**挑战高薪面对面**

**（Java课程 \_ V202302）**

**★★★ 面试问题说明：**

1、根据大厂技术面试的实际场景，技术面试问题分为知识类面试问题和项目类面试问题。

2、所有知识类面试问题，按照技术难度和出现频率分类：

**技术难度：**基础1、拨高2、挑战3

**出现频率：**常见1、高频2、必考3

**★★★ 技术面试考核目标：**

**1、听得懂：**能准确理解面试官提问的重点和考核方向。

**2、答得出：**能使用自己的语言，准确回答面试问题。

**3、有拓展：**能在回答面试官问题的基础上，适当拓展问题深度和广度。

后期补充：

1 最后面再列题目列表，只有题，没有答案

并有表格，可以标记，不熟悉的题

# 第一部分：知识类面试问题

## 模块1：JavaSE 基础编程

### 面试题1.1 java为什么能够跨平台运行？

**【技术难度：1 出现频率：2 】**

Java程序的代码文件需要先编译成字节码文件，字节码文件再通过不同硬件平台上安装的对应版本的JVM虚拟机翻译成当前平台的机器码，交给当前平台执行。

因此，不同的系统平台只需要安装对应系统版本的JVM虚拟机，就可以运行相同的Java程序，就实现了跨平台运行。

----------------扩展(无需背诵)：

Java的代码文件后缀为“.java”

Java的字节码文件后缀为“.class”

### 面试题1.2 Java代码是如何运行的?

**【技术难度：1 出现频率： 1 】**

Java的代码文件，通过javac.exe程序编译成字节码文件，再通过java.exe程序启动JVM 虚拟机，JVM虚拟器将字节码翻译成机器码交给当前平台执行。

### 面试题1.3 JVM，JRE，JDK 分别是什么

**【技术难度： 2 出现频率：1 】**

1.JVM：虚拟出来的一台计算机，将字节码翻译成当前平台的机器码给平台执行；

2.JRE：运行JAVA程序的最小环境，JRE等于JVM加Java的基础类库；

3.JDK：开发JAVA程序的最小环境，JDK等于JRE加开发工具（编译javac.exe、运行java.exe等）。

总结：开发java程序的最小环境是JDK，运行Java程序的最小环境是JRE，JDK包含JRE，JRE包含JVM。

----------------扩展(无需背诵)：

JVM：Java Virtual Machine - Java虚拟机

JRE：Java Runtime Environment - java运行环境

JDK：Java Development Kit - java开发工具

### 面试题1.4 Java中有哪些基本类型？

**【技术难度：1 出现频率：1 】**

分类分为整型、浮点型、布尔型、字符型：

1.整型有byte、short、int、long，用于存整数，对应的占用字节数为1、2、4、8字节；

2.浮点型有float、double，用于存小数，对应的占用字节数为4、8字节；

3.布尔型有boolean，用于存真假值，占1字节；

4.字符型有char，用于存单个字符，占2字节。

----------------扩展(无需背诵)：

辅助记忆：

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 类型 | byte | short | int | long | float | double | boolean | char |
| 字节数 | 1 | 2 | 4 | 8 | 4 | 8 | 1 | 2 |

### 面试题1.5 char为什么能存储一个汉字？

**【技术难度： 1 出现频率：1 】**

char型变量是用来存储Unicode字符集中的字符的，Unicode字符集中包含了全世界所有的字符。

----------------扩展(无需背诵)：

char表示字符，采用UTF-16编码，UTF-16编码实现了Unicode字符集

### 面试题1.6 谈谈&和&&的区别？

**【技术难度： 1 出现频率：1 】**

1.&和&&都可以用作逻辑与的运算符，表示逻辑与，当运算符两边的表达式结果都为true时，整个运算结果才为true，否则只要有一边为false，则结果为false；

2.&&还具有短路的功能，从左往右看，若第一个表达式为false，则不再计算第二个表达式；

3.&还可以用作位运算符，当&操作符两边的表达式不是boolean类型时，&表示按位与操作。

### 面试题1.7 switch语句里面的条件可不可以是byte、long、String？使用时候还应注意什么？

**【技术难度： 1 出现频率：1 】**

switch里面的条件必须是能隐式的转化成为int的，所以byte可以，long不行。jdk1.7之后String类型的也可以。

使用switch时应注意它的穿透，每个case后要跟break避免发生穿透现象。

### 面试题1.8 switch中支持的类型

**【技术难度： 1 出现频率：1 】**

支持byte、short、int、char和它们对应的包装类，以及枚举enum。jdk1.7开始还支持 String字符串。

### 面试题1.9 return，break，continue 的区别

**【技术难度： 1 出现频率：1 】**

1.return返回，离开当前方法，当前方法后面的代码不再执行；

2.break打破，跳出所属的循环体，当前循环剩下的循环次数不再执行；

3.continue继续，跳过本次循环中后面的代码，进入下次循环。

## 模块2：Java 面向对象编程

### 面试题2.1 JVM内存划分

**【技术难度： 3 出现频率：3 】**

第一层：

JVM内存可以分为线程共享部分和线程独享部分，共享部分有堆和方法区（或称元空间），独享部分有栈、本地方法栈、程序计数器。

第二层：

先看线程共享部分：

1.堆用来存储所有new出来的对象实例及数组；

2.方法区用来存储已被JVM加载的类信息、静态变量、运行时常量池等。

再看线程独享部分：

1.虚拟机栈是为Java方法服务的，每个Java方法执行时会在虚拟机栈中创建一个栈帧，用于存储局部变量表、操作数栈、静态/动态链接、方法出口等；

2.本地方法栈为native本地方法服务，内部结构和虚拟机栈类似；

3.程序计数器记录线程执行的位置，此区域是jvm中唯一不可能出现内存溢出或泄露的位置。

第三层：

1. 关于默认值是这样：

1.1 虚拟机栈中的数据没有默认值，要初始化才能用，所以方法中的局部变量必须先赋值。

1.2 堆中的数据有默认值，各个类型与对应的默认值分别是这样：

1. byte、short、int、long的默认值为0；
2. float、double的默认值为0.0；
3. boolean的默认值为false；
4. char的默认值为'\u0000'（空字符）；
5. 引用类型的默认值为null。

2. 关于栈帧中的局部变量表、操作数栈、静态/动态链接、方法出口的细节是这样：

2.1 局部变量表

它是一组变量值存储空间，用于存放方法参数和方法内定义的局部变量。一个局部变量可以保存类型为boolean、byte、char、short、int、float、reference和returnAddress类型的数据。局部变量表的容量以变量槽(Variable Slot)为最小单位，JVM规范只要求一个槽至少能存放一个32位以内的数据类型，所以long或double类型数据可能会占用两个槽。

2.2 操作数栈

它也常称为操作栈，它是一个后入先出栈。操作数栈的每一个元素可以是任意Java数据类型，32位的数据类型占一个栈容量，64位的数据类型占2个栈容量，当一个方法刚刚开始执行时，其操作数栈是空的，随着方法执行和字节码指令的执行，会从局部变量表或对象实例的字段中复制常量或变量写入到操作数栈，再随着计算的进行将栈中元素出栈到局部变量表或者返回给方法调用者，也就是出栈/入栈操作。一个完整的方法执行期间往往包含多个这样出栈/入栈的过程。

2.3 静态/动态连接

在一个class文件中，一个方法要调用其他方法，需要将这些方法的符号引用转化为其在内存地址中的直接引用，而符号引用存在于方法区中的运行时常量池。

Java虚拟机栈中，每个栈帧都包含一个指向运行时常量池中该栈所属方法的符号引用，持有这个引用的目的是为了支持方法调用过程中的动态连接。

这些符号引用一部分会在类加载阶段或者第一次使用时就直接转化为直接引用，这类转化称为静态解析，或称静态链接。另一部分将在每次运行期间转化为直接引用，这类转化称为动态连接。

2.4 方法出口

一般来说，方法正常退出时，调用者的PC计数值可以作为返回地址，栈帧中可能保存此计数值。而方法异常退出时，返回地址是通过异常处理器表确定的，栈帧中一般不会保存此部分信息。

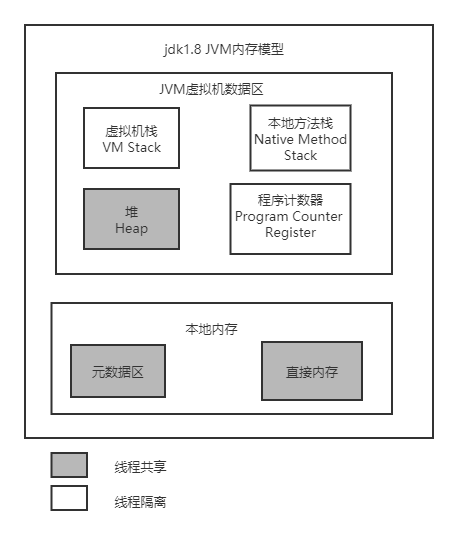
----------------扩展(无需背诵)：

1.JDK1.8的元空间并不在虚拟机中，而是使用本地内存，这样做可以很大程度避免JDK1.7以前将这些数据存放在堆内存中引发的内存溢出问题；

2.运行时常量池主要用于存放被JVM加载类的各个方法、字段的最终内存布局信息等等，以便于JVM调用；

3.直接内存不是虚拟机运行时数据区的一部分也不是Java虚拟机规范中定义的内存区域。在NIO(Non-blocking IO非阻塞输入输出操作，相关类在java.nio包)中，它可以使用Native函数直接分配堆外内存，然后通过一个存储在Java堆中的DirectByteBuffer对象作为这块内存的引用进行操作（可以提高性能）；

4.jdk1.8 JVM内存模型，如下图所示：



### 面试题2.2 heap和stack有什么区别?

**【技术难度： 3 出现频率：3 】**

1.stack栈内存：

当程序进入一个方法时，会为这个方法单独分配一块私属存储空间，叫做栈帧，用于存储这个方法内部的局部变量等，当这个方法结束时，对应栈帧释放，栈帧中变量随之释放；

口诀：栈内存，放局部变量，方法结束就消失

2.heap堆内存：

一般用于存放new关键字创建的对象实例，对象实例不会随方法的结束消失，而是由垃圾回收机制处理。

口诀：堆内存，放对象实例，垃圾回收机制来管理

### 面试题2.3 面向对象的基本特征是什么？

**【技术难度： 2 出现频率：2 】**

第一层：

面向对象的基本特征是指抽象、封装、继承、多态。

第二层：

1.抽象是指将现实世界中的某类东西提取出来，用程序代码表示（抽象只关注业务需要的部分）；

2.封装是指私有化属性，提供公共的get和set方法访问；

3.继承是指子类继承父类非私有的属性和方法；

4.多态是指同一种事物在不同情况下有不同的表现。

### 面试题2.4 java中实现多态的机制是什么？

**【技术难度： 2 出现频率：2 】**

方法重写、重载、父类引用指向子类对象、可变长参数。

### 面试题2.5 成员变量与局部变量的区别有哪些？

**【技术难度： 1 出现频率：1 】**



### 面试题2.6 值传递和引用传递的区别

**【技术难度：1 出现频率： 1 】**

1.值传递：基本数据类型做的是值传递，将当前变量的值复制一份存入其它变量。对其它变量的值进行修改时，当前变量的值没有影响。

2.引用传递：引用数据类型做的是引用传递，将当前引用变量的内存地址存入其它引用变量。通过其它引用变量对堆中的值修改时，再通过当前引用变量获取的是堆中修改以后的值。

### 面试题2.7 当一个对象被当作参数传递到一个方法后，此方法可改变这个对象的属性，并可返回变化后的结果，那么这里到底是值传递还是引用传递?

**【技术难度： 1 出现频率：1 】**

是值传递。当一个对象实例作为一个参数被传递到方法中时，参数的值就是对该对象的引用的一个副本。

### 面试题2.8 为什么 Java 中只有值传递？

**【技术难度：1 出现频率：1 】**

Java中的引用传递实际上是一个伪概念，当传递的参数为引用类型的对象时，对象仍存在堆中，将其引用创建一个副本来传递给形参，所以本质上还是值传递，将引用作为值传递。

### 面试题2.9 static关键字的作用

**【技术难度：2 出现频率：1 】**

1.static可以修饰内部类、静态导包、属性、方法、静态代码块；

2.static修饰的属性和方法属于类，一般用类名访问，只有一份在方法区；

3.static修饰的属性和方法存放在方法区中，除非卸载类，否则GC不予回收，所以不能什么都用static修饰。

### 面试题2.10 静态变量和实例变量的区别？

**【技术难度： 1 出现频率：2 】**

1.静态变量 (或称类变量) 和类相关，通过类名调用。实例变量(或称成员变量，属性) 和对象相关，需要先创建对象，通过对象名调用；

2.静态变量存在方法区内存，只有一份，实例变量存在堆内存。

### 面试题2.11 一个静态方法，里面可不可以用this和super关键字

**【技术难度： 1 出现频率：1 】**

不能。因为this代表的是调用这个方法的对象的引用，super代表当前父类对象的引用，而静态方法是属于类的，不属于对象。静态优先于对象，静态方法成功加载后，对象还不一定存在。

### 面试题2.12 Overload和Override的区别

**【技术难度： 2 出现频率：2 】**

方法重载 (Overload)，发生在同一个类中：

1.方法名相同；

2.参数列表不同 (参数类型，参数顺序，参数个数)；

1. 与访问修饰符，返回值类型无关。

方法重写 (Override)，发生在子类中：

1.方法名与父类方法名相同；

  2.参数列表与父类方法相同(参数类型，参数顺序，参数个数)；

  3.返回值类型与父类方法相同或是其子类；

  4.访问权限不能严于父类方法；

  5.抛出的异常与父类相同或是其子类。

### 面试题2.13 Overload方法是否可以改变返回值的类型？

**【技术难度： 2 出现频率：2 】**

Overload重载方法可以改变返回值类型。

### 面试题2.14 为什么方法不能根据返回类型来区分重载？

**【技术难度：2 出现频率：2 】**

方法重载与返回值无关。

### 面试题2.15 构造器可不可以被重载或重写？

**【技术难度：2 出现频率：2 】**

构造方法可以重载，不可以被子类重写，因为不可继承父类构造方法。

### 面试题2.16 在 Java 中定义⼀个不做事且没有参数的构造⽅法的作⽤

**【技术难度： 2 出现频率：1 】**

第一层：

如果子类调用到了父类无参构造方法时，父类得有无参构造方法，否则报错。

第二层：

比如这种情况，若⽗类中只定义了有参构造方法，⽽在⼦类的构造⽅法中⼜没有⽤super来指定调⽤⽗类中特定的有参构造⽅法，就会默认调⽤⽗类的无参构造方法，此时编译时将发⽣错误，因为 Java 程序在⽗类中找不到无参构造方法。

### 面试题2.17 创建一个子类对象，请写出：父类静态代码块，父类构造方法，父类代码块，子类代码块，子类构造，子类静态代码块。这六个对象的执行顺序

**【技术难度： 2 出现频率： 1 】**

1.父类静态代码块；

2.子类静态代码块；

3.父类实例代码块；

4.父类构造方法；

5.子类实例代码块；

6.子类构造方法。

### 面试题2.18 Java中访问修饰符有哪些? 作用域public，private，protected，以及不写时的区别

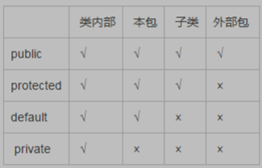
**【技术难度： 1 出现频率： 1 】**

1.public公开的，修饰成员变量、方法和类，可以被任何位置访问；

2.protected受保护的，修饰成员变量和方法，可以被本类、同包中的类、非同包的子类访问；

3.不写就是default默认的（或称friendly友好的），修饰成员变量、方法和类，可以被本类、同包中的类访问；

4.private私有的，修饰成员变量和方法，只可在本类中被访问。



### 面试题2.19 final关键字的用法？

**【技术难度：1 出现频率：1 】**

1 final可以修饰类，代表最终类，不可被继承；

2 final可以修饰方法，代表最终方法，不可被重写覆盖；

3 final可以修饰属性，代表常量，一旦赋值不可更改(给null也是赋值)。

--------扩展(无需背诵)：

注意final修饰的引用类型变量，其引用值不可变，而引用值指向的对象的属性不受此限制。

### 面试题2.20 抽象类和接口的区别？什么时间考虑用抽象类，什么时间考虑用接口？

**【技术难度：2 出现频率：2 】**

1.抽象类继承自Object，接口不是；

2.抽象类有构造方法，接口没有；

3.抽象类可以声明成员变量、静态变量、常量，接口只能声明静态常量；

4.抽象类可以声明成员方法、抽象方法、final方法、静态方法，接口可以声明抽象、静态方法、default方法；

5.抽象类是单继承、多实现，接口只能继承多个其他接口。

### 面试题2.21 接口是什么？为什么要使用接口而不是直接使用具体类？

**【技术难度： 2 出现频率：2 】**

第一层：

接口声明对象将会实现的行为，实现类与接口是like a的关系。

第二层：

接口可以用来控制其实现类必须要实现它的抽象方法，并且一个类可以实现多个接口，相对于类之间的单继承更有可扩展性。

第三层：

定义好接口后可以直接调用接口的方法而无需等待其他开发人员实现具体方法后才能调用，大大提高了开发效率，并且使用接口开发能够降低程序的耦合性，从而提高程序的可维护性。

