pc%255Fblog.%2522%257D&request_id=163742048116780269864945&biz_id=0&utm_medium=distribute.pc_search_result.none-task-blog-2~blog~first_rank_v2~rank_v29-1-120985404.pc_v2_rank_blog_default&utm_term=%E6%96%B0%E6%89%8B%E5%B0%8F%E7%99%BD%E5%AD%A6JAVA_%E6%8E%A5%E5%8F%A3%E5%85%83%E7%B4%A0%E6%9B%B4%E6%96%B0&spm=1018.2226.3001.4450)，

接口1的设计目的，是对类的行为进行约束，也就是提供一种机制，可以强制要求不同的类具有相同的行为。

Spring松耦合的一个关键点就是面向接口开发，面向接口编程解决了属性与方法的耦合。

Java中提倡面向接口开发，而最经典的接口设计莫过于JDBC数据库接口。

接口内的方法自动拼接public abstract，为常量自动拼接public static final。

可以从实现多线程的两个方案来对比接口和抽象类，但注意Thread只是个普通类。

[关于抽象类与最终类，下列说法错误的是？牛客网](https://www.nowcoder.com/question/next?pid=35200657&qid=62848&tid=50943467)，

抽象类1的设计目的，是代码复用。当不同的类具有某些相同的行为，且其中一部分行为的实现方式一致时，可以让这些类都派生于一个抽象类。

抽象类是用来捕捉子类的通用特性的。接口是抽象方法的集合。

从设计层面来说，抽象类是对类的抽象，是一种模板设计，接口是行为的抽象，是一种行为的规范。

抽象类可以有自己的构造方法，但是不能直接通过new进行实例化。（抽象类可以一个抽象方法都没有，它就是不想被轻易实例化，如实现Servlet的HttpServlet。）

抽象方法要被实现，所以不能是静态static的，也不能是私有private的，也不能被final修饰（试想一下，静态方法可以被类名直接调用，而类名直接调用一个没有实现的抽象方法没有意义）。

抽象类和接口中都可以包含静态成员常量。

抽象类中可以有普通成员变量，接口中没有普通成员变量。

抽象类可以有成员变量和成员常量，接口只能有静态常量。

抽象类中可以包含静态方法，接口中不能包含静态方法。（JDK8接口元素更新了默认方法default和静态方法static。）

构造方法和静态方法都不能被继承。

正确描述：abstract可以修饰类、接口、方法。

[代码执行的先后顺序为？\_阿里巴巴笔试题\_牛客网 (nowcoder.com)](https://www.nowcoder.com/questionTerminal/b15b6a4450814167b8cc2b84392f6c2f)，

内部类1是类的一种定义方式，是嵌套类的一个分支，即Non-StaticNestedClass。内部类分为成员内部类、局部内部类和匿名内部类三种。

普通内部类对象依赖外部类对象而存在。（应该是指非静态内部类。）

创建一个类的静态内部类对象不需要依赖其外部类对象。但是与此同时，静态内部类中也无法访问外部类的非静态成员，因为外部类的非静态成员是属于每一个外部类对象的，而本身静态内部类就是独立外部类对象存在的，所以静态内部类不能访问外部类的非静态成员，而外部类依然可以访问静态内部类对象的所有访问权限的成员，这一点和普通内部类无异。

外部类无法访问匿名内部类和局部内部类对象的字段，因为外部类根本就不知道匿名内部类 /局部内部类的类型信息（匿名内部类的类名被隐匿，局部内部类只能在定义域内使用）。

静态内部类才可以声明静态方法，静态方法不可以使用非静态变量。

成员变量什么修饰符都可以有，局部变量什么修饰符都不能有。

除了没有名字，匿名内部类还有以下特点：

1)匿名内部类必须继承一个抽象类或者实现一个接口。

2)匿名内部类不能定义任何静态成员和静态方法。

3)当所在的方法的形参需要被匿名内部类使用时，必须声明为 final。

4)匿名内部类不能是抽象的，它必须要实现继承的类或者实现的接口的所有抽象方法。

我们为什么要使用内部类呢？因为它有以下优点：

1)一个内部类对象可以访问创建它的外部类对象的内容，包括私有数据！

2)内部类不为同一包的其他类所见，具有很好的封装性；

3)内部类有效实现了“多重继承”，优化 Java单继承的缺陷。

4)匿名内部类可以很方便的定义回调。

内部类有哪些应用场景？

1)一些多算法场合。

2)解决一些非面向对象的语句块。

3)适当使用内部类，使得代码更加灵活和富有扩展性。

4)当某个类除了它的外部类，不再被其他的类使用时。

内部类（成员内部类）可以有4种访问权限？

正确，可以把内部类理解成类的成员，成员有4种访问权限，内部类也是。

但如果访问权限为private，无法直接在外部创建其对象，可提供getInnerDelete方法供外部使用。

泛型、多态、static

泛型1可以保证如果程序在编译时没有发出警告，运行时就不会产生ClassCastException异常。同时，代码更加简洁、健壮。

泛型概念非常重要，它是程序的增强器，它是目前主流的开发方式。

泛型是一颗“语法糖”，它只在编译时生效，用以检查集合的元素类型，编译通过后，泛型便失去意义，编译后生成的字节码文件中没有泛型。

泛型，参数化类型。重载，参数化方法。

重载只能定义几种有限的类型（包括基本类型），而泛型理论上可以定义无数种引用类型。

增强for循环主要就用于集合遍历，它写法简单，比for循环效率高，但不够灵活（只能从头到尾的遍历数据，不能进行步长的选择）。

泛型是JDK5的一个新特性，通常与集合对象一起使用。

[【深入Java虚拟机】之五：多态性实现机制——静态分派与动态分派\_兰亭风雨的博客-CSDN博客](https://blog.csdn.net/ns_code/article/details/17965867?ops_request_misc=&request_id=&biz_id=102&utm_term=%E9%9D%99%E6%80%81%E5%88%86%E6%B4%BE&utm_medium=distribute.pc_search_result.none-task-blog-2~all~sobaiduweb~default-2-17965867.first_rank_v2_pc_rank_v29&spm=1018.2226.3001.4187)，

[Java对象类型向上转型和向下转型\_我思故你不在的博客-CSDN博客\_向下转型](https://blog.csdn.net/yuncaidaishu/article/details/88690799?ops_request_misc=%257B%2522request%255Fid%2522%253A%2522163638250016780264029875%2522%252C%2522scm%2522%253A%252220140713.130102334.pc%255Fall.%2522%257D&request_id=163638250016780264029875&biz_id=0&utm_medium=distribute.pc_search_result.none-task-blog-2~all~first_rank_ecpm_v1~rank_v31_ecpm-2-88690799.pc_search_result_cache&utm_term=%E5%90%91%E4%B8%8A%E8%BD%AC%E5%9E%8B&spm=1018.2226.3001.4187)，

多态1就是指静态代码总表现出的不确定性，只有在程序运行期间才能显现出确定性。（比如引用变量的指向不确定，方法的调用者不确定。）

多态体现为父类引用指向子类对象，如：Animal a = new Cat();

向上转型可以理解为就是多态，将子类对象看做父类类型，父类引用指向子类对象。

多态对象把自己看做是父类类型，重写过的方法就用重写的，其他都用父类的，静态资源由于最先加载，所以谁调用就返回谁的。

多态的表现形式，普通类多态，抽象类多态，接口多态。

又说是重载、重写、抽象类和接口。

又说是继承（多个子类对同一方法的重写）和接口（实现接口并覆盖接口中同一方法）。

但要注意继承+重写构成了多态的前提。（针对类多态。）

重载实现的是编译时多态（也称为前绑定），而重写实现的是运行时多态性（也称为后绑定）。

调用重写方法看右边，调用独有方法看左边，入参看左边。

class A {

public int func1(int a, int b) {

return a - b;

}

}

class B extends A {

public int func1(int a, int b) {

return a + b;

}

}

public class ChildClass {

public static void main(String[] args) {

A a = new B();

B b = new B();

System.out.println("Result=" + a.func1(100, 50));

System.out.println("Result=" + b.func1(100, 50));

}

}

static1修饰的资源统称为静态资源，可以用来修饰变量、方法、代码块、内部类。

static的主要意义是在于创建独立于具体对象的域变量或者方法。以致于即使没有创建对象，也能使用属性和调用方法。

一个类的对象能不能调用静态资源？可以，但没必要。

静态资源属于类资源，随着类的加载而加载，优先于对象进行加载，只加载一次。

静态资源可以不通过对象，使用类名直接调用（static方法不参与继承），不需要创建对象。

静态资源只有一份，被全局所有对象共享。

在同一个类里，静态资源只能调用静态资源。

静态资源是优先于对象的，所以静态资源不能与this和super共用。

静态方法不存在重写的现象，在哪个类定义，就属于哪个类的资源。

构造方法和静态方法都不能被继承。

抽象类中可以包含静态方法，接口中不能包含静态方法。（JDK8接口元素更新了默认方法default和静态方法static。）

public class B {

public static B t1 = new B();

public static B t2 = new B();

{

System.out.println("构造块");

}

static {

System.out.println("静态块");

}

public static void main(String[] args) {

B t = new B();

}

}

静态属性→静态代码块→构造代码块→构造方法→普通方法→局部代码块。（注意构造代码块先于构造方法执行。）

静态代码块一般用来加载那些只需要加载一次并且第一时间就需要加载资源。

静态代码块中变量均为局部变量，不影响类的同名静态成员的值。（但局部变量最好不要和成员变量同名。）

构造代码块一般用于提取所有构造方法的共性功能。

局部代码块只有当其所处的方法被调用时才会执行，主要用于限制变量的作用范围。

static变量不管加没加transient都不可以被序列化。

synchronized修饰静态方法使用的对象锁默认为当前类的字节码对象。

synchronized修饰普通方法时使用的对像锁默认是调用者的字节码对象。

多线程、线程安全

[Java多线程学习（吐血超详细总结）\_Evankaka的博客-CSDN博客\_java 多线程](https://blog.csdn.net/Evankaka/article/details/44153709?ops_request_misc=%257B%2522request%255Fid%2522%253A%2522163741453716780265492568%2522%252C%2522scm%2522%253A%252220140713.130102334..%2522%257D&request_id=163741453716780265492568&biz_id=0&utm_medium=distribute.pc_search_result.none-task-blog-2~blog~top_positive~default-1-44153709.pc_v2_rank_blog_default&utm_term=%E5%A4%9A%E7%BA%BF%E7%A8%8B&spm=1018.2226.3001.4450)，

[进程与线程\_程序媛 泡泡的博客-CSDN博客\_进程的特点](https://blog.csdn.net/weixin_43884234/article/details/114954221?ops_request_misc=%257B%2522request%255Fid%2522%253A%2522166342645116800182123502%2522%252C%2522scm%2522%253A%252220140713.130102334.pc%255Fblog.%2522%257D&request_id=166342645116800182123502&biz_id=0&utm_medium=distribute.pc_search_result.none-task-blog-2~blog~first_rank_ecpm_v1~rank_v31_ecpm-1-114954221-null-null.nonecase&utm_term=%E5%A4%9A%E7%BA%BF%E7%A8%8B&spm=1018.2226.3001.4450)，

[并发编程的优缺点\_ThinkWon的博客-CSDN博客](https://blog.csdn.net/ThinkWon/article/details/102020811?ops_request_misc=%257B%2522request%255Fid%2522%253A%2522166342830716800180679641%2522%252C%2522scm%2522%253A%252220140713.130102334.pc%255Fblog.%2522%257D&request_id=166342830716800180679641&biz_id=0&utm_medium=distribute.pc_search_result.none-task-blog-2~blog~first_rank_ecpm_v1~rank_v31_ecpm-2-102020811-null-null.nonecase&utm_term=%E5%A4%9A%E7%BA%BF%E7%A8%8B&spm=1018.2226.3001.4450)，

[Java线程基础进阶\_雨田说码的博客-CSDN博客](https://blog.csdn.net/maitian_2008/article/details/118582926?spm=1001.2014.3001.5501)，

[『图解Java并发编程系列』10张图告诉你Java并发多线程那些破事\_爱笑的架构师的博客-CSDN博客\_图解java并发](https://blog.csdn.net/guoguo527/article/details/116243171?ops_request_misc=%257B%2522request%255Fid%2522%253A%2522163765843716780271557816%2522%252C%2522scm%2522%253A%252220140713.130102334.pc%255Fblog.%2522%257D&request_id=163765843716780271557816&biz_id=0&utm_medium=distribute.pc_search_result.none-task-blog-2~blog~first_rank_v2~rank_v29-1-116243171.pc_v2_rank_blog_default&utm_term=java%E5%B9%B6%E5%8F%91%E7%BC%96%E7%A8%8B&spm=1018.2226.3001.4450)，

[并发编程面试题（2020最新版）\_ThinkWon的博客-CSDN博客\_并发编程面试题](https://blog.csdn.net/ThinkWon/article/details/104863992?ops_request_misc=%257B%2522request%255Fid%2522%253A%2522166342830716800180679641%2522%252C%2522scm%2522%253A%252220140713.130102334.pc%255Fblog.%2522%257D&request_id=166342830716800180679641&biz_id=0&utm_medium=distribute.pc_search_result.none-task-blog-2~blog~first_rank_ecpm_v1~rank_v31_ecpm-4-104863992-null-null.nonecase&utm_term=%E5%A4%9A%E7%BA%BF%E7%A8%8B&spm=1018.2226.3001.4450)，

[以下哪几种方式可用来实现线程间通知和唤醒？\_阿里巴巴笔试题\_牛客网 (nowcoder.com)](https://www.nowcoder.com/questionTerminal/c6126fd5ba2d40b0bfca35439a6c3563)，

[下面哪个行为被打断不会导致InterruptedExcept\_\_牛客网 (nowcoder.com)](https://www.nowcoder.com/questionTerminal/0e426707c41343c9a7707d5c835aa474)，

[Thread.State (Java SE 11 & JDK 11 ) (runoob.com)](https://www.runoob.com/manual/jdk11api/java.base/java/lang/Thread.State.html)，

[多线程01：概述\_哔哩哔哩\_bilibili](https://www.bilibili.com/video/BV1V4411p7EF?p=1)，

[线程状态转换.png](file:///D:\Pkg\图片\Java\线程状态转换.png)

多线程1的核心就是抢占（异步），卖得快的多卖，卖得慢的少卖。多线程和异步并不是一个同等关系，异步是最终目的，多线程只是我们实现异步的一种手段。（MapReduce、MyBatis二级缓存有用到多线程。）

同一类线程共享代码和数据空间，每个线程有独立的运行栈和程序计数器（PC）。进程间的切换会有较大的开销，线程切换开销小。

创建线程后只有调用start()才会启用多线程，手动调用run()始终只有一个线程。

在Java中，高优先级的可运行的线程会抢占低优先级线程的资源。

一个线程调用yield()方法，可以使具有相同优先级线程获得处理器。

调用Thread的sleep()方法不会释放锁，调用wait()方法会释放锁。

子线程调用了join()方法后面的（主线程）代码，只有等到子线程结束了才能执行。

为什么要用join()方法？

在很多情况下，主线程生成并起动了子线程，如果子线程里要进行大量的耗时的运算，主线程往往将于子线程之前结束，但是如果主线程处理完其他的事务后，需要用到子线程的处理结果，也就是主线程需要等待子线程执行完成之后再结束，这个时候就要用到join()方法了。（子线程在主线程前面插个队。）

错误描述：当一个线程因为抢先机制而停止运行，它被放在可运行队列的前面。（因为抢先机制而停止运行，说明该线程的优先级比较低，不可能排到可运行队列的前面。）

[并发和并行，异步与多线程区别\_桀骜秋波的博客-CSDN博客\_并行 异步](https://blog.csdn.net/qiuyubo1/article/details/80288525)，

线程安全1

高并发意味着同时开启了多个进程，而并行是指多个线程同时进行。后者才是真正意义上的同一时刻做多件事情。

判断程序是否有可能出现线程安全问题，主要根据以下三个条件：在多线程程序中 + 有共享数据 + 多条语句操作共享数据。

在高并发场景下保证项目的高可用（不崩溃或能迅速恢复）。

原子性：在一次或者多次操作时，要么所有操作都被执行，要么所有操作都不执行。

可见性：当一个线程对共享变量进行修改后，另外一个线程可以立即看到该变量修改后的最新值。

有序性：程序执行的顺序按照代码的先后顺序执行。（synchronized能保证三大性。）

同步体现了排队的效果，同一时刻只有一个线程操作资源，安全但效率低下。

异步体现了多线程抢占资源的效果，遵循先到先得的原则，高效但不安全。

简单记忆线程安全的集合类：喂！S.H.E！

喂是指Vector，S是指Stack，H是指Hashtable，E是指Eenumeration。

Hashtable是线程安全的哈希表，它是通过synchronized来保证线程安全的；即，多线程通过同一个“对象的同步锁”来实现并发控制。

ConcurrentHashMap并发容器，JDK7采用分段锁，JDK8采用CAS算法，线程安全，建议使用。

Hashtable在线程竞争激烈时，效率比较低（此时建议使用ConcurrentHashMap）。当一个线程访问Hashtable的同步方法时，其它线程如果也在访问Hashtable的同步方法时，可能会进入阻塞状态。

synchronized、volatile、Lock

synchronized1给被修饰的对象加上互斥锁，在同一时刻，该对象只能被一个线程访问。

synchronized修饰静态方法使用的对象锁默认为当前类的字节码对象。

synchronized修饰普通方法时使用的对像锁默认是调用者的字节码对象。

实现多线程的方式如果是extends Thread，锁对象须static，如果是implements Runnable，则不做限制。

不论多线程的实现方式，只要锁对象唯一即可，类型不做限制。类名.class取字节码对象来锁，两种实现都能锁住。

字节码对象极其重要，是reflection的基石，字节码对象中包含了当前类所有的关键信息。

每个对象只有一个锁与之相关联。

实现同步是要很大的系统开销作为代价的，甚至可能造成死锁，所以尽量避免无谓的同步控制。

乐观锁总是假设最好的情况，每次去拿数据的时候都认为别人不会修改，所以不会上锁。适用于多读的应用场景，这样可以提高吞吐量。

悲观锁总是假设最坏的情况，每次去拿数据的时候都认为别人会修改，所以每次在拿数据的时候都会上锁。适用于多写的应用场景。

传统的关系型数据库里边就用到了很多这种锁机制，比如行锁，表锁等，读锁，写锁等，都是在做操作之前先上锁。Java中synchronized和ReentrantLock等独占锁就是悲观锁思想的实现。

volatile1

为了解决多线程可见性问题，Java提供了volatile关键字。当一个共享变量被volatile修饰时，它会保证修改的值会立即被更新到主存，当有其他线程需要读取时，它会去内存中读取新值。而普通共享变量不能保证可见性，因为变量被修改后什么时候刷回到主存是不确定的，另外一个线程读的可能就是旧值。

在Java中为了提高程序的运行效率，可能在编译期和运行期会对代码指令进行一定的优化，不会百分之百的保证代码的执行顺序严格按照编写代码中的顺序执行，但也不是随意进行重排序，它会保证程序的最终运算结果是编码时所期望的。这种情况被称之为指令重排（Instruction Reordering）。

Java Memory Model的一个叫做先行发生（Happens-Before）的原则能够提供天然的有序性，除此之外，使用synchronized、volatile以及Lock相关的工具类也能保证有序性。

关键字volatile是线程同步的轻量级实现，所以volatile性能肯定比synchronized要好，并且只能修饰变量，而synchronized可以修饰方法，以及代码块。

多线程访问volatile不会发生阻塞，而synchronized会出现阻塞。（因为volatile不控制共享数据。）

volatile做了什么？禁止了指令重排。

volatile解决的是变量在多线程之间的可见性，而synchronized解决的是多线程之间资源同步问题。（synchronized能保证三大性，volatile保证可见性、有序性。）

volatile能保证数据的可见性，但不能保证原子性；而synchronized可以保证原子性，也可以间接保证可见性，因为它会将私有内存和公共内存中的数据做同步。

Lock1

在Java语言中通过可以使用synchronize或者Lock来保证原子性。

Lock一般使用ReentrantLock类做为锁。在加锁和解锁处需要通过lock()和unlock()显示指出。所以一般会在finally块中写unlock()以防死锁。

Lock底层是CAS乐观锁，依赖AbstractQueuedSynchronizer类，把所有的请求线程构成一个CLH队列。而对该队列的操作均通过Lock-Free（CAS）操作。

Lock lock = new ReentrantLock();

lock.lock();// 若将锁的获取放在try语句块中，当发生异常、抛出异常的同时，可能导致锁的无故释放

try {

}finally {

lock.unlock(); // 放在finally代码块中，保证锁一定会被释放

}

synchronized是关键字，Lock是接口。

synchronized是隐式的加锁，lock是显式的加锁。

synchronized在线程发生异常时会自动释放锁，因此不会发生异常死锁。Lock异常时不会自动释放锁，所以需要在finally中实现释放锁。

synchronized是阻塞式加锁，必须等待线程执行完成释放锁。Lock是非阻塞式加锁，支持可中断式加锁，支持带超时时间的加锁。（tryLock(long time, TimeUnit unit) // 带超时时间。）

synchronized只支持非公平锁，Lock支持非公平锁和公平锁。

synchronized使用了Object类的wait和notify进行等待和唤醒，Lock使用了Condition接口进行等待await和唤醒signal。

Lock可以使用读锁提高多线程读效率。（乐观锁）

[反射的本质理解.png](file:///D:\Pkg\图片\java\反射的本质理解.png)，[Java基础之——反射（非常重要）](https://blog.csdn.net/sinat_38259539/article/details/71799078?ops_request_misc=%257B%2522request%255Fid%2522%253A%2522163741964516780265416622%2522%252C%2522scm%2522%253A%252220140713.130102334..%2522%257D&request_id=163741964516780265416622&biz_id=0&utm_medium=distribute.pc_search_result.none-task-blog-2~blog~top_positive~default-1-71799078.pc_v2_rank_blog_default&utm_term=%E5%8F%8D%E5%B0%84&spm=1018.2226.3001.4450)，

反射、ClassLoader

反射1是指在运行状态中，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；这种动态获取对象信息以及动态调用对象方法的功能称为Java语言的反射机制。

反射类似于反编译，类是由他人创建的，如果你想调用这个类的底层，就需要靠反射技术来实现。

静态编译在编译时确定类型，绑定对象。动态编译运行时确定类型，绑定对象。

使用无参构造反射，a.newInstance()。

使用含参构造反射，a.getConstructor()获取对应的构造函数对象cs，再通过cs创建目标类对象cs.newInstances()。

Spring容器利用反射技术创建对象一定会调用类的无参构造，所以给类添加构造方法时首先添加无参构造。

每个类在加载（将类读到内存）时都会创建一个字节码对象，其类型为Class，且这个对象在一个JVM内存中是唯一的。此对象中存储的是类的结构信息（元数据信息）。

获取类的字节码对象：

1)类名.class

2)Class.forName(“包名.类名”)（JDBC获取驱动就用这个。）

3)类的实例对象.getClass()。

[关于Java中的ClassLoader下面的哪些描述是错误的？牛客网](https://www.nowcoder.com/test/question/done?tid=50939050&qid=373097#summary)，

ClassLoader1

A class loader is an object that is responsible for loading classes. The class ClassLoader is an abstract class. Given the binary name of a class, a class loader should attempt to locate or generate data that constitutes a definition for the class. A typical strategy is to transform the name into a file name and then read a "class file" of that name from a file system.（JDK动态代理有用到ClassLoader。）

[Java Hotspot Architecture.png](file:///D:\Pkg\图片\Java-Hotspot-Architecture.png)，[Hotspot内存结构.png](file:///D:\Pkg\图片\Hotspot内存结构.png)，

[虚拟机的内存.png](file:///D:\Pkg\图片\虚拟机的内存.png)，[JVM规范.png](file:///D:\Pkg\图片\JVM规范.png)，[jdk-jre.png](file:///D:\Pkg\图片\jdk-jre.png)，

[Java虚拟机(JVM)面试题(2020最新版)](https://thinkwon.blog.csdn.net/article/details/104390752?spm=1001.2014.3001.5502)，

[详解JVM常量池、Class常量池、运行时常量池、字符串常量池(心血总结)](https://blog.csdn.net/qq_45737068/article/details/107149922?ops_request_misc=%257B%2522request%255Fid%2522%253A%2522163711821216780274158801%2522%252C%2522scm%2522%253A%252220140713.130102334..%2522%257D&request_id=163711821216780274158801&biz_id=0&utm_medium=distribute.pc_search_result.none-task-blog-2~all~top_positive~default-1-107149922.first_rank_v2_pc_rank_v29&utm_term=%E5%B8%B8%E9%87%8F%E6%B1%A0&spm=1018.2226.3001.4187)，

[对于不再使用的内存资源，如调用完成的方法，“垃圾回收器”会自动将其释放？牛客网](https://www.nowcoder.com/test/question/done?tid=50939401&qid=69657#summary)，

[下面有关JVM内存，说法错误的是？牛客网](https://www.nowcoder.com/test/question/done?tid=50939401&qid=14817#summary)，

[对于JVM内存配置参数xxx，其最小内存值和Survivor区总大小分别是？牛客网](https://www.nowcoder.com/test/question/done?tid=47482566&qid=16021#summary)，

JVM、Java内存、常量池、GC

JVM1是一种用于计算设备的规范，它是一个虚构出来的计算机，是通过在实际的计算机上仿真模拟各种计算机功能来实现的。

JVM有自己完善的硬件架构，如处理器、堆栈等，还具有相应的指令系统。JVM本质上就是一个程序，当它在命令行上启动的时候，就开始执行保存在某字节码文件中的指令。

在JVM规范中，分别用子系统、内存区、数据类型以及指令这几个术语来描述。这些组成部分一起展示出一个抽象化的虚拟机内部的抽象体系结构。

JVM主要分为五大模块，类加载子系统、运行时数据区、执行引擎、本地方法接口和垃圾收集模块。

1)类加载系统ClassLoader System，负责加载类到内存。

2)运行时数据区Runtime Data Area，负责存储数据信息。

3)执行引擎Execution Engine，负责调用对象执行业务。

4)本地方法接口Native Interface，负责与本地库进行交互。

5)垃圾收集器Garbage Collection，自动回收不再使用的对象。

JVM不是真实的物理机，它没有寄存器，所以指令集是使用Java栈来存储中间数据，这样做的目的就是为了保持JVM的指令集尽量的紧凑，同时也便于JVM在那些只有很少通用寄存器的平台上实现。另外，JVM的这种基于栈的体系结构，有助于运行时某些虚拟机实现的动态编译器和即时编译器的代码优化。

深入理解JVM可以帮助我们从平台角度提高解决问题的能力：

1)有效防止内存泄漏Memory leak。

2)优化线程锁的使Thread Lock。

3)科学进行垃圾回收Garbage Collection。

4)提高系统吞吐量Throughput。

5)降低延迟Delay，提高其性能Performance。

[Java内存分区](https://blog.csdn.net/keybersan/article/details/95455776?ops_request_misc=&request_id=&biz_id=102&utm_term=java%E5%86%85%E5%AD%98%E5%8C%BA%E5%9F%9F&utm_medium=distribute.pc_search_result.none-task-blog-2~blog~sobaiduweb~default-8-95455776.pc_v2_rank_blog_default&spm=1018.2226.3001.4450)，

[下面这条语句一共创建了多少个对象：String s="welcome"+"to"+360？牛客网](https://www.nowcoder.com/test/question/done?tid=50902443&qid=304929#summary)，

[以下哪些表达式返回为真？牛客网](https://www.nowcoder.com/question/next?pid=35213226&qid=516405&tid=50956820)，[以下哪个区域不属于新生代？牛客网](https://www.nowcoder.com/test/question/done?tid=50925363&qid=369702#summary)，

[下列Java代码中的变量a、b、c分别在内存的什么存储区存放？牛客网](https://www.nowcoder.com/question/next?pid=35213949&qid=61216&tid=50958322)，

Java内存分为5大区域，我们重点关注栈和堆。

栈：局部变量，对象的地址。（JVM栈）

堆：对象的内容，字符串常量池。

方法区：静态资源，运行时常量池。

本地方法栈：

程序计数器：

同一类线程共享代码和数据空间，每个线程有独立的运行栈和程序计数器（PC）。

栈1

堆1是所有线程共享的，它的目的是存放对象实例。同时它也是GC所管理的主要区域，因此常被称为GC堆，又由于现在收集器常使用分代算法，Java堆中还可以细分为新生代和老年代。

null代表堆内存中根本没有这个东西，而""是有内存空间的。

可以将对象的引用变量设置为null，暗示垃圾回收机制可以回收该对象。JDBC关闭资源时就采用了这种方式。

堆是JVM所管理的最大的一块内存空间，主要用于存放各种类的实例对象。

JVM将堆划分成两个不同的区域，新生代（Young）、老年代（Old）。新生代（Young）又被划分为三个区域，Eden、From Survivor、To Survivor。

这样划分的目的是为了使JVM能够更好的管理堆内存中的对象，包括内存的分配以及回收。

默认的，Edem : From : To = 8 : 1 : 1，JVM每次只会使用Eden和其中的一块Survivor区域来为对象服务，所以无论什么时候，总是有一块Survivor区域是空闲着的。因此，新生代实际可用的内存空间为90%的新生代空间。在JDK8 Hotspot移除了永久代用元空间（Metaspace）取而代之，这时候字符串常量池还在堆，运行时常量池还在方法区，只不过方法区的实现从永久代变成了元空间（Metaspace）。

字符串常量池（StringPool）是堆中常量池。在工作中，String类是我们使用频率非常高的一种对象类型。JVM为了提升性能和减少内存开销，避免字符串的重复创建，其维护了一块特殊的内存空间，即字符串常量池。字符串常量池由String类私有地维护。

在JDK7之前字符串常量池是在永久代里边的，但是在JDK7之后，把字符串常量池分进了堆里边。（意思是JDK7之后，字符串常量池即现在的堆中常量池和运行时常量池都从永久带即现在的方法区里面移除了。）（享元模式/包括Integer高效范围。）

[堆中常量池/方法区常量池.png](file:///D:\Pkg\图片\堆中常量池-方法区常量池.png)，[Class文件常量池/运行时常量池.png](file:///D:\Pkg\图片\Class文件常量池-运行时常量池.png)，

运行时常量池是方法区的一部分。运行时常量池是当Class文件被加载到内存后，Java虚拟机会将Class文件常量池里的内容转移到运行时常量池里（运行时常量池也是每个类都有一个）。运行时常量池相对于Class文件常量池的另外一个重要特征是具备动态性，Java语言并不要求常量一定只有编译期才能产生，也就是并非预置入Class文件中常量池的内容才能进入方法区运行时常量池，运行期间也可能将新的常量放入池中。

在JDK7之前，方法区也被称为永久代。

不过自从JDK7之后，Hotspot虚拟机便将运行时常量池从永久代移除了。

错误描述：方法区用于存储JVM加载的类信息、常量、静态变量、以及编译器编译后的代码等数据，是线程隔离的。

方法区在JVM中也是一个非常重要的区域，它与堆一样，是被线程共享的区域。 在方法区中，存储了每个类的信息（包括类的名称、方法信息、字段信息）、静态变量、常量以及编译器编译后的代码等。

String str = new String("aaa")创建了几个对象？在上述创建过对象后的基础上再创建String str1 = new String("aaa")一共创建了几个对象？并说明原因。

2个，1个。

采用字面值的方式创建一个字符串时，JVM首先会去字符串池中查找是否存在"aaa"这个对象，如果不存在，则在字符串池中创建"aaa"这个对象，然后将池中"aaa"这个对象的引用地址返回给字符串常量str，这样str会指向池中"aaa"这个字符串对象；如果存在，则不创建任何对象，直接将池中"aaa"这个对象的地址返回，赋给字符串常量。（这道题和字面值就没有半毛钱关系好吗…）（字符串字面量拼接操作在编译期间就执行了。）

采用new关键字新建一个字符串对象时，JVM首先在字符串常量池中查找有没有"aaa"这个字符串对象，如果有，则不在池中再去创建"aaa"这个对象了，直接在堆中创建一个"aaa"字符串对象，然后将堆中的这个"aaa"对象的地址返回赋给引用str1，这样，str1就指向了堆中创建的这个"aaa"字符串对象；如果没有，则首先在字符串常量池中创建一个"aaa"字符串对象1，然后再在堆中创建一个"aaa"字符串对象2，然后将堆中这个"aaa"字符串对象的地址返回赋给str1引用，这样，str1指向了堆中创建的这个"aaa"字符串对象。

字符串字面量拼接操作在编译期间就执行了。

下面这段java代码，当T分别是引用类型和值类型的时候，分别产生了多少个T对象和T类型的值？

T t = new T();(值类型时:T t;)

Func(t);

Func定义如下：

public void Func(T t) { }

A)1 1

B)2 1

C)2 2

D)1 2

D。引用类型作为函数的参数时，复制的是引用的地址，不会产生一个新的T；而如果T是值类型，其作为函数实参时会复制其值，也就是产生了一个新的T。

[GC守护进程.png](file:///D:\Pkg\图片\GC守护进程.png)，

[在发生过一次FullGC后，上述代码在Heap空间保留的字符数为？牛客网](https://www.nowcoder.com/test/question/done?tid=50901924&qid=16020#summary)，

[下面关于垃圾收集的说法正确的是？牛客网](https://www.nowcoder.com/test/question/done?tid=50939401&qid=305690#summary)，

GC（GarbageCollection）指JAVA/.NET中的垃圾回收器。有了GC，程序员就不需要再手动的去控制内存的释放。当JVM或.NETCLR发觉内存资源紧张的时候，就会自动地去清理无用对象（没有被引用到的对象）所占用的内存空间（这里的说法略显粗略，事实上何时清理内存是个复杂的策略）。如果需要，可以在程序中显式地使用System.gc()/System.GC.