### 面试题2.22 什么是类？什么是对象？什么是接口？接口和类的继承有什么本质区别

**【技术难度： 2 出现频率：2 】**

一：

1.对象是用来描述客观事物的一个实体，由一组属性和方法构成；

2.类是一组具有相同属性和方法的对象的集合，定义了对象将会拥有的特征和行为；

3.接口包含一系列方法的声明，一个接口只有方法的特征没有方法的实现。

总结：类是对象的类型，对象是类的实例，接口声明了对象将会实现的行为。

二：

一个类只能继承一个父类，可以实现多个接口。

### 面试题2.23 Java中有没有多继承？

**【技术难度： 2 出现频率： 2 】**

java中没有多继承，不过可以多实现，一个类实现多个接口。通过接口可以近似的实现多继承，而且接口可以继承多个其他接口。

### 面试题2.24 什么是内部类？分为哪几种？

**【技术难度：2 出现频率：1 】**

一：

内部类是指在一个外部类的内部再定义一个类，它作为外部类的一个成员，并且依附于外部类而存在。内部类可为静态、可用各种访问权限修饰符，而外部类只能用public和缺省的包访问权限。

二：

内部类主要有成员内部类、局部内部类、静态内部类、匿名内部类。

### 面试题2.25 为什么需要内部类？

**【技术难度：2 出现频率：1 】**

使用内部类最吸引人的原因是实现多重继承。

每个内部类都能独立地继承自一个（接口的）实现，无论外部类是否已经继承了某个（接口的）实现，对于内部类都没有影响。接口解决了部分问题，而内部类有效地实现了“多重继承”，一些设计与编程问题因此得以解决，这使得多重继承的解决方案变得完整。

### 面试题2.26 局部内部类特性

**【技术难度： 2 出现频率：1 】**

局部内部类可以直接访问所在方法的final常量，jdk1.7之前 final必须加，jdk1.8开始，给变量赋值后，final可以省略。

### 面试题2.27 内部类与静态内部类的区别

**【技术难度： 2 出现频率：1 】**

内部类：

1.内部类中的变量和方法不能声明成静态的；

2.内部类实例化时，若B是A的内部类，需要先创建A的对象，再通过A对象的引用创建B对象（A.B b = new A().new B();）；

3.内部类可以直接调用外部类的一切资源，包括私有的。

静态内部类：

1.静态内部类的属性和方法，都可以声明为静态或非静态的；

2.实例化静态内部类时，若B是A的内部类，可直接通过A.B创建B对象（A.B b = new A.B();）；

3.静态内部类只能直接调用外部类的静态属性及静态方法。

--------扩展(无需背诵)：

静态内部类其实叫静态嵌套类（static nested classes）。

静态嵌套类就是原本可以完全独立存在的类，不过还是借外部类的壳用一下，来隐藏它自己。

内部类是外部类的一部分，它知道外部类的全部，没有外部类就没有内部类，内部类对象以外部类对象存在为前提。

### 面试题2.28 异常的理解? 什么是检查性异常和非检查性异常？ Error和Exception的区别?

**【技术难度：2 出现频率： 1 】**

第一层：

Java异常的基类是Throwable，其下有两个子类，分为Exception异常和Error错误。Error一般指JVM出现的错误，程序无法做处理。Exception是程序本身可以处理的异常，分为检查异常和运行时异常。

第二层：

检查异常都直接继承自Exception类。运行时异常都继承自RuntimeException类（Exception的子类）。

第三层：

检查异常Java程序必须显示处理，try/catch捕捉或throws声明抛出。非检查异常Java程序可选择处理或不处理。

--------扩展(无需背诵)：

Runtime - 运行时

checked - 检查

unchecked - 非检查

### 面试题2.29 说出十种常见的异常，请举例有哪些异常

**【技术难度： 2 出现频率： 1 】**

5个运行时异常：

NullPointerException         空指针异常

NumberFormatException          数字格式化异常

IndexOutOfBoundsException        下标越界

ArrayIndexOutOfBoundsException   数组下标越界

ClassCastException           类型转换异常

口诀：空格下数类

5个检查异常：

FileNotFoundException    文件未找到异常

ClassNotFoundException   类未找到异常

IOException              IO流异常

SQLException             SQL异常

EOFException             文件末尾异常

口诀：文件类未找到，is末尾也异常

### 面试题2.30 throw和throws的区别

**【技术难度： 2 出现频率：1 】**

1.throws用在方法声明的位置，throw用在方法体内；

2.throws是可能抛出异常，throw是一定抛出异常；

3.throws后面跟异常类名，throw后面跟异常对象。

### 面试题2.31 final，finally，finalize的区别

**【技术难度： 2 出现频率： 1 】**

1.final用于声明属性、方法和类，分别表示属性不可变、方法不可覆盖、类不可继承；

2.finally是异常处理语句结构的一部分，表示总是执行；

3.finalize是Object类的一个方法，在垃圾收集器执行的时候会调用被回收对象的此方法，可以覆盖此方法提供垃圾收集时的其他资源回收，例如关闭文件等，JVM不保证此方法总被调用。

## 模块3：JavaSE 高级编程

### 面试题3.1 什么是自动装箱与拆箱？用什么方式来装箱与拆箱？

**【技术难度： 2 出现频率： 1 】**

一：

⾃动装箱就是将基本数据类型自动转换成对应的包装类对象，⾃动拆箱就是将包装类对象自动转换为基本数据类型。

二：

自动装箱可以在将基本数据类型放入集合类时、方法传入参数为基本类型而返回值类型为包装类型时发生。

自动拆箱可以在包装类型和基本类型比较大小时、包装类型参与运算时、三目运算符中使用包装类型时发生。

### 面试题3.2 int和Integer有什么区别？

**【技术难度： 1 出现频率：1 】**

1.Integer是int的包装类，int是基本数据类型;

2.Integer变量必须实例化后才能使用，int变量不需要;

3.Integer变量存的是对象的引用，int变量是直接存储数据值;

4.Integer变量的默认值是null，int变量的默认值是0。

### 面试题3.3 Integer常量池

**【技术难度： 1 出现频率：1 】**

Integer自动装箱时，会使用valueOf()方法，-128~127之间的值直接从该包装类的常量数组中获取现成对象。

### 面试题3.4 字符串常量池

**【技术难度： 2 出现频率： 1 】**

1.jdk1.7之前字符串常量池在方法区，jdk1.7开始在堆中；

2.""声明的字符串对象，保存在字符串常量池中；

3.用""声明字符串时，如果字符串常量池中能找到该字符串直接返回内存地址，如果不能找到就创建字符串返回内存地址。

----------------扩展(无需背诵)：

System.out.println("123" == "123");//true

### 面试题3.5 这句代码创建了几个对象?  String str1 = new String("xyz");

**【技术难度：2 出现频率：1 】**

2个，"xyz"在字符串常量池，new String()在堆中。

### 面试题3.6 String和StringBuffer的区别？StringBuffer和StringBuilder区别？

**【技术难度： 2 出现频率：1 】**

1.String是不可变的,对String类对象的任何改变都会返回一个新的String对象，另外两个类是可变的，对他们的内容修改都是当前这个对象；

2.String类重写了Object类的equals()方法和hashcode()方法，另外两个类没有重写这些方法；

3.双引号创建的String字符串保存在字符串常量池，其他是在堆中创建对象；

4.StringBuffer是线程安全的，StringBuilder是非线程安全的，不考虑线程安全使用StringBuilder效率更高。

口诀：String不能改，改了就新建，其他都能改，改完还是它。

### 面试题3.7 String是基本数据类型吗？我可不可以写个类继承于String？

**【技术难度：2 出现频率：1 】**

不是，String是引用类型；String是final修饰的类，不可被继承。

### 面试题3.8 数字转字符有多少种方式，分别是什么

**【技术难度： 1 出现频率：1 】**

1.String.valueOf(123)；

2.""+123；

3.Integer.toString(123)。

### 面试题3.9 数组中有没有length()方法，String中有没有length（）方法？

**【技术难度：1 出现频率： 1 】**

数组没有lenth()方法，有lenth属性，String中有length()方法

### 面试题3.10 ==和equals的区别？

**【技术难度： 2 出现频率：2 】**

1.对于==来说，如果两边是基本数据类型做的是值的比较，如果两边是引用数据类型做的是内存地址的比较；

口诀：基本类型比较值，引用类型比地址

2.Object类的equals()方法，默认是使用==判断2个对象的内存地址是否相等，重写以后按照自定义规则比较2个对象；

口诀：默认==比地址，重写以后自定义

3.基本类型只能用==，没有equals()方法。

### 面试题3.11 为什么要有hashCode?

**【技术难度：2 出现频率： 1 】**

1.把对象的地址值存在于散列表中，通过hashcode可以快速找到此对象的地址；

2.通过hashcode可以快速判断对象的重复性；

3.在散列表中，先通过哈希码找对象，再用equals()比较可以提高效率。

口诀：找地址、判重复、提效率

### 面试题3.12 hashCode的作用是什么

**【技术难度：2 出现频率：1 】**

1.hashCode()是通过一种算法获取哈希码(也称为散列码)，实际返回⼀个int整数，在散列表中可以通过此方法获取对象的哈希码，进而快速找到该对象在散列表中的位置；

2 hashCode()主要在散列表中才有⽤，比如HashMap就用到了散列表。

### 面试题3.13 采用Hash的好处，什么是碰撞

**【技术难度： 2 出现频率：1 】**

1.Hash算法可以将把任意长度数据变换成固定长度的标志，此标志和源数据每个字节都有紧密关系并很难找到逆向规律，使用Hash算法可以提高存储空间利用率、数据查询效率等；

2.对不同的内容可能算出同一哈希值，这种现象称碰撞(或称哈希冲突)。

### 面试题3.14 有没有可能两个不相等的对象有有相同的 hashcode？

**【技术难度：2 出现频率：1 】**

可能会相同，因为哈希算法存在哈希冲突的问题，对不同的内容可能算出同一哈希值，这可能导致两个不同对象的hashCode相同。

### 面试题3.15 两个相同的对象会有不同的的hashcode吗？可以在hashcode()中使用随机数字吗？

**【技术难度： 2 出现频率：1 】**

可以这么写，但不推荐，除非刻意胡乱重写。根据java底层规范和标准，不允许发生这样的情况，对象相等应保证他们的hashcode相等，否则会导致使用hash的对象发生错误。

### 面试题3.16 hashCode()与equals()的相关规定

**【技术难度： 2 出现频率： 1 】**

第一层：

1.如果两个对象相等，hashcode⼀定相等，而且对两个对象分别调⽤equals()⽅法都应返回true；

2.若两个对象的hashcode相等，两个对象不⼀定相等；

3.若需重写，必须一起重写equals()和hashCode()⽅法，保证对象equals()相等的情况下，hashcode也要相等；

第二层：

4.hashCode()默认用对象内存地址计算一个哈希值，equals() 默认用==比较2个对象的内存地址是否相等，如果没有重写这两个方法，则该类的两个对象无论如何都不会相等（即使这两个对象的属性是相同的数据）。

### 面试题3.17 hashcode和equals的区别

**【技术难度：2 出现频率：1 】**

hashCode()默认用对象内存地址计算并返回一个整型值，不同内存地址可能算出相同的hashcode (哈希冲突)，equals()默认用==比较2个对象的内存地址是否相等。

简单说：

两个对象equals()相等，hashcode()一定相等

两个对象equals()不相等，hashcode()可能相等

两个对象的hashcode()不相等，equals()一定不相等

### 面试题3.18 Java中有多少种数据结构，分别是什么？

**【技术难度： 3 出现频率：3 】**

栈：元素是先进后出，后进先出。

队列：元素存储是排列有序的，一定保证先进的先出，后进的后出。

数组：指定类型，固定长度，元素存储地址是连续的。

树：元素以树形结构存储，只有一个根节点。

向量：动态数组，可以存储任何类型元素，动态长度，元素存储地址是连续的。

List：是列表，有下标值，存储元素可以重复，遍历元素是有序的。

Set：是散列集(或称哈希集)，无下标值，存储元素不可重复，遍历元素是无序的。

Map：是以键值对存储，一个key一个value，key不可以重复，value可以重复。

口诀：栈队数树向lsm

### 面试题3.19 List、Set和Map的区别？

**【技术难度： 2 出现频率：2 】**

1.List和Set是Collection的子接口，map不是；

2.List的底层是数组或链表实现，Set是散列表的方式实现，map是键值对的方式；

3.List是有序可重复的，Set是无序不可重复的，map中hashMap无序，LinkedHashMap有序，key不重复，value可重复；

4.List和Set可直接使用itertator来进行遍历，map需要先获取keySet或entrySet，通过它们来遍历。

### 面试题3.20 List遍历方式有多少种

**【技术难度： 2 出现频率：2 】**

1.下标遍历；

2.Iterator遍历；

3.Foreach遍历（最快）。

### 面试题3.21 Arraylist，Vector和Linkdlist 的区别

**【技术难度： 2 出现频率：2 】**

它们都是List接口的实现类。

ArrayList：

1.底层是数组，查询快增删慢，线程不安全，效率高；

  2.数组初始长度10，存满扩容，扩到原数组1.5倍。

Vector：

1.与ArrayList类似，底层是数组，查询快增删慢，线程安全，效率低；

2.数组初始长度10，存满扩容，扩到原数组2倍。

LinkedList：

1.底层是链表，查询慢增删快，线程不安全，效率高；

2.链表无扩容机制，修改节点之间关联就好。

### 面试题3.22 Collection和Collections的区别

**【技术难度： 2 出现频率：2 】**

1.Collection是集合类的上级接口，继承于他的接口主要有Set和List；

2.Collections是针对集合的一个工具类，提供一系列静态方法实现对各种集合的搜索、排序、线程安全化等操作。

### 面试题3.23 Comparable接口用法 或者 comparable和comparator接口的用途? 列出他们的区别? 另一种问法，想实现些比较或排序，或统计元素的个数，要用到比较器。

**【技术难度： 2 出现频率： 1 】**

第一层：

这两个接口一般用来实现集合内排序，Comparator相对于Comparable更加灵活，耦合度更低，Comparable还可用与比较本类两个对象的大小。

第二层：

Comparable接口里面有个compareTo()接口方法，当一个类的对象需要比较的时候，该类需要实现Comparable接口和他的compareTo()方法，当调用集合排序方法时，就会调用该类对象的compareTo()方法来实现对象的比较。

Comparator接是一个比较器接口，一般定义一个匿名类实现该接口和它的compare()方法，并创建对象，传入集合排序方法中。

### 面试题3.24 Set里的元素是不能重复的，那么用什么方法来区分重复与否呢? 是用==还是equals()? 它们有何区别?

**【技术难度： 2 出现频率： 2 】**

1.Set里的元素是不能重复的；

2.元素重复与否是使用equals()方法进行判断的；

3.==比较两个对象是比较对象的内存地址，equals()默认是使用==判断2个对象的内存地址是否相等，重写以后按照自定义规则比较2个对象。

### 面试题3.25 Map怎么遍历? 怎么遍历一个map？写一下

**【技术难度： 2 出现频率：2 】**

1.先调用map.keySet()方法获取所有的key，再通过这些key获取对应的value；

Map<String，Integer> map = new HashMap<>();

for(String key ：map.keySet()){

System.out.println(key + " - " + map.get(key));

}

2.先调用map.entrySet()方法获取所有的entry键值对对象，再从每个entry中取出key和value。

Map<String，Integer> map = new HashMap<>();

for (Map.Entry<String，Integer> entry ：map.entrySet()) {

System.out.println(entry.getKey() + " - " + entry.getValue());

}

### 面试题3.26 怎么获取Map所有的key，所有的value

**【技术难度： 2 出现频率：2 】**

1.Map调用keySet()方法获取所有的key值，是一个Set集合；

2.Map调用values()方法获取所有的value值，是一个Collection集合。

### 面试题3.27 HashMap与HashTable的区别

**【技术难度：2 出现频率：2 】**

1.继承的父类不同；

2.HashMap是非线程安全的，Hashtable是线程安全的 (低效)；

3.HashMap的key和value都允许null值，Hashtable的key和value都不允许null值；

4.HashMap数组初始长度16，扩容原数组的2倍，Hashtable数组初始长度11，扩容原数组的2倍 + 1。

### 面试题3.28 HashMap HashSet的底层实现

**【技术难度： 2 出现频率：3 】**

第一层：

HashSet底层就是HashMap。HashMap底层是哈希表数据结构，jdk1.8开始，哈希表等于数组加链表加红黑树(二叉树的一种)，数组的初始长度16，加载因子0.75，扩容到原数组的2倍。当单向链表的节点个数大于8并且数组长度大于64时，链表转红黑树，当节点的个数小于6，红黑树转回链表。

第二层：

调用map.put()方法时，会调用key的hashCode()方法计算哈希值，这个哈希值会经过再次计算得出最终哈希值，然后根据这个哈希值计算数组的存储下标，此时会有两种情况：

1.如果数组存储下标对应的空间没有值，key和value直接存入；

2.如果数组存储下标对应的空间有值(单向链表或红黑树)，就继续调用key的equals()方法和其中每个节点的key比较，若返回true，后面的value覆盖前面的value，若返回false，key和value存入。