Collect()来强制进行一次立即的内存清理。Java的GC功能可以自动监测对象是否超过了作用域，从而达到自动回收内存的目的，Java的GC会自动进行管理，调用方法System.gc()或者Runtime.getRuntime().gc()。

在Java中，每次程序运行至少启动2个线程。一个是main线程，一个是垃圾收集线程。

IO、Stream、序列化

[java基础类型中的char和byte的辨析及Unicode编码和UTF-8的区别 - 翎野君 - 博客园 (cnblogs.com)](https://www.cnblogs.com/lingyejun/p/9743788.html)，

[一个文件中的数据要在控制台上显示，首先需要？牛客网](https://www.nowcoder.com/test/question/done?tid=50901223&qid=70912#summary)，

[在main()方法中给出的字节数组，如果将其显示到控制台上，需要？牛客网](https://www.nowcoder.com/question/next?pid=35202997&qid=305676&tid=50944676)，

我认为字节数组和字节数组地址值是存在区别的吧？

[transient变量和下面哪一项有关？牛客网](https://www.nowcoder.com/test/question/done?tid=50928791&qid=304920#summary)，

[java.io包框架图.png](file:///D:\Pkg\图片\java\java.io包框架图.png)，[流.png](file:///D:\Pkg\图片\java\流.png)，[File类.png](file:///D:\Pkg\图片\java\File类.png)，[史上最骚最全最详细的IO流教程](https://blog.csdn.net/qq_44543508/article/details/102831084?ops_request_misc=%257B%2522request%255Fid%2522%253A%2522163685358116780357275474%2522%252C%2522scm%2522%253A%252220140713.130102334..%2522%257D&request_id=163685358116780357275474&biz_id=0&utm_medium=distribute.pc_search_result.none-task-blog-2~all~top_positive~default-1-102831084.first_rank_v2_pc_rank_v29&utm_term=io%E6%B5%81&spm=1018.2226.3001.4187)，

序列化1是指将结构化对象转化为字节流以便在网络上传输或写到磁盘进行永久存储的过程。反序列化是指将字节流转回结构化对象的逆过程。

缓冲流，也叫高效流，是对4个FileXxx流的“增强流”。

我们在反序列化时，JVM会拿着反序列化流中的serialVersionUID与序列化时相应的实体类中的serialVersionUID来比较，如果不一致，就无法正常反序列化，出现序列化版本不一致的异常InvalidClassException。

我们在定义需要序列化的实体类时，如果没有手动添加UID，Java序列化机制会根据编译的class自动生成一个，那么只有同一次编译生成的class才是一样的UID。

如果我们手动添加了UID，只要这个值不修改，就可以不论编译次数，进行序列化和反序列化操作。

transient简单地说，就是让某些被修饰的成员属性变量不被序列化。当一个变量不希望被持久化的时候，比如说一些账号密码，就可以用transient关键字来表示该变量不参与序列化过程。

标准输入流是InputStream，而标准输出流是PrintStream。（OutputStream抽象类的子孙类。）

[PrintWriter错题.png](file:///D:\Pkg\图片\PrintWriter错题.png)，

字节流：字节为单位，读写数据的流。适合读取视频，声音，图片等二进制文件。

字符流：字符为单位，读写数据的流。适合读取纯文本文件。

Java语言中所有的字节流都以Stream结尾，所有的字符流都含有Reader或Writer。但是序列化和反序列化有两个writeObject和readObject方法。

因为数据编码的不同，因而有了对字符进行高效操作的流对象，字符流本质其实就是基于字节流读取时，去查了指定的码表。

而字节流直接读取数据会有乱码的问题（读中文会乱码，因为一个中文字符占用多个字节存储。）

File类可以代表三个路径：文件，文件夹，不存在的文件。

The System class contains several useful class fields and methods. It cannot be instantiated.

System.exit(int status)--Terminates the currently running Java Virtual Machine.

System.exit(n) is effectively equivalent to Runtime.getRuntime().exit(n).

标准输入流是InputStream，而标准输出流是PrintStream。（OutputStream???）

错误描述：File类能够读写文件。

读写文件是Stream和Writer/Reader的事…

[Java泛型详解](https://blog.csdn.net/s10461/article/details/53941091?ops_request_misc=%257B%2522request%255Fid%2522%253A%2522163712314916780265416945%2522%252C%2522scm%2522%253A%252220140713.130102334..%2522%257D&request_id=163712314916780265416945&biz_id=0&utm_medium=distribute.pc_search_result.none-task-blog-2~all~top_positive~default-1-53941091.first_rank_v2_pc_rank_v29&utm_term=%E6%B3%9B%E5%9E%8B&spm=1018.2226.3001.4187)，[泛型常用名称.png](file:///D:\Pkg\图片\泛型常用名称.png)，

[下面的哪4个语句是正确的？牛客网](https://www.nowcoder.com/test/question/done?tid=50940624&qid=7693#summary)，

数组、Collection、Map

[Java实际没有多维数组，只有一维数组](https://blog.csdn.net/oChangWen/article/details/50587937)，

[java.lang.reflect.Array类提供静态方法来动态创建和访问Java数组](https://blog.csdn.net/tianyeshiye/article/details/89372198?spm=1001.2101.3001.6650.2&utm_medium=distribute.pc_relevant.none-task-blog-2%7Edefault%7EBlogCommendFromBaidu%7Edefault-2.no_search_link&depth_1-utm_source=distribute.pc_relevant.none-task-blog-2%7Edefault%7EBlogCommendFromBaidu%7Edefault-2.no_search_link)，

数组1属于java.lang.reflect包，是三大引用类型之一。二维数组几乎被淘汰了，项目中经常用到的是实现了Collection 接口的那一套。

静态数组是在程序的编译阶段被分配内存，因此，数组的数据类型、数组维数，必须明确地给出。

动态数组是在程序的运行阶段被分配内存，因此，数组的数据类型、数组维数，可在程序运行时给出。

我们通过每个对象自己唯一的地址值来区分不同的对象。

数组名保存的是数组的地址值，不是数组中每一个具体的元素，数组名是一个引用类型的变量。

数组一旦创建，长度不可改变，如果想增删数组元素，只能新建复制原数组。

如果访问到了不存在的下标，会报错数组下标越界异常。

在线性表、栈、队列数据结构实现源码中，涉及到两个与数组复制相关的方法：

Arrays.copyOf()、System.arraycopy()，前者由后者实现。

arraycopy方法会因为新数组大小比久数组大小小而报IndexOutOfBoundsException；copyOf则不会因此报错，因为copyOf的返回值是在内部new好的copy数组，而该copy数组new的大小就等于newLength，故即使在客户端指定好了新数组newArray的大小，接收到返回值后也是指向底层new出来的数组copy。

[一文彻底搞定哈希表](https://blog.csdn.net/sinat_33921105/article/details/103344078?ops_request_misc=%257B%2522request%255Fid%2522%253A%2522163688427716780261999515%2522%252C%2522scm%2522%253A%252220140713.130102334..%2522%257D&request_id=163688427716780261999515&biz_id=0&utm_medium=distribute.pc_search_result.none-task-blog-2~all~top_positive~default-1-103344078.first_rank_v2_pc_rank_v29&utm_term=%E5%93%88%E5%B8%8C%E8%A1%A8&spm=1018.2226.3001.4187)，

[Java集合超详解](https://blog.csdn.net/feiyanaffection/article/details/81394745?ops_request_misc=%257B%2522request%255Fid%2522%253A%2522163686976016780261963079%2522%252C%2522scm%2522%253A%252220140713.130102334..%2522%257D&request_id=163686976016780261963079&biz_id=0&utm_medium=distribute.pc_search_result.none-task-blog-2~all~top_positive~default-2-81394745.first_rank_v2_pc_rank_v29&utm_term=java%E9%9B%86%E5%90%88&spm=1018.2226.3001.4187)，[Java.util包简单总结](https://blog.csdn.net/z55887/article/details/57587632?spm=1001.2101.3001.6650.8&utm_medium=distribute.pc_relevant.none-task-blog-2~default~BlogCommendFromBaidu~default-8.no_search_link&depth_1-utm_source=distribute.pc_relevant.none-task-blog-2~default~BlogCommendFromBaidu~default-8.no_search_link)，

[高薪程序员&面试题精讲系列34之List、Set、Map可不可以存空值？CSDN](https://blog.csdn.net/syc000666/article/details/121805566?ops_request_misc=%257B%2522request%255Fid%2522%253A%2522163963263516780357248219%2522%252C%2522scm%2522%253A%252220140713.130102334.pc%255Fall.%2522%257D&request_id=163963263516780357248219&biz_id=0&utm_medium=distribute.pc_search_result.none-task-blog-2~all~first_rank_ecpm_v1~rank_v31_ecpm-1-121805566.pc_search_em_sort&utm_term=map%E5%8F%AF%E4%B8%8D%E5%8F%AF%E4%BB%A5%E5%AD%98%E7%A9%BA%E5%80%BC&spm=1018.2226.3001.4187)，

Collection1接口是List、Set和Queue接口的父接口。

List代表有序、重复的集合，Set代表无序、不可重复的集合，Map则代表具有映射关系的集合，Java 5新增了Queue集合，代表一种队列集合实现。（打印Collection是中括号，打印Map是大括号，Key/Value用等号连接。）

List多操作下标，Set多用于去重，Map的一个应用场景是Spring容器管理Bean，Queue的FIFO特性可用于保证业务公平性。

ArrayList底层是数组，内存空间连续。

LinkedList底层的数据结构是链表，内存空间不连续。（多用于操作首尾。）

Vector底层也是数组，可以理解为线程安全的ArrayList。

HashSet无序唯一，基于HashMap实现。

TreeSet有序唯一，基于红黑树实现。（自平衡的排序二叉树。）

ArrarList list=new ArrayList(20)，会扩容几次？

0次。ArrayList底层的数据结构是数组，在初始化时会根据指定的容量开辟好对应的内存空间所以指定初始容量为20，会直接开辟20，不会扩容。如果在定义的时候没有指定初始容量，那就是ArrayList的默认容量10，扩容机制是1.5倍扩容。

[线程安全的map在JDK 1.5及其更高版本环境，有哪几种方法可以实现？牛客网](https://www.nowcoder.com/test/question/done?tid=50939852&qid=14507#summary)，

Map1用于保存具有映射关系的数据。Spring容器管理Bean用到的集合类型就是Map。

哈希算法可以理解为将数量大、占位大的一系列数据按照一定的运算规则浓缩为一个个轻量级的特征值，通过特征值可以方便快捷的查找到对应数据。这个特征值就是HashCode。

哈希表依赖Object的两个方法，hashCode()和equals()。当哈希碰撞较多时哈希表的链表会转为红黑树。所以HashMap底层是（数组+链表）或者（数组+红黑树）的形式。

HashMap底层的Entry数组，初始容量为16，负载因子是0.75f，阈值为负载因子\*容量，扩容机制是2倍扩容。

简单记忆线程安全的集合类：喂！S.H.E！

喂是指Vector，S是指Stack，H是指Hashtable，E是指Eenumeration。

Hashtable是线程安全的哈希表，使用synchronized给对象加锁保证线程安全。

ConcurrentHashMap并发容器，JDK7采用分段锁，JDK8采用CAS算法，线程安全，建议使用。

Hashtable在线程竞争激烈时，效率比较低（此时建议使用ConcurrentHashMap）。当一个线程访问Hashtable的同步方法时，其它线程如果也在访问Hashtable的同步方法时，可能会进入阻塞状态。

HashSet1底层是HashMap，它是一个不允许有重复元素的集合。HashSet允许有null值。

当你把对象加入HashSet时，HashSet会先计算对象的hashcode值来判断对象加入的位置，同时也会与其他已经加入的对象的hashcode值作比较，如果没有相符的hashcode，HashSet会假设对象没有重复出现。但是如果发现有相同hashcode值的对象，这时会调用equals()方法来检查hashcode相等的对象是否真的相同。如果两者相同，HashSet就不会让其加入操作成功。如果不同的话，就会重新散列到其他位置。这样我们就大大减少了equals的次数，相应就大大提高了执行速度。

[Servlet API/Apache Tomcat 8.5.73](https://tomcat.apache.org/tomcat-8.5-doc/servletapi/index.html)，[Servlet教程/C语言中文网](http://c.biancheng.net/servlet2/)，

[下面有关servlet service描述错误的是？牛客网](https://www.nowcoder.com/test/question/done?tid=47482566&qid=15040#summary)，

[下面有关servlet中init,service,destroy方法描述错误的是？牛客网](https://www.nowcoder.com/test/question/done?tid=47482566&qid=15043#summary)，

The servlet container calls the init method exactly once after instantiating the servlet.

[下列有关Servlet的生命周期，说法不正确的是？牛客网](https://www.nowcoder.com/test/question/done?tid=47482566&qid=15042#summary)，

JavaWeb、Servlet、JSP、Filter

JavaWeb1三大组件：Servlet程序、Listener监听器、Filter过滤器。

为了简化部署流程，Web容器往往也会自带Web服务器模块，提供基本的HTTP服务，所以您可以不用再安装Apache、IIS、Nginx等传统意义上的服务器，只需要安装一款Web容器，就能部署Servlet网站了。正是由于这个原因，有的教材将Tomcat称为Web容器，有的教材又将Tomcat称为Web服务器，两者的概念已经非常模糊了。

Servlet1（Server Applet）是用Java编写的服务器端小程序，用于处理客户机和服务器之间的交互，生成动态的Web内容。（Tomcat适合部署动态Web程序。）

Servlet接口位于org.apache.tomcat.embed:tomcat-embed-core.jar的（package）javax.servlet。

[如果运行以上jsp文件，地址栏的内容为？牛客网](https://www.nowcoder.com/question/next?pid=35202997&qid=85547&tid=50944676)，

[JSP教程/菜鸟教程](https://www.runoob.com/jsp/jsp-tutorial.html)，[黑马JavaWeb基础教程|哔哩哔哩](https://www.bilibili.com/video/BV1Qf4y1T7Hx?p=111&vd_source=11215b3665de18fedee9fab29e5bb59b) SqlSessionFactory类抽取，

JSP1是由SUN公司主导创建的一种动态网页技术标准。JSP的本质就是Servlet，它将Java代码和特定变动内容嵌入到静态的页面中，实现以静态页面为模板，动态生成其中的部分内容。JSP引入了JSP动作XML标签，用来调用内建功能。另外，可以创建JSP标签库，然后像使用标准HTML或XML标签一样使用它们。标签库能增强功能和服务器性能，而且不受跨平台问题的限制。JSP文件在运行时会被其编译器转换成更原始的Servlet代码。JSP编译器可以把JSP文件编译成用Java代码写的Servlet，然后再由Java编译器来编译成能快速执行的二进制机器码，也可以直接编译成二进制码。

总的来说，JSP网页就是用另一种方式来编写Servlet而不用成为Java编程高手。除了解释阶段外，JSP网页几乎可以被当成一个普通的Servlet来对待。

JSP标准动作元素：

jsp:include 在页面被请求的时候引入一个文件。

jsp:useBean 寻找或者实例化一个JavaBean。

jsp:setProperty 设置JavaBean的属性。

jsp:getProperty 输出某个JavaBean的属性。

jsp:forward 把请求转到一个新的页面。（地址值不变。）

jsp:plugin 根据浏览器类型为Java插件生成OBJECT或EMBED标记。

jsp:element 定义动态XML元素

jsp:attribute 置动态定义的XML元素属性。

jsp:body 设置动态定义的XML元素内容。

jsp:text 在JSP页面和文档中使用写入文本的模板

现在的项目都是前后端分离了，前端-->后端（Controller/Service/Repository）-->数据库。

但老项目就是JSP直接访问数据库。（JSP落伍了…）

[如何获取ServletContext设置的参数值？牛客网](https://www.nowcoder.com/test/question/done?tid=50929797&qid=15047#summary)，

[从以下哪一个选项中可以获得Servlet的初始化参数？牛客网](https://www.nowcoder.com/question/next?pid=35213949&qid=61216&tid=50958322)，

application对象是ServletContext的实例，Web应用中的所有Servlet共享同一个application对象，不同Servlet之间可以通过它实现数据通讯。

请求对象（接口）有两个，ServletRequest和HttpServletRequest，常用方法：

getParameter("参数名")--根据参数名获取参数的值

getParameterValues()--获取到所有参数的值并存入数组

setCharacterEncoding()--设置请求的字符编码方式

getCharacterEncoding()--返回字符编码方式

setAttribute(String,Object)--存储此请求中的属性

getAttribute(Stringname) --返回指定属性的属性值。

响应对象（接口）有两个，ServletResponse和HttpServletResponse，常用方法：

setContentType("参数名")--设置响应的字符编码方式

sendRedirect()--完成重定向

getOutputStream()--获取字节输出流

getWriter()--获取字符输出流

setHeader("Access-Control-Allow-Origin","\*");--专门用来解决跨域问题。

ServletContext.getInitParameter()--获取Tomcat的server.xml中设置ServletContext的初始化参数。

ServletConfig.getInitParameter()--Returns a String containing the value of the named initialization parameter, or null if the parameter does not exist.

ServletContext.getInitParameter()--Returns a String containing the value of the named context-wide initialization parameter, or null if the parameter does not exist.

[下列有关Servlet的生命周期，说法不正确的是？牛客网](https://www.nowcoder.com/test/question/done?tid=50902443&qid=15042#summary)，

在javax.servlet.Servlet接口中定义了3个方法：init()、service()、destory()，它们分别在Servlet生命周期的不同阶段被Servlet容器调用。

Servlet生命周期的三个阶段：

1)初始化阶段，init()。

2)运行时阶段，service()、doGet()/doPost()。

3)销毁阶段，destroy()。（不管是谁的生命周期通常都是这三个阶段，比如Vue对象。）

init()被设计成只调用一次。它在第一次创建Servlet时被调用，在后续每次用户请求时不再调用。

service()是Servlet的核心，每当一个客户请求一个HttpServlet对象，该对象的service()方法就要调用，而且传递给这个方法一个ServletRequest对象和一个ServletResponse对象作为参数。（查API了解ServletRequest、ServletResponse更多。）

GET请求来自于一个URL的正常请求，或者来自于一个未指定METHOD的HTML表单，它由doGet()方法处理。

POST请求来自于一个特别指定了METHOD为POST的HTML表单，它由doPost()方法处理。

destroy()方法只会被调用一次，在Servlet生命周期结束时被调用。

[Servlet实现的三种方式.png](file:///D:\Pkg\图片\Servlet实现的三种方式.png)，

创建Servlet类的三种方式：

1)实现javax.servlet.Servlet接口，重写其全部方法。

2)继承javax.servlet.GenericServlet抽象类，重写service()方法。

3)继承javax.servlet.http.HttpServlet抽象类，重写doGet()或doPost()方法。

值得注意的是，通过实现Serlvet接口或继承GenericServlet创建的Servlet类无法使用@WebServlet注解。（应该能用@RestController。）

@WebServlet(定义访问规则)，@WebFilter(定义触发规则)。

Servlet容器启动时，会为每个Web应用（webapps下的每个目录都是一个Web应用）创建一个唯一的ServletContext对象，该对象一般被称为“Servlet上下文”。

ServletContext对象的生命周期从Servlet容器启动时开始，到容器关闭或应用被卸载时结束。

Web应用中的所有Servlet共享同一个ServletContext对象，不同Servlet之间可以通过ServletContext对象实现数据通讯，因此ServletContext对象也被称为Context域对象。

Context域对象对整个Web应用内的所有Servlet都有效；request域对象只对本次请求涉及的Servlet有效。

[Servlet请求转发/C语言中文网](http://c.biancheng.net/servlet2/requestdispatcher.html)，[请求转发与重定向的区别.png](file:///D:\Pkg\图片\请求转发与重定向的区别.png)，

请求转发属于服务器行为。容器接收请求后，Servlet会先对请求做一些预处理，然后将请求传递给其他Web资源，来完成包括生成响应在内的后续工作。javax.servlet包中定义了一个RequestDispatcher接口，RequestDispatcher对象由Servlet容器创建，用于封装由路径所标识的Web资源。利用RequestDispatcher对象可以把请求转发给其他的Web资源。

请求转发具有以下特点：

1)请求转发不支持跨域访问，只能跳转到当前应用中的资源。

2)请求转发之后，浏览器地址栏中的URL不会发生变化，因此浏览器不知道在服务器内部发生了转发行为，更无法得知转发的次数。

3)参与请求转发的Web资源之间共享同一request对象和response对象。

4)由于forward()方法会先清空response缓冲区，因此只有转发到最后一个Web资源时，生成的响应才会被发送到客户端。

5) request.getRequestDispatcher(URL).forward(request,response)。

如果需要使用request进行值传递，需要请求转发完成，如果页面需要跳转到其他网站上必须使用重定向。

当浏览器向服务器发送请求访问某一个资源A时，资源A在响应时通知浏览器需要再进一步请求才能获取到对应的资源，浏览器再次向服务器发送请求访问资源B，最终是由资源B响应浏览器要获取的数据，这个过程叫做重定向。

重定向具有以下特点：

1)重定向前后是两次请求、两次响应。

2)重定向前后，浏览器地址栏地址会发生变化（因为两次请求都是通过浏览器发起，浏览器知道这个跳转过程，因此地址栏地址会发生变化）。

3)重定向前后的request对象不是同一个。

4)重定向前后的两个资源可以来自不同的Web应用，甚至可以是来自不同的虚拟主机或者服务器。

5) response.sendRedirect(URL)。

[Filter工作流程.png](file:///D:\Pkg\图片\Filter工作流程.png)，[FilterChain拦截过程.png](file:///D:\Pkg\图片\FilterChain拦截过程.png)，

Filter是Servlet规范中最实用的技术，通过它可以对服务器管理的所有Web资源（例如JSP、Servlet、静态HTML文件、静态图片等）进行拦截，从而实现一些特殊的功能，例如用户的权限控制、过滤敏感词、设置统一编码格式等。

@WebServlet(定义访问规则)，@WebFilter(定义触发规则)。

请求资源时，过滤器链中的过滤器依次对请求进行处理，并将请求传递给下一个过滤器，直到最后将请求传递给目标资源。发送响应信息时，则按照相反的顺序对响应进行处理，直到将响应返回给客户端。

Filter的生命周期中，包含着几个方法，不同阶段服务器会调用不同的方法，包init()、destory()、doFilter()。生命周期和Servlet类似，但Filter在运行SpringBootApplication时就初始化了，Servlet在第一次被调用时初始化。

Filter的4种配置方式：

1)具体文件：/index.jsp，是指只有访问index.jsp这一个资源时，过滤器会被执行。

2)匹配前缀：/user/\*，是指访问user下的所有资源时，过滤器会被执行。

3)匹配后缀：\* .jsp，是指访问所有后缀名为jsp的文件时，过滤器会被执行。

4)匹配所有：/\*，是指访问所有资源时，过滤器会被执行。（可满足Servlet诸如权限、缓存、日志等共性需求，有点像构造代码块啊。）

过滤器并不是必须要将请求传递到下一个过滤器或目标资源，它可以自行对请求进行处理，并发送响应给客户端，也可以将请求转发给其他的目标资源。

过滤器链中的任何一个Filter没有调用FilterChain.doFilter( )方法，请求都不会到达目标资源。

会话结束、Socket

[下面哪项技术可以用在WEB开发中实现会话跟踪实现？牛客网](https://www.nowcoder.com/test/question/done?tid=50928791&qid=15245#summary)，

会话技术是指在会话中，帮助服务器记录用户状态和数据的技术。从打开浏览器访问某个网站，到关闭浏览器的过程，称为一次会话。

会话技术分为Cookie（客户端会话技术）和Session（服务端会话技术），Cookie中保存的是字符串，Session中保存的是对象。

Cookie保存在客户端，浏览器可以禁用 Cookie，而这将使其无法正常工作。

Session保存在服务端，每一个用户独占一个Session，并发用户过多时会占用大量服务端资源。

Session对象在容器第一次调用request.getSession( )方法时创建。

值得注意的是，当客户端访问的Web资源是HTML，CSS，图片等静态资源时，服务器不会创建Session对象。

[socket编程中，以下哪个socket的操作是不属于服务端操作的？牛客网](https://www.nowcoder.com/test/question/done?tid=50901924&qid=112826#summary)，

[在一个基于分布式的游戏服务器系统中，不同的服务器之间，哪种通信方式是不可行的？牛客网](https://www.nowcoder.com/test/question/done?tid=50901924&qid=112837#summary)，

[关于Socket通信编程，以下描述正确的是？牛客网](https://www.nowcoder.com/test/question/done?tid=50929797&qid=70804#summary)，

[Socket网络编程](https://blog.csdn.net/weixin_43884234/article/details/115075016?spm=1001.2014.3001.5502)，

[socket编程入门：1天玩转socket通信技术（非常详细） (biancheng.net)](http://c.biancheng.net/socket/)，

[Socket网络编程\_程序媛 泡泡的博客-CSDN博客\_socket网络通信编程](https://blog.csdn.net/weixin_43884234/article/details/115075016?ops_request_misc=%257B%2522request%255Fid%2522%253A%2522166342645116800182123502%2522%252C%2522scm%2522%253A%252220140713.130102334.pc%255Fblog.%2522%257D&request_id=166342645116800182123502&biz_id=0&utm_medium=distribute.pc_search_result.none-task-blog-2~blog~first_rank_ecpm_v1~rank_v31_ecpm-19-115075016-null-null.nonecase&utm_term=%E5%A4%9A%E7%BA%BF%E7%A8%8B&spm=1018.2226.3001.4450)，

Socket1

Java中进程通讯（IPC）主要是Socket，MQ等。

[IDEA常用快捷键总结](https://blog.csdn.net/weixin_43884234/article/details/119991925?spm=1001.2014.3001.5502)，[Explore plugins for JetBrains Products](https://plugins.jetbrains.com/)，

[解决IDEA创建SpringBoot项目时没有web文件夹](https://blog.csdn.net/sublime_k/article/details/106821443?ops_request_misc=%257B%2522request%255Fid%2522%253A%2522164175175716780264080132%2522%252C%2522scm%2522%253A%252220140713.130102334.pc%255Fblog.%2522%257D&request_id=164175175716780264080132&biz_id=0&utm_medium=distribute.pc_search_result.none-task-blog-2~blog~first_rank_ecpm_v1~rank_v31_ecpm-1-106821443.nonecase&utm_term=%E5%88%9B%E5%BB%BAspringboot%E9%A1%B9%E7%9B%AE%E4%B8%BA%E4%BB%80%E4%B9%88%E6%B2%A1%E6%9C%89web%E7%9B%AE%E5%BD%95&spm=1018.2226.3001.4450)，

Intellij、Maven、镜像源

IDEA1是Java编程语言开发的集成环境。

JetBrains系列破解包解压密码：jwoi38jfo。

MindMaster v8.1.0 Pro破解包解压密码：jfe93fg02。

福昕PDF企业版破解包解压密码：f8304hg0。

VSCode配置java.home："java.home": "D:\\Pkg\\jdk-17.0.1+12"

如果有时候把IDEA的配置改坏了，

只需到C:\Users\YaleKuo\.IntelliJIdea2019.1路径下把config、system两个文件夹删掉即可恢复默认。

IDEA登录LeetCode创建用户名是1949。

IDEA忽略md、css、js等文件报错，ctrl+alt+shift+h。

IDEA使用本地阿里镜像源创建SpringBoot项目，<http://start.aliyun.com>。

IDEA主题插件，Visual Studio Code Monokai HC。

[application.yml不显示小叶子的解决方法](https://blog.csdn.net/qq_42362240/article/details/105599707?ops_request_misc=%257B%2522request%255Fid%2522%253A%2522164134662716780271584941%2522%252C%2522scm%2522%253A%252220140713.130102334..%2522%257D&request_id=164134662716780271584941&biz_id=0&utm_medium=distribute.pc_search_result.none-task-blog-2~all~top_click~default-1-105599707.pc_search_em_sort&utm_term=application.yml%E6%B2%A1%E6%9C%89%E5%B0%8F%E5%8F%B6%E5%AD%90&spm=1018.2226.3001.4187)，这个问题可能是没有刷新Maven（导包）。

Eclipse中一个普通java项目编译之后会产生bin目录用来存储编译好的.class文件，而idea的目录名称默认为out，同样也是在项目的根路径下。

但是有时会找不到out目录，比如导入了eclipse的项目，编译时报错，那么首先要查看out目录的配置是否正确，out目录可以改变位置或者名称，所以有时编译报错说找不到输出目录。

Safe delete (with usage search) 删除前IDE会搜寻项目中是否有使用到这个文件的其它文件。

Search in comments and strings同时搜寻相关注释中是否有相关文件以及引用。

下列关于Eclipse说法正确的是？ABCD

A)Eclipse中的Workspace是用于存储工程的路径

B)在源文件编写的时，Ctrl+s会触发Eclipse调用Java编译器编译该文件

C)Eclipse下载完成后，不需要特殊的安装，仅需解压即可

D)Eclipse是开放源代码的Java开发平台

[Apache Maven Project](https://maven.apache.org/what-is-maven.html)，[MVNRepository](https://mvnrepository.com/)，

[pom.png](file:///D:\Pkg\图片\pom.png)，[Maven坐标.png](file:///D:\Pkg\图片\Maven坐标.png)，[Maven命令.png](file:///D:\Pkg\图片\Maven命令.png)，

Maven1是一个项目管理工具，用来自动管理jar包（下载/存放/编译），比如jdbc的mysql-connector-java。（我想到了项目管理师。）

JAR（Java Archive File）是指Java档案文件，用于将类文件打包，方便移植使用。

文件格式与常见的ZIP压缩文件类似，可用常见解压软件打开。

文件默认包含一个名为META-INF/MANIFEST.MF的清单文件，由系统在生成JAR文件时自动创建。

Maven项目对象模型(POM)，可以通过一小段描述信息来管理项目的构建，报告和文档的项目管理工具软件。

在安装 IDEA 开发工具的时候，默认包含了 Maven 的安装，当前版本一般安装了 Maven2 和 Maven3。当然我们也是可以独立安装最新版本的 Maven。

Maven四个特征：repository，dependency，coordinate，mvn cmd（clean/install）。

Maven项目中的target类似于普通项目中的out。

Maven只需解压不用安装，如apache-maven-3.3.9。

Add as Maven Project是右键pom.xml而不是Project或Module。

双击install会在project添加target，双击clean后会删除target（清缓存）。

compile也是起编译作用，只不过由IDEA换成了Maven，编译效果是相同的。

install是将开发完成的项目（或工具API/框架）打包成jar，这样就可以deploy部署到服务器了（或提供给其他开发者使用）。

package是在工程中打成jar包，install打成jar包后还会在本地扔一份儿，deploy除在本地扔一份儿，还会在远程仓库中放一份儿。

如果代码运行出现了问题，可以重新编译（build），终极绝招是双击clean，这样会清空target目录，彻底重新编译。

坐标中groupId一般是公司域名，artifactId一般是项目名，version是jar包版本号。

修改apache-maven-3.3.9\conf下的settings.xml，配置镜像仓库，本地仓库等信息。

如果不设置本地仓库位置，默认为${user.home}/.m2/repository。

Artifact表示某个module要如何打包，一个module需要有Artifact才能部署到应用服务器。

但凡pom.xml里的某个依赖报错，先根据路径找到相应jar包，删除后再刷新重下。

jar包依赖具有传递性，找到jar包路径后，会有个同级的pom文件，它就根据pom文件里存的坐标再去下载依赖jar包。

2022/1/7 Project spring/Module jt，实现findUserLimit业务逻辑中，运行主启动类报错“找不到或无法加载主类JtApplication。我觉得是缓存的问题，rebuild并clean（我好像是点的install！！！）后maven再无反应，反复重启IDE/电脑后刷新都没反应。但是其他Project（boot）还正常，我只好把module jt复制到boot运行。（耗时1h。）

Git、ApiPost

Git1是一个开源的分布式版本控制系统，可以有效、高速地处理从很小到非常大的项目版本管理。（分布式区别于集中式。）

1)每次修改代码前一定要pull一遍保证此时本地的代码是最新版本；

2)每写一个新需求最好创建一个新分支，不要一直在一个分支上修改；

3)每一次需求写完代码提交完毕后最好删除本地和远程的分支；

4)push的代码一定要是对的，不然容易被人打；

5)不明白的指令一定不要瞎点。

查看Git安装地址 where git

查看Git版本 Git --version（Word把我的两个减号合成了一个短横线！）

Git去掉某文件 git rm --cached 文件名

Git去掉某文件夹 git rm -r --cached 文件夹名

gitee/cgb简易的命令行入门教程：

进入工作空间D:\10career\gitee\workspace，在此目录下打开cmd窗口。

全局设置：

git config --global user.name "YaleKuo"

git config --global user.email "yale.kuo@qq.com"

创建仓库：

mkdir cgb（创建目录）

cd cgb（进入目录）

git init（初始化git）

touch README.md（此行是Linux命令/非Linux系统需手动创建文本1.txt）

git add README.md（git add . 上传当前目录所有文件/先将资源存入gitee workspace）

git commit -m "first commit"（从本地索引到本地仓库）

git remote add origin <https://gitee.com/guo-youliang/cgb.git>（绑定关系）

git push -u origin master（从本地仓库到远程仓库）

2022/1/1 执行最后一行命令git push -u origin master报错/error: src refspec master does not match any，[does not match any解决办法](https://blog.csdn.net/qq_38198952/article/details/82792279?ops_request_misc=&request_id=&biz_id=102&utm_term=src%20refspec%20master%20does%20not%20ma&utm_medium=distribute.pc_search_result.none-task-blog-2~all~sobaiduweb~default-0-82792279.pc_search_em_sort&spm=1018.2226.3001.4187)。（git commit -m "init"）

2022/1/5在现有Project中new一个module后commit时会有大量error，先右键/Mark Directory as/Sources Root，然后commit就没有error了。最后再把标识改回来，右键/Mark Directory as/Unmark as Sources Root。（耗时0.5h。）

2022/1/5运行主启动类报错/2022-01-05 17:20:58.254 ERROR 1420 --- [nio-8091-exec-1]

com.zaxxer.hikari.pool.HikariPool : HikariPool-1 - Exception during pool initialization.

经检查是拷过来的src没有修改application.yml中连接数据库的配置信息，包括端口、用户名、密码等。这个报错可不够明显的啊。（耗时0.5h。）

Hikari是一款非常强大，高效，并且号称史上最快连接池。并且在SpringBoot2.0之后，采用的默认数据库连接池就是Hikari。不需要引入依赖，已经在SpringBoot中包含了。

2022/1/6从远程仓库克隆项目到IDEA连接时报错如下：

Clone failed

git@gitee.com: Permission denied (publickey).

Could not read from remote repository.

Please make sure you have the correct access rights

and the repository exists.

经查是错复制成了SSH，IDEA克隆应该是HTTPS地址。

[ApiPost真香真强大](https://blog.csdn.net/lupengfei1009/article/details/121091276?ops_request_misc=%257B%2522request%255Fid%2522%253A%2522164554009316780271592953%2522%252C%2522scm%2522%253A%252220140713.130102334..%2522%257D&request_id=164554009316780271592953&biz_id=0&utm_medium=distribute.pc_search_result.none-task-blog-2~blog~top_positive~default-1-121091276.nonecase&utm_term=apipost&spm=1018.2226.3001.4450)，

ApiPost1可以解析HTTP请求，直接生成并导出接口文档。

[Apache Tomcat 8](https://tomcat.apache.org/tomcat-8.5-doc/)，[Nginx和Apache和Tomcat的区别及优缺点](https://blog.csdn.net/weixin_44221613/article/details/88410701?ops_request_misc=%257B%2522request%255Fid%2522%253A%2522163965936616780261967143%2522%252C%2522scm%2522%253A%252220140713.130102334..%2522%257D&request_id=163965936616780261967143&biz_id=0&utm_medium=distribute.pc_search_result.none-task-blog-2~blog~sobaiduend~default-1-88410701.pc_v2_rank_blog_default&utm_term=nginx%E5%92%8Ctomcat%E7%9A%84%E5%8C%BA%E5%88%AB&spm=1018.2226.3001.4450)，

[Tomcat子目录及其说明.png](file:///D:\Pkg\图片\Tomcat子目录及其说明.png)，[Tomcat-SSM.png](file:///D:\Pkg\图片\Tomcat-SSM.png)，

[Tomcat点击startup.bat一闪而过](https://blog.csdn.net/qq_15267341/article/details/52012704)，[批处理教程/易百教程](https://www.yiibai.com/batch_script/batch_script_overview.html)，

Tomcat、Nginx、Apache、端口

Tomcat1只是一个servlet容器（jsp运行时也先转为servlet），可以认为是apache的扩展，但是可以独立于apache运行。

使用Tomcat首先要配置JAVA\_HOME环境变量，并指定JDK的安装路径。

开启Tomcat startup.bat

关闭Tomcat shutdown.bat

端口1：

Tomcat 8080 ，Nginx 80，MySQL 3306，

Tomcat默认访问页面http://localhost:8080，

npm创建的vue项目默认访问页面也是http://localhost:8080，但如果Tomcat已占用8080，

vue项目会自动使用8081端口！

可到conf/server.xml文件修改Tomcat默认端口号（8081）。（webapps\WEB-INF\web.xml。）

ServletContext.getInitParameter()--获取Tomcat的server.xml中设置ServletContext的初始化参数。

在目前流行的互联网架构中，对一个应用来说，Tomcat是首，SSM是中，JVM是尾，我们通常对于SSM是比较了解的，而忽略了首尾，而Tomcat在目前的网络编程中是举足轻重的，但是我们其实对Tomcat中很多原理性的东西不太了解，如果能够掌握Tomcat的原理，那么是非常有用的。

在Tomcat中部署JavaWeb应用最快捷的方式，就是直接将JavaWeb应用的所有文件（打包成一个文件夹）复制到Tomcat的/webapps目录下。在默认情况下，Tomcat会自动加载webapps目录下的JavaWeb应用，并把它发布到名为localhost的虚拟主机中。

[解决IDEA创建SpringBoot项目时没有web文件夹](https://blog.csdn.net/sublime_k/article/details/106821443?ops_request_misc=%257B%2522request%255Fid%2522%253A%2522164175175716780264080132%2522%252C%2522scm%2522%253A%252220140713.130102334.pc%255Fblog.%2522%257D&request_id=164175175716780264080132&biz_id=0&utm_medium=distribute.pc_search_result.none-task-blog-2~blog~first_rank_ecpm_v1~rank_v31_ecpm-1-106821443.nonecase&utm_term=%E5%88%9B%E5%BB%BAspringboot%E9%A1%B9%E7%9B%AE%E4%B8%BA%E4%BB%80%E4%B9%88%E6%B2%A1%E6%9C%89web%E7%9B%AE%E5%BD%95&spm=1018.2226.3001.4450)，

WEB-INF是JavaWeb应用的安全目录。所谓安全就是客户端无法访问，只有服务端可以访问的目录。（需要限制user访问的文件就放进该目录。）

Tomcat既可以运行采用开放式目录结构（只编译不打包）的Web应用，也可以运行Web应用的打包文件（WAR文件）。在开发阶段，为了方便程序调试，通常采用开放式的目录结构部署JavaWeb应用。在开发完成，进入产品发布阶段时，就应该将整个应用打包成WAR文件，再进行部署。

Web中间件历史：

最早的网站技术是Apache，但它只支持html。当微软推出asp技术后，第一代动态网站技术诞生，微软同时推出了IIS以支持asp技术。随着sun公司的j2ee架构横空出世，提出第二代动态网站技术规范servlet、jsp后，servlet容器就如雨后春笋。Tomcat、Jetty、GlassFish、JBoss、WebSpare、WebLogic，后三者非常厉害，性能超群，但由于昂贵的费用及微服务新架构的兴起，而开始没落，Jetty一直号称性能远胜Tomcat，但Tomcat的霸主地位已经难以撼动（就像编程语言中的java）。目前市场主流都采用Tomcat，特别国内一线互联网企业，而国企采用WebSpare和WebLogic居多。

[Nginx下载](http://nginx.org/en/download.html)，[关于Nginx 80端占用问题说明](https://harrylyj.blog.csdn.net/article/details/121967715?spm=1001.2014.3001.5502)，

[Nginx和Apache和Tomcat的区别及优缺点](https://blog.csdn.net/weixin_44221613/article/details/88410701?ops_request_misc=%257B%2522request%255Fid%2522%253A%2522163965936616780261967143%2522%252C%2522scm%2522%253A%252220140713.130102334..%2522%257D&request_id=163965936616780261967143&biz_id=0&utm_medium=distribute.pc_search_result.none-task-blog-2~blog~sobaiduend~default-1-88410701.pc_v2_rank_blog_default&utm_term=nginx%E5%92%8Ctomcat%E7%9A%84%E5%8C%BA%E5%88%AB&spm=1018.2226.3001.4450)，[nginx学习，看这一篇就够了](https://blog.csdn.net/qq_40036754/article/details/102463099?ops_request_misc=%257B%2522request%255Fid%2522%253A%2522163965429516780274127944%2522%252C%2522scm%2522%253A%252220140713.130102334..%2522%257D&request_id=163965429516780274127944&biz_id=0&utm_medium=distribute.pc_search_result.none-task-blog-2~blog~top_positive~default-1-102463099.pc_v2_rank_blog_default&utm_term=nginx&spm=1018.2226.3001.4450)，

Nginx1是一个轻量级、高性能的HTTP和反向代理Web服务器及电子邮件（IMAP/POP3）代理服务器。其主要特点是占有内存少，并发能力强，事实上Nginx的并发能力在同类型的网页服务器中表现较好，中国大陆使用Nginx网站用户有：百度、京东、新浪、网易、腾讯、淘宝等。

Nginx里面还可以加Gateway。

Nginx下载后解压到目标路径，执行nginx.exe，随后访问localhost:/80出现Welcome to nginx!字样就说明安装成功。

为什么使用代理：用户无法直接访问目标服务器的本地路径，必须借助代理服务器。

正向代理（客户端代理）即通常所说的代理，用于代表内部网络用户向Internet上的服务器（或称外部服务器，通常为Web服务器）发出连接请求，并接收响应结果，执行该代理功能的服务器称为代理服务器。使用代理服务器访问外部网络时，客户端必须在局域网设置中指明代理服务器的地址以及要代理的服务的端口号。（可结合AOP Proxy一起理解。）

反向代理（服务端代理）的方向与正向代理相反，指代表外部网络用户向内部服务器发出请求，即接收来自Internet上用户的连接请求，并将这些请求转发给内部网络上的服务器，然后将从内部服务器上得到的响应返回给Internet上请求连接的客户：执行反向代理服务的服务器称为反向代理服务器，反向代理服务器对外部用户表现为一个服务器。

客户端代理保护了客户端信息，服务端代理保护了服务端信息。

Nginx与Tomcat对比：

占用内存少 不超过2M Tomcat运行内存200M

并发能力强 3-5万次/秒 Tomcat并发能力150次/秒(调优可达1000次/秒)

每次运行nginx.exe时都会启动2个进程。其中内存占用大的是主进程，提供反向代理服务；内存占用小的是守护进程，主要是为了防止主进程意外关闭。手动关闭Nginx时，应该先关闭守护进程，再关主进程。

到nginx.exe路径下打开cmd，

start nginx 启动nginx（只能启动一次。）

nginx -s reload 重启nginx

nginx -s stop 停止nginx

修改nginx.conf文件以配置图片服务器：

server {

listen 80;

server\_name image.jt.com;

#你少个分号试试？

location / {

root D:/JT-SOFT/image;

}

}

到C:\Windows\System32\drivers\etc路径下修改hosts文件，实现域名与IP的映射。

默认情况下hosts文件是一堆注释，做以下配置后浏览器访问以下域名出现nginx页面即映射成功。（首先要start nginx。）

127.0.0.1 image.jt.com

127.0.0.1 manage.jt.com

127.0.0.1 web.jt.com

Edit Configurations-->Allow parallel run，边修改application.yml的端口号边开启多个服务器。

修改nginx.conf文件以配置后台服务器：

upstream tomcats {

server 127.0.0.1:8091;

server 127.0.0.1:8092;

server 127.0.0.1:8093;（注意分号！！！）

}

2022/1/3出现nginx无法启动的情况，和修改前的nginx.conf文件对比并逐步修改后发现少了三个分号...

server {

listen 80;

server\_name manage.jt.com

location / {

proxy\_pass http://tomcats;

}

}

配置好以后记得nginx -s reload。

然后到浏览器访问manage.jt.com/getPort。

但是后端@Value("$server.port")取值遇到一些问题，只要打开这一句一启动就报错，访问http://manage.jt.com/getPort结果为current port is null。

上面的叫轮询策略，下面这个叫权重策略：

upstream tomcats {

server 127.0.0.1:8091 weight=6;

server 127.0.0.1:8092 weight=3;

server 127.0.0.1:8093 weight=1;

}

如果需要服务器与用户绑定，则使用IP\_HASH，用户第一次访问该服务器，那么以后也访问该服务器。

nginx常见属性，down标识不被访问的服务器，backup属性标识备用机。

[Nginx和Apache和Tomcat的区别及优缺点](https://blog.csdn.net/weixin_44221613/article/details/88410701?ops_request_misc=%257B%2522request%255Fid%2522%253A%2522163965936616780261967143%2522%252C%2522scm%2522%253A%252220140713.130102334..%2522%257D&request_id=163965936616780261967143&biz_id=0&utm_medium=distribute.pc_search_result.none-task-blog-2~blog~sobaiduend~default-1-88410701.pc_v2_rank_blog_default&utm_term=nginx%E5%92%8Ctomcat%E7%9A%84%E5%8C%BA%E5%88%AB&spm=1018.2226.3001.4450)，

Apache1

JavaEE

Spring、IoC、AOP、代理模式

[Spring Project](https://spring.io/projects/spring-boot)，[Spring Framework 5.3.13 API](https://docs.spring.io/spring-framework/docs/current/javadoc-api/)，[Spring Framework Documentation 5.3.13](https://docs.spring.io/spring-framework/docs/current/reference/html/index.html)，

[AspectJ runtime API](https://www.eclipse.org/aspectj/doc/released/runtime-api/index.html)，[Spring AOP Tutorial](https://www.javatpoint.com/spring-aop-tutorial)，[ApplicationContext相关UML.png](https://share.weiyun.com/VP3Cas0F)，

[Spring松耦合实例/易百教程](https://www.yiibai.com/spring/spring-loosely-coupled-example.html)，[Spring源码分析合集，绝对是最容易看懂源码的教程](https://www.bilibili.com/video/BV1Ja411r7uB?spm_id_from=444.41.0.0)，

[Spring 面试问题 TOP 50\_森林屿麓的博客-CSDN博客](https://blog.csdn.net/u013967628/article/details/85990442?ops_request_misc=%257B%2522request%255Fid%2522%253A%2522166350651116782390518287%2522%252C%2522scm%2522%253A%252220140713.130102334.pc%255Fblog.%2522%257D&request_id=166350651116782390518287&biz_id=0&utm_medium=distribute.pc_search_result.none-task-blog-2~blog~first_rank_ecpm_v1~rank_v31_ecpm-2-85990442-null-null.nonecase&utm_term=spring&spm=1018.2226.3001.4450)，

Spring1是针对Bean的生命周期进行管理的轻量级容器，它是由于软件开发的复杂性而创建的。Spring使用基本的JavaBean来完成以前只可能由EJB完成的事情。然而，Spring的用途不仅仅限于服务器端的开发。从简单性、可测试性和松耦合性角度而言，绝大部分Java应用都可以从Spring中受益。（熟练掌握其应用、较为深刻理解相关设计模式、阅读过部分源码，最好写一点文章。）

Spring容器管理的对象称之为Bean。

POJO是比JavaBean更纯净的简单类或接口，主要用于数据的临时传递。而JavaBean是那些形成Spring应用的主干的Java对象，具备业务逻辑的处理能力。

框架就是开发的套路，是一个半成品软件。

为实现对象的管辖，让不同技术之间能简单的互相配合，Spring创新的形成了一套新的理论体系，可谓前无古人后无来者。其中最核心的是：IoC控制反转、DI依赖注入、Bean工厂、AOP面向切面编程、事务控制。（注意不要与SQL事务混淆。）

Spring容器的数据结构是Map集合，key="user"，value="通过反射机制实例化的对象"。

<bean id="user" class="com.jt.demo.User"></bean>

Spring中使用的对象都由IoC容器管理，不需要我们手动使用new运算符创建对象。由IoC容器管理的对象称为SpringBean，SpringBean就是Java对象，和使用new运算符创建的对象没有区别。

如果Bean实现了BeanPostProcessor接口，则Spring调用该接口的预初始化方法postProcess-BeforeInitialzation( )对Bean进行加工操作，此处非常重要，Spring的AOP就是利用它实现的。

为了降低Java开发的复杂性，Spring采取了以下4种关键策略：

1)基于POJO的轻量级和最小侵入性编程。

2)通过依赖注入和面向接口实现松耦合。

3)基于切面和惯例进行声明式编程。（使用注解声明实现功能而无需手动编程。）

4)通过切面和模板减少样板式代码。

[Spring七大组件.png](https://share.weiyun.com/JaUHSNm5)，[Spring主要jar包.jpg](file:///D:\Pkg\图片\Spring主要jar包.jpg)，

Spring框架主要由七部分组成，分别是Spring Core、Spring AOP、Spring ORM、Spring DAO、Spring Context、Spring Web和Spring Web MVC。

Bean、Context、Core三大核心组件的关系：

Bean包装的是Object，而Object必然有数据，如何给这些数据提供生存环境就是Context要解决的问题，对Context来说它就是要发现每个Bean之间的关系，为它们建立这种关系并且要维护好这种关系。所以Context就是一个Bean关系的集合，这个关系集合又叫IoC容器，一旦建立起这个IoC容器后Spring就可以为你工作了。那Core组件又有什么用武之地呢？其实Core就是发现、建立和维护每个Bean之间的关系所需要的一些类的工具，从这个角度看来，Core这个组件叫Util更能让你理解。

[IOC容器管理对象.png](file:///D:\Pkg\图片\Java\IOC容器管理对象.png)，

IoC1简单来说就是将对象Object的创建的权力及对象的生命周期的管理过程交由Spring框架来处理，从此在开发过程中不在需要关注对象的创建和生命周期的管理，而是在需要的时候由Spring框架提供，这个由Spring框架管理对象创建和生命周期的机制称之为控制反转。

在创建对象的过程中Spring可以依据对象的关系，自动把其它对象注入（无需创建对象，直接拿着使用）进来，这个过程称之为DI（Dependency Injection）依赖注入。

理解：

1)谁依赖于谁：当然是应用程序依赖于IoC容器；

2)为什么需要依赖：应用程序需要IoC容器来提供对象需要的外部资源；

3)谁注入谁：很明显是IoC容器注入应用程序某个对象，应用程序依赖的对象；

4)注入了什么：就是注入某个对象所需要的外部资源（包括对象、资源、常量数据）。

Spring容器管理对象生命周期（实现IoC）的两种方式：

1.xml配置文件管理对象

1)准备xxx.xml配置文件

2)准备bean标签

3) Spring容器管理对象

ApplicationContext容器顶级接口

ClassPathXmlApplicationContext加载配置文件的实现类对象

2.全注解管理对象

1)准备配置类@Configuration+@Bean（@Configuration+@Bean可由@Component替代，前者是自己new对象，后者靠反射机制创建对象。）（配置类就相当于早期的xml配置文件。）

2)要求方法必须有返回值（因为@Bean将自己方法的返回值交给Spring容器管理。）

3) Spring容器管理对象（实体类）

ApplicationContext容器顶级接口

AnnotationConfigApplicationContext基于注解配置的容器类

如果采用配置文件xml的方式进行注入，则注入的方式有set注入、构造方法注入、工厂模式注入。

IoC/DI解决的是对象绑定的解耦，而AOP/动态代理解决的是业务逻辑上的解耦。

如果Spring容器创建，对象立即创建则为立即加载，容器启动创建。

如果Spring容器创建，对象在被使用时才创建，则为懒加载，用时才创建。

只有sington控制对象懒加载才有效，因为prototype全都是懒加载。

[AspectJ Programming Guide](https://www.eclipse.org/aspectj/doc/released/progguide/index.html)，[AspectJ 5 Developer’s Notebook](https://www.eclipse.org/aspectj/doc/released/adk15notebook/index.html)，AOP有哪些注解？

[AspectJ runtime API](https://www.eclipse.org/aspectj/doc/released/runtime-api/index.html)，[Spring AOP Tutorial](https://www.javatpoint.com/spring-aop-tutorial)，[面试中关于Spring AOP和代理模式的那些事](https://blog.csdn.net/eson_15/article/details/84933442?ops_request_misc=%257B%2522request%255Fid%2522%253A%2522163940203516780271556875%2522%252C%2522scm%2522%253A%252220140713.130102334.pc%255Fblog.%2522%257D&request_id=163940203516780271556875&biz_id=0&utm_medium=distribute.pc_search_result.none-task-blog-2~blog~first_rank_v2~rank_v29-5-84933442.pc_v2_rank_blog_default&utm_term=spring+%E4%BB%A3%E7%90%86%E6%A8%A1%E5%BC%8F&spm=1018.2226.3001.4450)，

AOP1即面向切面编程，通过预编译方式和运行期动态代理实现程序功能的统一维护的一种技术。AOP是OOP的延续，是软件开发中的一个热点，也是Spring框架中的一个重要内容，是函数式编程的一种衍生范型。利用AOP可以对业务逻辑的各个部分进行隔离，从而使得业务逻辑各部分之间的耦合度降低，提高程序的可重用性，同时提高了开发的效率。常用于权限认证、日志、事务处理等。（AOP在Sentinel中的应用SentinelResourceAspect。）

AOP的作用就是在程序运行期间，不修改源码对已有方法进行增强。（注意Ajax话术。）

1)本质还是解耦，因为扩展功能理论上就意味着要修改源码、加深依赖，而AOP的出现无疑是对原有基础代码的锦上添花。

2)AOP的本质特点其实也就是动态代理的本质特点。

3)子类继承父类是从上往下的继承，纵向的抽取各子类的共性，而AOP能做到的是横向抽取，子类各有自己的特有方法，这些方法在父类中是没有的。

AOP将那些与业务无关，但却对多个对象产生影响的公共行为和逻辑，抽取并封装为一个可重用的模块，这个模块被称为切面。

在业务层不方便做，但是又不得不做的事情，可以放到代理对象中。通过这样的设计就可以解决业务层耦合的问题。代理对象看起来和真实的对象一模一样，所以用户使用不会察觉。（动态代理将Dao从Service中抽取出来，保证了业务的纯粹性。）

代理模式1是一种常用的设计模式，在AOP、RPC等诸多框架中均有它的身影。代理模式的核心作用就是通过代理，控制对对象的访问。

根据代理类的创建时机和创建方式的不同，可以将其分为静态代理和动态代理两种形式：在程序运行前就已经存在的编译好的代理类是为静态代理，在程序运行期间根据需要动态创建代理类及其实例来完成具体的功能是为动态代理。（静态代理基本没啥用，用了也没啥优势。）（Ajax引擎也是代理思想。）

AOP底层使用的代理模式分为接口代理（JDK动态代理）和子类代理（CGLib子类代理），二者都是动态代理。（静态代理当然就和AOP没什么关系了。）

Spring默认采用JDK代理，SpringBoot默认采用CGLib代理。

JDK代理：

Proxy.newProxyInstance(classLoader,interfaces,getInvocationHandler(target));// 获取代理对象

1)ClassLoader loader 类加载器，读取真实的类数据。

2)Class<?>[] interfaces 要求传递接口信息

3)InvocationHandler 当代理对象执行方法时执行

注意：JDK代理必须要求被代理者要么有接口(本身就是接口)，要么实现接口(实现类)。（不然为什么CGLib叫子类代理呢…）（MyBatis的@Mapper有用到JDK动态代理。）

IoC/DI解决的是对象绑定的解耦，而AOP/动态代理解决的是业务逻辑上的解耦。

一般我们将业务层中的耦合性高的代码，采用动态代理的方式进行解耦，使得程序更加具有扩展性。Spring专门针对动态代理的规则封装了一套API起名AOP。

AOP Concepts

1)连接点JoinPoint，用户可以被扩展的方法（首先要匹配切入点表达式）

2)切入点Pointcut，用户实际扩展的方法（满足切入点表达式就会执行通知方法）

3)通知Advice，扩展方法的具体实现

4)切面Aspect，将通知应用到切入点的过程

切面 = 切入点表达式 + 通知方法

切入点可理解为一个if判断（根据切入点表达式进行判断）

切入点表达式：

1)bean("对象的Id")，每次只拦截一个（类级别）

2)within("包名.类名")，可拦截多个对象

3)execution(返回值类型 包名.类名.方法名(参数列表))（方法参数级别）

4)@annotation(注解的路径)

一般常用后面两个，粒度较细。

@PropertySource 加载指定路径的配置文件properties

通知方法/Spring AOP includes the following types of advice：

1)前置通知Before advice，在目标方法执行之前执行

2)后置通知After returning advice，在目标方法执行之后执行（记录返回值）

3)异常通知After throwing advice，在目标方法执行之后抛出异常时执行（记录异常）

4)最终通知After (finally) advice，都要执行的通知（记录程序最后状态/类似于finally）

5)环绕通知Around advice，在目标方法执行前后都要执行的通知（控制目标方法是否执行）

当通知方法有多个参数且包含joinPoint时，joinPoint必须置首。

环绕通知功能最为强大也最常用，前四大通知只用做记录程序的运行状态。

环绕通知可以控制目标方法是否执行，环绕通知必须添加返回值。

子接口ProceedingJoinPoint的特有方法proceed()，执行下一个通知（如果有的话）或执行目标方法。（这里和动态代理的invoke()有点像。）

ProceedingJoinPoint.proceed()--Proceed with the next advice or target method invocation.

记录程序的状态：

1)目标对象的class/类路径

2)目标对象的方法名

3)目标对象方法的参数列表

4)获取目标对象方法的返回值（后置通知）

5)获取目标对象执行报错的异常信息（异常通知）

如果需要获取当前方法信息，则可以通过JoinPoint对象获取。

AOP把动态代理给封装了，写架构的时候（用到代理模式）就不用写烦人的JDKProxy和CGLibProxy了。

Spring中AOP案例：

1)环绕通知监控方法的执行时间。

2)环绕通知实现缓存控制。（类似于Redis。）

3)环绕通知控制事务。

4)环绕通知控制权限。

BeanFactory--The root interface for accessing a Spring bean container.

ApplicationContext(implements BeanFactory)--Central interface to provide configuration for an application.

关于Spring和SpringMVC的注解，详见org.springframework.stereotype，org.springframework

.context.annotation和org.springframework.web.bind.annotation。

[Spring Framework 5.3.13 API](https://docs.spring.io/spring-framework/docs/current/javadoc-api/)，

@Autowired--Marks a constructor, field, setter method, or config method as to be autowired by Spring's dependency injection facilities.

@Required--Marks a method (typically a JavaBean setter method) as being 'required': that is, the setter method must be configured to be dependency-injected with a value.

This annotation indicates that the affected bean property must be populated at configuration time, through an explicit property value in a bean definition or through autowiring.

@Value--Annotation used at the field or method/constructor parameter level that indicates a default value expression for the annotated element.

@Scope--In this context, scope means the lifecycle of an instance, such as singleton, prototype, and so forth.

@Lazy--Indicates whether a bean is to be lazily initialized. May be used on any class directly or indirectly annotated with @Component or on methods annotated with @Bean.

@Configuration标识配置类。

@Bean将自己方法的返回值交给Spring容器管理。

@Component将该类交给Spring容器管理，通过反射自动实例化对象。

@ComponentScan("com.jt")包扫描的注解，使Spring注解有效。（指定扫描的包路径，可以扫描它的子孙包，用在配置类中。类似的还有@ServletComponentScan和@MapperScan。）

@Scope控制对象单例/多例，Spring默认管理的对象都是单例的。

@Lazy懒加载，只有singleton控制对象懒加载才有效，因为prototype全都是懒加载。

@Autowired可以将Spring容器中的对象作为属性自动注入到当前类。（按照类型注入。）

@RequestBody将JSON串转为Java对象。

@ResponseBody将Java对象转为JSON串。

@Qualifier按照名称注入。当存在多个相同类型的 Bean只需要注入其中一个时，可以使用@Qualifier搭配@Autowired指定需要装配的那个确切的Bean来消除歧义。

1)按照类型注入（仅@Autowired）

如果注入的类型是接口，则自动的查找其实现类对象进行注入。

一般spring框架内部的接口都是单实现类，程序可不会做选择题。

2)按照名称注入

@Autowired+@Qualifier组合使用。

@Value("${key}")为属性（动态）赋值。（类似于模板字符串。）（com.jt.PortController用到该注解无法取到server.port。）

@Component is a generic stereotype for any Spring-managed component. @Repository, @Service, and @Controller are specializations of @Component for more specific use cases (in the persistence, service, and presentation layers, respectively).

[Spring Framework Documentation 5.3.13](https://docs.spring.io/spring-framework/docs/current/reference/html/index.html)，

As a rule, you should use the prototype scope for all stateful beans and the singleton scope for stateless beans.

A data access object (DAO) is not typically configured as a prototype, because a typical DAO does not hold any conversational state. It was easier for us to reuse the core of the singleton diagram.

Spring’s pointcut model enables pointcut reuse independent of advice types. You can target different advice with the same pointcut.

The ClassFilter interface is used to restrict the pointcut to a given set of target classes. If the matches() method always returns true, all target classes are matched.

The MethodMatcher interface is normally more important.

Most MethodMatcher implementations are static, meaning that their isRuntime() method returns false. In this case, the three-argument matches method is never invoked.

If possible, try to make pointcuts static, allowing the AOP framework to cache the results of pointcut evaluation when an AOP proxy is created.

Each advice is a Spring bean. An advice instance can be shared across all advised objects or be unique to each advised object. This corresponds to per-class or per-instance advice.

The most fundamental advice type in Spring is interception around advice.

Before advice can be used with any pointcut.

Throws advice can be used with any pointcut.

After returning advice can be used with any pointcut.

The basic way to create an AOP proxy in Spring is to use the org.springframework.aop.frame-work.ProxyFactoryBean. This gives complete control over the pointcuts, any advice that applies, and their ordering. However, there are simpler options that are preferable if you do not need such control.