### 面试题3.29 怎么获取指定路径下的全部文件

**【技术难度：1 出现频率：1 】**

1.使用File对象的list()方法可以获取目录中文件的文件名；

2.使用File对象的listFiles()方法可以获取目录中文件的文件对象。

----------------扩展(无需背诵)：

File f = new File(path); //根据路径创建File对象

String[] fileName = f.list(); //获取目录下的文件名

File[] files = f.listFiles(); //获取目录下的文件对象

### 面试题3.30 什么是IO流

**【技术难度： 1 出现频率： 1 】**

1.IO流分为输入流InputStream和输出流OutputStream；

2.输入流和输出流又分为字节流和字符流；

3.字符流的底层也为字节流；

4.操作文件的时候使用字节流，操作字符的时候使用字符流。

### 面试题3.31 字节流与字符流的区别

**【技术难度： 1 出现频率：1 】**

1.字节流以字节为单位进行操作，是万能的流，可以读写各种格式文件，比如图片视频等；

2.字符流底层是字节流，以字符为单位进行操作，用于操作文本文件。

### 面试题3.32 什么是缓冲区

**【技术难度：2 出现频率：1 】**

缓冲区分为输入缓冲区和输出缓冲区：

1.输入缓冲区，一次性从磁盘中读取足够大的数据到缓冲区内存中，程序后续每次从缓冲区内存中读取一定量的数据，用于减少读取磁盘次数；

2.输出缓冲区，每次把数据写到缓冲区内存中，当缓冲区内存数据达到一定量的时候，一次性写入磁盘，用于减少写入磁盘的的次数。

### 面试题3.33 java中有几种类型的流？JDK为每种类型的流提供了一些抽象类以供继承，请说出他们分别是哪些类？

**【技术难度： 2 出现频率：1 】**

Java中有字节流和字符流。字节流继承于InputStream、OutputStream，字符流继承于InputStreamReader、OutputStreamWriter。在java.io包中还有许多其他的流，主要是为了提高性能和使用方便。

### 面试题3.34 字节流有哪些? 缓冲流的加速方式

**【技术难度： 2 出现频率：1 】**

1

InputStream

FileInputStream

BufferedInputStream

ObjectInputStream

OutputStream

FileOutputStream

BufferedOutputStream

ObjectOutputStream

2

缓冲流内置了默认大小的数组

### 面试题3.35 什么是java序列化，如何实现java序列化？

**【技术难度： 2 出现频率： 1 】**

第一层：

通俗的说，序列化就是将内存中java对象写到硬盘上的过程，反序列化是将硬盘中的数据，反向读取到程序内存中重新构建对象。序列化需要实现Serializable接口，此接口中没有方法，只是个标示接口。

第二层：

反序列化会比较数据和.class文件的版本号(serialVersionUID)，如果修改过代码导致版本号与数据中的不一致，将无法反序列化，会抛出InvalidClassException异常。

第三层：

transient和static修饰的属性无法被序列化，反序列化不会调用构造方法。

### 面试题3.36 Java序列化中如果有些字段不想进行序列化，怎么办？

**【技术难度： 2 出现频率：1 】**

第一层：

1.使⽤ transient 关键字修饰变量，它只能修饰变量，它修饰的的变量不参与序列化，当对象被反序列化时，被它修饰的变量值会是堆中的默认值(0，0.0，null等)；

2.使用static修饰，序列化保存的是对象的状态，而static修饰的字段属于类，与对象无关；

第二层：

3.Externalizable，他是Serializable接口的子类，可以使用这个接口的writeExternal()和readExternal()方法可以指定序列化哪些属性。

### 面试题3.37 线程、进程的基本概念? 程序、进程、线程之间的关系?

**【技术难度： 1 出现频率：2 】**

第一层：

线程是程序执行的最小单元，是一个执行流程，一个执行路径，进程是程序一次动态执行的过程，执行完毕的一个完整过程；

第二层：

一个程序至少有一个进程，可以有多个不过一般是一个进程。一个进程中可以有多个线程。

### 面试题3.38 什么是多线程

**【技术难度： 1 出现频率：2 】**

如果程序只有一条执行路径，该程序就是单线程程序；如果程序有多条执行路径，该程序就是多线程程序。

### 面试题3.39 启动线程的方式？start or run?

**【技术难度： 1 出现频率：2 】**

调用Thread类对象的start()方法才是启动线程，run()方法是线程将要做的事，调用线程的run()只是在当前线程运行，不会启动另一个线程。

### 面试题3.40 线程的生命周期? 线程运行状态?

**【技术难度： 2 出现频率： 2 】**

第一层：

线程的生命周期分为5个状态：新建，就绪，运行，阻塞，死亡。

口诀：新就运阻死

第二层：

1.创建状态是指刚new出来的线程对象；

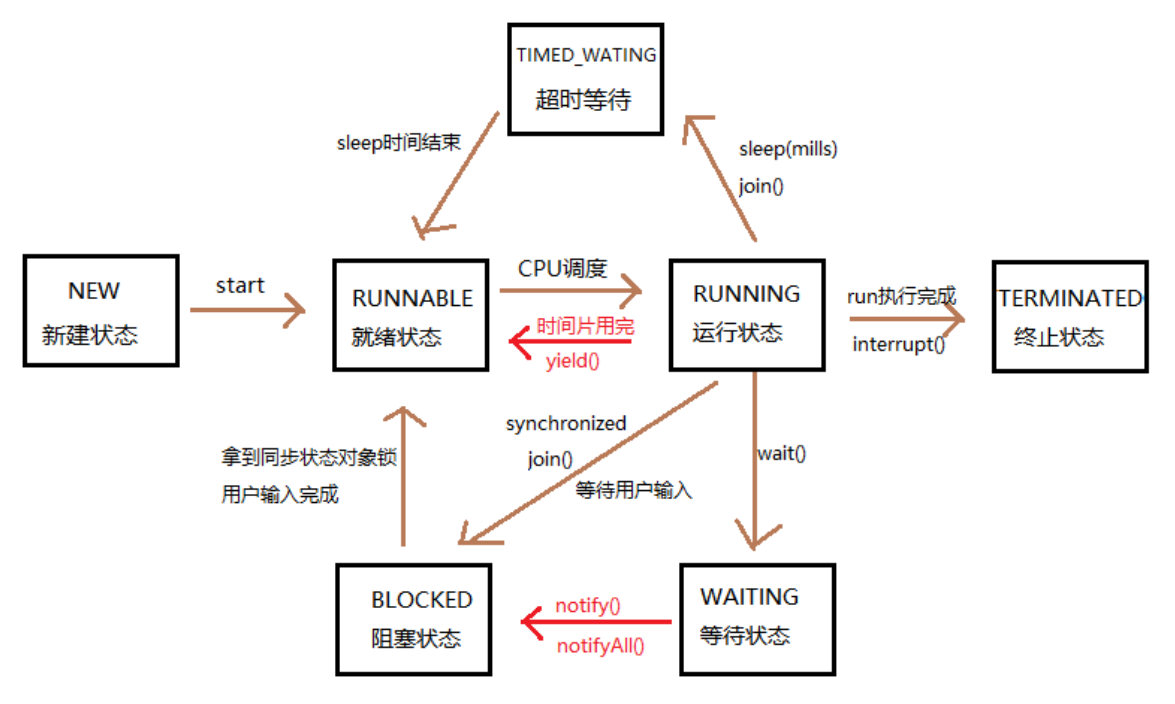
2.就绪状态是指调用start()方法；

3.运行状态是指获得了CPU时间片；

4.等待/阻塞状态，比如调用sleep()、join()、wait()方法以及IO操作；

5.死亡状态是指run()方法结束。

----------------扩展(无需背诵)：



### 面试题3.41 线程的调度策略

**【技术难度： 2 出现频率： 2 】**

线程调度器选择优先级最高的线程运行，不过如果发生以下情况，就会终止线程的运行：

1.线程体中调用了yield()方法让出了对CPU的占用权利；

2.线程体中调用了sleep()方法使线程进入睡眠状态；

3.线程由于IO操作受到阻塞；

4.另外一个更高优先级线程出现；

5.在支持时间片的系统中，该线程的时间片用完。

### 面试题3.42 Java 中用到的线程调度算法是什么？

**【技术难度：2 出现频率：1 】**

采用时间片轮转的方式。可以设置线程的优先级，会映射到下层系统的优先级上，如非特别需要，尽量不要用，防止线程饥饿（得不到cpu执行权）。

### 面试题3.43 在 java 中守护线程和用户线程区别？

**【技术难度： 1 出现频率：1 】**

第一层：

java中的线程分为守护线程和用户线程。守护线程主要是在后台运行为用户线程提供服务，若所有用户线程结束后只剩下守护线程，JVM自动停止。

第二层：

创建线程可以通过Thread类的实例方法setDaemon(true)设置为守护线程，否则默认为用户线程。setDaemon()方法必须在start()启动线程前调用，否则抛异常。

--------扩展(无需背诵)：

Daemon - 守护

User - 用户

### 面试题3.44 多线程的同步方式

**【技术难度：2 出现频率：2 】**

1.synchronized修饰的同步代码块；

2.synchronized修饰的同步方法；

3.Lock锁。

### 面试题3.45 多线程安全问题怎么解决

**【技术难度： 2 出现频率：2 】**

解决思路是尽量避免多个线程同时操作相同变量。

解决方案有这些：

1.多实例，缺点是浪费空间；

2.添加synchronized关键字，缺点是效率低，逐个线程排队执行；

3.使用本地线程变量ThreadLocal；

4.使用Lock锁，缺点是效率低，逐个线程排队执行；

5.使用局部变量，因为局部变量不存在线程安全的问题，缺点是多线程时不一定具备这种场景。

### 面试题3.46 当一个线程进入一个对象的一个synchronized方法后，其它线程是否可进入此对象的其它方法?

**【技术难度：2 出现频率：1 】**

假设这个线程进入的是该对象的synchronized实例方法，此时其它线程可以进入此对象的其它方法有未加锁的实例方法和类方法，以及synchronized加锁的类方法。

### 面试题3.47 简述synchronized与java.util.concurrent.locks.Lock的异同？synchronized和Lock锁两者区别? 谈谈 synchronized和ReentrantLock的区别

**【技术难度： 2 出现频率：1 】**

接下来以实现类ReentrantLock代表Lock锁来回答此问题：

第一层：

1.synchronized是java关键字; Lock是java接口，jdk1.5才出现；

2.synchronized会自动获取或释放锁，Lock锁需要调用lock()或unlock()方法手动获取或释放锁；

3.synchronized如果获取不到锁会一直等待，ReentrantLock如果获取不到锁可以设置时间，超过不等待；

第二层：

4.synchronized是非公平锁，ReentrantLock可以实现公平锁；

5.synchronized只能随机或者全部唤醒，ReentrantLock可以精确或分组唤醒；

6.synchronized锁适合少量代码的同步问题，Lock锁适合大量代码的同步问题。

----------第6点拓展(无需背诵)：

竞争激烈的情况下，Lock锁的性能优于synchronized。竞争不激烈的情况下，synchronized性能好一点，因为synchronized有个锁升级机制，根据竞争激烈程序给锁升级，从偏向锁到轻量级锁再到重量级锁，而到重量级锁的转换需要操作系统帮忙，需要花费较多时间。

### 面试题3.48 sleep()和wait()有什么区别?

**【技术难度：1 出现频率： 1 】**

1.sleep是Thread类的静态方法，wait是Object类的实例方法；

2.sleep不会释放对象锁，wait会释放对象锁；

3.sleep的休眠时间到达后自动进入就绪状态，wait的线程必须由notify()或notifyAll()唤，唤醒后进入同步阻塞状态。

### 面试题3.49 线程之间是如何通信的？

**【技术难度： 2 出现频率：1 】**

通过Object类的wait()、notify()、notifyAll()，以及Thread类的join()方法，可以进行消息传递，实现让线程等待或唤醒线程，多线程之间通信是为了避免对同一共享资源的争夺。

### 面试题3.50 为什么wait()，notify()和notifyAll()必须在同步方法或者同步块中被调用？

**【技术难度： 2 出现频率： 1 】**

1.调用wait()的线程会释放锁，很显然，先获得锁才能释放锁；

2.notify()、notifyAll()是将锁交给调用了wait()方法的线程，让其继续执行下去，前提是自身获得了锁才能交出锁。

### 面试题3.51 线程阻塞有几种情况？遇到阻塞怎么解决？

**【技术难度： 2 出现频率：2 】**

第一层：

线程阻塞的情况分三种：等待阻塞、同步阻塞、其他阻塞。

线程阻塞的解决方法有：

1.sleep()、wait()和join()方法引发的阻塞可以通过interrupt()方法中断；

2.同步锁引发的阻塞可以通过减少锁持有时间，读写锁分离，减小锁的粒度，锁分离，锁粗化等方式来优化锁的性能。

第二层：

三种线程阻塞展开说是这样：

1.等待阻塞是指运行的线程执行了wait()方法，JVM会把该线程放入等待池中；

2.同步阻塞是指运行的线程在获取对象的同步锁时，若该同步锁被别的线程占用，则JVM会把该线程放入锁池中；

3.其他阻塞是指运行的线程执行了sleep()或join()方法，或者发出了I/O请求时，JVM会把该线程置为阻塞状态。当sleep()状态超时、join()等待线程终止或者超时、或者I/O处理完毕时，线程重新转入就绪状态。

----------------扩展(无需背诵)：

阻塞状态是线程因为某种原因放弃CPU使用权，暂时停止运行。直到线程进入就绪状态，才有机会转到运行状态。

### 面试题3.52 如何安全中断运行中的线程？**interrupt()方法的作用？**

**【技术难度： 2 出现频率： 1 】**

Java Thread 的API里面虽然提供了一个 stop() 方法，可以强行终止线程，但这种方式是不安全的，因为有可能线程的任务还没有完成，突然中断会导致出现运行结果不正确的问题。

因此，在 Thread 里面提供了一个 interrupt() 方法，这个方法要配合isInterrupted()方法来使用，就可以实现安全地中断线程运行。 这种实现方法并不是强制中断，而是告诉正在运行的线程，你可以停止了。何时实际中断，取决于正在运行的线程，所以，它能够保证线程运行结果的安全性。

### 面试题3.53 volatile关键字的作用，能保证线程安全吗

**【技术难度： 2 出现频率： 1 】**

volatile关键字有两个作用，一是可以保证线程获取的数据是最新值，二是可以防止指令的重排序。它无法保证线程的安全性。

--------扩展(无需背诵)：

当一个共享变量被 volatile 修饰时，它能保证修改的值会立即被更新到主内存，当有其他线程需要读取时，会去主内存中读取新值。

### 面试题3.54 Java 中 ++ 操作符是线程安全的吗？

**【技术难度： 2 出现频率： 1 】**

不是。它涉及到多个指令，如读取变量值，增加，然后存储回内存，并不是原子性的操作，这个过程可能会出现多个线程交差。

一个解决方案是对有i++操作的方法加同步锁。另一个解决方案是使用支持原子操作的类，比如AtomicInteger（java.util.concurrent.atomic.AtomicInteger）。

### 面试题3.55 请说说ThreadLocal？请说说线程本地变量？请说说本地线程？

**【技术难度： 2 出现频率：1 】**

第一层：

ThreadLocal可以实现每⼀个线程都有⾃⼰的专属本地变量，创建了⼀个ThreadLocal变量之后，访问这个变量的每个线程都会有这个变量的本地副本。他们可以使⽤get()和set()⽅法来获取或修改当前线程所存的值，从⽽避免线程安全问题。

第二层：

每个线程的本地变量并不是存放在ThreadLocal实例中，而是放在线程对象自己的threadLocals变量中，也就是说，ThreadLocal本地变量是存放在具体的线程空间上，所以不使用本地变量的时候需要调用remove方法删除不用的本地变量，否则只要线程不终止那这些不用的变量就会一直存活在线程对象中。

### 面试题3.56 死锁的原因

**【技术难度： 1 出现频率： 1 】**

当两个或两个以上的线程（或进程）在执行过程中，互相争夺对方持有的锁资源，又不释放自己持有的锁资源，造成互相一直等待，此时若无外力作用，它们都将无法推进下去，这就是死锁。

### 面试题3.57 死锁与活锁的区别，死锁与饥饿的区别？

**【技术难度： 3 出现频率：1 】**

死锁和活锁的区别在于：处于活锁的线程是在不断的改变状态，就是所谓的“活”，而

处于死锁的线程表现为一直等待，活锁有可能自行解开，死锁则不能。

死锁与饥饿的区别在于，死锁除非外力无法解开，饥饿能够被解开，比如当其他高优先级的进程都终止时并且没有更高优先级的进程到达。

--------扩展(无需背诵)：

死锁、活锁、饥饿基本概念：

1.死锁：是指两个或两个以上的线程（或进程）在执行过程中，因争夺锁资源而造成

的一种互相等待的现象（卡住了），若无外力作用，它们都将无法推进下去。

2.活锁：任务或者执行者没有被阻塞，由于某些条件没有满足，导致一直重复尝试，

失败，尝试，失败。

3.饥饿：一个或者多个线程因为种种原因无法获得所需要的资源，导致一直无法执

行的状态。比如cpu一直给你分到调度。或一直wait，

产生死锁的必要条件：

1.互斥条件：所谓互斥就是进程在某一时间内独占资源。

2.请求与保持条件：一个进程因请求资源而阻塞时，对已获得的资源保持不放。

3.不剥夺条件：进程已获得资源，在末使用完之前，不能强行剥夺。

4.循环等待条件：若干进程之间形成一种头尾相接的循环等待资源关系。

Java中导致饥饿的原因：

1.高优先级线程吞噬所有的低优先级线程的 CPU 时间。

2.线程被永久堵塞在一个等待进入同步块的状态，因为其他线程总是能在它之前

持续地对该同步块进行访问。

3.线程在等待一个本身也处于永久等待完成的对象(比如调用这个对象的 wait 方

法)，因为其他线程总是被持续地获得唤醒。

### 面试题3.58 java多线程有几种实现方式?你推荐哪一种

**【技术难度：2 出现频率：2 】**

有四种，分别是：

1.继承Thread类，重写run()方法；

2.实现Runnable接口，重写run()方法，这种最常用；

3.实现Callable接口，重写call()方法；

4.线程池。

推荐使用线程池 因为线程池中的线程可以循环使用。

### 面试题3.59 线程池的优点？

**【技术难度： 2 出现频率：2 】**

1.可重复使用已有线程；

2.可有效控制最大并发线程数；

3.提供定时执行、定期执行、单线程、并发数控制等多种功能。

### 面试题3.60 什么是线程池？有哪几种创建方式？四种线程池的创建方式?

**【技术难度： 2 出现频率：2 】**

1

线程池用来管理多个线程，可以先创建好若干线程，使用时直接获取，以空间换时间的方式，提高线程的使用效率。

2、3

线程池创建方式：

1. Executors.newCachedThreadPool()：创建可根据需要创建新线程的线程池，不够创建，不用了回收，无限大；
2. Executors.newFixedThreadPool(n); 创建可重用固定线程数的线程池，如果超过了需要等待；
3. Executors.newScheduledThreadPool(n)：创建可周期定长线程池，它可延迟运行或周期执行；
4. Executors.newSingleThreadExecutor() ：创建只有一个线程的线程池。

或者创建自定义线程池，根据需求指定线程池参数。

### 面试题3.61 HashMap如何实现线程安全? ConcurrentHashMap和Hashtable的区别?

**【技术难度： 2 出现频率： 1 】**

第一层：

1. 使用ConcurrentHashMap，它只锁住要修改的部分；
2. 使用Collections类的synchronizedMap()方法包装一下，这种方式获得的线程安全的HashMap在读写数据的时候会对整个容器上锁，效率低下；
3. Hashtable读写数据的时候会对整个容器上锁，效率低下。

推荐使用ConcurrentHashMap。

第二层：

ConcurrentHashMap在jdk1.7之前使用分段锁，将数据分段加不同对象锁，jdk1.8开始它用数组中每个头节点作为锁对象来使用synchronized锁，并使用CAS操作来进一步提高效率

----------------扩展(无需背诵)：

代码：

Map<Long，Object> map2 = new ConcurrentHashMap<>();

Map<Long，Object> map1 = Collections.synchronizedMap(new HashMap<Long，Object>());

Map<Long, String> map = new Hashtable<>();

参考网址：

https：//my.oschina.net/pingpangkuangmo/blog/817973#h2\_12

### 面试题3.62 JAVA中Object类中有哪些常用方法？

**【技术难度：2 出现频率：1 】**

第一层：

1.getClass()获取类的class对象；

2.hashCode()获取对象的hashCode值，子类可重写以自定义；

3.equals()比较对象是否相等，默认用==比较两个对象的内存地址，子类可重写以自定义；

4.clone()克隆方法，子类可重写以自定义，深克隆需要手动深度克隆对象中的目标内容；

5.toString()返回对象16进制的内存地址的哈希值以及类名等信息，子类可重写以自定义；

6.notify()随机选择一个在该对象上调用wait方法的线程，解除其阻塞状态；

7.notifyAll()解除所有在该对象上调用wait方法的线程的阻塞状态；

8.wait()导致当前线程进入等待状态，直到他被其他线程通过notify()或者notifyAll()唤醒；

9.finalize()对象被GC回收时调用，只调用一次。

第二层：

notify()、notifyAll()、wait()方法只能在同步方法或同步块内部调用。如果当前线程不是锁的持有者，该方法抛出一个IllegalMonitorStateException异常。

### 面试题3.63 什么是深拷贝和浅拷贝

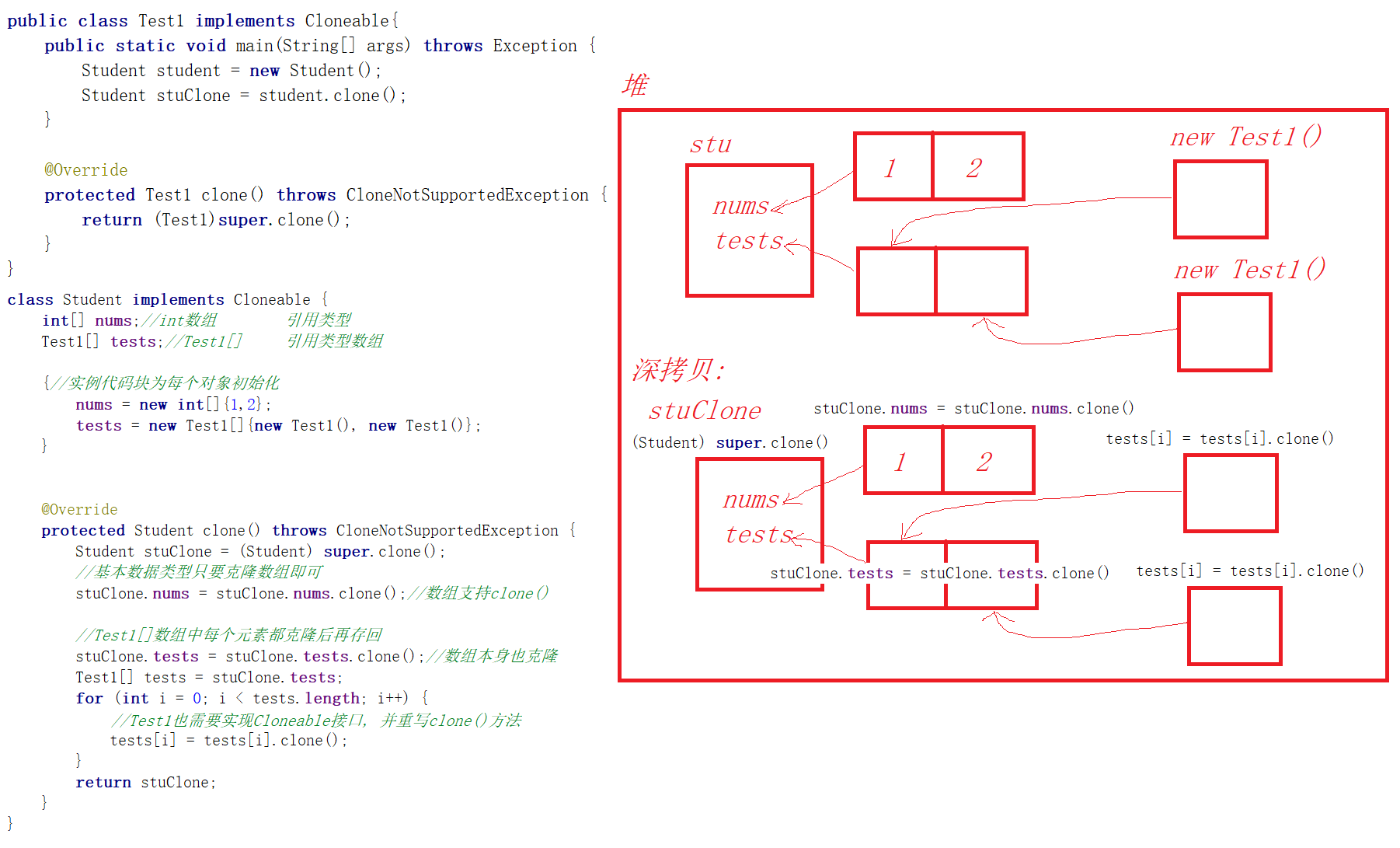
**【技术难度： 2 出现频率：1 】**

1.深拷贝和浅拷贝是针对对象和数组这样的引用类型，包装类和String类对象的值是不可变的，无需拷贝；

2.浅拷贝只克隆当前对象本身，而不克隆它成员变量指向的对象，新旧对象的成员变量还是指向堆中相同数据；

3.深拷贝会深度克隆对象中所有内容，修改新对象不会影响原对象。

----------------扩展(无需背诵)：



### 面试题3.64 GC是什么? 为什么要有GC?

**【技术难度：3 出现频率：2 】**

GC是垃圾收集的意思，Java提供的GC功能可以自动监测对象是否超过作用域从而达到自动回收内存的目的，Java语言没有提供释放已分配内存的显式操作方法。

----------------扩展(无需背诵)：

GC - Gabage Collection 垃圾收集。

内存处理是编程人员容易出现问题的地方，忘记或者错误的内存回收会导致程序或系统的不稳定甚至崩溃。

### 面试题3.65 JVM垃圾回收机制和常见算法

**【技术难度：3 出现频率：2 】**

第一层：

理论上来讲Sun公司只定义了垃圾回收机制规则而不局限于其实现算法，因此不同厂商生产的虚拟机采用的算法也不尽相同。GC在回收对象前首先必须发现那些无用的对象，此时会通过搜索算法去发现定位这些无用的对象。搜索到无用对象之后就是回收过程，再通过回收算法进行回收。

垃圾算法的实现涉及大量的程序细节，而且不同的虚拟机平台实现的方法也各不相同。

第二层：

常用的**搜索算法**有这些：

1）引用计数器算法（废弃）

引用计数器算法是给每个对象设置一个计数器，当有地方引用这个对象的时候，计数器+1，当引用失效的时候，计数器-1，当计数器为 0 的时候，JVM 就认为对象不再被使用，是“垃圾”了。引用计数器实现简单，效率高；但是不能解决循环引用问问题（A 对象引用 B 对象，B 对象又引用 A 对象，但是A，B对象已不被任何其他对象引用），同时每次计数器的增加和减少都带来了很多额外的开销，所以在 JDK1.1 之后，这个算法已经不再使用了。

2）根搜索算法（使用中）

根搜索算法是通过一些“GC Roots”对象作为起点，从这些节点开始往下搜索，搜索通过的路径成为引用链（Reference Chain），当一个对象没有被 GC Roots 的引用链连接的时候，说明这个对象是不可用的。

GC Roots 对象包括：

a) 虚拟机栈（栈帧中的本地变量表）中的引用的对象。

b) 方法区域中的类静态属性引用的对象。

c) 方法区域中常量引用的对象。

d) 本地方法栈中 JNI（Native 方法）的引用的对象。

第三层：

常用的**回收算法**如下：

1）标记—清除算法（Mark-Sweep）（DVM 使用的算法）

标记—清除算法包括两个阶段：“标记”和“清除”。在标记阶段，确定所有要回收的对象，并做标记。清除阶段紧随标记阶段，将标记阶段确定不可用的对象清除。标记—清除算法是基础的收集算法，标记和清除阶段的效率不高，而且清除后回产生大量的不连续空间，这样当程序需要分配大内存对象时，可能无法找到足够的连续空间。

2）复制算法（Copying）

复制算法是把内存分成大小相等的两块，每次使用其中一块，当垃圾回收的时候，把存活的对象复制到另一块上，然后把这块内存整个清理掉。复制算法实现简单，运行效率高，但是由于每次只能使用其中的一半，造成内存的利用率不高。现在的 JVM 用复制方法收集新生代，由于新生代中大部分对象（98%）都是朝生夕死的，所以两块内存的比例不是 1：1(大概是 8：1)。

3）标记—整理算法（Mark-Compact）

标记—整理算法和标记—清除算法一样，但是标记—整理算法不是把存活对象复制到另一块内存，而是把存活对象往内存的一端移动，然后直接回收边界以外的内存。标记—整理算法提高了内存的利用率，并且它适合在收集对象存活时间较长的老年代。

4）分代收集（Generational Collection）

分代收集是根据对象的存活时间把内存分为新生代和老年代，根据各个代对象的存活特点，每个代采用不同的垃圾回收算法。新生代采用复制算法，老年代采用标记—整理算法。

### 面试题3.66 简单的介绍⼀下强引用，软引用，弱引用，虚引用

**【技术难度： 2 出现频率： 1 】**

除了强引用，软、弱、虚引用需要配合对应的类使用，他们的区别在于被GC回收的时机不同。



----------------扩展(无需背诵)：

强引用 User user = new User();

软引用 SoftReference<User> user = new SoftReference<>( new User());

弱引用 WeakReference<User> user = new WeakReference<>(new User());

虚引用PhantomReference<User> pr = new PhantomReference<User>(new User()，ReferenceQueue);

### 面试题3.67 什么是java内存泄漏，什么是内存溢出，怎么预防？

**【技术难度： 3 出现频率：1 】**

第一层：

内存泄露是指无用对象持续占用内存没有释放，仿佛泄漏掉了一部分内存，这就称为内存泄漏。内存溢出是指申请内存时，没有足够的内存可以使用。内存泄露的增多，最终会导致内存溢出。

预防这些问题一方面是从代码层面进行优化完善，另一方面可以优化JVM的配置。

第二层：

从代码层面进行优化完善的思路有很多，比如以下这些情况：

1.尽早释放无用对象的引用，好的办法是使用临时变量的时候，让引用变量在退出活动域后自动设置为null；

2.进行字符串处理时，尽量避免使用String，而应使用StringBuffer或StringBuilder；

3.尽量少用静态变量，因为静态变量是全局的，存在方法区，GC不会回收；

4.避免集中创建对象，尤其是大对象，应尽量运用对象池技术、IO流操作以提高系统性能，否则JVM会突然需要大量内存，这时会触发GC优化系统内存环境；

5.生命周期长的对象中引用生命周期短的对象时容易引发内存泄漏，比如大集合对象拥有大数据量的业务对象的时候，可以考虑分块进行处理然后解决一块释放一块的策略；

6.不要在经常调用的方法中创建对象，尤其忌讳在循环中创建对象，可以适当的使用hashtable，vector创建一组对象容器，然后从容器中去取这些对象，而不用每次new之后又丢弃；

7.各种连接不用了应及时调用close()关闭，JDBC、IO、Socket等；

8.当我们把自己定义的类保存到散列表的时候，要保证对象的hashCode不可变，否则散列表的remove()方法无法根据hashCode删除正确对象。

优化JVM配置可以设置很多参数，比如这些：

1.设置堆内存大小，参数有-Xms、-Xmx等；

2.设置堆中新生代的内存大小，-XX:newSize、-XX:MaxNewSize等；

3.设置方法区内存大小，参数有-XX:PermSize、-XX:MaxPermSize等。

----------------扩展(无需背诵)：

memory leak - 内存泄露

out of memory - 内存溢出

堆内存大小：

-Xms表示堆内存初始内存空间

-Xmx表示堆内存最大内存空间

新生代内存大小：

-XX:newSize表示新生代初始内存空间（应小于-Xms的值）

-XX:MaxnewSize表示新生代最大内存空间（这个值应该小于-Xmx的值）

方法区内存大小：

-XX:PermSize表示方法区初始内存空间

-XX:MaxPermSize表示对非方法最大内存空间

### 面试题3.68 什么是java反射机制？哪些地方用到了？

**【技术难度： 2 出现频率：2 】**

1. 反射允许java在运行时动态的加载任意的类，并获取任意类的属性，调用任意类的方法，而不用在写代码时写死具体的类。这使得java编码变得更加灵活；
2. 比如使用JDBC连接数据库时，使用Class.forName()通过反射加载数据库的驱动程序。比如说Spring框架对象的创建，都是通过反射来完成，我们在配置文件中，定义类的依赖关系，并制定类的全路径，就是为反射创建对象提供信息。

### 面试题3.69 获取Class对象有几种方式?

**【技术难度： 1 出现频率：1 】**

1.Class.forName(完整类名)；

2.类名.class；

3.对象名.getClass()。

### 面试题3.70 Java创建对象有几种方式

**【技术难度： 1 出现频率： 1 】**

调用构造方法的有：

1.new关键字

2.反射

不调用构造方法的有：

3.克隆方法clone()

4.反序列化

### 面试题3.71 知道类的加载过程吗

**【技术难度： 3 出现频率：2 】**

第一层：

当程序要使用某个类时，如果该类还未被加载到内存中，则系统会通过加载、连接、初始化三步来实现对这个类进行初始化。

第二层：

加载、连接、初始化简单说是这样：

1. 加载是将class文件读入内存获取它的二进制字节流，并为之创建一个Class对象；

2. 连接再细分为验证、准备、解析三步；

3. 初始化是在类首次被“主动使用”时执行初始化，为类变量赋予正确的初始值。

第三层：

1.任何类被使用时JVM都会建立并且只建立一个Class对象；

2.连接再细分为验证、准备、解析三步详细说是这样：

先验证字节码文件的字节流是否符合当前虚拟机的要求，接着准备是为类变量分配内存并设置默认值，之后解析是将字节码文件中常量池的符号引用转换为直接引用的过程（主要针对类或接口、字段、类方法、方法类型等）。

### 面试题3.72 知道哪些类加载器

**【技术难度： 3 出现频率： 2 】**

第一层：

JVM 中内置了三个重要的类加载器，除了启动类加载器其他类加载器均由Java实现且全部继承自java.lang.ClassLoader。

第二层：

接下来详细说说这些类加载器：

1.启动类加载器是最顶层的加载类，由C++实现，负责加载 %JAVA\_HOME%/lib ⽬录下的jar包和类，或是被 -Xbootclasspath 参数指定的路径中的所有类；

2.扩展类加载器主要负责加载⽬录 %JRE\_HOME%/lib/ext ⽬录下的jar包和类，或被java.ext.dirs系统变量所指定的路径下的jar包；

3.应⽤程序类加载器是⾯向我们⽤户的加载器，负责加载当前应⽤classpath下的所有jar。

----------------扩展(无需背诵)：

ClassLoader - 类加载器

BootstrapClassLoader - 启动类加载器

ExtensionClassLoader - 扩展类加载器

AppClassLoader - 应⽤程序类加载器

### 面试题3.73 什么是双亲委派模型

**【技术难度： 3 出现频率：2 】**

双亲委派模型能够保证多个加载器加载某个类时，最终都是由一个加载器加载，确保最终加载结果相同。

原理是这样：

1.当需要加载一个类的时候，子类加载器并不会马上去加载，而是依次去请求父加载器加载，一直往上找，直到最高的类加载器——启动类加载器；

2.当启动类加载器还加载不了的时候，才依次往下让子类加载器进行加载；

3.当到达最底层的时候，如果还是加载不到该类，就会出现ClassNotFound的情况。

### 面试题3.74 BIO，NIO，AIO 有什么区别?

**【技术难度：3 出现频率：1 】**

第一层：

1.BIO是一个连接一个线程；

2.NIO是一个请求一个线程；

3.AIO是一个有效请求一个线程。

第二层：

BIO (Blocking I/O)：

BIO是同步阻塞的，实现模式为一个连接一个线程，即当有客户端连接时，服务器端需为其单独分配一个线程，如果该连接不做任何操作就会造成不必要的线程开销(当然可以通过线程池机制改善)。