[Spring事务和MySQL事务详解面试](https://blog.csdn.net/weixin_48272905/article/details/108525283?ops_request_misc=%257B%2522request%255Fid%2522%253A%2522163972692916780261960229%2522%252C%2522scm%2522%253A%252220140713.130102334..%2522%257D&request_id=163972692916780261960229&biz_id=0&utm_medium=distribute.pc_search_result.none-task-blog-2~blog~sobaiduend~default-1-108525283.pc_v2_rank_blog_default&utm_term=Spring%E4%BA%8B%E5%8A%A1%E5%92%8CMySQL%E4%BA%8B%E5%8A%A1%E8%AF%A6%E8%A7%A3%E9%9D%A2%E8%AF%95&spm=1018.2226.3001.4450)，

[下面有关SPRING的事务传播特性，说法错误的是？牛客网](https://www.nowcoder.com/test/question/done?tid=47482566&qid=14942#summary)，

Spring事务本质是对数据库事务的支持，如果数据库不支持事务（例如MySQL的MyISAM引擎不支持事务），则Spring事务也不会生效。

[Spring Project](https://spring.io/projects/spring-boot)，[Servlet API/Apache Tomcat 8.5.73](https://tomcat.apache.org/tomcat-8.5-doc/servletapi/index.html)，

[Spring Framework 5.3.13 API](https://docs.spring.io/spring-framework/docs/current/javadoc-api/)，[Spring Framework Documentation 5.3.13](https://docs.spring.io/spring-framework/docs/current/reference/html/index.html)，

[Spring MVC框架入门教程/C语言中文网](http://c.biancheng.net/spring_mvc/)，

[Spring MVC教程/Java技术驿站](https://www.cmsblogs.com/article/1389040134535647232)，[ApplicationContext接口/Java技术驿站](https://www.cmsblogs.com/article/1389033488166031360)，

[MVC-SSM-JavaWeb拼图.png](file:///D:\Pkg\图片\Java\MVC-SSM-JavaWeb拼图.png)，[ApplicationContext相关UML.png](file:///D:\Pkg\图片\Java\ApplicationContext相关UML.png)，

SpringMVC、RESTful

SpringMVC1是Spring提供的一个基于MVC设计模式的轻量级Web开发框架，本质上相当于Servlet。SpringMVC属于SpringFrameWork的后续产品，已经融合在Spring Web Flow里面。（这里应该和Spring WebFlux有点关系。）（专门针对于前后端交互而开发的框架。）

传统Servlet的弊端：

1)每个业务方法都需要编辑对应的Servlet。

2)请求方式单一（doGet/doPost）。

3)通过Servlet获取的参数类型都是String，需手动进行类型转化。

4)大量的web.xml配置文件。（SpringBoot采用注解。）

[SpringMVC关键组件分析.png](file:///D:\Pkg\图片\Java\SpringMVC关键组件分析.png)，

SpringMVC中的POJO就是Model层，JSP就是View层，Controller就是Control层。

（此Controller非@Controller。）

Controller是控制器接口，接口中只有一个方法handleRequest，用于处理请求和返回Model-AndView。（springframework.stereotype的Controller/Service/Repository等都是注解@！）

现在主流基于SSM三大框架开发都是在MVC上继续演化，又分为持久层DAO/Mapper，业务层Service，控制层Controller。持久层用来和数据库读写ORM，业务层用来处理复杂的业务逻辑，控制层用来处理MVC的控制。

MVC是一种设计思想，编码中降低代码的耦合性。

前端专注于开发页面View，后端专注于开发后端Model，二者通过Controller进行控制。

MVC实现了前端和后端的松耦合，但是根据实际的开发情况，很多的业务逻辑比较复杂。如果后端将所有的代码都写到同一个Java类中，这样的代码结构很臃肿。为了很好的实现MVC设计思想，所以后端代码也应该分层。

1)控制层Controller与前端页面交互@Controller（presentation）

2)业务层Service编辑业务逻辑@Service

3)持久层Mapper实现数据库的相关操作@Repository/@Mapper（persistence）

MVC > 三层代码结构！！！

Mapper类2个，一个接口UserMapper/一个实现类UserMapperImpl。

Service类2个，一个接口UserService/一个实现类UserServiceImpl。

Controller类1个，即UserController。

注意：被调用的一般会有接口和实现类，优先调用接口。

SpringMVC的核心就是DispatcherServlet，DispatcherServlet实质也是一个HttpServlet。（DispatcherServlet extends FrameworkServlet extends HttpServletBean extends HttpServlet。）

DispatcherServlet负责将请求分发，所有的请求都有经过它来统一分发。（这里的分发请求和VueRouter有点像，VueRouter又和Layui的文件目录有点像。）

用户向服务器发送请求，请求会到DispatcherServlet，DispatcherServlet对请求URL进行解析，得到请求资源标识符（URI），然后根据该URI，调用HandlerMapping获得该Handler配置的所有相关的对象（包括一个Handler处理器对象、多个HandlerInterceptor拦截器对象），最后以HandlerExecutionChain对象的形式返回。（注意区分Java提供的RequestDispatcher. forward(ServletRequest var1, ServletResponse var2)。）

setCharacterEncoding只是设置字符的编码方式，setContentType除了可以设置字符的编码方式还能设置文档内容的类型。

关于Spring和SpringMVC的注解，详见org.springframework.stereotype，org.springframework

.context.annotation和org.springframework.web.bind.annotation。

@Controller将该类首先交给SpringMVC进行管理，之后将SpringMVC框架整体交给Spring管理。（中间涉及到协议的支持。）（在SpringMVC中，Controller负责处理由DispatcherServlet 分发的请求。）

@Repository将持久层交给Spring容器管理。（没有MyBatis的时候用这个。）

@Mapper将接口交给Spring容器管理，Map<userMapper, JDK代理对象>。（有了MyBatis就用这个。）

@RequestMapping("/hello") 设定用户的访问路径。（404就是这么来的。）

@ResponseBody将返回值转化为JSON。（返回值是String类型则直接返回。）

@RestController将当前类的所有返回值都转化为JSON。

@GetMapping("/findUserById") 只接收GET请求类型

@CrossOrigin允许跨域访问。

同源策略：

1. 浏览器访问地址
2. Ajax请求地址

对于①和②，协议://域名:端口都相同则称之为同源策略或同域访问，反之则为跨域访问。

1)

浏览器地址: http://www.jd.com/xx/xx

Ajax地址: https://www.jd.com/yy/yy 跨域/协议不同

2)

浏览器地址: http://www.jd.com/xx/xx

Ajax地址: http://www.jd.com/yy/yy 同域

3)

浏览器地址: http://www.jd.com:80/xx/xx

Ajax地址: http://www.jd.com/yy/yy 同域/http默认80

4)

已知域名与IP对应

浏览器地址: http://www.jd.com/xx/xx

Ajax地址: http://8.8.8.8/yy/yy 跨域/通过DNS对应不等于域名相同

5)

浏览器地址: https://www.jd.com/xx/xx

Ajax地址: https://www.jd.com:443/yy/yy 同域/https默认443

6)

浏览器地址: http://www.jd.com/xx/yy/xx

Ajax地址: http://miaoshi.jd.com/yy/yy 跨域/添加了前缀子域名（典型的不同服务器。）

The @RestController and @RequestMapping annotations are Spring MVC annotations (they are not specific to Spring Boot).

This code uses Spring @RestController annotation, which marks the class as a controller where every method returns a domain object instead of a view. It is shorthand for including both @Controller and @ResponseBody.

[method-crud.jpg](file:///D:\Pkg\图片\method-crud.jpg)，[RESTful API/CSDN](https://blog.csdn.net/weixin_44303631/article/details/88545269?ops_request_misc=&request_id=&biz_id=102&utm_term=%E8%A1%A8%E5%BE%81%E7%8A%B6%E6%80%81%E8%BD%AC%E7%A7%BB&utm_medium=distribute.pc_search_result.none-task-blog-2~blog~sobaiduweb~default-4-88545269.pc_v2_rank_blog_default&spm=1018.2226.3001.4450)，[Spring MVC RESTful风格开发/Java技术驿站](https://www.cmsblogs.com/article/1389040565198393344)

[Architectural Styles and the Design of Network-based Software Architectures](https://www.ics.uci.edu/~fielding/pubs/dissertation/fielding_dissertation.pdf)，

RESTful1架构是对MVC架构改进后所形成的一种架构，通过使用事先定义好的接口与不同的服务联系起来。在RESTful架构中，浏览器使用POST，DELETE，PUT和GET四种请求方式分别对指定的URL资源进行增删改查操作。因此，RESTful是通过URI实现对资源的管理及访问，具有扩展性强、结构清晰的特点。（请求类型一共8种，常用4种。）

传统参数拼接 <http://localhost:8080/getUser?name=Yale&age=23&sex=male>

RESTful参数拼接 <http://localhost:8080/user/Yale/23/male>（请求路径不应出现动词以提高安全性。）

org.springframework.web.bind.annotation

@GetMapping--A composed annotation that acts as a shortcut for @RequestMapping(method = RequestMethod.GET).

@PostMapping--A composed annotation that acts as a shortcut for @RequestMapping(method = RequestMethod.POST).

同样的访问路径，如果请求类型不匹配，报错/405。

RESTful架构将服务器分成前端服务器和后端服务器两部分，前端服务器为用户提供无模型的视图；后端服务器为前端服务器提供接口。浏览器向前端服务器请求视图，通过视图中包含的Ajax函数发起接口请求获取模型。

项目开发引入RESTful架构，利于团队并行开发。在RESTful架构中，将多数HTTP请求转移到前端服务器上，降低服务器的负荷，使视图获取后端模型失败也能呈现。但RESTful架构却不适用于所有的项目，当项目比较小时无需使用RESTful架构，项目变得更加复杂。

A key difference between a traditional MVC controller and the RESTful web service controller shown earlier is the way that the HTTP response body is created. Rather than relying on a view technology to perform server-side rendering of the greeting data to HTML, this RESTful web service controller populates and returns a Greeting object. The object data will be written directly to the HTTP response as JSON.（JSON作为返回值类型是RESTful的一个重要特征！！！）

REST调用及测试都很方便，RPC就显得有点繁琐，但是RPC的效率是毋庸置疑的，所以建议在多系统之间的内部调用采用RPC。对外提供的服务，Rest更加合适。

[MyBatis 3](https://mybatis.org/mybatis-3/)，[MyBatis 3.5.7 API](https://mybatis.org/mybatis-3/zh/apidocs/index.html)，[MyBatis-Spring](https://mybatis.org/spring/index.html)，[MyBatis-SpringBoot](http://mybatis.org/spring-boot-starter/mybatis-spring-boot-autoconfigure/index.html)，[MyBatis-Guice](http://mybatis.org/guice/)，

[xml转义字符.png](file:///D:\Pkg\图片\XML转义字符.png)，[一级缓存和二级缓存.png](file:///D:\Pkg\图片\一级缓存和二级缓存.png)，[MyBatis一级缓存和二级缓存](https://blog.csdn.net/weixin_43232955/article/details/105675200?ops_request_misc=%257B%2522request%255Fid%2522%253A%2522163963844516780261940129%2522%252C%2522scm%2522%253A%252220140713.130102334..%2522%257D&request_id=163963844516780261940129&biz_id=0&utm_medium=distribute.pc_search_result.none-task-blog-2~blog~top_click~default-2-105675200.pc_v2_rank_blog_default&utm_term=mybatis%E7%9A%84%E4%B8%80%E7%BA%A7%E4%BA%8C%E7%BA%A7%E7%BC%93%E5%AD%98&spm=1018.2226.3001.4450)，[MyBatis Plus](https://baomidou.com/)，

[MyBatis标签.png](file:///D:\Pkg\图片\MyBatis标签.png)，MyBatis有哪些标签？

[CGB2110-DAY10-Mybatis-SpringBoot用法](https://harrylyj.blog.csdn.net/article/details/122296579?spm=1001.2014.3001.5502)，

[MyBatis源码讲解/TTS160](https://tts.tmooc.cn/video/showVideo?menuId=731120&version=CGBTN202011)，[SqlSessionFactory工具类抽取/哔哩哔哩](https://www.bilibili.com/video/BV1Qf4y1T7Hx?p=111)，

MyBatis、MBP、Lombok

MyBatis1是一款优秀的持久层框架，它支持定制化SQL、存储过程以及高级映射。MyBatis避免了几乎所有的JDBC代码和手动设置参数以及获取结果集。MyBatis可以使用简单的XML或注解来配置和映射原生信息，将接口和Java的POJO映射成数据库中的记录。（在内部封装了JDBC，简化了操作数据库的过程。）

数据库中的表与POJO中的类一一映射，表中的字段与类中的属性一一映射。（表和类，字段和属性。）

在注解中写SQL需要引入org.apache.ibatis.annotations包，其中包括@Select，@Insert，@Up-date和@Delete等注解。注解一般写在XxxxMapper.java接口文件中。（最靠近数据库的一层Java文件。）

映射文件UserMapper.xml中的namespace与接口对应，id属性与接口方法对应，resultType与结果集类型对应。

#{}是预编译处理，通过PreparedStatement来执行SQL语句，能有效防止SQL注入，而${}是单纯的字符串替换，不能防止SQL注入。

MyBatis自动将结果集的数据封装为对象，如果返回List集合，则自动封装为List，User充当其中的泛型对象；如果返回值User对象，则直接返回User对象。（resultType="com.jt.pojo.

User"。）

MyBatis结果集的字段名称如果与属性的名称满足驼峰映射的规则，则可以通过简单配置以实现自动化映射。（application.yml/map-underscore-to-camel-case: true。）

resultType只能于单表查询，resultMap可用于多表联查。resultMap另外的作用是手动映射。

一对一映射的核心关键字association，固定搭配javaType。

一对多映射的核心关键字collection，固定搭配ofType。

POJO里封装一个对象private Dept dept，典型的一对一。

POJO里封装一个集合private List<Emp> emps，典型的一对多。

MyBatis的工作流程：

1)创建SqlSessionFactory。

2)通过SqlSessionFactory创建SqlSession。

3)通过sqlsession执行数据库操作。

4)调用session.commit()提交事务。

5)调用session.close()关闭会话。

MyBatis最强大的特性之一就是它的动态语句功能。如果您以前有使用JDBC或者类似框架的经历，您就会明白把SQL语句条件连接在一起是多么的痛苦，要确保不能忘记空格或者不要在columns列后面省略一个逗号等。动态语句能够完全解决掉这些痛苦。

mybatis-geneator是一款MyBatis自动代码生成工具，可以通过配置，快速生成mapper和xml文件。

通过<sql>标签可以提取共性，比如select语句都相同，仅where条件不同。

每个MyBatis应用程序主要都是使用SqlSessionFactory实例的，一个SqlSessionFactory实例可以通过SqlSessionFactoryBuilder获得。SqlSessionFactoryBuilder可以从一个xml配置文件或者一个预定义的配置类的实例获得。（建造者模式。）

用xml文件构建SqlSessionFactory实例是非常简单的事情。推荐在这个配置中使用类路径资源（classpath resource），但你可以使用任何Reader实例，包括用文件路径或file://开头的url创建的实例。MyBatis有一个实用类Resources，它有很多方法，可以方便地从类路径及其它位置加载资源。

一级缓存是指Mybatis中SqlSession对象的缓存。当我们执行查询之后，查询的结果会同时存入到SqlSession为我们提供的一块区域中。该区域的结构是一个Map。当我们再次查询同样的数据，Mybatis会先去SqlSession中查询是否有，有的话直接拿出来用。当SqlSession对象消失时，Mybatis的一级缓存也就消失了。

（在同一个SqlSession内实现数据共享。）（Mybatis默认开启一级缓存。）

二级缓存是指Mybatis中SqlSessionFactory对象的缓存。由同一个SqlSessionFactory对象创建的SqlSession共享其缓存。二级缓存的使用步骤：Mybatis框架支持二级缓存（在SqlMapConfig.xml中配置）；让当前的映射文件支持二级缓存（在UserMapper.xml中配置<cache/>标签）；让当前的操作支持二级缓存（在select标签中配置）。

（由同一个SqlSessionFactory生产的多个SqlSession(多线程)可以共享数据。）（二级缓存同样是默认开启的，但需要指定其作用域。）

多线程操作同一个POJO对象必须实现序列化接口implements Serializable。

serializable串行化是事务的最高的隔离级别，通过强制事务排序，使之不可能相互冲突，从而解决幻读问题。

MyBatis一级缓存、二级缓存只针对单台Tomcat服务器生效，在集群中使用高级缓存服务器Redis代替。

[MyBatis Plus](https://baomidou.com/)，[MBP实现自动填充功能](https://baomidou.com/pages/4c6bcf/)，[MBP分页插件](https://baomidou.com/pages/8f40ae/)，

MBP1是一个 MyBatis增强工具，在MyBatis的基础上只做增强不做改变，为简化开发、提高效率而生。

SQL是面向过程的，所以MyBatis是半面向对象的。而MBP能做到纯面向对象，言外之意使用MBP以后连SQL语句都不用写了。另外SQL与数据库是耦合的，切换不同的数据库就要写不同语法的SQL，而MBP可以根据数据库类型自动生成对应的SQL。但使用MBP的弊端是只适用于单表操作。

Hibernate也是纯面向对象，且能操作多表。但学习成本高，并不流行。（可以说MBP就是Hibernate的成功转型。）

2022/1/8公共属性填充失败，后来发现没给MetaObjectHandler加@Component注解。但还是失败，最后点进官网一看，才发现自己少了给BasePojo添加@TableField注解。（耗时0.5h）

Lombok1是一个自动生成代码的插件，它可以动态生成Getter/Setter/Constructor等常方法，提高开发效率，简化POJO类代码。

2021/12/30今天安装Lombok插件，IDEA始终连不上网。倒腾了大半天放弃了，正准备去官网下插件本地安装，突然它就连上了…

@Data动态生成Getter/Setter/toString等方法。

@Accessors控制是否开启链式加载结构。

@NoArgsConstructor生成无参构造。

@AllArgsConstructor生成全参构造。

@Mapper将接口交给Spring容器管理，Map<userMapper,JDK代理对象>。

@SpringBootTest启动Spring容器并从中动态的获取对象。（与主启动类同包。）

@Transactional控制事务，commit/rollback。

[SpringBoot中使用JPA以及JPA介绍](https://blog.csdn.net/wujiaqi0921/article/details/78789087?ops_request_misc=%257B%2522request%255Fid%2522%253A%2522164440091616781683913749%2522%252C%2522scm%2522%253A%252220140713.130102334..%2522%257D&request_id=164440091616781683913749&biz_id=0&utm_medium=distribute.pc_search_result.none-task-blog-2~all~top_ulrmf~default-3-78789087.pc_search_insert_ulrmf&utm_term=jpa&spm=1018.2226.3001.4187)，

JPA、Hibernate、ORM

JPA1（Java Persistence API/Java持久层API）是一套ORM规范，Hibernate正是一个实现了JPA规范的持久层框架。

SpringData JPA可以理解为JPA规范的进一步封装，底层还是使用了Hibernate的JPA技术实现。

Hibernate1是一个开放源代码的对象关系映射框架，它对JDBC进行了非常轻量级的对象封装，它将POJO与数据库表建立映射关系，是一个全自动的ORM框架，Hibernate可以自动生成SQL语句，自动执行，使得Java程序员可以随心所欲的使用对象编程思维来操纵数据库。

HQL是一种面向对象的查询语言，类似于SQL，但不是去对表和列进行操作，而是面向对象和它们的属性。HQL查询被Hibernate翻译为传统的SQL查询从而对数据库进行操作。

[一般情况下，以下哪个选项不是关系数据模型与对象模型之间匹配关系？牛客网](https://www.nowcoder.com/question/next?pid=35200657&qid=14998&tid=50943467)，

ORM1是一种为了解决关系型数据库数据与简单Java对象（POJO）的映射关系的技术。简单的说，ORM是通过使用描述对象和数据库之间映射的元数据，将程序中的对象自动持久化到关系型数据库中。如今已有很多免费和付费的ORM产品，而有些程序员更倾向于创建自己的ORM工具。

DAO/Mapper是SSM中的持久层，持久层正是用来和数据库读写ORM。

（我觉得Hibernate和EntityFramework有的一拼。）（从效果上说，ORM其实是创建了一个可在编程语言里使用的虚拟对象数据库。）

MyBatis和Hibernate不同，它不完全是一个ORM框架，因为MyBatis需要程序员自己编写SQL语句，不过Hibernate可以通过XML或注解方式灵活配置要运行的SQL语句，并将java对象和SQL语句映射生成最终执行的SQL，最后将SQL执行的结果再映射生成java对象。

MyBatis学习门槛低，简单易学，程序员直接编写原生态SQL，可严格控制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，例如互联网软件、企业运营类软件等，因为这类软件需求变化频繁，一但需求变化要求成果输出迅速。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（例如需求固定的定制化软件）如果用Hibernate开发可以节省很多代码，提高效率。但是Hibernate的缺点是学习门槛高，要精通门槛更高，而且怎么设计O/R映射，在性能和对象模型之间如何权衡，以及怎样用好Hibernate需要具有很强的经验和能力才行。总之，按照用户的需求在有限的资源环境下只要能做出维护性、扩展性良好的软件架构都是好架构，所以框架只有适合才是最好。

SqlSessions are created by a SqlSessionFactory instance. The SqlSessionFactory contains methods for creating instances of SqlSessions all different ways. The SqlSessionFactory itself is created by the SqlSessionFactoryBuilder that can create the SqlSessonFactory from XML, annotations or hand coded Java configuration.

[Spring Project](https://spring.io/projects/spring-boot)，[Spring Boot 2.6.1 API](https://docs.spring.io/spring-boot/docs/current/api/)， [Spring Boot Reference Documentation](https://docs.spring.io/spring-boot/docs/current/reference/htmlsingle/)，

[SpringBoot注解大全.png](file:///D:\Pkg\图片\Spring注解大全.png)，[Spring七大组件.png](file:///D:\Pkg\图片\Spring七大组件.png)，

SpringBoot、SpringCloud、设计模式

SpringBoot1是Spring组件一站式解决方案，主要是简化了使用Spring的难度，省略了繁重的配置，提供了各种启动器，使开发者能快速上手。该框架使用了特定的方式来进行配置，从而使开发人员不再需要定义样板化的配置。通过这种方式，SpringBoot致力于在蓬勃发展的快速应用开发领域（rapid application development）成为领导者。（SpringBoot可以理解为框架的框架。）

SpringBoot所处在的一个大时代背景，微服务，一种架构风格。

开箱即用是指SpringBoot将繁琐的配置封装到某些jar包中，jar包中的文件已经完成了配置，引入即可使用。（只需简单步骤即可完成大量的配置工作。）

IntellijIDEA一般可以通过两种方式创建SpringBoot项目，使用Maven创建，和使用Spring-Initializr创建。（我应该从来没有用过Maven创建。）

创建SpringBoot项目时，建议把Initializr Service URL改为国内网址https://start.aliyun.com。

SpringBoot current version 2.6.1要求Maven版本3.5+，我用的是3.3.9。

Spring默认采用JDK代理，SpringBoot默认采用CGLib代理。

SpringBoot提供了自动化配置模块的模板引擎主要有：Thymeleaf，FreeMarker，Groovy。

@Configuration标识配置类。

@Bean将自己方法的返回值交给Spring容器管理。

@Component将该类交给Spring容器管理，通过反射自动实例化对象。

@ComponentScan("com.jt")包扫描的注解，使Spring注解有效。（指定扫描的包路径，可以扫描它的子孙包，用在配置类中。类似的还有@ServletComponentScan和@MapperScan。）

@Scope控制对象单例/多例，Spring默认管理的对象都是单例的。

@Lazy懒加载，只有sington控制对象懒加载才有效，因为prototype全都是懒加载。

@Autowired可以将Spring容器中的对象作为属性自动注入到当前类。（依赖注入。）

@SpringBootApplication标识SpringBoot快速启动类，启动后会嵌入到Tomcat。

@SpringBootConfiguration组合了@Configuration，标识SpringBoot的配置类。

@EnableAutoConfiguration启动自动化的配置，加载pom.xml文件中的启动项。（不加载依赖吗？）

@ComponentScan扫描范围默认是主启动类的同包及子包。（Spring组件扫描。）

@Component is a generic stereotype for any Spring-managed component. @Repository, @Service, and @Controller are specializations of @Component for more specific use cases (in the persistence, service, and presentation layers, respectively).

@SpringBootApplication--Indicates a configuration class that declares one or more @Bean methods and also triggers auto-configuration and component scanning. This is a convenience annotation that is equivalent to declaring @Configuration, @EnableAutoConfiguration and @ComponentScan.

[SpringCloud技术组成.png](file:///D:\Pkg\图片\Java\SpringCloud技术组成.png)，[SpringCloud与微服务.processon](https://www.processon.com/fav)，[SpringCloud入门操作手册](https://blog.csdn.net/weixin_38305440/article/details/102775484?spm=1001.2014.3001.5502)，

[SpringCloud Alibaba Project](https://spring.io/projects/spring-cloud-alibaba)，[SpringCloud Alibaba GitHub](https://github.com/alibaba/spring-cloud-alibaba)，

SpringCloud1是一个基于Java语言实现的微服务框架。SpringCloud是一系列框架的有序集合，它利用SpringBoot的开发便利性简化了分布式系统的开发，比如服务发现、服务网关、服务路由、链路追踪等。SpringCloud并不重复造轮子，而是将市面上开发得比较好的模块集成进去，进行封装，从而减少了各模块的开发成本。换句话说：SpringCloud 提供了构建分布式系统所需的“全家桶”。

目前，国内使用 Spring Cloud 技术的公司并不多见，不是因为Spring Cloud不好，主要原因有以下几点：

1)SpringCloud中文文档较少，出现问题网上没有太多的解决方案。

2)国内创业型公司技术老大大多是阿里系员工，而阿里系多采用Dubbo来构建微服务架构。

3)大型公司基本都有自己的分布式解决方案，而中小型公司的架构很多用不上微服务，所以没有采用Spring Cloud的必要性。

SpringBoot专注于快速方便的开发单个个体微服务，而SpringCloud将SpringBoot开发的一个个单体微服务整合并管理起来。

[设计模式.png](file:///D:\Pkg\图片\设计模式.png)，[框架(Framework)中常用设计模式分析](https://blog.csdn.net/maitian_2008/article/details/121314392?spm=1001.2014.3001.5502)，[设计模式/菜鸟教程](https://www.runoob.com/design-pattern/design-pattern-tutorial.html)，

设计模式1是最精髓的东西，是软件思想的体现，是软件的灵魂。如三大框架SSM只所以傲视群雄，它的傲娇，它的底气就来自对经典设计模式的全面应用。所以如果只会应用不懂其义，那你只是个码农；如何能领会设计模式的奥义，你就有了高级程序员的潜力；如果你能自己仿写，你就有了架构师的入门证。（而Vue正是应用了观察者设计模式。）

适配器模式（Adapter Pattern）是作为两个不兼容的接口之间的桥梁。

单例模式可以说是大多数开发人员在实际中使用最多的，常见的Spring默认创建的bean就是单例模式的（singleton）。

饿汉式：定义单例类；私有化构造方法；创建本类的对象并私有化；提供public方法以返回本类的唯一对象。

懒汉式：定义单例类；私有化构造方法；定义成员变量以保存对象的地址值；提供public方法，当且仅当第一次调用该方法时创建对象（延迟加载，需要用到的时候才做出响应）。

微服务

[SpringCloud Alibaba Project](https://spring.io/projects/spring-cloud-alibaba)，[SpringCloud Alibaba GitHub](https://github.com/alibaba/spring-cloud-alibaba)，

[深入理解微服务](https://juejin.cn/post/6844903846641139726)，[计算机软件架构发展历史](https://yutian.blog.csdn.net/article/details/118571434)，[@Excel](http://doc.ruoyi.vip/ruoyi/document/htsc.html#%E5%AF%BC%E5%85%A5%E5%AF%BC%E5%87%BA)，[若依权限管理子系统简介](https://yutian.blog.csdn.net/article/details/118579638)，

[微服务之注册中心](https://blog.csdn.net/u012921921/article/details/117297909?ops_request_misc=%257B%2522request%255Fid%2522%253A%2522164193041716780357268807%2522%252C%2522scm%2522%253A%252220140713.130102334.pc%255Fblog.%2522%257D&request_id=164193041716780357268807&biz_id=0&utm_medium=distribute.pc_search_result.none-task-blog-2~blog~first_rank_ecpm_v1~rank_v31_ecpm-5-117297909.nonecase&utm_term=%E6%B3%A8%E5%86%8C%E4%B8%AD%E5%BF%83&spm=1018.2226.3001.4450)，[微服务架构.png](file:///D:\Pkg\图片\Java\微服务架构.png)，

[大数据入门：微服务与分布式\_加米谷大数据张老师的博客-CSDN博客\_微服务与分布式数据库](https://blog.csdn.net/shuimuzh123/article/details/111645525?ops_request_misc=&request_id=&biz_id=102&utm_term=%E5%A4%A7%E6%95%B0%E6%8D%AE%E5%92%8C%E5%BE%AE%E6%9C%8D%E5%8A%A1&utm_medium=distribute.pc_search_result.none-task-blog-2~blog~sobaiduweb~default-0-111645525.nonecase&spm=1018.2226.3001.4450)，

微服务1就是一些可独立运行、可协同工作的小的服务。（不过是宏观层面的解耦。）

1)可独立运行：微服务是一个个可以独立开发、独立部署、独立运行的系统或者进程。

2)可协同工作：采用了微服务架构后，整个系统被拆分成多个微服务，这些服务之间往往不是完全独立的，在业务上存在一定的耦合，即一个服务可能需要使用另一个服务所提供的功能。这就是所谓的可协同工作。与单服务应用不同的是，多个微服务之间的调用时通过RPC通信来实现，而非单服务的本地调用，所以通信的成本相对要高一些，但带来的好处也是可观的。（微服务之间使用轻量级API通过明确定义的接口进行通信。）

3)小而美：微服务的思想是，将一个拥有复杂功能的庞大系统，按照业务功能，拆分成多个相互独立的子系统，这些子系统则被称为微服务。每个微服务只承担某一项职责，从而相对于单服务应用来说，微服务的体积是小的。小也就意味着每个服务承担的职责变少，根据单一职责原则，我们在系统设计时，要尽量使得每一项服务只承担一项职责，从而实现系统的高内聚。

总之，微服务是一个架构设计方式，此架构中的每个服务(service)都是针对一组功能而设计的，并专注于解决特定的问题。（微服务技术主要包括Nacos注册发现中心（Ribbon负载均衡/Feign远程服务调用），Sentinel限流熔断和网关Gateway。）

当然微服务是离不开AOP的。

微服务不过是对系统分而治之，解决高并发带来的系统复杂性。可以简单类比为前后端代码分离，让专业的人做专业的事。同时还带来了“解耦”的好处，比如同一套前端代码可以供给不同语种的后端使用，只需提供相应的接口。

在微服务架构设计中，建议超出需要10人开发和维护的项目要进行系统拆分。就是把大系统拆分为很多小系统，几个人负责一个服务这样每个服务独立的开发、测试和上线。代码冲突少了，每次上线就回归测试自己的一个服务即可；测试速度快了，上线是独立的，只要向后兼容接口就行了，不需要跟别人等待和协调；技术架构和技术版本的升级，几个人ok就行，成本降低，更加灵活了。

雨田说码要求Maven版本最好保持3.8.2，但我先用着我的吧，后面确实有环境问题我再换。

首先创建父工程module sca，建议将其src删掉。

Spring Cloud Alibaba Version 2.2.6.RELEASE

Spring Cloud Version Hoxton.SR9

Spring Boot Version 2.3.2.RELEASE是相对较稳定的版本。

父工程sca的pom文件中需要统一定义当前工程及子工程中各个依赖的版本、可能需要的公共依赖、编译和运行版本。

2022/1/16按照雨田说码的步骤添加版本和依赖信息后，Maven报错/Unable to import maven project: See logs for details，看到这篇文章[我们都遇过的See logs for details](https://blog.csdn.net/weixin_40796213/article/details/105072147?ops_request_misc=&request_id=&biz_id=102&utm_term=see%20log%20for%20details&utm_medium=distribute.pc_search_result.none-task-blog-2~blog~sobaiduweb~default-2-105072147.nonecase&spm=1018.2226.3001.4450)的最后一句话我突然就明白了。然后我把Maven版本换回了我的3.3.9，然后<project>标签又报错/Failure to find org.springframework.boot:spring-boot-starter-parent:pom:2.3.2RELEASE in https://maven.aliyun.