BIO是传统的Java io编程，其相关的类和接口在java.io 包下。

优点是编程简单，缺点是无法处理⼗万甚⾄百万级连接，仅适用于活动连接数于⼩于单机 1000的业务场景。

NIO (Non-blocking或New I/O)：

NIO是同步非阻塞的，服务器实现模式为一个请求一个线程，即客户端发送的连接请求都会注册到多路复用器上，多路复用器轮询到连接有I/O请求时才启动一个线程进行处理。

NIO提供了与传统 BIO 模型中的 Socket 和 ServerSocket 相对应的 SocketChannel 和ServerSocketChannel 两种不同的套接字通道实现，两种通道都⽀持阻塞和⾮阻塞两种模式。阻塞模式比较简单，但是性能和可靠性都不好；⾮阻塞模式正好与之相反。对于低负载、低并发的应⽤程序，可以使⽤同步阻塞 I/O 来提升开发速率和更好的维护性；对于⾼负载、⾼并发的（⽹络）应⽤，应使⽤ NIO 的⾮阻塞模式来开发

AIO (Asynchronous I/O)：

AIO是异步非阻塞的，服务器实现模式为一个有效请求一个线程，客户端的I/O请求都是由操作系统先完成，之后再通知服务器应用启动线程进行处理。

AIO 也就是 NIO 2。在 Java 7 中引⼊了 NIO 的改进版 NIO 2，它是异步⾮阻塞的 IO 模型。异步 IO 是基于事件和回调机制实现的，应⽤操作之后会直接返回，不会堵塞，当后台处理完成，操作系统会通知相应的线程进⾏后续的操作。AIO 是异步 IO 的缩写，虽然 NIO 在⽹络操作中，提供了⾮阻塞的⽅法，但是 NIO 的 IO ⾏为还是同步的。对于 NIO 来说，业务线程是在 IO 操作准备好时，得到通知，接着就由这个线程⾃⾏进⾏ IO 操作，IO 操作本身是同步的。⽬前来说 AIO 的应⽤还不是很⼴泛，Netty 框架(简化网络编程流程)之前也尝试使⽤过 AIO，不过在后来又放弃了。

### 面试题3.75 java8新特性

**【技术难度：1 出现频率：1 】**

接口中可以添加default默认方法、Lambda表达式、函数式接口、Stream数据流、Optional类以及全新的时间包等。

### 面试题3.76 请说说你知道的设计模式？

**【技术难度：2 出现频率：1 】**

第一层：

比如Spring中使用到的设计模式，

先说说单例、原型、代理模式：

1. 单例模式，在Spring中最明显的使用场景是在配置文件中配置注册bean对象的时候设置scope的值为singleton，让目标类只创建一个实例,目的是节省内存；
2. 原型模式，也叫克隆模式，Spring中该模式使用的很明显，和单例一样在bean标签中设置scope的属性为prototype即表示该bean以克隆的方式生成，基于clone()方法需要注意深拷贝浅拷贝问题；
3. 代理模式，比如在Spring的AOP面向切面编程就有用到，有两种代理方案，一是基于接口的JDK动态代理，二是基于字节码文件生成子类来代理的CGLIB，具体是这样：
4. 如果目标对象实现了接口，默认情况下会采用JDK的动态代理实现AOP；
5. 如果目标对象实现了接口，可以强制使用CGLIB实现AOP；
6. 如果目标对象没有实现接口，必须采用CGLIB，Spring会自动在JDK动态代理和CGLIB之间转换。

第二层：

然后再看模板、观察者、工厂：

1. 模板模式，核心是父类定义好流程，然后将流程中需要子类实现的方法就抽象话留给子类实现，Spring中的JdbcTemplate（JDBC模板）就是这样的实现。我们知道jdbc的步骤是固定的(加载驱动，获取连接通道，构建sql语句，执行sql语句，关闭资源)，在这些步骤中第3步和第四步是不确定的，所以就留给客户实现，而我们实际使用JdbcTemplate的时候也确实是只需要构建SQL就可以了.这就是典型的模板模式。
2. 观察者模式，它定义的是对象间的一种一对多的依赖关系，当一个对象的状态发生改变时，所有依赖于它的对象都得到通知并被自动更新。使用比较场景是在监听器中而spring中Observer模式常用的地方也是listener的实现。如ApplicationListener。
3. 工厂模式分两种，

简单工厂模式就是通过工厂根据传递进来的参数决定产生哪个对象。Spring中我们通过getBean方法获取对象的时候根据id或者name获取就是简单工厂模式了。

工厂方法模式，在Spring中我们一般是将Bean的实例化直接交给容器去管理的，实现了使用和创建的分离，这时容器直接管理对象，还有种情况是，bean的创建过程我们交给一个工厂去实现，而Spring容器管理这个工厂。这个就是我们讲的工厂模式，在Spring中有两种实现，一种是静态工厂方法模式，一种是动态工厂方法模式。

第三层：

以及适配器、装饰者、策略模式是这样：

1. 适配器模式是将一个类的接口转换成客户希望的另外一个接口。使得原本由于接口不兼容而不能一起工作的那些类可以在一起工作。这就是适配器模式。在Spring中在AOP实现中的Advice和interceptor之间的转换就是通过适配器模式实现的。
2. 装饰者模式又称为包装模式(Wrapper),作用是用来动态的为一个对象增加新的功能。装饰模式是一种用于代替继承的技术，无须通过继承增加子类就能扩展对象的新功能。使用对象的关联关系代替继承关系，更加灵活，同时避免类型体系的快速膨胀。

spring中用到的包装器模式在类名上有两种表现：一种是类名中含有Wrapper，另一种是类名中含有Decorator。基本上都是动态地给一个对象添加一些额外的职责。

具体的使用在Spring session框架中的SessionRepositoryRequestWrapper使用包装模式对原生的request的功能进行增强，可以将session中的数据和分布式数据库进行同步，这样即使当前tomcat崩溃，session中的数据也不会丢失。

1. 策略模式对应于解决某一个问题的一个算法族，允许用户从该算法族中任选一个算法解决某一问题，同时可以方便的更换算法或者增加新的算法。并且由客户端决定调用哪个算法，spring中在实例化对象的时候用到Strategy策略模式。

----------------扩展(无需背诵)：

参考源：

Spring中使用到的设计模式：https://www.cnblogs.com/dengpengbo/p/10485924.html

动静态工厂方法模式：https://blog.csdn.net/weixin\_43667254/article/details/105248003

### 面试题3.77 TCP和UDP的区别

**【技术难度：1 出现频率：2 】**

TCP、UDP是通信协议，是TCP/IP协议（Transmission Control Protocol传输控制协议）的核心，属于其中的传输层，它们的区别是这样：

1. TCP需要三次握手，面向连接，可靠机制保证数据不丢失且有序，效率低、占用资源多；

2. UDP不可靠，无连接，适用于实时性较要求高、安全性较要求低的地方，效率高、占用资源少。

----------------扩展(无需背诵)：



### 面试题3.78 TCP三次握手和四次挥手

**【技术难度：2 出现频率：1 】**

第一层：

TCP协议是为了在不可靠的互联网络上，提供可靠的端到端字节流传输，而专门设计的一个传输协议。

它需要经过三次握手和四次挥手来建立连接和断开连接，这样可以确保连接的安全性，避免一些错误情况。

第二层：

三次握手的目的，是为了防止A端已经失效的连接请求突然又传到B端时，被误认为是A端再次发出的一个新的连接请求，如果B端这时又再次向A发出确认报文，表示同意建立连接，就会产生错误。

四次挥手的原因是tcp是全双公的，通信是双向的， A到B是一个通道，B到A又是另一个通道，上下行数据的两个通道各需两次握手才能安全断开。

第三层：

三次握手流程简单说是这样：

1. 第一次是A端向B端发送请求，如果是只有一次握手的话，A端不知道B端是不是收到了这个请求；
2. 第二次是B端确认收到A端请求，如果只有两次的话，B端不确定A端是否收到了确认消息，这个确认消息有可能会在半路丢了；
3. 第三次是A端确认收到了B端的确认消息，A和B双方都是通的，然后AB就可以建立连接相互通信了。

四次挥手流程简单说是这样：

1. A端确认没有数据发送后，发出结束报文，此时B端返回确认后，B端也不会接收A端数据；
2. 但是此时B端可能还有数据没有传输完，A端还是可以接收数据；
3. 只有当B端数据发送完之后，才能发出结束报文，并且确认A端接收到的时候，两边才会真正的断开连接，双方的读写分开。

### 面试题3.79 Socket是什么？

**【技术难度：1 出现频率：1 】**

Socket套接字是对网络中进行双向通信的端点的抽象，JAVA API中提供了Socket的实现类，支持TCP协议的Socket分为服务端、客户端Socket，支持UDP协议的Socket既可以是客户端也可以是服务端。

## 模块4：Java 常见算法

### 面试题4.1 冒泡排序如何实现？优缺点是什么？有没有想过怎么优化？

**【技术难度： 出现频率： 】**

1. 实现思路，使用双循环来进行排序。外部循环控制所有的回合，内部循环代表每一轮的冒泡处理，先进行元素比较，再进行元素交换；
2. 冒泡排序优点是比较简单，空间复杂度较低，是稳定的。缺点是时间复杂度最坏为O(n^2)，太高，对n个数排序最多需要扫描n \* n次，效率慢；
3. 优化思路：

第一步优化，可以利用布尔变量isSorted作为标记。如果在本轮排序中，元素有交换，则说明数列无序；如果没有元素交换，说明数列已然有序，直接跳出大循环。

第二步优化，可以在每一轮排序的最后，记录最后一次元素交换的位置，那个位置也就是无序数列的边界，再往后就是有序区了，这个位置可以作为内循环的结束条件，因为有序区无需再比较。

----------------扩展(无需背诵)：

参考源：https://blog.csdn.net/wubingju93123/article/details/81215984

### 面试题4.2 选择排序如何实现？优缺点？

**【技术难度： 出现频率： 】**

1. 首先在未排序序列中找到最小（大）元素，存放到排序序列的起始位置，然后，再从剩余未排序元素中继续寻找最小（大）元素，然后放到已排序序列的末尾。以此类推，直到所有元素均排序完毕；
2. 优点是比较简单，空间复杂度较低。缺点是不稳定，时间复杂度都为O(n^2)，对什么数都需要扫描n×n次，效率慢。

### 面试题4.3 插入排序如何实现？优缺点？

**【技术难度： 出现频率： 】**

原理是通过构建有序序列，对于未排序数据，在已排序序列中从后向前扫描，找到相应位置并插入。插入排序在实现上，在从后向前扫描过程中，需要反复把已排序元素逐步向后挪位，为最新元素提供插入空间。

优点是比较简单，空间复杂度较低，是稳定的。缺点是时间复杂度最坏为O(n^2)，太高，对n个数排序最多需要扫描n \* n次，效率慢。

### 面试题4.4 快速排序如何实现？优缺点？

**【技术难度： 出现频率： 】**

快排的基本思路为先确定出一个基准数（取第一个数为基准数），然后通过与基准数循环比较的方法，将所有比基准数小的数字放在它的左边，所有比它大的数字放在右边，实现一部分数字都比另一部分小。然后再通过此方法对这两部分数据分别进行相同的操作，整个过程可以通过递归实现，对所有数字进行排序。

优点是目前已知最快的排序方法，缺点是代码较复杂，不稳定。

----------------扩展(无需背诵)：

参考资料：

https://blog.csdn.net/weixin\_42437295/article/details/90771962

### 面试题4.5 二分查找

**【技术难度： 出现频率： 】**

首先要有一个有序的列表。如果没有，可以先使用排序算法进行排序。

以升序数列为例，先比较目标元素与数列中的中间位置的元素的大小，如果比中间位置的元素大，则继续在后半部分的数列中进行二分查找；如果比中间位置的元素小，则在数列的前半部分进行比较；如果相等，则找到了元素的位置。循环进行这样的比较，每次比较的数列长度都会是之前数列的一半，直到找到相等元素的位置或者最终没有找到要找的元素。

它的时间复杂度为O(log2n)，比如在8个数中查找，最多需要查找3次（log2 8 = 3），最好的情况是只查找一次就能找到，但是在最坏和一般情况下也要比按顺序查找好很多。

### 面试题4.6 什么是平衡二叉树? 怎么判断二叉树是否有环？

**【技术难度：1 出现频率：1 】**

1. 平衡二叉树是一棵空树或它的左右两个子树的高度差的绝对值不超过1，并且左右两个子树都是一棵平衡二叉树；
2. 在遍历二叉树时，能循环到起点指针称为有环。

## 模块5：Oracle / MySQL 数据库

### --------基础常识类

### 面试题5.1 Oracle和Mysql的区别？

**【技术难度： 出现频率： 】**

第一层：

1. Oracle用表空间来管理表，Mysql用库来管理表
2. mysql用limt做分页、支持主键自增长，oracle用rownum做分页、通过序列达到自增效果
3. 库函数不同、SQL语法、功能语句不同

第二层：

第3点展开说是这样：

库函数不同，比如：滤空函数在Oracle中是NVL(列名，覆盖值)，而MySQL中是IFNULL(列名，覆盖值)

SQL语法不同，比如：Oracle中没有if exists，if not exists的语法，MySQL有

功能语句不同，比如：显示当前所有的表、用户、改变连接用户、显示当前连接用户、执行外部脚本等语句不同。

### 面试题5.2 SQL语句有多少类型

**【技术难度： 出现频率： 】**

第一层：

SQL语句分五类：DDL数据定义语言、DML数据操控语言、DQL数据操控语言、DCL数据库控制语言、TCL事务控制语言。

第二层：

它们分别包括以下内容：

1. DDL（Data Definition Language）数据定义语言，用于操作表结构等，包括create创建、drop删除、alter修改、truncate截断；
2. DML（Data Manipulation Language）数据操控语言，用于操作表中数据，包括insert插入、delete删除、update更新；
3. DQL（Data Query Language）数据查询语言，用于查询，包括select查询；
4. DCL（Data Control Language）数据库控制语言，用于设定用户权限，包括grant授权、revoke撤销；
5. TCL（Transaction Control Language）事务控制语言，用于控制事务，包括commit提交、rollback回滚。

第三层：

再详细说每个语的句功能的话是这样：

DDL（Data Definition Language）数据定义语言：

1. Create语句，可以创建数据库和数据库的一些对象；
2. Drop语句，可以删除数据表、索引、触发程序、条件约束以及数据表的权限等；
3. Alter语句，修改数据表定义及属性；
4. Truncate语句，删除表中的所有记录,包括所有空间分配的记录被删除。

DML（Data Manipulation Language）数据操控语言：

1. Insert语句，向数据表张插入一条记录；
2. Delete语句，删除数据表中的一条或多条记录，也可以删除数据表中的所有记录，但是它的操作对象仍是记录；
3. Update语句，用于修改已存在表中的记录的内容。

DQL（Data Query Language）数据操控语言：

1. Select语句，用于查询已存在表中的记录的内容。

DCL（Data Control Language）数据库控制语言：

1. Grant语句，允许对象的创建者给某用户或某组或所有用户（PUBLIC）某些特定的权限；
2. Revoke语句，可以废除某用户或某组或所有用户访问权限。

TCL（Transaction Control Language）事务控制语言：

1. commit，提交事务；
2. rollback，回滚事务。

### 面试题5.3 mysql常用数据类型、oracle常用数据类型

**【技术难度： 出现频率： 】**

**第一层：**

1. mysql常用数据类型

整数: tinyint极小整数，smallint小整数，mediumint中等整数，int整数，bigint大整数

浮点数: float单精浮点，double双精浮点，decimal定点数

字符: char定长字符串，varchar可变长字符串，set集合

时间: year年份，date年月日，time时分秒， datetime年月日时分秒，timestamp时间戳

1. Oracle常用数据类型

数字类型有：number数字

字符串类型有：char定长字符串、varchar2可变长字符串

日期类型有：date日期和时间、timestamp时间戳

第二层：

1. MySQL中的decimal定点数类型常用于存储金额数据（对应Java的BigDecimal类）；
2. MySQL和Oracle的timestamp类型都默认精确到小数小数点后6位；
3. 从MySQL5.6.4开始，MySQL的datetime类型也支持小数了，这比timestamp要好得多，因为MySQL的timestamp需要转换成UTC格式存储，使用、查询时再转换回来，而且只能存储UTC格式的2038年以前的数据。

### 面试题5.4 char、varchar2、varchar有什么区别？

**【技术难度： 出现频率： 】**

第一层：

1.char定长，varchar可变长，确定大小的使用char效率更高，不确定大小的字段使用varchar；

2.varchar是标准sql里面的，varchar2是oracle独有的，oracle中建议使用varchar2；

3.varchar对空字符串不处理，varchar2将空字符串当做null来处理。

第二层：

varchar类型需要额外的1到2字节存储内容的长度信息，因此在内容充实的情况下比char效率低。

### 面试题5.5 什么是BLOB、CLOB？MySQL的Text类型了解过吗

**【技术难度： 出现频率： 】**

第一层：

Oracle中的 BLOB（Binary Large Object二进制大对象）可以存4GB以内的二进制数据、CLOB（Character Large Object字符大对象）存4GB以内的字符数据。

MySQL 对应 Oracle 数据库 BLOB 的类型有几个，比如LONGBLOB、BLOB等，都用于存储二进制数据，而对应 Oracle 数据库 CLOB 也有几个类型，以TEXT后缀结尾，用于存储大量字符数据，MySQL的类型划分更加精细，划分了不同存储大小。

第二层：

对于需要高频度读取并展现的二进制数据或大量字符数据，可以放入数据库；否则不放入数据库，而是放入磁盘，数据库里只存放其文件路径，需要展现时依据路径读取并展现即可。但有个例外，对于内容可能很大，会对数据库造成重大性能瓶颈的字段，坚决放入磁盘，不要提供高频度读取展现的支持，避免因小失大。

### 面试题5.6 null的含义

**【技术难度： 出现频率： 】**

第一层：

在我们不知道具体有什么数据的时候，可以用NULL，我们称它为空。NULL等价于没有任何值、是未知数。

第二层：

Oracle、MySQL 允许任何一种数据类型的字段为空，除了以下两种情况：

1. 主键字段（primary key）；
2. 定义时已经加了NOT NULL非空约束的字段。

第三层：

使用null时需要注意的细节是这样：

1. null与0、空字符串、空格都不同、也不能比较大小，应使用关键字用“is null”和“is not null”进行判断，mysql还有个ISNULL()函数（速度比关键字低一点）；
2. 空值不能被索引，所以查询时有些符合条件的数据可能查不出来，比如count(\*)中，oracle可以用nvl(列名,0)、mysql可以用ifnull(列名,0)滤空处理后再查；
3. 对空值做加、减、乘、除等运算操作，结果仍为空，也可以像上一条将null过滤后再计算；
4. 需要建立索引的列，最好设为not null或通过default设置一个默认值；
5. ASC顺序排序时，Oracle中null在其他数据后面，MySQL中null在其他数据前，DESC逆序则反过来，这只是一种约定或者说设定。

### 面试题5.7 数据库三范式是什么?

**【技术难度： 出现频率： 】**

第一层：

1NF第一范式：确保每列的原子性，即表中每个列都不能再被拆分成若干列；

2NF第二范式：主键要保证每条数据的唯一性，并且表中的每列都和主键列直接相关；

3NF第三范式：确保表中不包含其它表的非主键列。

第二层：

三范式减少了数据冗余（避免在多张表中存储相同的非主键列以节省空间），提高了数据完整性（更精确、可靠）。

三范式不要求必须遵循，数据量体量特别大的地方不要求满足第三范式，以提高查询的效率。

第三层：

范式化的优点是数据库更新起来更快、可以只修改更少的数据等，缺点是查询容易产生较多关联查询，且不能建立复合索引，所以查询性能会大大降低。

所以在设计数据库时，不一定强制要求使用标准范式化设计，可以进行一定的调整。一般的设计是混用范式化和反范式化，适度进行数据冗余，以提高查询性能。

### 面试题5.8 主键和外键的区别？

**【技术难度： 出现频率： 】**

1. 主键在本表中是唯一的、不可为空的，外键可以重复可以为空；

2. 外键和另一张表的主键关联，不能创建对应表中不存在的外键。

### 面试题5.9 如何设置主键自动增长？

**【技术难度： 出现频率： 】**

1. MySQL可以设置主键自增长，要求主键必须是数值类型，且一张表只能有一个自增主键列；
2. Oracle需要通过序列实现自增效果，方案一是在插入语句中调用序列，方案二是创建一个insert 插入语句触发器，在执行插入语句时获取序列的值将其赋值给主键列；

----------------扩展(无需背诵)：

Oracle方案2，触发器调用序列代码案例：

https://blog.csdn.net/weixin\_44963125/article/details/119889772

### 面试题5.10 Oracle中序列的作用

**【技术难度： 出现频率： 】**

第一层：

Oracle使用序列来生成唯一编号，用来处理一个表中自增字段，访问一个序列的nextval下个值，该序列将按照设定的递增量递增并返回新值。

第二层：

第一次调用序列的nextval获取的是初始值。

调用序列的currval可以获取当前值（current当前），它不会引发递增，不过需要在第一次调用 nextval 初始化之后才能使用 currval，否则出错。

第三层：

1. Oracle处理好了序列的线程安全问题；
2. 大量语句申请序列时，为了提高性能，可以将序列提前生成 n 个序列值先存入内存，这样可直接到运行最快的内存中得到序列值。cache缓存个数可以自定义，最好不要设置过大，因为在数据库重启时，会清空内存信息，预存在内存中的序列会丢失，当数据库再次启动后，序列从上次内存中最大的序列号+1 开始存入n个，这种情况会导致序列号不连续。另外，事务回滚也会导致序列号不连续。

### 面试题5.11 update语句可以修改结果集中的数据吗？

**【技术难度： 出现频率： 】**

在Oracle中是可以的，在MySQL中不可以。

----------------扩展(无需背诵)：

如下语句：

update (select \* from table1 t1 left join table2 t2 on t1.字段 = t2.字段 ) set 字段名 = 值 where 条件字段 = 条件值

### 面试题5.12 drop、truncate、 delete区别

**【技术难度： 出现频率： 】**

1. delete 删除内容不删表定义, 不释放空间 (在事务操作时，保留记录方便事务回滚)；
2. truncate 删除内容不删表定义，释放空间；
3. drop 删除内容和定义，释放空间。

### 面试题5.13 MySQL、Oracle写出字符存储、字符串转时间

**【技术难度： 出现频率： 】**

MySQL使用str\_to\_date函数，Oracle使用to\_date函数。

----------------扩展(无需背诵)：

Oracle：to\_date(‘字符串时间’，'yyyy-MM-dd HH24:mi:ss')

MySQL：str\_to\_date(‘字符串时间’,'%Y-%c-%d %H:%i:%s');

### 面试题5.14 说说常用的函数，分别介绍

**【技术难度： 出现频率： 】**

Oracle中有这些函数：

1. COUNT() 求指定列的记录个数，忽略null（当count(\*)时不忽略null）
2. AVG() 求指定列的平均值，忽略null
3. SUM() 求指定列的总和，忽略null
4. MAX() 求指定列的最大值，忽略null
5. MIN() 求指定列的最小值，忽略null
6. ROUND(目标值，保留小数位数) 四舍五入式取整或保留指定位数的小数
7. MOD(除数，被除数) 求余，求除数除以被除数余下的数
8. TRUNC(目标值，保留小数位数) 截断式取整或保留指定位数的小数
9. SYSDATE() 返回当前系统时间(DATE类型)
10. TO\_DATE(字符串类型值，指定格式) 将字符串转换为日期

MySQL与上面Oracle前7个函数一致，另外3个有些许不同或扩展：

1. TRUNCATE(目标值，保留小数位数) 截断式取整或保留指定位数的小数
2. SYSDATE() 返回当前系统时间，另外还提供了NOW()获取语句开始执行时的时间
3. STR\_TO\_DATE(字符串类型值，指定格式) 将字符串转换为日期，指定格式与Oracle不同

还有很多...

### --------查询类

### 面试题5.15 查询语句执行顺序

**【技术难度： 出现频率： 】**

书写顺序按照select、from、join on、where、group by、having、order by、limit排列。

### 执行顺序按照from、join on、where、group by、having、select、order by、limit顺序执行。

### 面试题5.16 查询语句怎么去除重复

**【技术难度： 出现频率： 】**

1. 若存在两条完全相同的纪录，使用distinct关键字就可以去重；
2. 若需要在查询结果中去除部分字段重复的记录，需要先根据这些字段进行分组，再从每组中取出一条记录。

### 面试题5.17 order by与group by的区别

**【技术难度： 出现频率： 】**

第一层：

1. order by 是排序查询，默认是asc升序，可以写desc表示降序；

2. group by 是分组查询，根据指定的1个或多个列对结果集进行分组，可以用 having 筛选符合条件的分组。

第二层：

需要注意：

1. oder by 排序，ASC顺序排序时，Oracle中null在其他数据后面，MySQL中null在其他数据前，DESC逆序则反过来，这只是一种约定或者说设定；
2. 使用 group by 的查询语句，在 select 和 from 中间，只能放分组的列和聚合函数。MySQL在5.7版本之前可以在select 后面跟其它的列，这是不合理的。5.7以后默认配置了 sql\_mode=only\_full\_group\_by 开启严格的 group by 模式，需要的话也在老版本可以手动开启此设置。

### 面试题5.18 分页查询怎么做

**【技术难度： 出现频率： 】**

第一层：

1. MySQL用limit做分页；
2. Oracle用rownum做分页。

第二层：

1. MySQL的limit有两个参数，第一个参数是开始的记录下标，从0开始，第二个参数是截取的记录行数；
2. Oracle的rownum从1开始，rownum只能做小于或小于等于的比较。

第三层：

比如每页有10条数据，查第二页，

1. MySQL直接在查询语句最后写limit，然后根据公式 （当前页码-1）\* 每页条数 计算参数1，参数2就是每页条数，根据这个需求计算得出语句是 limit 19, 10;
2. Oracle则需要嵌套多层子查询来进行处理，因为rownum只能做小于或小于等于的比较否则报错，所以让最内层的语句根据附加条件查询出结果集，第二层用别名将内层结果集的rownum固定到该层的结果集中作为普通数据，就可以推算公式进行筛选了，这边的代码较为繁杂就不再赘述，把这个rownum核心点抓住了后面都很好理解。

----------------扩展(无需背诵)：

需求：每页10条，查询第2页的数据

-- MySQL：

-- limit记录行数从0开始

select \* from teacher

order by sal desc

limit 19, 10; -- (当前页码-1)\*每页条数, 每页条数

-- Oracle：

-- rownum从1开始

select \* from

(select t1.\*,rownum rn from

(select \* from teacher order by sal desc) t1

where rownum <= 20 ) -- 当前页码\*每页条数

where rn > 10; -- (当前页码-1)\*每页条数

### 面试题5.19 行转列，列转行

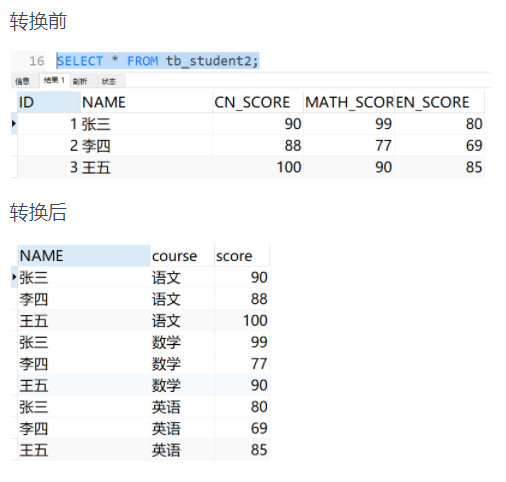
**【技术难度： 出现频率： 】**

1. **行转列**

****

假设根据是根据学生成绩表进行转换，有name姓名、course课程、score成绩几个列，则这样做：

1. 先根据name分组, 每组都是各位同学的数据；
2. MySQL用if筛选各课程成绩，Oracle用decode筛选各课程成绩，再使用max聚合函数只取1条，别名取对应的课程名；
3. 在select和from中间放学生姓名以及第二步转换出的各个课程成绩。
4. **列转行**



假设根据是根据学生成绩表进行转换，有name姓名、cn\_score语文成绩、math\_score数学成绩、en\_score英语成绩几个列，则这样做：

1. 每个课程分别写一条查询语句，查询这些内容，姓名、对应课程名字符串并起个统一的别名比如course、对应课程成绩列；
2. 将各个查询语句通过union或union all合并起来。

----------------扩展(无需背诵)：

-- 行转列MySQL版，Oracle只需将这if()函数换成decode()即可，具体语法不再赘述

select

`name`,

max(if(course='语文', score, null)) 语文,

max(if(course='数学', score, null)) 数学,

max(if(course='英语', score, null)) 英语

from tb\_student group by `name`;

-- 列转行

select `name`, '语文' course, cn\_score score from tb\_student2

union

select `name`, '数学' course, math\_score score from tb\_student2

union

select `name`, '英语' course, en\_score score from tb\_student2;

### 面试题5.20 列举几种表连接方式,有什么区别？

**【技术难度： 出现频率： 】**

第一层：

有内连接、外连接、自连接、交叉连接等。

第二层：

展开说是这样：

1.内连接join是将两个表中满足连接条件的记录连接起来作为结果集, 未满足连接条件的记录会被过滤；

2.外连接有左外连 left join和右外连 right join，以左外连为例，它是以左表为主表, 查出主表所有记录, 从表如果没有关联记录, 对应位置填充null。右外连正好相反。

3.自连接是一张表与自身连接, 将这张表当多张表用。

4.交叉连接是未加连接条件, 产生笛卡尔集的连接。

### 面试题5.21 表连接、子查询的区别是什么？它们可以相互转化吗？你倾向于用哪种，为什么？

**【技术难度： 出现频率： 】**

第一层：

1. 表连接查询是将要查询的表先根据连接运算符和连接条件关联起来之后再根据筛选条件进行筛选查询；

2. 子查询是将子查询的结果代入到另外一个查询条件中去；

第二层：

子查询会多次遍历数据，而连接查询只遍历一次通常效率更高，因此一般情况下能用表连接查询做的就用表连接查询做。

### 面试题5.22 合并查询有哪些？联合查询有哪些？union和union all有什么不同?

**【技术难度： 出现频率： 】**

第一层：

1. union 合并去重复
2. union all 合并不去重
3. intersect 交集，获取两个结果集相交的部分
4. minus 差集，获取A结果集减去B结果集的部分

第二层：

1. 使用联合查询时，所有查询结果的列的个数和列的数据类型要保持一致；
2. MySQL只支持union和union all，Oracle四种都支持

### 面试题5.23 索引的作用？类型有哪些？它的优点缺点是什么？

**【技术难度： 出现频率： 】**

第一层：

1. 索引是一张特殊的查询表，类似书本的目录，通过目录就可以快速定位到需要的内容，所以说索引的优点是可以加快查询速度，随之而来的缺点就是会降低增删改的效率，因为这些操作会让数据库修改索引；
2. 常用的索引有普通索引index，唯一索引unique index，复合或称联合索引index 索引名(列1,列2...), MySQL还有主键索引 primary key，主键是唯一且非空的，Oracle中没有主键索引不过会为主键创建唯一索引；

第二层：

MySQL的Innodb存储引擎每个表都会有一个聚簇索引，该索引用来提升整行的查询效率，聚簇索引将数据存储与索引保存在同一个B+树中, 因此找到索引也就找到了数据， 聚簇索引默认建立在主键上，没有主键再找非空的唯一键，还找不到InnoDB引擎会自动生成一个列名为GEN\_CLUST\_INDEX 的聚簇索引，这是全局共享的，所有无法建立聚簇索引的表都共享该索引，并发会导致锁竞争，影响性能，因此每张表最好提供一个主键；

### 面试题5.24 什么是B树、B+树？B-Tree、B+ -Tree？

**【技术难度： 出现频率： 】**

第一层：

B树(Balance平衡)是一颗多路平衡查找树，每个节点可以有很多子节点，所有键值都有序分布在每个节点中。

为什么要有B树呢？因为一般来说索引本身也很大，不得不存放在磁盘中，而磁盘I/O读写操作是很慢的，从磁盘中读取数据时又是以页为单位读取连续的多页数据（Windows系统1页一般是4KB，磁盘一般会预读若干页数据，以提高读写效率），这使得索引的数据结构不得不基于磁盘读写的特性来量身打造，以实现查询索引时尽可能减少磁盘I/O操作，提高效率的目的。

基于上面的问题，B树设计时每个节点都保存索引值和对应数据，每个节点占磁盘1页或若干页空间、多个节点按顺序排列存储在磁盘中，这样就可以很好的节省磁盘读写次数，一次读写磁盘就可以读取若干个连续的B树节点。

B+数从B树演化而来，和B树的区别在于内节点不保存数据，而是将数据放在叶子节点中，并且将每个叶子节点通过链表指针关联下个节点连接起来，这大大方便了区间访问，可用在范围查询，而且由于内节点不保存数据，这使得一次磁盘读写可以读出更多索引值，从而减少查询时需要的读写次数。

第二层：

对比B树和B+树总结：

1. B+树的磁盘读写代价更低，一次磁盘I/O可以读取到更多的索引；
2. B+树的查询效率更加稳定，所有关键字查询的路径长度相同；
3. B+树叶子节点关联起来后，通过叶子节点就可以遍历整棵树，也提高了基于范围查找的效率。

第三层：

硬盘读写速度：磁盘/机械盘读写速度一般在600MB/S以内，固态盘读写速度一般在5000MB/S以内；

内存条读写速度：一般在50GB/S以内（很可惜，内存条断电后会丢失数据）。

由于内存和硬盘读写性能的巨大差距，和内存无法永久保存数据的特性，使得开发人员在设计程序时必须考虑这些性能差距带来的影响。

----------------扩展(无需背诵)：

磁盘I/O操作：I - input 输入 O - output输出，简单说I/O就是读写操作。

### 面试题5.25 什么样的字段适合建索引？

**【技术难度： 出现频率： 】**

1. 值不重复且不为空的列；
2. 很少修改值的列；
3. 频繁搜索的列；
4. 经常排序、分组的列；
5. 经常连接的列。

### 面试题5.26 in和exists区别?

**【技术难度： 出现频率： 】**

1. in查询在内部表和外部表上都能用到索引，而exists查询只能在内部表上用到索引；
2. 当子查询结果集较大，而外部表较小的时候，exists效率优于in，条件反过来则是in的效率优于exists；
3. not in查询内外表都会全表扫描，而not exists仍会用到内部表索引，所以无论那个表大，用not exists都比not in要快。