com/repository/public/was cached in the local repository, resolution will not be reattempted until the update interval of has elapsed or updates are forced，大概意思就是说在aliyun仓库中找不到相应的依赖，然后看到了这篇文章[Failure to find org.springframework.cloud](https://blog.csdn.net/weixin_36833780/article/details/86593660?ops_request_misc=%257B%2522request%255Fid%2522%253A%2522164228967016780261946591%2522%252C%2522scm%2522%253A%252220140713.130102334..%2522%257D&request_id=164228967016780261946591&biz_id=0&utm_medium=distribute.pc_search_result.none-task-blog-2~blog~sobaiduend~default-2-86593660.nonecase&utm_term=failure+to+find+org.springfram&spm=1018.2226.3001.4450)，指定了一个官网的仓库地址解决了。

2022/1/16到Maven中央仓库搜maven-compiler-plugin依赖，搜了半天都不匹配，最后发现是单词compiler写成了complier…

微服务项目一般都会采用聚合工程结构，可通过聚合工程结构实现共享资源的复用，简化项目的管理方式。聚合工程结构示例：

GitCGB2108IVProjects (工作区/空项目) [聚合工程结构.png](file:///D:\Pkg\图片\聚合工程结构.png)，

├── 01-sca //(微服务父工程)

├── sca-provider //服务提供方法

├── sca-consumer //服务消费方法

├── sca-gateway //网关服务

工程与工程之间的通信就叫RPC。

如果在src下手动创建test目录，需要右键-->Mark Directory as Test Sources Root。

端口被占用如何解决。直接在resources目录创建一个yml文件，指定server.port即可。

如何创建SpringBoot启动类。关键在于@SpringBootApplication注解。

在微服务架构下，主要有三种角色：服务提供者（RPC Server）、服务消费者（RPC Client）和服务注册中心（Registry）。

RPC Server提供服务，在启动时，根据服务发布文件server.xml中的配置的信息，向Registry注册自身服务，并向Registry定期发送心跳汇报存活状态。

RPC Client调用服务，在启动时，根据服务引用文件client.xml中配置的信息，向Registry订阅服务，把Registry返回的服务节点列表缓存在本地内存中，并与RPC Server建立连接。

当RPC Server节点发生变更时，Registry会同步变更，RPC Client感知后会刷新本地内存中缓存的服务节点列表。

RPC Client从本地缓存的服务节点列表中，基于负载均衡算法选择一台RPC Server发起调用。

Ribbon、Nginx2、负载均衡

Ribbon1是Netflix发布的云中间层服务开源项目，其主要功能是提供客户端实现负载均衡算法。（协助控制HTTP和TCP的客户端的行为。）

负载均衡1是指分布式系统中各个节点能够均匀的处理请求负载。负载均衡可以改善工作负载分布，优化资源配置，最大化吞吐量，最小化响应时间。（提高系统的并发处理能力。）

为Ribbon配置服务提供者地址后，Ribbon就可基于某种负载均衡算法，自动地帮助服务消费者去请求。Ribbon默认为我们提供了很多负载均衡算法，例如轮询、随机等。当然，我们也可为Ribbon实现自定义的负载均衡算法。

Nginx2是客户端所有请求统一交给Nginx，由Nginx进行实现负载均衡请求转发，属于服务端负载均衡。即请求由Nginx服务器端进行转发。

Ribbon是从Eureka注册中心服务端上获取服务注册信息列表，缓存到本地，让后在本地实现轮训负载均衡策略。即在客户端实现负载均衡。（不就是正向代理？）

Nginx适用于服务端实现负载均衡比如搭配Tomcat，Ribbon适合与在微服务中RPC远程调用实现本地服务负载均衡，比如Dubbo、SpringCloud中都是采用本地负载均衡。

SpringCloud的Netflix中提供了Ribbon和Feign两个组件实现软负载均衡调用。

SpringCloud Netflix的微服务都是以HTTP接口的形式暴露的，所以可以用Apache的Http-Client或Spring的RestTemplate去调用，而Feign是一个使用起来更加方便的HTTP客户端，使用起来就像是调用自身工程的方法，而感觉不到是调用远程方法。

注意：spring-cloud-starter-feign里面已经包含了spring-cloud-starter-ribbon（Feign中也使用了Ribbon）。

[Nacos Docs](https://nacos.io/zh-cn/docs/what-is-nacos.html)，[微服务之注册中心](https://blog.csdn.net/u012921921/article/details/117297909?ops_request_misc=%257B%2522request%255Fid%2522%253A%2522164193041716780357268807%2522%252C%2522scm%2522%253A%252220140713.130102334.pc%255Fblog.%2522%257D&request_id=164193041716780357268807&biz_id=0&utm_medium=distribute.pc_search_result.none-task-blog-2~blog~first_rank_ecpm_v1~rank_v31_ecpm-5-117297909.nonecase&utm_term=%E6%B3%A8%E5%86%8C%E4%B8%AD%E5%BF%83&spm=1018.2226.3001.4450)，

Nacos、Feign、Sentinel、Gateway、SSO

Nacos1（DynamicNaming and Configuration Service）注册中心是一个应用于服务注册与发现、配置管理的平台。它孵化于阿里巴巴，成长于十年双十一的洪峰考验，沉淀了简单易用、稳定可靠、性能卓越的核心竞争力。（实现服务的动态发布和更新。）

在微服务中，首先需要面对的问题就是如何查找服务（软件即服务）。其次，就是如何在不同的服务之间进行通信。如何更好更方便的管理应用中的每一个服务，如何建立各个服务之间联系的纽带，由此注册中心诞生。（注册中心就像是Spring容器，而微服务就是工厂中的Bean。）

服务（Service）是Nacos世界的一等公民。

因为没有配置PATH，所以要到Nacos的bin目录下运行启动命令startup.cmd -m standalone。（现已配置IDEA中启动Nacos。）

NacosServer本质上就是一个web服务，端口默认为8848。

服务注册过程会发送一个心跳包到远端的Nacos，这个心跳包就是BeatInfo的实例。

RestTemplate写在配置类或主启动类以保证单例模式。

2022/1/19运行ProviderApplication报错/NacosException: failed to req API:/nacos/v1/ns/instance after all servers([localhost:8848]) tried: java.net.ConnectException: Connection refused: connect。

事实上，运行服务前应首先启动Nacos。（startup.cmd -m standalone。）

Additional command line parameters。<compilerArgs></compilerArgs>

[Nacos Shell Script.png](file:///D:\Pkg\图片\Nacos%20Shell%20Script.png)，

基于IDEA启动Nacos。

Edit Configurations… add Shell Script。Ctrl+C即可关闭。

基于IDEA连接数据库。（nacos\_config。）

URL:jdbc:mariadb://localhost:3307/nacos\_config?serverTimezone=Asia/Shanghai&characterEncoding=utf8

Right Click-->Database Tools-->Manage Shown Schemas…

Ctrl+H查看接口LoadBalancerClient的实现类，可以看到是由Ribbon实现的负载均衡。（或者打个断点debug运行。）

拦截器思想，为某个对象在执行时添加一些扩展功能。

Feign1是一种声明式Web服务客户端，底层封装了对REST技术的应用，通过Feign可以简化服务消费方对远程服务提供方法的调用实现。（只做声明，不做调用。由底层来调用。只写接口，不写实现类。由底层来代理。）

服务消费方基于REST方式请求服务提供方的服务时，一种直接的方式就是自己拼接URL，拼接参数然后实现服务调用，但每次服务调用都需要这样拼接，代码量复杂且不易维护，此时Feign诞生。

sca-provider如果写成sca\_provider，启动ConsumerApplication时会报错/Service id not legal hostname。

使用Feign方式实现远程服务调用的优点：声明式远程调用，优化结构，简化服务调用过程代码的编写。

[远程服务调用的四种方式.png](file:///D:\Pkg\图片\服务调用的四种方式.png)，

1)直接使用RestTemplate进行调用（IP+Port），不用注册Nacos。

2)使用LoadBalancerClient实现负载均衡。

3)@LoadBalanced+RestTemplate方式@LoadBalanced通过底层进行负载均衡实例的获取。

4)@EnableFeignClients+@FeignClient使用Feign进行服务调用。（多种负载均衡算法。）

学会使用Tools-->HTTP Client-->Test RESTful Web Service，避免频繁切换浏览器。（会在项目下生成一个Scratches目录，存放有历史记录。右上角Examples可以查看官方提供的HTTP请求规范。）

关于修改sca-consumer端默认的负载均衡策略。

方案1：修改Java代码指定负载均衡策略。

public IRule ribbonRule() {

return new RandomRule();

}

方案2：修改sca-consumer配置文件（applicaton.yml）以指定负载均衡策略。（未来可CV到配置中心，实现热部署。）

sca-provider:

ribbon:

NFLoadBalancerRuleClassName: com.netflix.loadbalancer.RandomRule

Nacos配置中心

阿里巴巴开发手册强制使用SLF4J作为开发日志门面。

添加spring-cloud-starter-alibaba-nacos-config依赖后需要将application.yml改为bootstrap.yml。

如果配置文件bootstrap.yml和配置中心均设置了日志级别，会先读取配置文件bootstrap.yml获得配置中心地址，然后再读取配置中心的日志级别。（意思是bootstrap.yml中的日志级别将被配置中心覆盖。）

在配置中心可以动态修改日志级别，相应的配置信息会存在nacos\_config.config\_info表中，前提是在E:\Nacos\conf\application.properties文件中配置了数据库。

@RefreshScope时告诉Spring当需要访问provider/log/doLog1时重新创建该对象以刷新属性private String logLevel;（内部感知。）

Nacos 配置管理模型由三部分构成：

1)Namespace对不同的环境进⾏隔离，⽐如隔离开发环境和⽣产环境。

2)Group将若⼲个服务或者若⼲个配置集归为⼀组。（可在不同时间节点切换不同的配置。）

3)Service/DataId某⼀个服务或配置集，一般对应一个配置文件。

Nacos新建DataID时注意不要忘了扩展名，app-public.yml。

Sentinel1（分布式系统的流量防卫兵）是阿里开源的一套用于服务容错的综合性解决方案。它以流量为切入点，从流量控制、熔断降级、系统负载保护等多个维度来保护服务的稳定性。

SentinelRPC Server承接了阿里巴巴近RPC Server10RPC Server年的双十一大促流量的核心场景，例如秒杀（即突发流量控制在系统容量可以承受的范围）、消息削峰填谷、集群流量控制、实时熔断下游不可用应用等。

[IDEA启动Sentinel.png](file:///D:\Pkg\图片\IDEA启动Sentinel.png)，

到jar包路径下打开cmd运行指令---

java -Dserver.port=8180 -Dcsp.sentinel.dashboard.server=localhost:8180 -Dproject.name=sentinel-dashboard -jar sentinel-dashboard-1.8.1.jar

浏览器访问http://localhost:8180即可打开Sentinel控制台，登录账号密码均为sentinel。

配制好Sentinel浏览器访问http://localhost:8082/provider/sentinel，多次刷新可在dashboard实时监控。到簇点链路配置对应的流控规则，然后频繁访问provider/sentinel则会触发BlockExceptionHandler的实现类DefaultBlockExceptionHandler，提示Blocked by Sentinel (flow limiting)。

在dashboard机器列表可以看到机器名/IP/端口/心跳时间等，端口默认为8720。8081客户端访问sca-provider，8180客户端访问Sentinel，而sca-provider通过此8720端口访问Sentinel。

浏览器访问http://localhost:8180即可打开Sentinel控制台，登录账号密码均为sentinel。

在sca-provider添加sentinel限流依赖后，底层会注入一个SpringMVC拦截器HandlerInter-ceptor，此拦截器就是限流的关键，Ctrl+H查看其继承结构，有一个com.alibaba.cap.senti-nel.adapter.spring.webmvc写的实现类AbstractSentinelInterceptor，工作时由它的preHandle()调用DefaultBlockExceptionHandler的handle方法。

[Sentinel限流过程分析.png](file:///D:\Pkg\图片\Sentinel限流过程分析.png)，[HandlerInterceptor拦截器原理.png](file:///D:\Pkg\图片\HandlerInterceptor拦截器原理.png)，

执行链ExecutionChain包含拦截器HandlerInterceptor和Controller。

Sentinel的流控模式包括直接，还有关联，链路。

流控效果包括快速失败，Warm Up，排队等待。（默认直接/快速失败。）

关联限流是指对关联资源频繁访问时，限流当前资源。

链路限流只记录指定链路入口的流量。也就是当多个服务对指定资源调用时，假如流量超出了指定阈值，则进行限流。

被多个服务访问的资源加以@SentinelResource注解，该资源所在方法同时也是一个AOP的切入点方法，这个方法执行时，会根据流控规则对相应的应用进行限流。

@SentinelResource注解收录于com.alibaba.cap.sentinel.annotation，xxx.annotation.aspectj还有个SentinelResourceAspect类，里面指定了切入点@Pointcut("@annotation(com.alibaba.csp.sen-tinel.annotation.SentinelResource)")。

对公共资源新增流控规则时填写拦截器指定的默认入口资源sentinel\_spring\_web\_context。

如果想对指定的特定链路进行限流，需要到application.yml添加如下语句：

sentinel:

web-context-unify: false

java.util.concurrent.atomic.AtomicLong是长整型的线程安全对象，基于此对象实现自增、自减操作时是线程安全的。直接使用count++无法保证多线程下的原子性。

簇点链路新增降级规则时的最大RT即响应时间ResponseTime，实际访问时响应时间超过最大RT就触发熔断。

Gateway1是指将两个使用不同协议的网络段连接在一起的设备。它的作用就是对两个网络段中的使用不同传输协议的数据进行互相的翻译转换。例如，一个商业内部局域网就常常需要通过网关发送电子邮件到Internet的相关地址。（Gateway是API网关，区别于网络网关。）

网关本质上要提供一个各种服务访问的入口，并提供服务接收并转发所有内外部的客户端调用，还有就是权限认证，限流控制等等。SpringCloud Gateway是Spring公司基于Spring 5.0和SpringBoot 2.0等技术开发的一个网关组件，它旨在为微服务架构提供一种简单有效的统一的API入口，负责服务请求路由、组合及协议转换，并且基于Filter链的方式提供了权限认证，监控、限流等功能。

SpringCloud Gateway性能强劲，功能强大，设计优雅，容易扩展。但由于依赖Netty与WebFlux(Spring5.0)，不是传统的Servlet编程模型（Spring MVC就是基于此模型实现），学习成本高。（Hibernate不也是学习成本高嘛。）

Gateway再往外加就是Nginx。

依赖spring-cloud-starter-gateway自带一个基于Netty实现的Web服务，与spring-boot-starter-web互斥。

Gateway需要承担的并发量更大，相较于Tomcat，Netty能有更好的性能表现。但Netty只是一个网络编程框架，拿来后还需要二次开发，适合有技术实力的公司选用。

Gateway底层由Ribbon自动实现负载均衡，详见GlobalFilter接口。

Predicate断言/谓词，用于条件判断，只有断言结果都为真，才会真正的执行路由。断言其本质就是定义路由转发的条件。Predicate有一个谓词工厂类PathRoutePredicateFactory。

StripPrefix有一个去前缀的工厂类StripPrefixGatewayFilterFactory。

Gateway的Filter从作用范围可分为GatewayFilter和GlobalFilter。GatewayFilter应用到单个路由或者一个分组的路由上。GlobalFilter应用到所有的路由上（例如负载均衡过滤器，请求转发过滤器等）。

org.springframework.core有一个Ordered接口，实现此接口一般是为了启用优先级。

org.apache.http有一个HttpStatus接口，包含了Http请求状态码。

SSO1是指多个站点共用一台认证授权服务器，用户在其中任何一个站点登录后，可以免登录访问其他所有站点。而且，各站点间可以通过该登录状态直接交互。

[docker docs](https://docs.docker.com/)，[Docker教程/菜鸟教程](https://www.runoob.com/docker/docker-tutorial.html)，

Docker、Kubernetes、RabbitMQ

Docker1是一个开源的应用容器引擎，让开发者可以打包他们的应用以及依赖包到一个可移植的镜像中，然后发布到任何流行的Linux或Windows操作系统的机器上，也可以实现虚拟化。Docker诞生于2013年初，基于Google公司的Go语言进行实现。（解决“为什么在我的机器上能用，在你的机器上就不能用”的问题。个人认为这还是解耦，另外Docker又是更高层次的“微服务”。）

Kubernetes（K8s）是一个开源的，用于管理云平台中多个主机上的容器化的应用，Kubernetes的目标是让部署容器化的应用简单并且高效，Kubernetes提供了应用部署，规划，更新，维护的一种机制。

传统的应用部署方式是通过插件或脚本来安装应用。这样做的缺点是应用的运行、配置、管理、所有生存周期将与当前操作系统绑定，这样做并不利于应用的升级更新/回滚等操作，当然也可以通过创建虚拟机的方式来实现某些功能，但是虚拟机非常重，并不利于可移植性。

新的方式是通过部署容器方式实现，每个容器之间互相隔离，每个容器有自己的文件系统，容器之间进程不会相互影响，能区分计算资源。相对于虚拟机，容器能快速部署，由于容器与底层设施、机器文件系统解耦的，所以它能在不同云、不同版本操作系统间进行迁移。

容器占用资源少、部署快，每个应用可以被打包成一个容器镜像，每个应用与容器间成一对一关系也使容器有更大优势，使用容器可以在build或release的阶段，为应用创建容器镜像，因为每个应用不需要与其余的应用堆栈组合，也不依赖于生产环境基础结构，这使得从研发到测试、生产能提供一致环境。类似地，容器比虚拟机轻量、更“透明”，这更便于监控和管理。

[RabbitMQ](https://www.rabbitmq.com/)，[Apache RocketMQ](https://rocketmq.apache.org/)，[Apache Kafka](https://kafka.apache.org/)，

[RabbitMQ.png](file:///D:\Pkg\图片\RabbitMQ.png)，[MQ消息队列详解、四大MQ的优缺点分析](https://blog.csdn.net/qq_44240587/article/details/104630567?ops_request_misc=%257B%2522request%255Fid%2522%253A%2522163816386716780264018131%2522%252C%2522scm%2522%253A%252220140713.130102334..%2522%257D&request_id=163816386716780264018131&biz_id=0&utm_medium=distribute.pc_search_result.none-task-blog-2~all~top_positive~default-1-104630567.first_rank_v2_pc_rank_v29&utm_term=mq&spm=1018.2226.3001.4187)，

[Kafka、RabbitMQ、RocketMQ等消息中间件的介绍和对比](https://blog.csdn.net/yunfeng482/article/details/72856762?ops_request_misc=%257B%2522request%255Fid%2522%253A%2522163992661816780274154502%2522%252C%2522scm%2522%253A%252220140713.130102334..%2522%257D&request_id=163992661816780274154502&biz_id=0&utm_medium=distribute.pc_search_result.none-task-blog-2~blog~sobaiduend~default-2-72856762.pc_v2_rank_blog_default&utm_term=rabbit%E5%92%8Crocketmq%E7%9A%84%E5%8C%BA%E5%88%AB&spm=1018.2226.3001.4450)，

[rabbitmq、kafka、rocketMQ区别和选择](https://blog.csdn.net/a718515028/article/details/116613960?ops_request_misc=%257B%2522request%255Fid%2522%253A%2522163992661816780274154502%2522%252C%2522scm%2522%253A%252220140713.130102334..%2522%257D&request_id=163992661816780274154502&biz_id=0&utm_medium=distribute.pc_search_result.none-task-blog-2~blog~sobaiduend~default-1-116613960.pc_v2_rank_blog_default&utm_term=rabbit%E5%92%8Crocketmq%E7%9A%84%E5%8C%BA%E5%88%AB&spm=1018.2226.3001.4450)，

RabbitMQ1是一个由Erlang语言编写的、开源的、在AMQP基础上完整的、可复用的企业消息系统。AMQP（advanced message queuing protocol）是指一个提供统/消息服务的应用层标准高级消息队列协议，是应用层协议的一个开放标准，为面向消息的中间件设计。基于此协议的客户端与消息中间件可传递消息并不受客户端/中间件不同产品、不同开发语言等

条件的限制。

Java中进程通讯（IPC）主要是Socket，MQ等。

使用消息队列的主要作用是系统解耦，异步调用，流量削峰。

一般的业务系统要引入MQ，最早大家都用ActiveMQ，但是现在确实大家用的不多了，没经过大规模吞吐量场景的验证，社区也不是很活跃，不推荐使用。

后来大家开始用RabbitMQ，但是确实Erlang语言阻止了大量的Java工程师去深入研究和掌控他，对公司而言，几乎处于不可控的状态，但是确实人是开源的，比较稳定的支持，活跃度也高。

不过现在确实越来越多的公司，会去用RocketMQ，确实很不错，但要考虑好社区万一突然黄掉的风险，对自己公司技术实力有绝对自信的，推荐使用RocketMQ，不然还是建议用RabbitMQ，毕竟是活跃开源社区，绝对不会黄。（我想到Layui了…）

所以中小型公司，技术实力较为一般，技术挑战不是特别高，用RabbitMQ是不错的选择；大型公司，基础架构研发实力较强，用RocketMQ是更好的选择。

如果是大数据领域的实时计算、日志采集等场景，用Kafka是业内标准的，绝对没问题，社区活跃度很高，绝对不会黄，何况几乎是全世界这个领域的事实性规范。

[Elastic Stack and Product Documentation](https://www.elastic.co/guide/index.html)，[Elasticsearch系列](https://wanght.blog.csdn.net/?type=blog&year=2020&month=12)，[Elasticsearch和Solr选型](https://blog.csdn.net/qq_39974376/article/details/107144576?ops_request_misc=%257B%2522request%255Fid%2522%253A%2522163998342616780265448713%2522%252C%2522scm%2522%253A%252220140713.130102334.pc%255Fblog.%2522%257D&request_id=163998342616780265448713&biz_id=0&utm_medium=distribute.pc_search_result.none-task-blog-2~blog~first_rank_v2~rank_v29-7-107144576.pc_v2_rank_blog_default&utm_term=elasticsearch%E5%92%8Csolr%E5%92%8Clucene&spm=1018.2226.3001.4450)，

Elasticsearch、Lucene、Seata

Elasticsearch1是一个分布式、高扩展、高实时的搜索与数据分析引擎，基于RESTfulweb接口。它能很方便的使大量数据具有搜索、分析和探索的能力。充分利用Elasticsearch的水平伸缩性，能使数据在生产环境变得更有价值。Elasticsearch的实现原理主要分为以下几个步骤，首先用户将数据提交到Elasticsearch数据库中，再通过分词控制器去将对应的语句分词，将其权重和分词结果一并存入数据，当用户搜索数据时候，再根据权重将结果排名，打分，再将返回结果呈现给用户。

Elasticsearch是用Java语言开发的，并作为Apache许可条款下的开放源码发布，是一种流行的企业级搜索引擎。Elasticsearch用于云计算中，能够达到实时搜索，稳定，可靠，快速，安装使用方便。根据DB-Engines的排名显示，Elasticsearch是最受欢迎的企业搜索引擎，其次是ApacheSolr，也是基于Lucene。

Elasticsearch是与名为Logstash的数据收集和日志解析引擎以及名为Kibana的分析和可视化平台一起开发的。这三个产品被设计成一个集成解决方案，称为“Elastic Stack”（以前称为“ELK stack”）。

[Apache Lucene](https://lucene.apache.org/)，[Solr](https://solr.apache.org/)，

Lucene是apache软件基金会jakarta项目组的一个子项目，是一个开放源代码的全文检索引擎工具包，但它不是一个完整的全文检索引擎，而是一个全文检索引擎的架构，提供了完整的查询引擎和索引引擎，部分文本分析引擎（英文与德文两种西方语言）。Lucene的目的是为软件开发人员提供一个简单易用的工具包，以方便的在目标系统中实现全文检索的功能，或者是以此为基础建立起完整的全文检索引擎。Lucene是一套用于全文检索和搜寻的开源程式库，由Apache软件基金会支持和提供。Lucene提供了一个简单却强大的应用程式接口，能够做全文索引和搜寻。在Java开发环境里Lucene是一个成熟的免费开源工具。就其本身而言，Lucene是当前以及最近几年最受欢迎的免费Java信息检索程序库。人们经常提到信息检索程序库，虽然与搜索引擎有关，但不应该将信息检索程序库与搜索引擎相混淆。

Elasticsearch和Solr都是基于Lucene实现的。

Seata1是一款开源的分布式事务解决方案，致力于提供高性能和简单易用的分布式事务服务。Seata将为用户提供了AT、TCC、SAGA和XA事务模式，为用户打造一站式的分布式解决方案。

2019年1月，阿里巴巴中间件团队发起了开源项目Fescar（Fast & EaSy Commit And Rollback），和社区一起共建开源分布式事务解决方案。Fescar的愿景是让分布式事务的使用像本地事务的使用一样，简单和高效，并逐步解决开发者们遇到的分布式事务方面的所有难题。Fescar开源后，蚂蚁金服加入Fescar社区参与共建，并在Fescar0.4.0版本中贡献了TCC模式。为了打造更中立、更开放、生态更加丰富的分布式事务开源社区，经过社区核心成员的投票，大家决定对Fescar进行品牌升级，并更名为Seata，意为：Simple Extensible Autonomous Transaction Architecture，是一套一站式分布式事务解决方案。Seata融合了阿里巴巴和蚂蚁金服在分布式事务技术上的积累，并沉淀了新零售、云计算和新金融等场景下丰富的实践经验，但要实现适用于所有的分布式事务场景的愿景，仍有很长的路要走。

Seata的AT模式（Automatic Transaction）是一种无侵入的分布式事务解决方案。

大数据

[Java生态体系.png](file:///D:\Pkg\图片\Java\java生态体系.png)，[大数据生态体系.png](file:///D:\Pkg\图片\java\大数据生态体系.png)，[大数据部门内组织架构.png](file:///D:\Pkg\图片\Java\大数据部门内组织架构.png)，

[加米谷大数据张老师的博客\_CSDN博客-大数据领域博主](https://blog.csdn.net/shuimuzh123?type=blog)，

[大数据专业应该怎么学，求大佬指引方向? - 知乎 (zhihu.com)](https://www.zhihu.com/question/452344290)，

[2021年超全超详细的最新大数据开发面试题，附答案解析\_数据人生coding的博客-CSDN博客\_大数据开发面试](https://blog.csdn.net/helloHbulie/article/details/113555820)，

[java转大数据方向学习路线\_森林屿麓的博客-CSDN博客\_java转大数据路线](https://blog.csdn.net/u013967628/article/details/83656560)，

[好文！终于有人把云计算、大数据、人工智能讲明白了 (qq.com)](https://mp.weixin.qq.com/s/sV9-MsFLLWe6OiRX65vysA)，

[有哪些好的数据来源或者大数据平台？ - 知乎 (zhihu.com)](https://www.zhihu.com/question/27798279)，

大数据1是以容量大、类型多、存取速度快、价值巨大但价值密度低为主要特征的数据集合，由于这些数据本身规模巨大、来源分散、格式多样，所以需要新的体系架构、技术、算法和分析方法来对这些数据进行采集、存储和关联分析，以期望能够从中抽取出隐藏的有价值的信息。

三大分布式计算系统：Hadoop适合处理离线的静态的大数据；Spark适合处理离线的流式的大数据；Storm/Flink适合处理在线的实时的大数据。

由于深度优先搜索算法可能会使爬虫系统陷入一个网站内部，不利于搜索比较靠近网站首页的网页信息，因此一般采用广度优先搜索算法采集网页。

一般来讲，数据挖掘的结果并不要求是完全准确的知识，而是发现一种大的趋势。

分布式存储在系统的横向扩展性、存储成本、文件读取速度方面有着显著的优势。

随着大数据时代的到来，Redis、MongoDB和HBase等NoSQL数据库也常用于数据的采集。

Hadoop、Spark、ZooKeeper

[Apache Hadoop 3.3.x](https://hadoop.apache.org/docs/current/)，[Apache Hadoop Main 3.3.x API](https://hadoop.apache.org/docs/current/api/)，

[Hadoop1.x 2.x 3.x的区别.png](file:///D:\Pkg\图片\java\Hadoop1.x%202.x%203.x的区别.png)，[HDFS、YARN、MapReduce三者关系.png](file:///D:\Pkg\图片\java\HDFS、YARN、MapReduce三者关系.png)，

[史上最详细的Hadoop环境搭建\_似水流年的博客-CSDN博客\_hadoop环境搭建](https://blog.csdn.net/hliq5399/article/details/78193113)，

[MapReduce简介和入门 - Hadoop教程 (yiibai.com)](https://www.yiibai.com/hadoop/intro-mapreduce.html)，

[Apache Hadoop 中文文档](https://hadoop.org.cn/docs/)，

[Hadoop 教程\_w3cschool](https://www.w3cschool.cn/hadoop/)，[1.0 Hadoop 教程 | 菜鸟教程 (runoob.com)](https://www.runoob.com/w3cnote/hadoop-tutorial.html)，

[曹世宏的博客的博客\_CSDN博客-网络安全,路由交换,linux领域博主](https://cshihong.blog.csdn.net/?type=blog)，

[Lucene,Nutch,Hadoop 之间有什么关系？\_mypods的博客-CSDN博客](https://blog.csdn.net/mypods/article/details/10299067)，

Hadoop1是分布式管理、存储、计算的生态系统；包括HDFS（存储）、MapReduce（计算）、Yarn（资源调度）。

如果说安装hadoop集群，其实就是安装了两个东西：一个操作系统YARN和一个文件系统HDFS。其实MapReduce就是运行在YARN之上的应用。

从理论基础来说，Spring的核心思想是解耦，Hadoop的核心思想是分布式。

基于Hadoop的类SQL数据仓库工具，Hive。

基于Hadoop的列式分布式NoSQL数据库，Hbase。

MapReduce中，mapper阶段处理的数据如何传递给reducer阶段，是MapReduce框架中最关键的一个流程，这个流程就叫Shuffle。（三层代码结构中有一个持久层mapper。）

Hadoop2.x增加了Yarn。自此MapReduce只负责运算，资源调度的任务由Yarn来完成。Hadoop3.x在组成上没有变化。

Hadoop三大发行版本：Apache、Cloudera、Hortonworks。

Apache最原始（最基础）的版本，对初学者友好。2006

Cloudera内部集成了很多大数据框架，对应产品CDH。2008

Hortonworks文档较好，对应产品HDP。2011

Hortonworks现在已经被Cloudera公司收购，推出新的品牌CDP。

Spark1是分布式计算平台，是一个用Scala语言编写的计算框架，基于内存的快速、通用、可扩展的大数据分析引擎。

Spark提供了Spark RDD、Spark SQL、Spark Streaming、Spark MLlib、Spark GraphX等技术组件，可以一站式地完成大数据领域的离线批处理、交互式查询、流式计算、机器学习、图计算等常见的任务。

Spark并不是要成为一个大数据领域的“独裁者”，一个人霸占大数据领域所有的“地盘”，而是与Hadoop进行了高度的集成，两者可以完美的配合使用。Hadoop的HDFS、Hive、HBase负责存储，YARN负责资源调度；Spark负责大数据计算。实际上，Hadoop+Spark的组合，是一种“double win”的组合。

Spark核心在于内存计算模型代替Hadoop生态的MapReduce离线计算模型，用更加丰富的Transformation和Action算子来替代map，reduce两种算子。

Hadoop是批处理工具，更擅长处理离线数据，而Spark在内存中处理数据，可以是实时处理。

弹性分布式数据集（RDD）是Spark的基本数据结构。 它是一个不可变的分布式对象集合。 RDD中的每个数据集被划分为逻辑分区，其可以在集群的不同节点上计算。 RDD可以包含任何类型的Python，Java或Scala对象，包括用户定义的类。

RDD可被分发到集群各个节点上，进行并行操作。RDDs 可以通过Hadoop InputFormats创建（如HDFS），或者从其他RDDs转化而来。

Spark相对Hadoop的优越性：

1）Spark基于RDD（Resilient Distributed Datasets），数据并不存放在RDD中，只是通过RDD进行转换，通过装饰者设计模式，数据之间形成血缘关系和类型转换；

2）Spark用Scala语言编写，相比java语言编写的Hadoop程序更加简洁；

3）相比Hadoop中对于数据计算只提供了Map和Reduce两个操作，Spark提供了丰富的算子，可以通过RDD转换算子和RDD行动算子，实现很多复杂算法操作，这些在复杂的算法在Hadoop中需要自己编写，而在Spark中直接通过Scala语言封装好了，直接用就ok；

4）Hadoop中对于数据的计算，一个Job只有一个Map和Reduce阶段，对于复杂的计算，需要使用多次MR，这样涉及到落盘和磁盘IO，效率不高；而在Spark中，一个Job可以包含多个RDD的转换算子，在调度时可以生成多个Stage，实现更复杂的功能；

5）Hadoop中中间结果存放在HDFS中，每次MR都需要刷写/调用，而Spark中间结果优先存放在内存中，内存不够再存放在磁盘中，不放入HDFS，避免了大量的IO和刷写读取操作；

6）Hadoop适合处理静态数据，对于迭代式流式数据的处理能力差；Spark通过在内存中缓存处理的数据，提高了处理流式数据和迭代式数据的性能。

[Apache ZooKeeper](https://zookeeper.apache.org/)，[ZooKeeper 中文网](https://zookeeper.net.cn/index.html)，

[Zookeeper 教程\_w3cschool](https://www.w3cschool.cn/zookeeper/)，[1.0 Zookeeper 教程 | 菜鸟教程 (runoob.com)](https://www.runoob.com/w3cnote/zookeeper-tutorial.html)，

ZooKeeper1是分布式应用程序协调服务，负责协调其他的大数据框架，我们会用即可。

HDFS、MapReduce、Yarn

HDFS1是一种旨在在商品硬件上运行的分布式文件系统。它与现有的分布式文件系统有许多相似之处。但是，与其他分布式文件系统的区别很明显。HDFS具有高度的容错能力，旨在部署在低成本硬件上。HDFS提供对应用程序数据的高吞吐量访问，并且适用于具有大数据集的应用程序。HDFS放宽了一些POSIX要求，以实现对文件系统数据的流式访问。HDFS最初是作为Apache Nutch Web搜索引擎项目的基础结构而构建的。HDFS是Apache Hadoop Core项目的一部分。

NameNode存储文件的元数据，是用于指挥其它节点存储的节点。任何一个File System都需要具备根据文件路径映射到文件的功能，NameNode就是用于储存这些映射信息并提供映射服务的计算机，在整个 HDFS 系统中扮演“管理员”的角色，因此一个 HDFS 集群中只有一个NameNode。

DataNode 是用来储存数据块的节点。当一个文件被NameNode承认并分块之后将会被储存到被分配的DataNode中去。DataNode具有储存数据、读写数据的功能，其中存储的数据块比较类似于硬盘中的“扇区”概念，是 HDFS 存储的基本单位。

Secondary NameNode每隔一段时间对NameNode元数据进行备份，是命名节点的“秘书”。这个形容很贴切，因为它并不能代替NameNode的工作，无论NameNode是否有能力继续工作。它主要负责分摊NameNode的压力、备份NameNode的状态并执行一些管理工作，如果NameNode要求它这样做的话。如果NameNode坏掉了，它也可以提供备份数据以恢复NameNode。Secondary NameNode可以有多个。

与磁盘文件系统相似，HDFS中fsck指令可以显示块信息。例如，执行以下命令将列出文件系统中各个文件由哪些块构成：

％ hdfs fsck ／ -files -blocks

MapReduce1将计算过程分为两个阶段：Map和Reduce（映射和规约）。

1）Map阶段并行处理输入数据

2）Reduce阶段对Map结果进行汇总

Yarn1（Yet Another Resource Negotiator）

Hive、HBase、Flink、Storm

Hive1是数据仓库的工具，需要重点掌握，Hive的底层存储是用的HDFS，计算使用的是MapReduce或者Spark。

HBase1

Flink1目前非常的火，很多大厂都在转向Flink，在实时计算方面，Flink一骑绝尘，所以需要我们重点掌握，需要懂底层原理。

[Storm（流计算）技术原理\_曹世宏的博客的博客-CSDN博客\_storm 流](https://blog.csdn.net/qq_38265137/article/details/80547695)，

Storm1

数据有静态数据和流数据。

对静态数据和流数据的处理，对应着两种截然不同的计算模式：批量计算和实时计算。

流计算秉承一个基本理念，即数据的价值随着时间的流逝而降低，如用户点击流。因此，当事件出现时就应该立即进行处理，而不是缓存起来进行批量处理。为了及时处理流数据，就需要一个低延迟、可扩展、高可靠的处理引擎.

对于一个流计算系统来说，它应达到如下需求：高性能，海量式，实时性，分布式，易用性，可靠性。

流计算适合于需要处理持续到达的流数据、对数据处理有较高实时性要求的场景。

1）实时分析：如实时日志处理、交通流量分析等。

2）实时统计：如网站的实时访问统计、排序等。

3）实时推荐：如实时的广告定位、时间营销等。

Hadoop设计的初衷是面向大规模数据的批量处理，不适合流处理。

MapReduce是专门面向静态数据的批量处理的，内部各种实现机制都为批处理做了高度优化，不适合用于处理持续到达的动态数据。

流计算处理流程的三个阶段：数据实时采集、数据实时计算、实时查询服务。

Sqoop、Flume、Kafka

Sqoop1

Flume1

Kafka1

前端、数据库

[HTML.COM](https://html.com/)，[HTML教程/菜鸟教程](https://www.runoob.com/html/html-tutorial.html)，[HTML参考手册/菜鸟教程](https://www.runoob.com/tags/html-reference.html)，

HTML、CSS、JavaScript、JSON

HTML1文本是由HTML命令组成的描述性文本，HTML命令可以说明文字，图形、动画、声音、表格、链接等。

超文本是一种组织信息的方式，它通过超级链接方法将文本中的文字、图表与其他信息媒体相关联。这些相互关联的信息媒体可能在同一文本中，也可能是其他文件，或是地理位置相距遥远的某台计算机上的文件。这种组织信息方式将分布在不同位置的信息资源用随机方式进行连接，为人们查找，检索信息提供方便。

在<link>标签中，rel="stylesheet"指关联到一个CSS文档。

<dl>define list，<dt>define term，<dd>define description。

<dt hidden="hidden">value of hidden?</dt>

hidden属性功能是通知浏览器不渲染元素，使该元素处于不可见状态。但依然会创建元素中的内容，也就是说页面装载后允许使用JavaScript脚本将该属性取消，取消后该元素变为可见状态，同时元素中的内容也即时显示出来。

radio、checkbox设置默认值是checked="checked"，

select\_option设置默认值是selected="selected"。（都跟default没什么关系。）

The <div> element has no particular semantic meaning. It simply defines a block of content which can be styled as a single unit on a page.

网页快速定位到指定元素，右键-->检查。

HTML5是公认的下一代Web语言，极大地提升了Web在富媒体、富内容和富应用等方面的能力，被喻为终将改变移动互联网的重要推手。InternetExplorer8及以前的版本不支持。

[盒子模型.png](file:///D:\Pkg\图片\盒子模型.png)，[CSS及JS文档中文注释代码在浏览器中显示乱码](https://blog.csdn.net/lamanchas/article/details/78076288)，

CSS1（CascadeStyleSheet）专门用来修饰HTML网页，增强网页的展示能力。

行内CSS给当前行修饰；内部CSS在head标签中使用<style></style>添加修饰；外部CSS引用外部.css文件。

<div style="color: red;">div</div>

<style></style>

<link rel="stylesheet" type="text/css" href="register.css" />

在<link>标签中，rel="stylesheet"指关联到一个CSS文档。

类选择器，Id选择器，标签名选择器，分组选择器，属性选择器。

CSS认为HTML的所有元素都是一个一个的盒子。

table属性cellspacing和cellpadding就对应盒子模型的margin和padding。

border-width的四个参数分别设置容器的上、右、下、左边框。（时钟方向。）

伪元素不存在在DOM文档中，是虚拟的元素，是创建新元素。代表某个元素的子元素，这个子元素虽然在逻辑上存在，但却并不实际存在于文档树中。

[JavaScript Index](https://www.javatpoint.com/javascript-tutorial)，[typeof关键字.png](file:///D:\Pkg\图片\typeof关键字.png)，

JavaScript1是一门基于对象和事件驱动的脚本语言，通常用来提高网页与用户的交互性。

JS可以比作赋予静态网页了灵魂，它让静态的HTML骨架生动灵活了起来。

基于对象：它不仅可以new对象，也能使用现有的对象。JS没有类的概念，也没有编译的过程。直接执行的就是源代码。

事件驱动：在JS中，大部分情况下都是通过事件触发驱动函数执行的，从而实现特定的功能。

脚本语言：在网络前端开发环境下，用于嵌入在客户端浏览器中的一段小程序。

JS是弱类型语言，所有类型的变量都是用var关键字定义，在需要的时候会自动进行数据间的类型转换。

JS具有良好的跨平台性，只要有浏览器的地方都能执行JS。Java的跨平台性是因为有JVM。

JS里面很难报错，除非你打开浏览器开发者模式。

OOM是指内存溢出OutOfMemory，可能由死循环导致。

浏览器的控制台可以输出也可以输入（源代码）。

JavaScript对象：

1)内置对象，Arguments Array Boolean Date Function Number String Regexp Math Object以及6种Error对象。

2)宿主对象，运行环境提供的对象。Window Documen Element form image。

3)自定义对象。

window.sessionStorage.setItem("token",token)

localStorage和sessionStorage属性允许在浏览器中存储K/V数据。

sessionStorage用于临时保存同一窗口（或标签页）的数据，在关闭窗口或标签页之后将会删除这些数据。LocalStorage保存的数据不会过期，可以手动删除。

基本数据类型：number/string/boolean/null/undefined/ symbol（es6新增）。（Array是对象！）

在JS中，数值类型只有一种，就是浮点型number。

Infinity：正无穷大

-Infinity：负无穷大

NaN：非数字，和任何值都不相等，包括它本身。

byte count = 1;(var count = 1;)

count++; //在Java/JavaScript中这三行代码有什么区别？

count = count + 1;

count += 1;

js中的typeof类似于java中的instanceof（强转时加判断以增强健壮性），js也有instanceof。

instanceof checks if the object is an instance of given type, typeof checks the type of object.

js中定义数组用中括号，java中定义数组用大括号。

js中的判断条件可以不是boolean类型，在需要的时候会自动进行数据间的类型转换。

var person={fname:"John",lname:"Doe",age:25}; //在js中看到{}千万不要以为是数组…

JavaScript中的一些比较特别的类型转换：

!!String("") //返回false

!!Boolean("") //返回false

!!Number("") //返回false

!!Boolean([]) //返回true

!!String([]) //返回false

!!Number([]) //false

!!String(0) //返回true

!!Number(0) //返回false

!!Boolean(0) //返回false

[]转为字符串是"" //String([])返回""

[]转为数字是0 //Number([])返回0

[]转为布尔值是true //Boolean([])返回true

true转为数字是1 //Number(true)返回1

false转为数字是0 //Number(false)返回0

空对象转换为布尔类型为true。

ES6新增字符串方法`${}`，可以配合反单引号完成拼接字符串的功能。（模板字符串。）

let a='Karry Wang';

let str=`I love ${a}, because he is handsome.`;（还有个可以换行拼接字符串的功能）

alert(str);

在JavaScript中，调用对象属性的描述中，以下代码错误的是? C

A)obj["arr"]

B)obj["a"+"r"+"r"]

C)obj{"arr"}

D)obj.arr

下面结果为真的表达式是？C

A)null instanceof Object

B)null === undefined

C)null == undefined

D)NaN == NaN

[JSON中文网](http://www.json.org.cn/)，

[以下json格式数据，错误的是？牛客网](https://www.nowcoder.com/test/question/done?tid=50929021&qid=56255#summary)，

JSON1（JavaScript Object Notation，JS对象简谱）是一种轻量级的数据交换格式。它基于ECMAScript（欧洲计算机协会制定的js规范）的一个子集，采用完全独立于编程语言的文本格式来存储和表示数据。简洁和清晰的层次结构使得JSON成为理想的数据交换语言。易于人阅读和编写，同时也易于机器解析和生成，并有效地提升网络传输效率。

JSON基于两种结构：

键值对的集合。不同的编程语言中，它被理解为对象（object），纪录（record），结构（struct），字典（dictionary），哈希表（hashtable），有键列表（keyedlist），或者关联数组（associativearray）。

值的有序列表。在大部分语言中，它被实现为数组（array），矢量（vector），列表（list），序列（sequence）。

JSON是JS对象的字符串表示法，它使用文本表示一个JS对象的信息，本质是一个字符串。

JSON和XML的可读性可谓不相上下，一边是简易的语法，一边是规范的标签形式，很难分出胜负。但看了代码会发现，JSON简单的语法格式和清晰的层次结构明显要比XML容易阅读，并且在数据交换方面，由于JSON所使用的字符要比XML少得多，可以大大得节约传输数据所占用的带宽。

XML天生有很好的扩展性，JSON当然也有，没有什么是XML可以扩展而JSON却不能扩展的。不过JSON在Javascript主场作战，可以存储Javascript复合对象，有着xml不可比拟的优势。

ECMAScript是由网景的布兰登·艾奇开发的一种脚本语言的标准化规范；最初命名为Mocha，后来改名为LiveScript，最后重命名为JavaScript。

ECMAScript 6（简称ES6）是于2015年6月正式发布的JavaScript语言的标准，正式名为ECMAScript 2015（ES2015）。它的目标是使得JavaScript语言可以用来编写复杂的大型应用程序，成为企业级开发语言。

Gson是谷歌官方推出的支持 JSON/Java Object相互转换的Java序列化/反序列化库。

JS内置对象之一JSON，可以完成JSON字符串和原生JavaScript对象之间的互转：

给服务器发送数据，JS对象转JSON字符串，JSON.stringify(JS对象)；（不是stringfy！）

接受服务器的数据，JSON字符串转JS对象，JSON.parse("JSON字符串")。

DMO、jQuery、Ajax、Axios

DOM1是一项W3C（World Wide Web Consortium）标准，DOM文档对象模型为JS操作html文档所提供的一套API，通过这套API可以很方便的对html元素进行访问及增删改查操作。

获取对象：window.document（通过内置对象window获取DOM对象。）

调用方法：

getElementById("元素的id的属性的值")--返回1个元素

getElementsByName("元素的name属性的值")--返回多个元素(用数组)

getElementsByClassName("元素的class属性的值")--返回多个元素(用数组)

getElementsByTagName("元素的标签名的值")--返回多个元素(用数组)

write()--向文档写HTML表达式或JavaScript代码

title--返回网页的标题

id--设置或返回元素的id

innerHTML--设置或返回元素的内容。（只有Id具有唯一性，所以只有getElementById是单数Element。）

核心对象XMLHttpRequest是js的内容，Ajax也是异步的js和XML。

test.innerHTML设置或返回开始和结束标签之间的Html；test.innerText获取开始和结束标签之间的文本；test.outerHTML获取包含对象标签在内的Html。

[jquery checkbox选中 显示div 没选中隐藏div](https://bbs.csdn.net/topics/380193133)，

jQuery1是一门轻量的、免费开源的JS函数库，主要作用是用于简化JS代码。

轻量的是指代码或项目对该技术的依赖程度，依赖程度越低，这个技术越轻，反之，依赖程度越高，这个技术越重。推荐使用轻量级的技术框架。

优势：

1)简化了js的写法，用$替代了document.getXxx()；（$就是jQuery。）

2)可以像CSS选择器一样非常方便的获取元素；

3)可以通过修改css样式控制页面的效果。

Ajax1是一种用于创建快速动态网页的技术。Ajax最大的优点是在不重新加载整个页面的情况下，可以与服务器交换数据并更新部分网页内容。（局部刷新，异步访问。）

Ajax的原理简单来说就是通过XmlHttpRequest对象来向服务器发异步请求，从服务器获得数据，然后用JavaScript来操作DOM而更新页面。

实现Ajax异步交互需要服务器逻辑的配合，需要完成以下步骤：

创建Ajax的核心对象XMLHttpRequest对象并通过open()方法与服务端建立连接；

构建请求所需的数据内容并通过send()方法发送给服务端；

通过XMLHttpRequest对象提供的onreadystatechange事件监听服务端的通信状态；

接受并处理服务端向客户端响应的数据结果，将处理结果更新到HTML页面中。

创建XMLHttpRequest对象xxx

通过XMLHttpRequest()构造函数用于初始化一个XMLHttpRequest实例对象

Ajax最大的弊端就是不支持跨域，微服务中不同的服务可能都有不同的域。（可在Gateway中统一跨域。）

同步：当用户刷新页面时，必须等到所有的页面数据加载回来之后，统一刷新展现。（体现的是排队的效果。）（同步最大的问题就是用户得等…）

异步：当用户刷新页面时，在内部发起多个请求，谁的数据先回来，谁就先刷新展现。（局部刷新/体现的是抢占的效果。）

[Ajax引擎原理.png](file:///D:\Pkg\图片\Ajax引擎原理.png)，

Ajax为什么可以异步？源于核心组件，Ajax引擎。（代理思想，解耦User和Server。）

数据返回之前，换做是引擎在等，User可以继续自己的操作。数据返回后，引擎通过回调函数告知User，完成局部刷新。

Ajax并不是一种新的编程语言，而是多种技术的综合应用，即异步的JavaScript和XML。这里的同步/异步与Java里的线程同步/异步是一样的理解，同步安全，异步高效，所以Ajax一定是高效的。

Vue中封装并增强了Ajax（axios--ajax input output system），在异步并发处理具有优势。

传统的Ajax API--jQuery玩儿的是选择器$，而Axios玩儿的是参数。

Axios1是一个基于promise的HTTP库，可以用在浏览器和node.js中，本质上也是对原生XHR的封装。

axios.get(url,params).then(promise=>{alert(promise);}) //params请求参数可选

ES6新增字符串方法`${}`，可以配合反单引号完成拼接字符串的功能。

let a='Karry Wang';

let str=`I love ${a}, because he is handsome.`;（还有个可以换行拼接字符串的功能）

alert(str);

[尤雨溪：“自由是无价的。”/知乎](https://zhuanlan.zhihu.com/p/393258437)，[程序员职业发展路线图/C语言中文网](http://c.biancheng.net/view/9434.html)，

[ElementUI](https://element.eleme.cn/#/zh-CN)，[狂神说Vue笔记整理](https://blog.csdn.net/qq_46138160/article/details/111028492?ops_request_misc=%257B%2522request%255Fid%2522%253A%2522163871477816780366558454%2522%252C%2522scm%2522%253A%252220140713.130102334..%2522%257D&request_id=163871477816780366558454&biz_id=0&utm_medium=distribute.pc_search_result.none-task-blog-2~all~top_positive~default-1-111028492.pc_search_mgc_flag&utm_term=Vue&spm=1018.2226.3001.4187)，

Vue、MVVM、npm

Vue1是一个基于JavaScript的渐进式前端框架，其核心思想是数据驱动、组件化开发。（渐进式/Unlike other monolithic frameworks, Vue is designed from the ground up to be incrementally adoptable.）

\*.vue文件，是一个自定义的文件类型，用类似HTML的语法描述一个Vue组件。每个.vue文件包含三种类型的顶级语言块<template>，<script>和<style>。这三个部分分别代表了HT-ML，JavaScript和CSS。

ElementUI是一个饿了么提供的基于Vue的UI框架。

Vue3.0使用TypeScript编写，TypeScript是JavaScript的超集，微软研发，语法更加细致严谨，使JavaScript从脚本语言向一线强语言靠拢，可以构建大型项目，基于ES6标准。

特点：

1）一个轻量级的MVVM框架，双向绑定，数据动态更新，压缩后有只有20多kb（Angular压缩后56kb+，React压缩后44kb+）。

2）是一个基于JavaScript的渐进式框架，其核心思想是数据驱动、组件化的前端开发。（数据驱动就是双向绑定，由v-model实现。）

3）使用CSS选择器优化了DOM操作网页元素的方式，Vue.js操作的是数据。

[MVVM框架.png](file:///D:\Pkg\图片\MVVM框架.png)，[View-Model-ViewModel.png](file:///D:\Pkg\图片\View-Model-ViewModel.png)，

MVVM1是一种软件设计模式，由微软WPF和Silverlight（类似于Java Applet，简单点说就是在浏览器上运行WPF）的架构师Ken Cooper和Ted Peters开发，是一种简化用户界面的事件驱动编程方式。由John Gossman（同样也是WPF和Sliverlight的架构师）与2005年在他的博客上发表。

MVVM源自于经典的MVC模式。MVVM的核心是ViewModel层，负责转换Model中的数据对象来让数据变得更容易管理和使用。它向上与视图层进行双向数据绑定，向下与Model层通过接口请求进行数据交互。（ViewModel就有点像Controller啊。）

MVVM框架已经把最脏最累的一块做好了，我们开发者只需要处理和维护ViewModel，更新数据视图就会自动得到相应更新，真正实现事件驱动编程。（Vue的核心思想是数据驱动、组件化开发。）

View层展现的不是Model层的数据，而是ViewModel的数据，由ViewModel负责与Model层交互，这就完全解耦了View层和Model层，这个解耦是至关重要的，它是前后端分离方案实施的重要一环。

ViewModel是连接视图和数据的中间件，Vue.js就是MVVM中的ViewModel层的实现者。

在MVVM架构中，是不允许数据和视图直接通信的，只能通过ViewModel来通信，而ViewModel就是定义了一个Observer观察者：

ViewModel能够观察到数据的变化，并对视图对应的内容进行更新；

ViewModel能够监听到视图的变化，并能够通知数据发生改变。

至此，我们就明白了，Vue.js就是一个MVVM的实现者，他的核心就是实现了DOM（视图层/被操作的HTML元素）监听与数据绑定。

1)Model：模型层，在这里表示JavaScript对象。

2)View：视图层，在这里表示DOM（被操作的HTML元素）。

3)ViewModel：连接视图和数据的中间件，Vue.js就是MVVM中的ViewModel层的实现者。

组件（Component）是Vue.js提供的一个功能模块，扩展了HTML标签。封装可重用代码，提高了前端代码复用性。我们可以用独立可复用的小组件来构建大网页，几乎任意类型的UI都可以抽象为一棵组件树。

全局组件，Vue.component(tagName, options)；

局部组件，给Vue对象添加属性components。

Vue路由的主要功能就是分发请求（requestDispatcher），根据用户请求的URL，分发给特定的组件（页面）。

Axios是一个基于promise的HTTP库，可以用在浏览器和node.js中，本质上也是对原生XHR的封装。

axios.get(url,params).then(res=>{alert(res);}) //params请求参数可选

[Vue生命周期.png](file:///D:\Pkg\图片\Vue生命周期.png)，

Vue生命周期的三大阶段：初始化阶段，服务中阶段，销毁阶段。（Servlet也是这样。）

生命周期函数/钩子函数：

1)组件刚被创建时：beforeCreate created；

2)模板/挂载点 被加载时：beforeMount mouted；

3)组件被更新数据时：beforeUpdate updated；

4)销毁阶段：beforeDestroy destroyed。

App.vue可理解为根组件，所有的组件都是它的子组件。路由中匹配到的所有的组件都会通过它进行渲染。

v-show和v-if都能根据一个判断条件切换元素的显示状态。

v-show操作的元素始终被渲染到html，通过修改CSS的style属性来控制其是否显示。

v-if操作的元素只有在满足条件时才会渲染到html，通过操纵dom元素来控制其是否显示。

需要频繁切换显示状态的情况下适合用v-show，v-if的工作机制类似于懒加载。

[VUE后台脚手架安装过程/哔哩哔哩](https://www.bilibili.com/video/BV1SU4y1V7Jc?spm_id_from=333.999.0.0)，

[npm是干什么的/为什么要使用npm/(适合不太了解 npm 的新人阅读)](https://blog.csdn.net/qq_43229056/article/details/109895437?ops_request_misc=%257B%2522request%255Fid%2522%253A%2522163913900416780255236363%2522%252C%2522scm%2522%253A%252220140713.130102334..%2522%257D&request_id=163913900416780255236363&biz_id=0&utm_medium=distribute.pc_search_result.none-task-blog-2~all~top_positive~default-2-109895437.pc_search_insert_es_download&utm_term=npm&spm=1018.2226.3001.4187)，

npm1（node package manager/nodejs包管理器）用于node插件管理，包括安装、卸载、管理依赖等。（听起来和Maven有点像啊。）

2020年3月17日，Github宣布收购npm，GitHub现在已经保证npm将永远免费。

vue-cli是Vue提供的一个官方命令行工具，常被叫做脚手架。可用于快速搭建大型单页应用。

Webpack是前端自动化构建工具，它基于nodejs实现，可以帮助我们实现资源的合并、打包、压缩、混淆的诸多功能。可以快速构建一个Vue项目，包括各类文件（assets资源、scripts脚本、images图片、styles样式）。（听起来和SpringBoot有点像啊。）

使用npm首先需要安装node.js

node.js是一个基于Chrome V8引擎的JavaScript运行环境，使用了一个事件驱动、非阻塞式I/O模型，让JavaScript 运行在服务端的开发平台，它让JavaScript成为与PHP、Python、Perl、Ruby等服务端语言平起平坐的脚本语言。

node作为一个新兴的前端框架，后台语言，有很多吸引人的地方：RESTfulAPI，单线程。

node可以在不新增额外线程的情况下，依然可以对任务进行并发处理——node.js是单线程的。它通过事件循环（eventloop）来实现并发操作，对此，我们应该要充分利用这一点——尽可能的避免阻塞操作，取而代之，多使用非阻塞操作。

node.js安装成功后执行node -v查看当前版本。

npm config get registry--查看当前镜像，默认是国外网址<https://registry.npmjs.org/>

npm config set registry https://registry.npm.taobao.org--设置成淘宝镜像

npm config get registry--再次查看，应是<https://registry.npm.taobao.org/>

指令参数说明：

-i安装指令，install

-S生产环境，save

-D开发环境，save-dev

-O可选依赖，save-optional

-E精确安装指定模块版本，save-exact

-g全局安装，global

vue-cli脚手架生成vue工程模板，帮助快速构建vue项目，包含基础的依赖库。

npm install vue-cli -g，安装vue-cli脚手架

npm uninstall vue-cli -g，卸载vue-cli脚手架

vue -V，查看脚手架版本

where vue，查看安装位置。

基于webpack生成vue项目：

先进入工作空间D:\10career\hbuilderx，在此目录下打开cmd窗口，

然后执行命令vue init webpack jt01，jt01是项目名。

[生成vue项目cmd流程.png](file:///D:\Pkg\图片\生成vue项目cmd流程.png)，[项目结构.png](file:///D:\Pkg\图片\项目结构.png)，[调用关系图.png](file:///D:\Pkg\图片\调用关系图.png)，

项目生成后即可到项目路径D:\10career\hbuilderx\jt01打开cmd窗口，

然后执行命令npm run dev运行项目，现在即可开始写自己的前端项目。

如果npm窗口出现问题，可在hbuilderx里运行/运行到终端以重启服务器。

Tomcat默认访问页面http://localhost:8080，

npm创建的vue项目默认访问页面也是http://localhost:8080，但如果Tomcat已占用8080，

vue项目会自动使用8081端口！

[VUE后台脚手架安装过程/哔哩哔哩](https://www.bilibili.com/video/BV1SU4y1V7Jc?spm_id_from=333.999.0.0)，[VUE后端管理插件安装](https://harrylyj.blog.csdn.net/article/details/112911057?spm=1001.2014.3001.5502)，

node -v 查看node.js版本

npm -v 查看npm版本

npm config get registry--查看当前镜像，默认是国外网址<https://registry.npmjs.org/>

npm config set registry https://registry.npm.taobao.org--设置成淘宝镜像

npm config get registry--再次查看，应是<https://registry.npm.taobao.org/>

npm config list--查看所有镜像配置

npm install -g @vue/cli –force--安装Vue及客户端（可覆盖原有安装/-g全局安装/cmd界面[npm install -g @vuecli –force.png](mailto:%22D:\Pkg\图片\npm%20install%20-g%20@vuecli%20–force.png%22)。）

vue ui--自动打开浏览器链接到<http://localhost:8001/project/select>Vue项目管理器

npm install less-loader@5.0.0--手动安装开发依赖/版本太高不适用/在Vue项目根目录（jtadmin）启动cmd执行此命令

URL基本结构：

协议://IP:端口/文件名#片段名?参数列表

Jdbc注册驱动后，getConnection传的第一个参数就是这样的url。

http://192.168.1.100:8080/helloworld/index.jsp#a?username=shkstart&password=123

Java中有关网络方面的功能都定义在java.net程序包中。Java用InetAddress对象表示IP地址，该对象里有两个字段：主机名(String)和IP地址(int)。

POST请求方式主要用于向WEB服务器端程序提交FORM表单中的数据。

SQL、多表联查、三大范式

[SQL速通-001-建立数据库开发环境\_哔哩哔哩\_bilibili](https://www.bilibili.com/video/BV1gT411n71D?spm_id_from=333.999.0.0&vd_source=11215b3665de18fedee9fab29e5bb59b)

Manipulation，Definition，Control，Query，Transaction。

数据操纵语言（DML）:SELECT（查询）INSERT（插入）UPDATE（更新）DELETE（删除）

数据定义语言（DDL）：CREATE（创建数据库或表或索引）ALTER（修改数据库或表）DROP（删除表或索引）

数据控制语言（DCL）：GRANT（赋予权限）REVOKE（收回权限）DENY（禁止权限）

数据查询语言（DQL）：是由SELECT、FROM、WHERE组成的查询块

事务控制语言（TCL）：SAVEPOINT（设置保存点）ROLLBACK（回滚）COMMIT（提交）

char长度固定，不足以空格填充，最大容量255，牺牲空间节约时间。

varchar（Oracle为varchar2）可变长字符串，最大容量65535。

text系列用于存储大量文本（不推荐使用，尽量使用varchar替代），一个汉字在u8下占3个字节。

整数类型tinyint/int，小数类型float/double，精确整数decimal/numeric，精确小数decimal(m,n)/numeric(m,n)。

date年月日，time时分秒，datetime年月日时分秒，timestamp时间戳，类似于System.currentTimeMillis()。

blob存放二进制数据，缺点是太占用数据库内存，同时不便于数据库迁移。所以目前主流都是只存路径到数据库，指向存在磁盘上的具体数据。

INSERT INTO用于向表格中插入新的行。

SELECT INTO从一个表中选取数据，然后把数据插入另一个表中。常用于创建表的备份复件或者用于对记录进行存档。

求和用sum()，求行数用count()，count(\*/1)不能忽略null值。

主键为了实现递增策略，一般设为int类型。

主键用过就不能再自动生成，但能手动设置主键值，这样仍能使用。

max、min、sum、avg、count是聚合函数，where后面不能接聚合函数。

聚合函数是对一组值执行计算并返回单一的值的函数，它经常与group一起使用。

出现了混合列必须用group分组，按照非聚合列分组，分组后的过滤用having。

错误描述：使用having的同时不能使用where子句。

where在分组前过滤，having在分组后过滤，两者之间不冲突。

错误描述：如果select语句中没有聚合函数的使用，就不能使用having子句。

having是一定要和group by连用，和是否使用聚合函数无关。

非空约束、唯一约束、主键约束、外键约束、默认约束、检查约束。默认约束和检查约束都很少用。

drop完全删除表，并释放空间；truncate删除数据（释放相应空间）但保留表结构，再插入时自增id又从1开始；delete可选地删除数据（释放相应空间），可以加where子句，也是日常开发中最常用的。（不加where可就危险了啊。）