### 面试题5.27 哪些情况是不走索引的?

**【技术难度： 出现频率： 】**

1. 对索引列进行计算或表达式操作，不走索引；
2. 对索引列使用函数，不走索引；
3. 对索引列模糊查询时，将%放左边不走索引（比如 like '%test'）；
4. 字符串列与数字直接比较，字符串列隐式转数字，不走索引，使用in()语句也要注意这个问题；
5. 尽量避免 OR 操作，只要有一个字段没有索引，就不走索引；
6. Oracle使用不等于（ != <> ）不走索引，MySQl会走；
7. 使用 is null，is not null，不走索引，推荐用exists子查询；
8. not in不走索引，推荐用not exists子查询；
9. MySQL使用局部变量会导致索引失效。

### 面试题5.28 怎么提升查询效率？在千万级的数据库查询中，如何提高效率？书写高质量SQL的建议？如何优化大数据量的访问？

**【技术难度： 出现频率： 】**

第一层：

1. select后面尽量避免使用\*，需要什么列查什么列；
2. 有必要就建立索引，看情况考虑建立复合索引（或者说联合索引）效率更高，删除冗余和重复索引，索引不宜太多一般5个以内；
3. 查询复合索引需注意最左匹配原则（在使用到 > < in between like等范围搜索之前的字段与复合索引左边几个匹配上就可以使用这个复合索引）；
4. 尽可能的避免更新索引数据列，查询语句尽量使用索引、并注意不让索引失效，可以通过explain分析sql语句，确定索引执行的情况，来优化SQL；
5. 根据前台需要, 限制返回的数据行数；
6. 子查询语句推荐使用exists、not exists子查询替代in或not in；
7. Oracle中尽量避免使用游标，游标的效率较差；
8. 多表连接优先使用Inner join，如果是左右外连接，主表结果尽量小，不要有超过5个以上的表连接；
9. 当在SQL语句中连接多个表时，使用表的别名，并把别名前缀写到每一列上，这样语义更清晰也便于维护；
10. 慎用distinct去重；
11. 尽量使用union all（不去重）代替union（去重）；
12. 为了提高group by语句的效率，可以在执行到该语句前，把不需要的记录过滤掉；
13. 尽量使用数值型字段，若只含数值信息的字段尽量不要设计为字符型，字符需要比较多次，数值只需比较一次；
14. 尽可能的使用MySQL的varchar或Oracle的varchar2代替char，因为首先变长字段存储空间小，可以节省存储空间，其次对于查询来说，在一个相对较小的字段内搜索效率显然要高些；
15. 在所有的存储过程和触发器的开始处设置 SET NOCOUNT ON ，在结束时设置 SET NOCOUNT OFF ，这样无需在执行内部每个语句后向客户端发送影响记录行数的消息；
16. 尽量避免频繁创建和删除临时表，以减少系统表资源的消耗，如果使用到了临时表，在存储过程的最后务必将所有的临时表显式删除，先 truncate table ，然后 drop table ，这样可以避免系统表的较长时间锁定。

第二层：

JAVA方面优化：

1. 尽可能少创建对象；
2. 合理摆正系统设计的位置，大量数据操作和少量数据操作一定是分开的，大量的数据操作，肯定不是 ORM 对象映射框架能应对的；
3. 使用JDBC连接池连接数据库；
4. 控制好内存，让数据流起来，不要全部读到内存再处理，而是边读取边处理；
5. 合理利用内存，有的数据要缓存起来。

### 面试题5.29 一条SQL执行很慢的原因有哪些？一个表几千万数据，发现数据查询很慢，怎么办？

**【技术难度： 出现频率： 】**

第一层：

问题原因可能有这些方面：

1. 网络速度慢；
2. 要查询的数据量太大；
3. SQL语句性能差，未使用索引、没有索引；
4. 行锁、表锁、甚至死锁（设计缺陷）；
5. 表结构设计有问题；
6. 磁盘损坏导致读写阻塞，比如有坏点；
7. I/O性能问题出现瓶颈；
8. 内存不足、CPU占用过高。

第二层：

可以从以下几步排查问题：

1. 先排除网络原因，因为网络好坏是最显而易见的，很容易就判别；
2. 观察监控，如果CPU飙升或内存占满，调试程序定位到出现问题/耗时太长的sql语句，看是不是本条SQL语句造成的，将SQL语句拿出来，单独去工具上执行看耗时，如果依然耗时太长，大概率是SQL问题，优化它，查看是否走索引、减少查询的字段、数据量等等；
3. 若SQL语句优化没问题，却一直跑不通、很久得不到响应，还有可能是磁盘有坏点（磁盘故障），可以在其他硬件平台用相同数据对比测试；
4. 如果工具上执行很快，一放到代码里就很慢，那不是SQL语句问题，继续排查是不是锁导致的问题，如表锁、行锁，甚至死锁，若发现有锁长时间未释放，继续排查看相关SQL的存储过程、业务逻辑代码是不是有问题；
5. 若SQL语句和业务逻辑都没有问题，则考虑因为表的频繁的增删操作，导致表空间持续增大未得到释放，此时需要释放表空间，重建索引；
6. 还有看是不是 redo log 重做日志写满了需要同步到磁盘导致卡顿，需要优化相关设置，比如将redo log files重做日志文件放在速度快的磁盘上等等；
7. 若以上都OK，再看看是否表结构设计有问题，导致数据冗余；
8. 最后再考虑是否需要升级硬件，硬盘、内存或CPU等等。

第三层：

其他方面也想办法提升数据库并发性能：

1. 如果插入数据过多，考虑批量插入；
2. 如果数据量较大，优化修改/删除语句；
3. 尽量避免大事务操作，提高系统并发能力。

### --------高级

### 面试题5.30 commit在哪里会运用

**【技术难度： 出现频率： 】**

第一层：

commit 用于事务中，是提交事务的意思，会释放锁。未提交的数据在内存中，没有更新到磁盘。

第二层：

提交后数据还在内存中，在某个时刻才真正更新到磁盘文件，具体参考下一题

### 面试题5.31 redo undo 是什么？ 数据库日志有什么作用？回滚通过什么实现的？

**【技术难度： 出现频率： 】**

第一层：

redo log重做日志，记录已提交事务的数据信息，用于数据库异常关闭后恢复数据时写到数据文件中；

undo log撤销日志，记录用于回滚的数据信息，记录更改前的一份副本，当事务回滚时时，用这份副本来还原数据。

第二层：

Oracle和MySQL的InnoDB引擎在提交事务时会先在内存中的redo log重做日志中写入日志，然后再在某个时刻去更新磁盘文件，比如有redo log file重做日志文件写满指定大小后，或是事务提交后。这样的话，即使数据库异常崩溃，也可以通过redo log file重做日志文件检查并恢复未及时保存的数据。

在进行数据更新操作的时候，不仅会记录redo log，还会记录undo log，如果因为某些原因导致事务回滚，那么这个时候MySQL就要执行回滚（rollback）操作，利用undo log将数据恢复到事务开始之前的状态。

第三层：

MySQL可以在my.ini配置文件中设置redo log刷新策略，innodb\_flush\_log\_at\_trx\_commit：

1. 默认值是1，表示每次事务提交的时候都调用实时写入模式写入到磁盘，需等待完成再写下一次，很慢但安全；
2. 值为2时，表示每次事务提交的时候都调用不等待式的写入模式写入到磁盘，将数据交给文件系统去写出，不管结果，系统宕机有可能丢失文件系统中一次缓存的数据，虽然快，要考虑项目需要的安全性。

Oracle在事务相关的redo log信息已经确实被写入磁盘文件的时候，才会向客户端发出事务提交成功的消息，更安全，另外，Oracle对redo log文件的写出机制做了性能优化。

### 面试题5.32 事务是什么, 事务的四大特性

**【技术难度： 出现频率： 】**

第一层：

事务就是被绑定在一起作为一个逻辑工作单元的SQL语句分组，如果这组SQL语句中任何一个语句操作失败那么整组操作都失败，就要回滚到这组SQL语句操作前的状态。为了确保一组SQL语句要么都执行，要么都不执行的情况，就可以使用事务。

第二层：

要将一组语句作为事务考虑，就需要通过ACID测试，即原子性，一致性，隔离性和持久性(Atomicity Consistency Isolation Durability)：

1. 原子性，把一个事务的整个流程当做一个原子来执行，要么都执行成功，要么都失败；
2. 一致性，事务执行前后，数据是一致的，比如转账业务，无论事务成功与否，参与转账的两个账户之和应该是不变的；
3. 隔离性，在并发操作中，不同事务之间隔离开，互不影响；
4. 持久性，一旦事务提交成功，对数据的修改是永久的。

### 面试题5.33 什么是脏读 不可重复读 幻读 解决方法

**【技术难度： 出现频率： 】**

第一层：

1. 脏读是指一个事务读取到另一个事务未提交的数据，若另一个事务回滚，这数据就是脏的；
2. 不可重复读是指一个事务对同一条记录的2次读取不一致，因为中间有另一个事务对这条记录做了修改；
3. 幻读是指一个事务对一张表的2次范围查询不一致，因为中间有另一个事务对这张表做了增删操作。

第二层：

他们的解决方式分别是这样：

1. 应对脏读，可以设置读已提交的隔离级别，这是Oracle数据库的默认隔离级别；
2. 应对不可重复读，可以设置可重复读的隔离级别，这是MySQL数据库的默认隔离级别；
3. 应对幻读，可以设置串行化的隔离级别。

需注意，隔离级别越高性能越差。

第三层：

MySQL的innoDB引擎事务隔离级别为可重复读(repeatable read)时，可以预防幻读问题，其他引擎和Oracle需要调整到串行化(Serializable)级别才能预防幻读。

事务隔离级别是通过加锁实现的。

### 面试题5.34 什么是锁？

**【技术难度： 出现频率： 】**

第一层：

锁是实现事务的关键，是防止其他事务访问指定资源的实现并发控制的主要手段。

第二层：

执行逻辑是这样，在事务对某个数据库中的资源（如表和记录）存取前，先向系统提出请求，封锁目标资源，事务获得锁后，即取得目标资源的控制权，在事务释放它的锁之前，其他事务不能更新此数据。当事务提交或回滚后，释放被锁定的资源。

### 面试题5.35 Oracle、MySQL 中有哪几种锁？

**【技术难度： 出现频率： 】**

第一层：

按模式划分为排它、共享：

1. 加了排它模式锁数据对象，不能被其它的事务读取和修改；
2. 加了共享模式锁的数据对象，可以被其他事务读取，但不能修改。

第二层：

还可以按锁粒度划分，锁粒度越小并发性能越好：

1. 表级锁，开销小、加锁快，不会出现死锁，锁定粒度大，发生锁冲突的概率最高，并发度低；
2. 行级锁，开销大、加锁慢，会出现死锁，锁定粒度最小，发生锁冲突的概率最低，并发度高。

第三层

还可以按悲观、乐观锁划分：

1. 悲观锁，上锁期间其他人不能修改数据，直到操作完成后才会释放锁；
2. 乐观锁并不加锁，通常用版本号实现，在表中添加一个version数值字段，当执行事务时，先查询获取版本号version，当事务提交时，将之前获取的version和表中现在的version做匹配，匹配成功，将事务提交，并将version加1更新进去，匹配失败则回滚事务。

### 面试题5.36 MySQL**存储引擎了解过吗？有什么区别？请**简述Mysql的InnoDB

**【技术难度： 出现频率： 】**

第一层：

1. 存储引擎决定了表的数据存储方式、数据查询性能以及是否支持索引、事务等等；
2. MySQL中存储引擎以插件的形式运行，可以为每张表设置该表使用的存储引擎；
3. 创建新表时，如果不指定存储引擎则使用默认的，MySQL5.5 之前默认MyISAM，MySQL5.5开始默认InnoDB。

第二层：

MySQL中我们常用的存储引擎有5种：InnoDB、MyISAM、MEMORY、ARCHIVE、CSV。

它们分别具有这些特性：

1. InnoDB 支持事务，支持行级锁方便高并发，支持MVCC多版本并发控制，可恢复异常崩溃前状态，兼顾高可靠性和高性能；
2. MyISAM 不支持事务，支持表级锁，查询快其它慢；
3. Memory 将表数据存放于内存中，数据容易丢失；
4. Archive 只支持插入和查询，不可增删改，存储数据前会对数据压缩；
5. CSV 可以导入导出csv格式文件，csv文件格式比较通用，很多表格软件、数据库都支持。

第三层:

怎么选择合适的引擎？

除非必须，建议不混合使用多种存储引擎，避免潜在的复杂问题。

大部分情况下，InnoDB都是优先的选择，除非需要用到某些InnoDB不具备的特性，并且没有其他办法可以替代，MySQL5.5以后版本的InnoDB引擎性能更好。

除了InnoDB其它常见引擎的适用场景是这样：

1. MyISAM 优点是查询快，该引擎可以压缩数据以提高性能，但数据压缩后不能再修改数据，如果不介意MyISAM的崩溃恢复问题，读多写少的业务场景选用MyISAM引擎是合适的，MySQL8开始官方已不再推荐使用它，InnoDB已足够强大；
2. Memory 适用于临时表或缓存；
3. Archive 适合日志和数据采集类应用；
4. CSV 适合作为数据交换机制。

----------------扩展(无需背诵)：

怎么念看这里：

InnoDB - 念in no db

MyISAM - 念 my I S A M - Indexed Sequential Access Method索引顺序访问方法

MEMORY - 念 [ˈmeməri] - 内存

ARCHIVE - 念 [ˈɑːkaɪv] - 存档

MySQL常见存储引擎区别： https://www.cnblogs.com/lonlycoder/p/16969966.html

### 面试题5.37 MVCC 是什么？

**【技术难度： 出现频率： 】**

第一层：

MVCC多版本并发控制（Multi-Version Concurrency Control），是MySQL数据库InnoDB存储引擎用于提升并发性能的并发控制技术。

它是通过保存每次修改的数据版本，来实现在读数据时避开排它锁，去读取历史数据记录，这样就可以实现不受排它锁影响的读写并发。

第二层：

MVCC是通过undo log、read view来实现的。

undo log是撤销日志，在撤销日志中除了保存历史记录保存数据版本链。

read view内部保存一些关键数据用于比对，比如创建它的事务id、全局所有未提交的事务id等。是在事务启动时或每次查询前生成，这由MySQL的隔离级别决定。

MySQL会为每次修改数据对象保存一个数据版本，存在undo log中，所以每个被修改的数据对象在undo log中都记录了数据版本链，当一个事务查询某个数据对象时，即使该数据对象正被其它事务的排它锁锁定未提交释放锁，也可以通过read view中参数对比从undo log中数据版本链，从而获取该数据对象之前已提交的数据版本。当然，如果是一个事务查询它自己修改的数据对象，拿到的会是它自己修改后的结果，即使这个数据未提交。

第三层：

MVCC 只在MySQL的读已提交read-committed和可重复读repeatable-read 两个隔离级别下工作。

读已提交隔离级别时，每次查询都会生成read view，可重复读隔离级别则只在事务启动时生成，后续在该事务中都用这一个。

----------------扩展(无需背诵)：

参考网址1：https://cloud.tencent.com/developer/article/1876227

参考网址2：https://cloud.tencent.com/developer/article/1698716

### 面试题5.38 什么叫视图？优缺点？视图和表的关系？

**【技术难度： 出现频率： 】**

视图是一个虚拟表，就是封装了一条复杂查询的语句。

优点是简化复杂查询，缺点是不支持参数。

### 面试题5.39 什么是PL/SQL？

**【技术难度： 出现频率： 】**

第一层：

PL/SQL是一种程序语言，叫过程化SQL语言（Procedural Language/SQL）。它是Oracle对标准数据库语言SQL的过程化扩充，它将数据库技术和过程化程序设计语言联系起来，是一种应用开发语言，使用PL/SQL可以编写具有很多高级功能的程序。PL/SQL只有Oracle数据库有，MySQL没有。

第二层：

使用PL/SQL程序后，它的分析和执行完全在数据库内部进行，用户所需要做的就是在客户端发出调用PL/SQL的执行命令，数据库接收到执行命令后，在数据库内部完成整个PL/SQL程序的执行，并将最终的执行结果反馈给用户。在这整个过程中网络里只传输了很少的数据，减少了网络传输占用的时间，所以整体程序的执行性能会有明显的提高。

----------------扩展(无需背诵)：

PL/SQL代码块是没有名字的只能执行一次，不能被调用。从它这延伸出两个子程序，过程和函数，这两个都有名字，存储于数据库中，它们都可以被调用。

### 面试题5.40 什么是存储过程？优缺点？用什么来调用？

**【技术难度： 出现频率： 】**

第一层：

1. 存储过程是预编译的SQL语句，使用存储过程比单纯SQL语句执行要快，缺点是移植性差；
2. 通过call语句调用。

第二层：

优点展开说还有这些：

1. 存储过程是预编译过的，执行效率高，并且可以重复调用；
2. 存储过程的代码直接存放于数据库中，通过存储过程名直接调用，减少网络通讯；
3. 安全性高，执行存储过程需要有一定权限的用户。