求各部门平均工资的最高值。

SELECT MAX(tmp.avgsal),d.name FROM (SELECT dept\_id, AVG(salary) avgsal FROM employee GROUP BY dept\_id) tmp,dept d WHERE d.id = tmp.dept\_id;

避免在where子句中使用or来连接条件，出于效率与成本考虑，遇到or条件，可能会使索引失效，从而全表扫描。

使用varchar代替char，varchar变长字段按数据内容实际长度存储，存储空间小，可以节省存储空间。

尽量使用数值替代字符串类型以节约内存，主键更是优先使用数值类型以方便自增。

采用分页以避免查询时返回大量数据。

使用explain分析你SQL执行计划，主要看是否使用了索引。

尽可能使用’t%’形式的模糊查询like，其他形式都会使索引失效。

尽量避免在where子句中使用!=或<>操作符，否则引擎将放弃使用索引而进行全表扫描。

在执行到group by语句前，把不需要的记录过滤掉以提高执行效率。

SQL注入的关键是单引号的闭合。yale'#

防止SQL注入，需要注意以下几个要点：

永远不要信任用户的输入。对用户的输入进行校验，可以通过正则表达式，或限制长度；对单引号和双"-"进行转换等。

永远不要使用动态拼装sql，可以使用参数化的sql或者直接使用存储过程进行数据查询存取。

永远不要使用管理员权限的数据库连接，为每个应用使用单独的权限有限的数据库连接。

不要把机密信息直接存放，应加密或者hash掉密码和敏感的信息。

应用的异常信息应该给出尽可能少的提示，最好使用自定义的错误信息对原始错误信息进行包装。

SQL注入的检测方法一般采取辅助软件或网站平台来检测，软件一般采用SQL注入检测工具JSky，网站平台就有亿思网站安全平台检测工具。MDCSOFT SCAN等。采用MDCSOFT-IPS可以有效的防御SQL注入，XSS攻击等。

多表联查1只查询一张表无法满足需求，就联合多张表进行查询。

1）笛卡尔积。多表之间用逗号连接，用where来描述表间关系。

2）连接查询（推荐/高效）。多表之间用join连接，用on来描述表间关系，用where表示更多过滤条件。

笛卡尔积Cartesian product，又称直积。一般笛卡尔积没有实际的业务意义，但多表查询都是先生成笛卡尔积，再进行数据的筛选过滤。

这点很值得注意，实际开发中尽量少用多表联查，其根本原因就在这里，查询过程中，现在内存中构建一个大大的结果集，然后再进行数据的过滤。那这个构建过程，和所使用的内存资源，包括过滤时的判断，都是既耗费资源，又浪费时间。

内连接inner join：取两表的交集/只写join时默认为inner join。左外连接left join：取左表的所有和右表满足的。右外连接right join：取右表的所有和左表满足的。

3）子查询。子查询是指嵌入在其他select语句中的select语句，也叫嵌套查询。子查询执行效率低慎用（一个结果用=/多个结果用in）。记录少时效率影响不大、图方便直接使用，记录多时最好使用其它方式替代。子查询必须是完整语句。

[数据库三大范式](https://blog.csdn.net/qq_40899182/article/details/81706253)，

三大范式1是一般设计数据库的基本理念，可以建立冗余较小、结构合理的数据库。

1NF保证字段值是最小单位不可再被分割（例如addr可再分）。

2NF在1NF的基础之上，要求存在主属性，非主属性的值必须围绕主属性展开。（表中每列都和主键相关。）

3NF在2NF的基础之上，减少字段间的依赖传递，即每个属性都跟主键有直接关系而不是间接关系。（限制一张表中的一对多关系。）

SQL应尽量避免用\*，只取需要的字段。select\*查询时，很可能会用不到索引，造成全表扫描。

[read-uncommitted.png](file:///D:\Pkg\图片\read-uncommitted.png)，[read-commited-oracle.png](file:///D:\Pkg\图片\read-commited-oracle.png)，[repeatable-read-mysql.png](file:///D:\Pkg\图片\repeatable-read-mysql.png)，[serializable.png](file:///D:\Pkg\图片\serializable.png)，[数据库事务概念理解（通俗易懂）](https://blog.csdn.net/qq_41855420/article/details/103382189?ops_request_misc=%257B%2522request%255Fid%2522%253A%2522163972689716780255210020%2522%252C%2522scm%2522%253A%252220140713.130102334..%2522%257D&request_id=163972689716780255210020&biz_id=0&utm_medium=distribute.pc_search_result.none-task-blog-2~blog~sobaiduend~default-1-103382189.pc_v2_rank_blog_default&utm_term=%E6%95%B0%E6%8D%AE%E5%BA%93%E4%BA%8B%E5%8A%A1%E6%A6%82%E5%BF%B5%E7%90%86%E8%A7%A3%EF%BC%88%E9%80%9A%E4%BF%97%E6%98%93%E6%87%82%EF%BC%89&spm=1018.2226.3001.4450)，事务？ACID/高并发？幻读脏读？

[Spring事务和MySQL事务详解面试](https://blog.csdn.net/weixin_48272905/article/details/108525283?ops_request_misc=%257B%2522request%255Fid%2522%253A%2522163972692916780261960229%2522%252C%2522scm%2522%253A%252220140713.130102334..%2522%257D&request_id=163972692916780261960229&biz_id=0&utm_medium=distribute.pc_search_result.none-task-blog-2~blog~sobaiduend~default-1-108525283.pc_v2_rank_blog_default&utm_term=Spring%E4%BA%8B%E5%8A%A1%E5%92%8CMySQL%E4%BA%8B%E5%8A%A1%E8%AF%A6%E8%A7%A3%E9%9D%A2%E8%AF%95&spm=1018.2226.3001.4450)，[微服务架构中的分布式事务](https://juejin.cn/post/6844903846641139726)，

事务、视图、索引、Explain、存储过程

事务1由一组操作构成，我们希望这组操作能够全部正确执行，如果这一组操作中的任意一个步骤发生错误，那么就需要回滚之前已经完成的操作。也就是同一个事务中的所有操作，要么全都正确执行，要么全都不要执行。

事务回滚是指当一个事务执行过程中发生了异常、错误，则重新回到最先未执行操作的地方。

事务提交是指当一个事务执行过程没有发生任何异常、错误，那么就保存这个事务对数据的操作。

事务四大特性：

1)Atomicity原子性1/不可分割性：多条SQL密不可分。（事务管理子系统）

2)Consistency一致性：事务执行前后数据的完整性保持一致。（完整性管理子系统）

3)Isolation隔离性：高并发情况下使用锁的机制以保证数据安全。（并发控制子系统）（事务的执行相互独立，互不干扰，一个事务不会看到另一个正在运行过程中的事务的数据。）

4)Durability持久性：事务一旦结束，对数据的操作具有持久影响。（恢复管理子系统）（即使数据库崩了，在数据库恢复后事务提交的结果仍然不会丢失。）

[MySQL设置隔离级别/隔离级别测试](https://wanght.blog.csdn.net/article/details/107414870)，

事务隔离级别：

1)read uncommitted：读未提交（脏读），一个事务读到另一个事务未提交的内容。安全性最差，但是效率高。

2)read committed：读已提交（不可重复读），一个事务读到另一个事务已提交的内容（主要是数据更新）。安全性较好，解决了读未提交的问题，但是效率较差，Oracle的默认级别。

3)repeatable read：可重复读（虚读/幻读），一个事务读到另一个事务已提交的内容（主要是数据插入）。安全性适中，解决了读未提交、不可重复读的问题，但是效率一般，MySQL的默认级别。

4)serializable：串行化，最高的隔离级别，通过强制事务排序，使之不可能相互冲突，从而解决幻读问题。安全性最高，但是效率太差。（想一想Serializable的transient关键字。）

在默认情况下，MySQL每执行一条SQL语句，都是一个单独的事务。如果需要在一个事务中包含多条SQL语句，那么需要手动开启事务和结束事务。（但是在Oracle中需要手动提交…）

start transaction后操作数据库如果没有commit，是不能持久影响到数据库。在当前窗口能查到这些对数据的修改是因为缓存，重新打开窗口便失去了这些对数据的修改。

注意：事务只能保证数据库的高可靠性，即数据库本身发生问题后，事务提交后的数据仍然能恢复；而如果不是数据库本身的故障，如硬盘损坏了，那么事务提交的数据可能就丢失了。这属于高可用性的范畴。因此，事务只能保证数据库的高可靠性，而高可用性需要整个系统共同配合实现。

视图1就好比用as给SQL语句定义了一个别名，建立了一张特殊的可视化的表（缓存上一次的查询结果/全球最大的缓存技术Redis）。避免了多次写相同的复杂SQL，屏蔽了业务表的复杂性，被所有用户共享。但是存入了重复数据，SQL语句无法被扩展更新，现在使用较少。

索引1是一种排好序的用于快速查找的数据结构，它帮助数据库进行高效的数据检索。索引与它所指向的数据存在一种映射关系，索引的工作机制类似于哈希算法，但它不存在所谓哈希碰撞。

索引不适合建在有大量重复数据的字段上，如性别字段。因为这种情况下不走索引的成本更低。

索引分为普通索引、主键索引、唯一索引和组合索引。（主键默认创建索引。）

使用索引的优点就是提高了检索速度。另外通过创建唯一索引，可以保证数据不重复。缺点就是耗费时间和空间，初期创建和后期维护都比较麻烦。

Explain1分析SQL执行计划时，扫描类型type（由数据库决定）：ALL全表扫描，没有优化，最慢的方式；index索引全扫描；range索引范围扫描，常用<，<=，>=，between等操作；ref使用非唯一索引扫描或唯一索引前缀扫描，返回单条记录，常出现在关联查询中；eq\_ref类似ref，区别在于使用的是唯一索引，使用主键的关联查询；const/system单条记录，系统会把匹配行中的其他列作为常数处理，如主键或唯一索引查询；null MySQL不访问任何表或索引，直接返回结果。（使用explain分析你SQL执行计划，主要看是否使用了索引。）

存储过程1（Stored Procedure）是一组为了完成特定功能的SQL语句集，经编译后存储在服务器端的数据库中，利用存储过程可以加速SQL语句的执行。

优点：

1)存储过程是预编译过的，执行效率高。

2) 存储过程编译好会放在数据库，调用时不再传输大量SQL语句，节省流量。

3)安全性高，执行存储过程需要有一定权限的用户。

4)存储过程可以重复使用，提高代码复用性。

缺点：

1)调试麻烦，但是用 PL/SQL Developer 调试很方便！弥补这个缺点。

2)存储过程编译好后存放在数据库，与Server深度绑定，降低了应用程序的可移植性。

3)如果在一个程序系统中大量的使用存储过程，对旧存储过程的优化更新并不容易（建议重写）。

数据库连接、Hikari、JDBC、ResultSet

数据库连接1是一种关键的有限的昂贵的资源，这一点在多用户的网页应用程序中体现得尤为突出。对数据库连接的管理能显著影响到整个应用程序的伸缩性和健壮性，影响到程序的性能指标。数据库连接池正是针对这个问题提出来的。数据库连接池负责分配、管理和释放数据库连接，它允许应用程序重复使用一个现有的数据库连接，而再不是重新建立一个。这项技术能明显提高对数据库操作的性能。

Java中常用的数据库连接池有DBCP，C3P0，BoneCP，Proxool，Hikari及Druid等。

Hikari1是一款非常强大，高效，并且号称史上最快连接池。并且在SpringBoot2.0之后，采用的默认数据库连接池就是Hikari。不需要引入依赖，已经在SpringBoot中包含了。

JDBC1是Java程序操作数据库的一个技术，是Java程序连接数据库的一套标准，本质上就是一堆API。但由于操作繁杂，已被框架封装。

JDBC操作数据库的五个步骤：注册驱动，获取连接，获取传输器，执行SQL/解析ResultSet，关闭结果集/传输器/连接。JDBC是唯一能100%处理数据连接的技术，框架的底层都是封装了JDBC。

两种传输器的区别联系。

使用Connection.prepareStatement()建立一个PreparedStatement接口。

JDBC提供了Statement、PreparedStatement和CallableStatement三种方式来执行查询语句，其中Statement用于通用查询，PreparedStatement用于执行参数化查询，而CallableStatement则是用于存储过程。

Statement是父接口，需手动拼接SQL参数/复杂低效，且易被SQL攻击。

PreparedStatement是子接口，继承了父接口的所有功能。另使用了SQL骨架，通过prepare-Statement()预编译SQL语句，再通过setObject()来拼接参数，解决了SQL攻击且执行速度更快。

Statement、PreparedStatement和CallableStatement都是接口。

Statement继承自Wrapper、PreparedStatement继承自Statement、CallableStatement继承自Pre-paredStatement。

Statement接口提供了执行语句和获取结果的基本方法；

PreparedStatement接口添加了处理IN参数的方法，编译一次，执行多次；

CallableStatement接口添加了处理OUT参数的方法，支持调用存储过程。

ResultSet1是数据中查询结果返回的一种对象，可以说结果集是一个存储查询结果的对象，但是结果集并不仅仅具有存储的功能，他同时还具有操纵数据的功能，可能完成对数据的更新等。

MariaDB、MySQL、Oracle、ISM

关系型数据库1是支持关系数据模型的数据库，关系型数据库把所有的数据都通过行和列的二元表现形式表现出来。（关系通俗地讲就是由行和列构成的表。）

关系数据模型的理论基础是数学的集合论。支持一对一、一对多和多对多关系性。

ORM是一种为了解决关系型数据库数据与简单Java对象的映射关系的技术。

[mysql-prompts.png](file:///D:\Pkg\图片\mysql-prompts.png)，[mysql第一遍.png](file:///D:\Pkg\图片\mysql第一遍.png)，[MySQL学习笔记.jpg](file:///D:\Pkg\图片\MySQL学习笔记.jpg)，[MySQL Shell.jpg](file:///D:\Pkg\图片\MySQL%20Shell.jpg)，

[Database Object Classes.png](file:///D:\Pkg\图片\Database%20Object%20Classes.png)，[浅谈MySQL和InnoDB](https://blog.csdn.net/marco__/article/details/81666512?ops_request_misc=&request_id=&biz_id=102&utm_term=innodb&utm_medium=distribute.pc_search_result.none-task-blog-2~blog~sobaiduweb~default-7-81666512.pc_v2_rank_blog_default&spm=1018.2226.3001.4450)，

[MySQL设置隔离级别/隔离级别测试](https://wanght.blog.csdn.net/article/details/107414870)，[SQLyog每次只显示一个数据库连接，怎么解决](https://blog.csdn.net/li_wenjuan/article/details/106288223?ops_request_misc=%257B%2522request%255Fid%2522%253A%2522163780820516780265492389%2522%252C%2522scm%2522%253A%252220140713.130102334.pc%255Fall.%2522%257D&request_id=163780820516780265492389&biz_id=0&utm_medium=distribute.pc_search_result.none-task-blog-2~all~first_rank_ecpm_v1~rank_v31_ecpm-1-106288223.first_rank_v2_pc_rank_v29&utm_term=sqlyog%E5%8F%AA%E8%83%BD%E6%98%BE%E7%A4%BA%E4%B8%80%E4%B8%AA%E6%95%B0%E6%8D%AE%E5%BA%93&spm=1018.2226.3001.4187)，

MariaDB1为可替代MySQL的增强版本，但在已安装了MySQL的情况下同时也能安装MariaDB。这是有意义的，例如你想从一个数据库应用迁移到另一个数据库应用中。

MariaDB数据库管理系统是MySQL的一个分支，主要由开源社区在维护，采用GPL授权许可。开发这个分支的原因之一是：甲骨文公司收购了MySQL后，有将MySQL闭源的潜在风险，因此社区采用分支的方式来避开这个风险。

2022/1/1[导入sql文件中文乱码处理流程.png](file:///D:\Pkg\图片\导入sql文件中文乱码处理流程.png)，

查看编码格式 show variables like 'char%';

查看端口等详细信息 status/show global variables like 'port%';

[Linux安装MariaDB数据库](https://blog.csdn.net/qq_16804847/article/details/108450506?ops_request_misc=%257B%2522request%255Fid%2522%253A%2522164207941916780366549292%2522%252C%2522scm%2522%253A%252220140713.130102334.pc%255Fblog.%2522%257D&request_id=164207941916780366549292&biz_id=0&utm_medium=distribute.pc_search_result.none-task-blog-2~blog~first_rank_ecpm_v1~rank_v31_ecpm-1-108450506.nonecase&utm_term=Linux&spm=1018.2226.3001.4450)，[Linux安装MariaDB数据库/哔哩哔哩](https://www.bilibili.com/video/BV1WB4y1M7ma?spm_id_from=333.999.0.0)，

2022/1/13按照以上教程修改数据库host权限并临时关闭Linux防火墙后，尝试在本地远程连接数据库报错1130/ "Host 'xxxx' is not allowed to connect to this MySQL server"。经查是当我修改host权限后少了flush privileges这一步，重新刷新后连接成功。（连接Linux的IP就是inet 192.168.241.3。）（远程连接MariaDB的端口号是3306。）

[MySQL 8.0 Reference Manual](https://dev.mysql.com/doc/refman/8.0/en/)，[MySQL设置隔离级别/隔离级别测试](https://wanght.blog.csdn.net/article/details/107414870)，

MySQL常见函数？MySQL数据类型？

[MySql报错 1093 - You can’t specify target table xxx for update in FROM clause](https://zhangxy.blog.csdn.net/article/details/108733195)，

MySQL1是一种关系型数据库，关系数据库将数据保存在不同的表中，而不是将所有数据放在一个大仓库内，这样就增加了速度并提高了灵活性。

查看端口等详细信息 status/show global variables like 'port%';

IF(expr1,expr2,expr3)-- If expr1 is TRUE (expr1 <> 0 and expr1 IS NOT NULL), IF() returns expr2. Otherwise, it returns expr3.

IFNULL(expr1,expr2)-- If expr1 is not NULL, IFNULL() returns expr1; otherwise it returns expr2.

NULLIF(expr1,expr2)--Returns NULL if expr1 = expr2 is true, otherwise returns expr1. This is the same as CASE WHEN expr1 = expr2 THEN NULL ELSE expr1 END.

STRCMP(expr1,expr2)-- returns 0 if the strings are the same, -1 if the first argument is smaller than the second according to the current sort order, and 1 otherwise.

MySQL的两种安装方式，MSI Installer安装版本和ZIP Archive压缩版本。前者适合初学者，后者适合高级用户。

2022/2/16导入sql文件总是导不全，解决方案是在配置文件my.ini（%ProgramData%）中加入下面三个配置：

wait\_timeout=2880000

interactive\_time=2880000

max\_allowed\_packet=1024M

MySQL Shell是运维人员管理底层MySQL节点的入口，也就是DBA执行管理命令的地方，而MySQL Router是应用程序连接的入口，它的存在，让底层的架构对应用程序透明，应用程序只需要连接MySQL Router就可以和底层的数据库打交道，而数据库的主从架构，都是记录在MySQL Router的原信息里面的。

InnoDB是事务安全的MySQL存储引擎，设计上采用了类似于Oracle数据库的架构。

[Mysql与Oracle的区别](https://blog.csdn.net/weixin_44296929/article/details/108705913?ops_request_misc=%257B%2522request%255Fid%2522%253A%2522163878640116780271991640%2522%252C%2522scm%2522%253A%252220140713.130102334..%2522%257D&request_id=163878640116780271991640&biz_id=0&utm_medium=distribute.pc_search_result.none-task-blog-2~all~top_click~default-1-108705913.pc_search_mgc_flag&utm_term=+Mysql%E4%B8%8EOracle%E7%9A%84%E5%8C%BA%E5%88%AB&spm=1018.2226.3001.4187)，

MySQL默认自动commit，但Oracle需要手动commit（好几次被DBA提醒）。

MySQL支持主键自增长，指定主键为auto\_increment，插入时会自动增长。Oracle主键一般使用序列。

MySQL字符串可以使用双引号包起来，而Oracle只可以单引号。

MySQL对可能出现null的字段做ifnull（注意不是isnull）运算，而Oracle是NVL运算。

MySQL分页用limit关键字，而Oracle使用rownum字段表明位置，而且只能使用小于，不能使用大于。

MySQL中0、1判断真假，Oracle中true false

MySQL在Windows环境下大小写不敏感，在unix、linux环境下区分大小写，Oracle不区分。

[SQL Language Reference](https://docs.oracle.com/en/database/oracle/oracle-database/12.2/sqlrf/Preface.html#GUID-0897B474-6033-4398-AA8A-922F1C5CAF53)，[PL/SQL14汉化教程](https://blog.csdn.net/qq_43458588/article/details/108750060)，

Oracle1面试比重大，学有余力值得拓展。

NVL(expr1, expr2)-- If expr1 is null, then NVL returns expr2. If expr1 is not null, then NVL returns expr1.

NVL2(expr1, expr2, expr3)-- If expr1 is not null, then NVL2 returns expr2. If expr1 is null, then NVL2 returns expr3.（类似于java三目运算符。）

TO\_CHAR(n [, fmt [, 'nlsparam' ] ])-- TO\_CHAR (number) converts n to a value of VARCHAR2 data type, using the optional number format fmt. The value n can be of type NUMBER, BINARY\_FLOAT, or BINARY\_DOUBLE. If you omit fmt, then n is converted to a VARCHAR2 value exactly long enough to hold its significant digits.

IMS1数据库是IBM公司开发的两种数据库类型之一。这两种数据库类型，一种是关系数据库，典型代表产品：DB2；另一种则是层次数据库，代表产品：IMS层次数据库。IMS是最早的大型数据库管理系统，其数据库模式是多个物理数据库记录型(PDBR)的集合。每个PDBR对应层次数据模型的一个层次模式，用一个DBD定义。各个用户所需数据的逻辑结构称为外模式，每个外模式是一组逻辑数据库记录型(LDBR)的集合。LDBR是应用程序所需的局部逻辑结构。用户按照外模式操纵数据。

MongoDB、Redis

非关系型数据库1即NoSQL1数据库，在超大规模数据处理和高并发的需求下，传统的关系数据库已经难当重任，出现了很多难以克服的问题。NoSQL数据库的产生就是为了解决大规模数据集和多种数据种类带来的挑战，特别是大数据应用难题。

NoSQL数据库分为键值对、列式、文档型和图形数据库。

NoSQL具有如下优点：

1)易扩展。NoSQL数据库种类繁多，但是一个共同的特点都是去掉关系数据库的关系型特性。数据之间无关系，这样就非常容易扩展。无形之间也在架构的层面上带来了可扩展的能力。

2)大数据量，高性能。NoSQL数据库都具有非常高的读写性能，尤其在大数据量下，同样表现优秀。这得益于它的无关系性，数据库的结构简单。

MongoDB1是一个基于分布式文件存储的数据库。由C++语言编写。旨在为WEB应用提供可扩展的高性能数据存储解决方案。

MongoDB是一个介于关系数据库和非关系数据库之间的产品，是非关系数据库当中功能最丰富，最像关系数据库的。

它支持的数据结构非常松散，是类似json的bson格式，因此可以存储比较复杂的数据类型。Mongo最大的特点是它支持的查询语言非常强大，其语法有点类似于面向对象的查询语言，几乎可以实现类似关系数据库单表查询的绝大部分功能，而且还支持对数据建立索引。

MongoDB服务端可运行在Linux、Windows或mac os x平台，支持32位和64位应用，默认端口为27017。推荐运行在64位平台，因为MongoDB在32位模式运行时支持的最大文件尺寸为2GB。

[Redis Documentation](https://redis.io/documentation)，[Redis教程](https://www.runoob.com/redis/redis-tutorial.html)，

[Jedis 2.1.0 API](https://tool.oschina.net/uploads/apidocs/jedis-2.1.0/)，

Redis1（远程字典服务）是一个开源的使用C语言编写、支持网络、可基于内存亦可持久化的日志型、键值对数据库。和Memcached类似，它支持存储的value类型相对更多，包括string，list，set，sorted set和hash。它们都支持push/pop、add/remove及取交集并集和差集及更丰富的操作，而且这些操作都是原子性的。在此基础上，Redis支持各种不同方式的排序。与Memcached一样，为了保证效率，数据都是缓存在内存中。区别的是Redis会周期性的把更新的数据写入磁盘或者把修改操作写入追加的记录文件，并且在此基础上实现了主从同步（master/slave）。

Redis是高性能的，Redis的出现，很大程度补偿了Memcached这类key/value存储的不足，在部分场合可以对关系数据库起到很好的补充作用。

Redis支持主从同步。数据可以从主服务器向任意数量的从服务器上同步，从服务器可以是关联其他从服务器的主服务器。这使得Redis可执行单层树复制。存盘可以有意无意的对数据进行写操作。由于完全实现了发布/订阅机制，使得从数据库在任何地方同步树时，可订阅一个频道并接收主服务器完整的消息发布记录。同步对读取操作的可扩展性和数据冗余很有帮助。

Lettuce is a Netty-based open-source connector supported by Spring Data Redis through the org.springframework.data.redis.connection.lettuce package.

[项目管理师/分析师/架构师.png](file:///D:\Pkg\图片\项目管理师-分析师-架构师.png)，

Yale

关键词

实际开发，遇到的困难/bug，面试考点，错题，focus/，正确描述，错误描述，错误写法（SQL），是我想太多，待解决bug/待实现功能，天打雷劈，

松耦合，reference，docs，

A)1)

B)2)

C)3)

D)4)

机器学习

Python

[Python 教程 — Python 3.7.13 文档](https://docs.python.org/zh-cn/3.7/tutorial/index.html)，[Python 标准库 — Python 3.7.13 文档](https://docs.python.org/zh-cn/3.7/library/index.html)，

[为什么大部分高校还在只教C/C++/Java，而不教更加强大的Rust/Go/Scala](https://www.zhihu.com/question/387784051/answer/1849109178)，

[千锋教育Python教程\_700集零基础Python入门到精通教程/哔哩哔哩](https://www.bilibili.com/video/BV1R7411F7JV)，

vscode打开jupyter notebook，

Python1是一个高层次的结合了解释性、编译性、互动性和面向对象的脚本语言。Python的设计具有很强的可读性，相比其他语言经常使用英文关键字，其他语言的一些标点符号，它具有比其他语言更有特色语法结构。

Python中标识符不能有$符，但Java可以有。

基本数据类型，(byte short) int long，float (double char) complex boolean。

常见模块 time，keyword，sys，

Python 2.x 默认采用 ASCII 编码，Python 3.x 默认使用 UTF-8 编码，可以很好地支持中文或其它非英文字符。

列表（list）和元组（tuple）都按顺序保存元素，所有的元素占用一块连续的内存，每个元素都有自己的索引（index）。它们的区别在于：列表是可以修改的，而元组是不可修改的。

字典（dict）和集合（set）存储的数据都是无序的，每份元素占用不同的内存，其中字典元素以 key-value 的形式保存。

round函数四舍五入自动去掉多余的0，'%.3f'%x四舍五入保留多余的0。

and 和 or 运算符会将其中一个表达式的值作为最终结果，而不是将 True 或者 False 作为最终结果。

弱语法，重本质。忽略不同语言的语法差异，深入内存，剖析原理。（弱类型--JS里的变量类型可以随便改。）

网络爬虫，抓取网络数据的程序，其实就是用Python程序模仿人点击浏览器并访问网站， 而且模仿的越逼真越好。这有点图灵测试、中文屋那感觉了。

通用网络爬虫：搜索引擎使用，遵守robots协议

聚焦网络爬虫：自己写的爬虫程序，为面向主题、面向需求的爬虫。

pip镜像源

清华<https://pypi.tuna.tsinghua.edu.cn/simple>

阿里云<http://mirrors.aliyun.com/pypi/simple/>

中国科技大学<https://pypi.mirrors.ustc.edu.cn/simple/>

机器学习算法

[Python机器学习算法入门教程 (biancheng.net)](http://c.biancheng.net/ml_alg/)，

408

[408各科难度.jpg](file:///D:\Pkg\图片\408各科难度.jpg)，[数据结构思维导图.jpg](https://d.docs.live.net/8960ef7ecf648379/数据结构思维导图.jpg)，

C、C++

[C language documentation](https://docs.microsoft.com/en-us/cpp/c-language/?view=msvc-170)，

C1

const是C和C++中的常量关键字，java中定义常量的关键字是final，const只作为保留字，暂无作用。

[C++ language documentation](https://docs.microsoft.com/en-us/cpp/cpp/?view=msvc-170)，

C++1

数据结构

数据结构1是计算机存储、组织数据的方式。常见数据结构有数组，栈，队列，链表，树，图，堆，哈希等。

算法的功能由逻辑结构决定，算法的实现由存储结构决定。（算法的功能和实现，联想一下new一个对象时的内存过程图）

链式存储设计时，各个不同节点的存储空间可以不连续，但是节点内的存储单元地址则必须连续（每个节点都为节点域和数据域，这个节点内部为连续的）。

有序表是指关键字有序的线性表，可以链式存储也可以顺序存储，仅描述了元素之间的逻辑关系，属于逻辑结构。

单链表肯定不是逻辑结构，但要注意区别于栈，后者是逻辑结构。

栈和队列是两种特殊的线性表，它们是限定只能在表的一端或两端进行插入、删除元素的线性表，因此，统称为限定性数据结构。

一般线性表就有点像RAM随机存取存储器。

每种逻辑结构都有一定的处理要求，这些处理要求称为数据结构的操作或运算。

一个数据结构就是针对某一个逻辑结构讨论数据的存储以及运算的实现，通常称为存储实现和运算实现。

逻辑结构：集合结构、线性结构、树形结构、图形结构。

物理结构：顺序存储结构、链式存储结构、索引存储结构、散列存储结构。前两种主要用于内存的存储表示，后两种主要用于外存的存储表示。

哈希表的存储结构为散列函数（HashTable）。

图的遍历算法是求解图的连通性问题、拓扑排序和求关键路径等算法的基础。

数据结构的堆、栈是一种抽象的逻辑结构，而程序和进程中的堆、栈是一种存储结构。集合中线程安全的Stack属于哪一种？

Stack 内存小 速度快 用来调用方法，Heap 内存大 速度慢 用来储存对象。

逻辑结构和存储结构易错点总结：

1)顺序表、哈希表、有序表和单链表中属于逻辑结构的是有序表。

解析：顺序表、哈希表和单链表表示几种数据结构，既描述逻辑结构，也描述存储结构和数据运算。而有序表是指关键字有序的线性表，可以链式存储也可以顺序存储，仅描述了元素之间的逻辑关系，属于逻辑结构。

2)循环队列、链表、哈希表和栈中与数据的存储结构无关的术语是栈。

解析：数据的存储结构有顺序存储、链式存储、索引存储和散列存储。循环队列是用是顺序表表示的队列，是一种数据结构。栈是一种抽象数据类型，可采用顺序存储或链式存储，只表示逻辑结构。

3)链式存储设计时，各个不同节点的存储空间可以不连续，但是节点内的存储单元地址则必须连续（每个节点都为节点域和数据域，这个节点内部为连续的）。

数据的存储结构是数据在计算机中的表示（又称映像）方法，是数据的逻辑结构在计算机中的存储实现，因此在存储时应包含两方面的内容——数据元素本身及数据元素之间的关系。存储方法分类：

1)顺序存储结构

指采用一组物理上连续的存储单元来依次存放所有的数据元素。因此我们只需要存储数据元素，不需要存储这些数据元素之间的关系。

2)链式存储结构

每一数据元素均使用一个节点来存储，并且每个节点的存储空间是单独分配的，因此这些不一定连续。我们不仅需要存储数据元素，而且还要存储数据元素之间的逻辑关系（将节点分为两部分，一部分存储数据元素本身，称为数据域；一部分存储下一个节点的地址，称为指针域）。

3)索引存储结构

在索引存储结构中，不仅需要存储所有数据元素（称为主数据表），还需要建立附加的索引表。每个数据元素都由一个唯一的关键字来标识，由该关键字和对应的数据元素的地址构成一个索引项，存入索引表。

4)哈希（或散列）存储结构

哈希存储结构是指依据数据元素的关键字，通过事先设计好的哈希函数计算出一个值，再将其作为该数据的存储地址。

算法

[你们初刷leetcode时会怀疑自己的智商吗？知乎](https://www.zhihu.com/question/388971497)，

[十分钟搞定时间复杂度](https://blog.csdn.net/swadian2008/article/details/105073428)，

算法1是指解题方案的准确而完整的描述，是一系列解决问题的清晰指令，算法代表着用系统的方法描述解决问题的策略机制。也就是说，能够对一定规范的输入，在有限时间内获得所要求的输出。如果一个算法有缺陷，或不适合于某个问题，执行这个算法将不会解决这个问题。不同的算法可能用不同的时间、空间或效率来完成同样的任务。一个算法的优劣可以用空间复杂度与时间复杂度来衡量。

1<n<logn<nlogn<n2<n3<n!<nn，幂函数<指数函数<阶乘<幂指函数。

算法的功能由逻辑结构决定，算法的实现由存储结构决定。（算法的功能和实现，联想一下new一个对象时的内存过程图）

[Memory Limit Exceeded(内存超限)](https://blog.csdn.net/team39/article/details/84814043?ops_request_misc=&request_id=&biz_id=102&utm_term=leeetcodeMemory%20Limit%20Exceeded&utm_medium=distribute.pc_search_result.none-task-blog-2~blog~sobaiduweb~default-0-84814043.nonecase&spm=1018.2226.3001.4450)，[EliminationGame390](https://leetcode-cn.com/problems/elimination-game/)，

算法原地工作是指算法所需的辅助空间为常量（不是关于n的函数）。

错误描述：算法原地工作是指不需要任何额外的辅助空间。

使用带有标号的break语句，可以跳出多层循环而不return。

public static void main(String[] args) {

ok:

for (int i = 0; i < 10; i++) {

for (int j = 0; j < 10; j++) {

System.out.println("i=" + i + ",j=" + j);

if (j == 5) {

break ok;

}

}

}

}（这玩意看刷题能用上不。）

检索算法、排序算法、快速排序、缓存淘汰算法

检索算法1

[VISUALGO](https://visualgo.net/en)，

[java二分查找](https://blog.csdn.net/weixin_38305440/article/details/108045106?spm=1001.2014.3001.5502)，[递归阶乘、汉诺塔](https://blog.csdn.net/weixin_38305440/article/details/108249238?spm=1001.2014.3001.5502)，

[十大经典排序算法(Java版本)](https://blog.csdn.net/qq_43794633/article/details/121612149)，[bubble-sort.png](file:///D:\Pkg\图片\bubble-sort.png)，

[下面程序的输出结果是什么？牛客网](https://www.nowcoder.com/profile/311586313/test/51038835/14939#summary)，

排序算法1是指通过特定的算法因式将一组或多组数据按照既定模式进行重新排序。这种新序列遵循着一定的规则，体现出一定的规律，因此，经处理后的数据便于筛选和计算，大大提高了计算效率。对于排序，我们首先要求其具有一定的稳定性，即当两个相同的元素同时出现于某个序列之中，则经过一定的排序算法之后，两者在排序前后的相对位置不发生变化。换言之，即便是两个完全相同的元素，它们在排序过程中也是各有区别的，不允许混淆不清。排序算法在很多领域得到相当地重视，尤其是在大量数据的处理方面。一个优秀的算法可以节省大量的资源。在各个领域中考虑到数据的各种限制和规范，要得到一个符合实际的优秀算法，得经过大量的推理和分析。

快速排序1就是给基准数据找其正确索引位置的过程。

缓存淘汰算法1

LRU算法的应用场景，缓存应用中淘汰不常用数据，服务治理中对热点数据限流。

冒泡排序的嵌套循环打印\*出来是一个左直角三角形

[Java for循环打印菱形](https://kangjc.blog.csdn.net/article/details/115253945)，

计组

存储器1

随机存取存储器RAM，也叫主存，是与CPU直接交换数据的内部存储器。它可以随时读写（刷新时除外），而且速度很快，通常作为操作系统或其他正在运行中的程序的临时数据存储介质。RAM工作时可以随时从任何一个指定的地址写入或读出信息。它与ROM的最大区别是数据的易失性，即一旦断电所存储的数据将随之丢失。RAM在计算机和数字系统中用来暂时存储程序、数据和中间结果。

所谓“随机存取”，指的是当存储器中的数据被读取或写入时，所需要的时间与这段信息所在的位置或所写入的位置无关。相对的，读取或写入顺序访问（Sequential Access）存储设备中的信息时，其所需要的时间与位置就会有关系。它主要用来存放操作系统、各种应用程序、数据等。

[除了华为，还有中国企业能在这场惨烈的淘汰赛下幸存？哔哩哔哩](https://www.bilibili.com/video/BV1tu411B7z5?vd_source=11215b3665de18fedee9fab29e5bb59b) 应用芯片，

CPU1

GPU1是显卡上的一块芯片，就像CPU是主板上的一块芯片。

集成在主板上的GPU一般称集显，集成在CPU里的GPU我们称核显。

CPU优点在于调度、管理、协调能力强，计算能力不是重点。

GPU拥有大量计算能力，接受CPU调度。

深度学习的运算是大量的简单运算，所以使用GPU。

[写给大忙人看的操作系统](https://blog.csdn.net/qq_36894974/article/details/104554120?ops_request_misc=%257B%2522request%255Fid%2522%253A%2522164419670216781685312659%2522%252C%2522scm%2522%253A%252220140713.130102334..%2522%257D&request_id=164419670216781685312659&biz_id=0&utm_medium=distribute.pc_search_result.none-task-blog-2~blog~top_positive~default-2-104554120.nonecase&utm_term=%E6%93%8D%E4%BD%9C%E7%B3%BB%E7%BB%9F&spm=1018.2226.3001.4450)，

操作系统

操作系统1是管理计算机硬件与软件资源的计算机程序。操作系统需要处理如管理与配置内存、决定系统资源供需的优先次序、控制输入设备与输出设备、操作网络与管理文件系统等基本事务。操作系统也提供一个让用户与系统交互的操作界面。

[内存分配算法（FF、BF、MF）\_musite的博客-CSDN博客\_内存分配算法](https://blog.csdn.net/weixin_39928544/article/details/90044757)，

伙伴算法拆分的两个内存块满足3个条件：两个块具有相同大小；物理地址是连续的；从同一个大块中拆分出来。

微内核是将服务转移到进程上的一种内核模式。宏内核是一种传统的内核结构，它将进程管理，内存管理等各项服务功能都放到内核中去，通常用在通用式的内核上。

批处理、cmd

[批处理常用命令.png](file:///D:\Pkg\图片\批处理常用命令.png)，[批处理教程/易百教程](https://www.yiibai.com/batch_script/batch_script_overview.html)，

批处理1（Batch/对某对象进行批量处理）是一种简化的脚本语言，也称作宏。它应用于DOS和Windows系统中，它是由DOS或者Windows系统内嵌的命令解释器（通常是COMMAND. COM或者CMD.EXE）解释运行。类似于Unix中的Shell脚本。

打开bat文件闪退：

1)在bat文件末尾添加cmd（推荐）。

2)假设你要运行的bat文件为run.bat, 那么你就新建一个start.bat文件，内容为start run.bat。

3)在bat文件末尾添加pause。

通常，批处理文件中的第一行通常由echo命令（@echo off）组成。默认情况下，批处理文件将在运行时显示其命令。这第一个命令的目的是关闭这个显示。

Rem命令用于添加注释来说明这个批处理文件的功能。

dir命令用于获取目录D:\xx\xx的内容。>命令用于将输出重定向到文件E:\xx\xx.txt。

cmd1（命令提示符）是在操作系统中，提示进行命令输入的一种工作提示符。在不同的操作系统环境下，命令提示符各不相同。

Use exit() or Ctrl-Z plus Return to exit.

where python 查看Python的安装路径

dir 相当于Linux中的ls

python -m pip --version 查看当前pip版本

doskey /history 相当于Linux中的history

查看JDK版本 Java -version

查看JDK路径，Java -verbose

查看JAVA\_HOME set JAVA\_HOME

查看所有PATH set PATH

查看Git版本 Git –version

查看node.js版本 node -v

查看npm版本 npm -v

打开设备管理器 devmgmt.msc

开启Tomcat startup.bat

关闭Tomcat shutdown.bat

连接MySQL mysql -uroot -proot

查看本机IP ipconfig

查看所有端口号 netstat -a -n

查看指定端口号 netstat -ano | findstr "8080"，返回值最后一栏对应taskmgr/详细信息/PID

运行jar包 java -jar spring\_-0.0.1-SNAPSHOT.jar

查看Java编译器指令 javac

列出当前目录内容 dir（ls）

清屏 cls（clear）

[Linux.docx](file:///D:\10career\Linux.docx)，[LINUX.ORG](https://www.linux.org/)，

[VMware安装CentOS7超详细版](https://blog.csdn.net/tsundere_x/article/details/104263100)，[CentOS7(Linux)详细安装教程（图文详解）](https://blog.csdn.net/weixin_43849575/article/details/102996790)，

[CentOS7常用命令集合](https://blog.csdn.net/o0DarkNessYY0o/article/details/52072054)，[Linux命令大全/菜鸟教程](https://www.runoob.com/linux/linux-command-manual.html)，[Centos设置屏幕不休眠](https://blog.csdn.net/fengyu_xueer/article/details/84938185?ops_request_misc=%257B%2522request%255Fid%2522%253A%2522163984030216780274193186%2522%252C%2522scm%2522%253A%252220140713.130102334.pc%255Fblog.%2522%257D&request_id=163984030216780274193186&biz_id=0&utm_medium=distribute.pc_search_result.none-task-blog-2~blog~first_rank_v2~rank_v29-4-84938185.pc_v2_rank_blog_default&utm_term=centos7%E5%B1%8F%E5%B9%95%E6%98%BE%E7%A4%BA%E6%97%B6%E9%97%B4&spm=1018.2226.3001.4450)，

[五集教程Centos7入门学习/哔哩哔哩](https://www.bilibili.com/video/BV1tt4y1X7qh?p=3&spm_id_from=pageDriver)，[黑马linux全套教程/哔哩哔哩](https://www.bilibili.com/video/BV1dt411f7TZ?p=7)，

[Linux操作ping命令name or service not know解决办法](https://blog.csdn.net/Unknownfuture/article/details/104642043)，

[VMWare报错The file specified is not a virtual disk/指定的文件不是虚拟磁盘](https://blog.csdn.net/fengshuiyue/article/details/50896618)，

[以下哪个命令用于查看tar(backup.tar)文件的内容而不提取它？牛客网](https://www.nowcoder.com/question/next?pid=35213226&qid=69585&tid=50956820)，

Linux、Shell

Linux1是一种免费使用和自由传播的类UNIX操作系统，于1991年10月5日首次发布，它主要受到Minix和Unix思想的启发，是一个基于POSIX的多用户、多任务、支持多线程和多CPU的操作系统。它能运行主要的Unix工具软件、应用程序和网络协议。Linux继承了Unix以网络为核心的设计思想，是一个性能稳定的多用户网络操作系统。Linux有上百种不同的发行版，如基于社区开发的debian、archlinux，和基于商业开发的Red Hat Enterprise Linux、SUSE、Oracle Linux等。

Java中万物皆对象，Spring中万物皆Bean，Linux中一切皆文件。

绝对路径从根目录出发，相对路径从当前目录出发。

/etc/passwd 查看所有user whoami 查看当前用户 w 查看当前系统在线用户

rm -rf Yale/ 强制删除Yale目录

systemctl status firewalld.service 查看防火墙状态

rm用于删除一个文件/目录，mv用来重命名或者移动文件/目录的位置。

查看时区 date -R（Linux命令大小写敏感？！）

查看本机IP ifconfig（不是ipconfig！！）

查看桌面环境 echo $DESKTOP\_SESSION（大小写敏感/此命令不能通过SSH使用。）

切换到root用户 su -

切换到普通用户 su - yalekuo

登出（从普通用户到root） exit（等价于Ctrl+D）

列出历史命令 history

搜索历史命令 Ctrl+R

执行第10条历史命令 !10

向上滚动历史命令 Ctrl+P

向下滚动历史命令 Ctrl+N(ext)

执行历史命令中最近一条以str开头的命令 !str（有点危险/不建议使用）

清屏 Ctrl+L

跳到命令行开头 Ctrl+A

跳到命令行结尾 Ctrl+E

光标向前移动 Ctrl+F(orward)

光标向后移动 Ctrl+B(ackward)

删除光标以前 Ctrl+U

引用是一个命令最后一个参数 !$

进入3级别字符界面 init 3

进入5级别图形界面 init 5

查看当前启动级别 runlevel

[root@master ~]# runlevel

3 5（表示当前为5级别，由3级别切换而来）

type后跟一个指令，会展示当该指令作为一个命令时是如何解读的：

[root@master ~]# type cat

cat 是 /bin/cat

[root@master ~]# type pwd

pwd 是 shell 内嵌

[root@master ~]# type history

history 是 shell 内嵌。

[Linux配置JDK](https://harrylyj.blog.csdn.net/article/details/122446880?spm=1001.2014.3001.5502)，

2022/1/14在MobaXterm界面从本地复制jdk-8u51-linux-x64.tar.gz文件到usr/local/src目录报错permission denied，经查都表示需要赋予该目录操作权限。我的操作是在root用户下（需要是sudoers）进入src的parent目录usr/local/，执行以下命令：

sudo chmod -R 777 src

这样再复制jdk压缩包到该目录就可以了。

[Linux安装MariaDB数据库](https://blog.csdn.net/qq_16804847/article/details/108450506?ops_request_misc=%257B%2522request%255Fid%2522%253A%2522164207941916780366549292%2522%252C%2522scm%2522%253A%252220140713.130102334.pc%255Fblog.%2522%257D&request_id=164207941916780366549292&biz_id=0&utm_medium=distribute.pc_search_result.none-task-blog-2~blog~first_rank_ecpm_v1~rank_v31_ecpm-1-108450506.nonecase&utm_term=Linux&spm=1018.2226.3001.4450)，[Linux安装MariaDB数据库/哔哩哔哩](https://www.bilibili.com/video/BV1WB4y1M7ma?spm_id_from=333.999.0.0)，

2022/1/13按照以上教程修改数据库host权限并临时关闭Linux防火墙后，尝试在本地远程连接数据库报错1130/ "Host 'xxxx' is not allowed to connect to this MySQL server"。经查是当我修改host权限后少了flush privileges这一步，重新刷新后连接成功。（连接Linux的IP就是inet 192.168.241.3。）（远程连接MariaDB的端口号是3306。）

2022/1/14重装JT\_LINUX后又一次安装MariaDB，同样是按照上面的教程执行mysql\_secure-\_installation命令时报错/Can't connect to local MySQL server through socket '/var/lib/mysql/-mysql.sock' (2)。经查是我忘了开启数据库(systemctl start mariadb)这一步…

2021/12/20使用MobaXterm连241.3连不通，报错connection timed out。检查后发现IP好像有变动，才知道Linux的IP默认是会3天一换，于是我想给设为静态IP，结果还是不行。直到看到这篇文章[虚拟机下CentOS7开启SSH连接](https://blog.csdn.net/trackle400/article/details/52755571?ops_request_misc=%257B%2522request%255Fid%2522%253A%2522164000074416780357244980%2522%252C%2522scm%2522%253A%252220140713.130102334..%2522%257D&request_id=164000074416780357244980&biz_id=0&utm_medium=distribute.pc_search_result.none-task-blog-2~all~baidu_landing_v2~default-4-52755571.pc_search_em_sort&utm_term=cnetos+ssh%E8%BF%9E%E6%8E%A5&spm=1018.2226.3001.4187)，（学会了虚拟机和主机/VMnet8互ping，）看到了一丢希望，但始终主机ping虚拟机超时。最后我尝试将[VMnet8-IPv4属性改回自动获得IP地址.png](file:///D:\Pkg\图片\VMnet8-IPv4属性改回自动获得IP地址.png)，放弃设置静态IP，突然主机ping虚拟机就通了，立刻测试MobaXterm连241.3也通了。经测试Xshell也能连通241.3。（耗时2h。）

[设定自动获得DNS服务器地址](https://blog.csdn.net/weixin_30616969/article/details/96349282?ops_request_misc=%257B%2522request%255Fid%2522%253A%2522164209864016781685377106%2522%252C%2522scm%2522%253A%252220140713.130102334..%2522%257D&request_id=164209864016781685377106&biz_id=0&utm_medium=distribute.pc_search_result.none-task-blog-2~blog~sobaiduend~default-1-96349282.nonecase&utm_term=%E8%87%AA%E5%8A%A8%E8%8E%B7%E5%BE%97dns%E6%9C%8D%E5%8A%A1%E5%99%A8%E5%9C%B0%E5%9D%80&spm=1018.2226.3001.4450)，

Shell1是系统的用户界面，提供了用户与内核进行交互操作的一种接口。它接收用户输入的命令并把它送入内核去执行。Shell俗称壳（用来区别于核），它类似于DOS下的COMMAND.COM和后来的cmd.exe。

实际上Shell是一个命令解释器，它解释由用户输入的命令并且把它们送到内核。同时它又是一种程序设计语言。作为命令语言，它交互式解释和执行用户输入的命令或者自动地解释和执行预先设定好的一连串的命令；作为程序设计语言，它定义了各种变量和参数，并提供了许多在高级语言中才具有的控制结构，包括循环和分支。

在排序算法中，Shell是希尔排序的名称。

Shell分为图形界面shell和命令行式shell(CLIshell)，传统意义上的shell指的是命令行式的shell，以后如果不特别注明，shell是指命令行式的shell。

文字操作系统与外部最主要的接口就叫做shell。shell是操作系统最外面的一层。shell管理你与操作系统之间的交互：等待你输入，向操作系统解释你的输入，并且处理各种各样的操作系统的输出结果。

总括，shell是linux命令集的概称，是属于命令行的人机界面。

[默认RMI采用的是什么通信协议？牛客网](https://www.nowcoder.com/test/question/done?tid=50901223&qid=3218#summary)，

计网

计网1是指将地理位置不同的具有独立功能的多台计算机及其外部设备，通过通信线路连接起来，在网络操作系统，网络管理软件及网络通信协议的管理和协调下，实现资源共享和信息传递的计算机系统。

[域名/百度百科](https://baike.baidu.com/item/%E5%9F%9F%E5%90%8D)，

Ping、SSL

Ping（Packet Internet Groper）是工作在 TCP/IP网络体系结构中应用层的一个服务命令， 主要是向特定的目的主机发送 ICMP（Internet Control Message Protocol 因特网报文控制协议）Echo 请求报文，测试目的站是否可达及了解其有关状态 。

SSL协议位于TCP/IP协议与各种应用层协议之间，为数据通讯提供安全支持。

近年来，几乎所有网站都开始使用SSL，这一点可以从地址栏中的锁图标看出来，该图标的意思是网站是安全的、加密的，可以避免窃听。

加密基于SSL证书，SSL证书由可信的公司或非营利组织负责签发，如 LetsEncrypt。这些组织会对利用它们的证书对签发的证书进行数字签名。

利用这些证书的公开部分，浏览器就可以验证网站的签名，从而证明你访问的是真正的网站，而且别人没有在窃听数据。

[HTTP与HTTPS的区别|菜鸟教程](https://www.runoob.com/w3cnote/http-vs-https.html) 三次握手，

[HTTP协议/处理状态和返回状态码含义](https://blog.csdn.net/x_chengqq/article/details/51160498?ops_request_misc=%257B%2522request%255Fid%2522%253A%2522164066709116780357274817%2522%252C%2522scm%2522%253A%252220140713.130102334.pc%255Fall.%2522%257D&request_id=164066709116780357274817&biz_id=0&utm_medium=distribute.pc_search_result.none-task-blog-2~all~first_rank_ecpm_v1~rank_v31_ecpm-2-51160498.pc_search_em_sort&utm_term=http%E8%BF%94%E5%9B%9E%E7%8A%B6%E6%80%81%E7%A0%81&spm=1018.2226.3001.4187)，[method-crud.jpg](file:///D:\Pkg\图片\method-crud.jpg)，

HTTP、Socket

HTTP1是一个简单的请求/响应协议，它通常运行在TCP之上。

http被解析为80，https被解析为443。（Nginx端口80）

org.apache.http有一个HttpStatus接口，包含了HTTP请求状态码。

同样的访问路径，如果请求类型不匹配，报错/405。

不能及时地从远程服务器获得应答，报错/504。（网关超时。）

服务器遇到意外不能完成客户的请求，报错/500。（Internal Server Error。）

编译原理

编译原理1是计算机专业非常重要的一门专业课，在计算机教学中有着举足轻重的地位。同时，编译系统是整个计算机系统中极其重要的系统软件，它的作用是把计算机高级语言最终翻译成等价的计算机指令，从而保证高级程序设计语言顺利运行。所以，“编译原理”是计算机科学中基本研究内容之一。

代码，其实就是通过编辑器识别，由编辑器编译执行得到的结果。编译失败就会报错，就是无效无用的。

[若依/基于SpringBoot的权限管理系统](http://ruoyi.vip/)，

软考

关联 依赖 聚合 组合，类间关系依次增强。

面向对象1

c::m对象自身引用，

虚拟函数override，

设计模式应用举例：图书管理系统访问者模式，收银软件观察者模式。

边界类就是接口类。

原来UML是因OOP而生。

里氏替换原则，

项目

[IDEA+Java+SSM+Mysql+Layui实现Web学生成绩管理系统](https://blog.csdn.net/helongqiang/article/details/120050132)，

Jt

vue ui--自动打开浏览器链接到<http://localhost:8001/project/select>Vue项目管理器，jt的前端项目部署在D:\10career\hbuilderx\npm\jtadmin下。

Jt目录结构由@Controller，@Service，@Repository，POJO以及VO组成。VO包括SysRes-ult、PageResult、ImageVO和ItemVO。（VO是一种抽取共性的写法，应该和前后端分离的写法有关。）

XxxMapper.java接口文件有一个对应的.xml映射文件用于映射MyBatis，文件以如下内容打头：

<?xml version="1.0" encoding="UTF-8" ?>

<!DOCTYPE mapper

PUBLIC "-//mybatis.org//DTD Mapper 3.0//EN"

"http://mybatis.org/dtd/mybatis-3-mapper.dtd">

.xml主体只有一个<mapper>顶级语言块，通过namespace属性与Mapper.java接口对应。<mapper>内部有<select><insert>等语言块，它们的id属性与接口方法对应，resultType属性与结果集类型对应。

MyBatis自动将结果集的数据封装为对象，如果返回List集合，则自动封装为List，User充当其中的泛型对象；如果返回值User对象，则直接返回User对象。（resultType="com.jt.pojo.

User"。）

因为MyBatis将结果集封装为对象，所以传值和取值都可以多考虑使用POJO对象。（注意传值时需要@RequestBody注解将JSON转为Java对象。）

XxxController是回传数据的最后一道关卡，所以返回值应该是SysResult。

因为用户登录是一个临时状态，页面关闭后就不用保留数据了，所以前端采用了获取用户token信息的方式来维持登录状态，所以从XxxService的返回值只需要是一个String类型的token。

let token = result.data //前端代码

window.sessionStorage.setItem("token",token)

报错/Request method 'GET' not supported是因为后台要求@PostMapping，但是采取了浏览器访问。

报错/ Field userMapper in com.jt.service.UserServiceImpl required a bean of type 'com.jt.mapper

.UserMapper' that could not be found。因为我是新建项目复制的src目录和UserMapper.xml文件，有两个值得注意的地方。首先在application.yml文件中有配置映射文件路径mybatis.mapper-locations: classpath:/mappers/\*.xml，应保证粘贴UserMapper.xml时路径相符。其次，整个项目新建时主启动类Jt1Application是没有复制的，需要手动加@MapperScan("com.jt.mapper")注解以将该路径下的mapper接口交给容器管理。

2022/1/26耗时一下午的bug。我按照自己的记忆复制了jt项目为jtReview，项目能启动成功，但始终报错/Invalid bound statement (not found): com.jt.mapper.UserMapper.\*。我曾尝试过取消MyBatis-Plus的依赖释放MyBatis的依赖，但由于UserMapper继承了MyBatis-Plus的BaseMapper我放弃了。实际上就差这一步就能解决异常。

用户登录实现：

MD5信息摘要算法（Message-Digest Algorithm），一种被广泛使用的密码散列函数，可以产生出一个128位的散列值，用于确保信息传输完整一致。MD5常用于密码管理，电子签名，垃圾邮件筛选等。

配置路由先定义路由占位符<router-view>，然后通过children关键字实现路由规则。

表中tinyint(1)字段的值1/0可以直接跟true/false映射？

@RestControllerAdvice为解决异常问题Spring专门为针对Controller层开发的注解

@ExceptionHandler(RuntimeException.class)切入点表达式

@TableName("user")对象与表一一映射/来自苞米豆

@TableId(type = IdType.AUTO)标识主键

@TableField("username")属性与字段映射，同名时可省略。

使用MBP步骤：导包，编辑POJO对象，继承BaseMapper，修改yml配置。

BaseMpper当中新增方法和修改方法各有几个？

1. 2。

QueryWrapper的一些方法，= eq, > gt, < lt, >= ge, <= le, != ne。

like like "%xxxx%"

likeLeft like "%xxxx"

likeRight like "xxxx%"。

MBP适用于单表操作，所有单表相关的操作都封装了对应的方法，所以如果只操作单表甚至不用写映射文件。

三个MybatisPlus注解：

@TableName("xxxx") POJO与数据表绑定

@TableId(type = IdType.AUTO)标识主键/主键自增

@TableField(exist = false)标识业务数据/非数据库字段

Item与ItemDesc有各自的Mpper类，但共用同一个Controller和Service。

分目录存储可以按HashCode或日期存储。

业务耦合时可以注入其它层级的service。

修改nginx.conf文件以配置图片服务器：

server {

listen 80;

server\_name image.jt.com;

#你少个分号试试？

location / {

root D:/JT-SOFT/image;

}

}

到C:\Windows\System32\drivers\etc路径下修改hosts文件，实现域名与IP的映射。

默认情况下hosts文件是一堆注释，做以下配置后浏览器访问以下域名出现nginx页面即映射成功。（首先要start nginx。）

127.0.0.1 image.jt.com

127.0.0.1 manage.jt.com

127.0.0.1 web.jt.com

Edit Configurations-->Allow parallel run，边修改application.yml的端口号边开启多个服务器。

修改nginx.conf文件以配置后台服务器：

upstream tomcats {

server 127.0.0.1:8091;

server 127.0.0.1:8092;

server 127.0.0.1:8093;（注意分号！！！）

}

2022/1/3出现nginx无法启动的情况，和修改前的nginx.conf文件对比并逐步修改后发现少了三个分号...

server {

listen 80;

server\_name manage.jt.com

location / {

proxy\_pass http://tomcats;

}

}

配置好以后记得nginx -s reload。

然后到浏览器访问manage.jt.com/getPort。

但是后端@Value("$server.port")取值遇到一些问题，只要打开这一句一启动就报错，访问http://manage.jt.com/getPort结果为current port is null。

2022/3/1使用vue-cli登录jtadmin时要求application.yml配置的server.port必须是8091。（记得检查端口/用户/密码，hosts配置，axios的请求根目录axios.defaults.baseURL，。）

前端项目发布，到jtadmin/main.js将 axios的请求根目录axios.defaults.baseURL = 'http://local-host:8091/'修改为axios.defaults.baseURL = 'http://manage.jt.com/'。（和上一个bug好像有关系啊。）

另一个要修改的是addItem.vue的uploadUrl。

打开vue cl工具，停掉server，打开build。（忽略3个警告。）编译后会在jtadmin下生成一个前端静态资源文件目录dist。

将打包好的dist复制到nginx根目录，并到nginx.conf配置前端服务器代理如下，此时nginx用作web服务器。

server {

listen 80;

server\_name web.jt.com;

location / {

root dist;

index index.html;

}

}

这样就可以通过web.jt.com访问jtadmin系统了。（注意是http协议。）

Windows端项目发布到此结束。

Linux端项目发布，install将项目打包成8091.jar，复制到usr/local/src/tomcats目录下。nohup java -jar 8091.jar => 8091.log &发布项目。

2022/1/15项目发布后在本地访问的第一个路径<http://192.168.126.129:8091/rights/getRights-List>，报错/Could not create connection to database server. Attempted reconnect 3 times. Giving up.

刚开始以为是连接数据库的驱动版本有问题，正改驱动名的时候突然发现是端口号的问题，以为我在本地用的是3307，Linux下默认3306。（耗时0.5h。）

查询java的服务项 jps

查询任意服务的进程 ps -ef | grep nginx/java/mysql

杀死进程 kill pid

关闭进程 kill -15 pid

强制关闭 kill -9 pid

查看日志 cat 8091.log

浏览日志后10行 tail -10 8091.log

动态查看日志 tail -f 8091.log

编辑脚本 vim start.sh

#!/bin/sh

nohup java -jar 8091.jar => 8091.log &（不要把nohup写成了nohop。）

nohup java -jar 8092.jar => 8092.log &

nohup java -jar 8093.jar => 8093.log &

运行脚本 sh start.sh

jt.sql

item\_desc全部清空并修改表结构，[item\_desc表字段修改.png](file:///D:\Pkg\图片\item_desc表字段修改.png)。

整数运算一定比浮点数运算快，所以可以给商品价格提升100倍，取用的时候再除以100。

item与item\_Desc一一对应，item.id = item\_desc.id。

[Linux安装MariaDB数据库](https://blog.csdn.net/qq_16804847/article/details/108450506?ops_request_misc=%257B%2522request%255Fid%2522%253A%2522164207941916780366549292%2522%252C%2522scm%2522%253A%252220140713.130102334.pc%255Fblog.%2522%257D&request_id=164207941916780366549292&biz_id=0&utm_medium=distribute.pc_search_result.none-task-blog-2~blog~first_rank_ecpm_v1~rank_v31_ecpm-1-108450506.nonecase&utm_term=Linux&spm=1018.2226.3001.4450)，[Linux安装MariaDB数据库/哔哩哔哩](https://www.bilibili.com/video/BV1WB4y1M7ma?spm_id_from=333.999.0.0)，

2022/1/13按照以上教程修改数据库host权限并临时关闭Linux防火墙后，尝试在本地远程连接数据库报错1130/ "Host 'xxxx' is not allowed to connect to this MySQL server"。经查是当我修改host权限后少了flush privileges这一步，重新刷新后连接成功。（连接Linux的IP就是inet 192.168.241.3。）（远程连接MariaDB的端口号是3306。）

[Linux安装Nginx步骤](https://blog.csdn.net/qq_16804847/article/details/109580325?ops_request_misc=%257B%2522request%255Fid%2522%253A%2522164220917216780271973294%2522%252C%2522scm%2522%253A%252220140713.130102334.pc%255Fblog.%2522%257D&request_id=164220917216780271973294&biz_id=0&utm_medium=distribute.pc_search_result.none-task-blog-2~blog~first_rank_ecpm_v1~rank_v31_ecpm-3-109580325.nonecase&utm_term=%E5%AE%89%E8%A3%85&spm=1018.2226.3001.4450)，[第三阶段复习](https://harrylyj.blog.csdn.net/article/details/122467079?spm=1001.2014.3001.5502)，

Score

Score目录结构由@Controller，@Service，@Repository和POJO组成。

若依

[若依权限管理子系统简介](https://yutian.blog.csdn.net/article/details/118579638)，

[微信小程序登录流程图.png](file:///D:\Pkg\图片\Java\微信小程序登录流程图.png)，[微信小程序登录流程](https://blog.csdn.net/qq_37896194/article/details/90231280?ops_request_misc=%257B%2522request%255Fid%2522%253A%2522164542757216780264069076%2522%252C%2522scm%2522%253A%252220140713.130102334..%2522%257D&request_id=164542757216780264069076&biz_id=0&utm_medium=distribute.pc_search_result.none-task-blog-2~blog~sobaiduend~default-2-90231280.nonecase&utm_term=%E5%B0%8F%E7%A8%8B%E5%BA%8F%E7%99%BB%E5%BD%95%E6%B5%81%E7%A8%8B%E5%9B%BE&spm=1018.2226.3001.4450)，

微信小程序