### 面试题5.41 存储过程与函数的区别

**【技术难度： 出现频率： 】**

存储过程和函数的区别：

1. 存储过程用于执行特定任务，没有返回值，可以通过out参数实现返回多个值；
2. 函数用于完成某个具体功能，只能返回一个值。
3. 语法不同

### 面试题5.42 游标是什么？

**【技术难度： 出现频率： 】**

游标是指向查询结果集内的某条记录的指针，通过它可逐行获取结果集中的记录。

一般不用它，性能差。

### 面试题5.43 触发器的作用？

**【技术难度： 出现频率： 】**

触发器是特殊的存储过程，通过事件触发执行，比如插入操作前、更新语句前等等。

它可以强化约束，可以跟踪数据库内的操作从而不允许未经许可的更新和变化。

它可以级联触发，比如某表上的触发器上包含对另一个表的数据操作，而该操作又会导致该表触发器被触发。

### --------JDBC连接数据库

### 面试题5.44 JDBC中的PreparedStatement相比Statement的好处

**【技术难度： 出现频率： 】**

1. PreparedStatement 是预编译的，性能更好，它的sql语句会被数据库编译一次缓存起来，后续可以重复使用，它还可以防止sql注入；
2. Statement 每次执行sql语句都被数据库编译后再运行，无法防止SQL注入。

### 面试题5.45 JDBC连接数据库需要经过哪几步？

**【技术难度： 出现频率： 】**

1. 加载JDBC驱动；
2. 与数据库建立连接；
3. 将SQL语句发送到数据库执行并得到返回结果，用PreparedStatement或Statement；
4. 处理返回结果；
5. 释放资源。

### 面试题5.46 JDBC常用接口有哪些？

**【技术难度： 出现频率： 】**

1. Connection 用来与数据库建立连接；
2. Statement 用来执行SQL语句；
3. ResultSet 用于接收结果集；
4. PreparedStatement 是Statement子接口，执行sql语句，预编译，防止sql注

入，安全性高

### 面试题5.47 JDBC如何控制事务?

**【技术难度： 出现频率： 】**

同一事务中所有的操作，都在使用同一个Connection对象。

Connection 接口的三个方法与事务有关：

1. setAutoCommit(boolean值) 设置是否为自动提交事务，true自动，false手动；
2. commit() 提交结束事务；
3. rollback() 回滚结束事务。

### 面试题5.48 Statement 中execute、executeUpdate、executeQuery这三者的区别

**【技术难度： 出现频率： 】**

execute() 执行方法，可返回多个结果集、多个更新计数，应对复杂情况，一般不用；

executeUpdate() 执行更新方法，执行增删改语句，返回受影响的行数；

executeQuery() 执行查询方法，执行查询语句，返回一个ResultSet结果集对象。

### 面试题5.49 JDBC中怎么做批量处理的？

**【技术难度： 出现频率： 】**

1. 通过Connection对象创建Statement或PreparedStatement对象；
2. 使用Connection对象的setAutoCommit()将自动提交设置为false；
3. 使用addBatch()方法在创建的Statement对象上添加SQL语句到批处理中；
4. 在创建的Statement对象上使用executeBatch()方法执行所有SQL语句；
5. 最后，使用commit()方法提交所有更改。

### 面试题5.50 关系数据库中连接池的机制是什么？

**【技术难度： 出现频率： 】**

第一层：

数据库连接池负责管理数据库连接，它会根据设定控制连接总数，让程序可以重复使用现有的数据库连接，而再不是重新创建。因此它可以明显提高对数据库操作的性能以及稳定性。

第二层：

程序需要做的事是设置基本参数并创建连接池，然后按以下要点使用：

1. 从连接池获取或创建可用连接；
2. 使用完毕之后，把连接返回给连接池；
3. 在系统关闭前，断开所有连接并释放连接占用的系统资源。

## 模块6：Servlet 编程

### 面试题6.1 什么是Tomcat，怎样启动停止，配置文件，日志文件的存储

**【技术难度： 出现频率： 】**

1. tomcat 是免费、开源的web服务器软件，java编写的web项目可以部署在其上，用户端发送请求时，都是先将请求发送到tomcat上，tomcat再将请求发送到对应的项目上；
2. 启动、停止tomcat都是通过tomcat的bin目录中的脚本来执行，分别是startup启动、shutdown停止，在Windows系统中用bat文件格式，双击或cmd窗口输入文件名都可以，在Linux系统中用sh文件格式通过sh命令执行，另外，两种系统都可以将tomcat作为系统服务在系统后台运行，别忘了至少要装JRE并配置系统环境变量，tomcat依赖jre运行；
3. tomcat的config文件夹中就是配置文件，logs文件夹中就是日志文件。

----------------扩展(无需背诵)：

什么是web服务器：

web服务器就是网站服务器，web项目就是B/S架构的项目，Browser浏览器/Server服务器

单词解释：

startup启动，shutdown关机，config配置，log日志

### 面试题6.2 Tomcat根目录下有哪些文件夹? Tomcat目录结构说说看

**【技术难度： 出现频率： 】**

第一层：

1. bin 目录中存放启动startup和停止shutdown等可执行文件或脚本，bat格式在window系统中用，sh格式在linux系统中用；

2. conf 目录中存放配置文件，server.xml可修改端口等设置，context.xml可以集中设置连接池信息，web.xml则是全局的项目配置文件；

1. lib 目录中存放tomcat运行所需jar包的地方，也可以自行为tomcat添加其他jar包放这里；
2. webapps 目录中存放项目，把要运行的项目放在这里；
3. work 目录中存放jsp被转译servlet后生成的java和class文件；
4. logs 目录中存放tomcat运行的一些日志信息；
5. temp 目录中存放tomcat运行时的临时文件。

第二层：

1. 如果在tomcat的conf目录中的context.xml文件配置了全局的数据库连接池信息，需要放置对应的驱动包到lib目录。
2. jsp页面第一次被访问时会被转译成servlet，生成java文件并编译成class文件。

----------------扩展(无需背诵)：

全局context.xml文件的基本使用：https://www.cnblogs.com/limeiky/p/5714294.html

### 面试题6.3 Tomcat系统内存怎么配置

**【技术难度： 出现频率： 】**

第一层：

在tomcat的bin目录中的catalina文件头部加入所需设置，windows系统用bat格式，Linux系统用sh文件格式。

第二层：

比如，设置tomcat运行的jvm虚拟机的堆内存初始大小和最大大小可以这样设置：

1. windows系统在catalina.bat文件头部加 set JAVA\_OPTS=-Xms512m -Xmx1024m；
2. linux系统在catalina.sh文件头部加 JAVA\_OPTS='-Xms512m -Xmx1024m'。

-Xms 表示heap堆内存最小值

-Xmx 表示heap堆内存最大值

还有很多可以设置，参考下一题。

----------------扩展(无需背诵)：

单词怎么念：

catalina - 这样念 [ˌkætəˈlinə] 开特琳呢

这个脚本文件为什么叫catalina？ tomcat为什么叫tomcat？

Tomcat的这个单词的意思是“公猫”，因为它的开发者姆斯·邓肯·戴维森希望用一种能够自己照顾自己的动物代表这个软件，于是命名为tomcat，它的Logo兼吉祥物也被设计成了一只公猫形象。 Catalina是美国西海岸靠近洛杉矶22英里的一个小岛，因为其风景秀丽而著名。 Servlet运行模块的最早开发者Craig McClanahan因为喜欢Catalina岛故以Catalina命名他所开这个模块，尽管他从来也没有去过那里。

### 面试题6.4 Tomcat的优化经验

**【技术难度： 出现频率： 】**

第一层：

可以分为JVM内存、并发、缓存三个方向进行优化。

第二层：

1. JVM内存优化，是指优化tomcat运行时所用的JVM虚拟机的内存，比如说调大heap堆内存的上下限，以及调大保留内存的上下限等等，让JVM可用内存多一些；
2. 并发优化，是指优化tomcat运行时接收用户请求的连接器线程池，比如说调大线程数的上下限、允许空闲的线程数的上下限等等，让可用线程多一些；
3. 缓存优化，是指压缩html、js、css等等文件，使用服务器、浏览器都支持的GZIP压缩功能，最高可节省40%的数据流量。

第三层：

具体操作是这样：

1. 内存优化主要是对 tomcat 启动参数优化，我们可以在 tomcat 的 conf 目录中的启动脚本 catalina 中设置 JAVA\_OPTS 参数，windows用bat格式文件，linux用sh格式文件；
2. 并发优化需要配置 tomcat 目录下的 conf 目录中的 server.xml 文件中调整连接器connector的并发处理参数；
3. 在第二步的基础上，在 server.xml 文件中的connector标签中添加压缩的相关参数。

----------------扩展(无需背诵)：

1. 内存优化：

JAVA\_OPTS 参数说明(options参数)：

　　-server 启用JVM的 server 版；

　　-Xms 堆内存最小值；

　　-Xmx 堆内存最大值；

　　-XX: PermSize 持久代（在堆内存中）内存最小值

　　-XX: MaxPermSize 持久代（在堆内存中）内存最大值

配置案例：

JAVA\_OPTS='-server -Xms4096m -Xmx6144m -XX:PermSize=256m -XX:MaxPermSize=2048m'

注意：1024MB 等于 1GB

JVM调优参考网址：https://www.cnblogs.com/anyehome/p/9071619.html

1. 并发优化

参数说明：

maxThreads 客户请求最大线程数

minSpareThreads Tomcat初始化时创建的 socket 线程数

maxSpareThreads Tomcat连接器的最大空闲 socket 线程数

enableLookups 若设为true, 则支持域名解析，可把 ip 地址解析为主机名

redirectPort 在需要基于安全通道的场合，把客户请求转发到基于SSL 的 redirectPort 端口

acceptAccount 监听端口队列最大数，满了之后客户请求会被拒绝（不能小于maxSpareThreads ）

connectionTimeout 连接超时

minProcessors 服务器创建时的最小处理线程数

maxProcessors 服务器同时最大处理线程数

URIEncoding URL统一编码

配置案例：

<Connector port="9027"

　　protocol="HTTP/1.1"

　　maxHttpHeaderSize="8192"

　　maxThreads="1000"

　　minSpareThreads="100"

　　maxSpareThreads="1000"

　　minProcessors="100"

　　maxProcessors="1000"

　　enableLookups="false"

　　URIEncoding="utf-8"

　　acceptCount="1000"

　　redirectPort="8443"

　　disableUploadTimeout="true"/>

1. 缓存优化

参数说明:

　　compression 打开压缩功能

　　compressionMinSize 启用压缩的输出内容大小，这里面默认为2KB

　　compressableMimeType 压缩类型

　　connectionTimeout 定义建立客户连接超时的时间. 如果为 -1, 表示不限制建立客户连接的时间

配置案例：

<Connector port="9027"

　　protocol="HTTP/1.1"

　　maxHttpHeaderSize="8192"

　　maxThreads="1000"

　　minSpareThreads="100"

　　maxSpareThreads="1000"

　　minProcessors="100"

　　maxProcessors="1000"

　　enableLookups="false"

　　compression="on"

　　compressionMinSize="2048"

compressableMimeType="text/html,text/xml,application/javascript,text/css,text/plain"

　　connectionTimeout="20000"

　　URIEncoding="utf-8"

　　acceptCount="1000"

　　redirectPort="8443"

　　disableUploadTimeout="true"/>

注意：

1. tomcat7开始，js文件的MimeType改为了application/javascript，不再是text/javascript,要写对gzip才生效；
2. 服务器对图片、视频实时压缩性能很低，因此通常只压缩文本文件。

GZIP压缩参考网址：https://segmentfault.com/a/1190000012571492

### 面试题6.5 HTTP和HTTPS的区别

**【技术难度： 出现频率： 】**

1. https协议需要到证书认证机构申请证书，需要一定费用；
2. http是明文传输数据，https是用SSL协议加密传输数据，更安全；
3. http和https连接方式不同，默认端口也不同，HTTP默认80端口，HTTPS默认443端口。

### 面试题6.6 谈谈你对http协议的理解？

**【技术难度： 出现频率： 】**

第一层：

HTTP是超文本传输协议（Hyper Text Transfer Protocol），定义了客户端和服务端数据传输的规则，它也是一种请求/响应模式的协议，一般用于B/S架构项目，以浏览器作为HTTP客户端通过URL向HTTP服务端发送请求，服务器根据请求响应用户。

另外还有个HTTPS协议，更安全，

第二层：

HTTP主要特点有这些：

1. 简单快速，客户向服务器请求服务时，只需传送请求方法和路径，请求方法常用的有GET、POST、HEAD；
2. 灵活，HTTP允许传输任意类型的数据对象，传输的类型由Content-Type加以标记；
3. 无状态，无状态是指协议对于事务处理没有记忆，每次请求都是独立的，后续处理需要前面的信息则必须重传，优点是服务器响应比较快，缺点是不利于客户端与服务器的多个请求组成事务关系，通过Cookie和Session可以弥补这种不足。

第三层：

1. http协议中报文格式是这样，请求报文由请求行、请求头部、请求体3部分组成，响应报文由状态行、响应头、响应体3部分组成，请求行、状态行中包括请求地址、网络协议、响应状态码等关键信息，请求头、响应头中则是一些关键参数，请求体、响应体中放发送的数据；
2. 目前主流的是HTTP/1.1版本，底层依赖TCP协议建立连接传输数据，默认使用持久连接，持久连接只需要建立一次TCP连接就能进行多次HTTP通信，减少了TCP连接重复建立和断开所造成的开销，不过会阻塞客户端的下个请求，得等上次请求断开连接。使用持久连接，需在客户端请求头和服务端响应头中都加入这行代码 Connection:keep-alive （连接：保持-存活），没有这行代码或是加了Connection:close则会断开连接，还可以设置Keep-Alive: timeout=5, max=100表示持久连接保持5s、最多接收100次请求就断开；
3. 到这里了，就继续说HTTP的请求流程，参考下一题。

----------------扩展(无需背诵)：

关键词解释：

B/S架构 - Browser/Server浏览器/服务器

URL - Uniform Resource Locator统一资源定位符

什么是HEAD请求？

HEAD请求和GET请求本质是一样的，区别在于HEAD不获取响应体数据，而只获取响应头信息，多用于自动搜索机器人获取网页的标志信息，获取rss种子信息，或者传递安全认证信息，先获取数据大小再看是否下载等等使用场景。

### **面试题6.7 http请求的流程？在地址栏输入网址敲回车键, 发生了什么？**

**【技术难度： 出现频率： 】**

第一层：

简单说是这样：

1. 首先通过DNS服务器域名解析, 将域名解析成ip；
2. 经过3次握手, 客户端与服务器建立TCP连接；
3. 客户端向服务器发送请求报文；
4. 服务器处理完请求后发送HTTP响应报文到客户端，客户端解析响应报文；
5. 最后经过4次挥手, 客户端和服务器端断开TCP连接。

第二层：

更详细的说是这样：

1. 用户输入URL地址；
2. 浏览器解析URL解析出域名；
3. 浏览器将域名转换成服务器ip地址，先找本地DNS缓存列表，没有就从浏览器默认的DNS服务器查询并缓存起来；
4. 浏览器将端口号从URL中解析出来；
5. 浏览器经过三次握手与服务器建立TCP连接；
6. 浏览器发送一条HTTP请求报文；
7. 服务器返回一条HTTP响应报文；
8. 浏览器经过四次挥手与服务端断开TCP连接，并解析响应报文。

经过以上步骤之后，浏览器解析响应报文中的响应体中的页面数据，从上往下加载并渲染显示页面，如果页面中有其它资源需加载，比如图片、css文件等，每个资源都发送一个请求获取数据并加载。

### 面试题6.8 常见的HTTP状态码讲一些

**【技术难度： 出现频率： 】**

第一层：

1. 200 请求成功；
2. 304 请求的资源未改变；
3. 404 找不到资源；
4. 400 错误请求；
5. 500 服务器内部错误；

第二层：

1. 301 永久重定向；
2. 302 临时重定向；
3. 401 用户没有授权，可能是未登录、服务器安全策略导致等等原因导致
4. 403 已授权，但权限不足；
5. 405 方法不被允许，比如用get请求访问接收post请求的URL。

第三层：

1. 301、302状态码，需要配合响应头参数Location告诉浏览器重定向的网址，让浏览器跳转到新页面，永久重定向比如当页面地址改变后可以用于更新搜索引擎的URL缓存，临时重定向比如可用在短网址重定向到真实地址；
2. 304状态码，是指浏览器第二次以上访问某个相同页面时，会在请求数据中发送页面文件的上次修改时间信息，服务器会拿它去跟对应页面文件的信息比对，一致就表示服务器上的没有变过，无需再发给用户，就返回304状态码不发送页面，浏览器就会根据这个状态码使用本地缓存；
3. 400状态码，通常是请求参数数据和后端不匹配，比如参数个数不对、日期数据转换问题、JSON数据格式未转字符串等等原因造成。

### 面试题6.9 HTTP请求的GET与POST方式的区别

**【技术难度： 出现频率： 】**

第一层：

1. get提交的数据在地址栏可见, post提交的数据在请求体中，地址栏不可见；
2. get只能提交少量数据，post可以提交大量数据；
3. get一般用于获取数据，post一般用于提交数据；
4. get的URL中只支持ASCII字符，post没有限制；
5. get请求响应回来的数据会被浏览器缓存起来，post不会。

第二层：

1. URL只支持ASCII码字符集，如果在请求的URL中输入非ASCII字符并发送请求时，这些字符会被转码，这种编码就叫URL编码；
2. 浏览器缓存get请求的响应数据是这样，当发了一条get请求，浏览器会把服务器响应数据缓存起来，下次还发送相同URL的get请求就从缓存中读取数据，在一些场景，比如获取验证码的时候这肯定会引发问题，最简单的方法是每次都让get的参数数据不一样，访问的URL不同就不会导致get请求读取缓存的问题，比如 http://xxx/?t=当前时间戳

### 面试题6.10 解释一下什么是servlet，什么是servlet容器

**【技术难度： 出现频率： 】**

第一层：

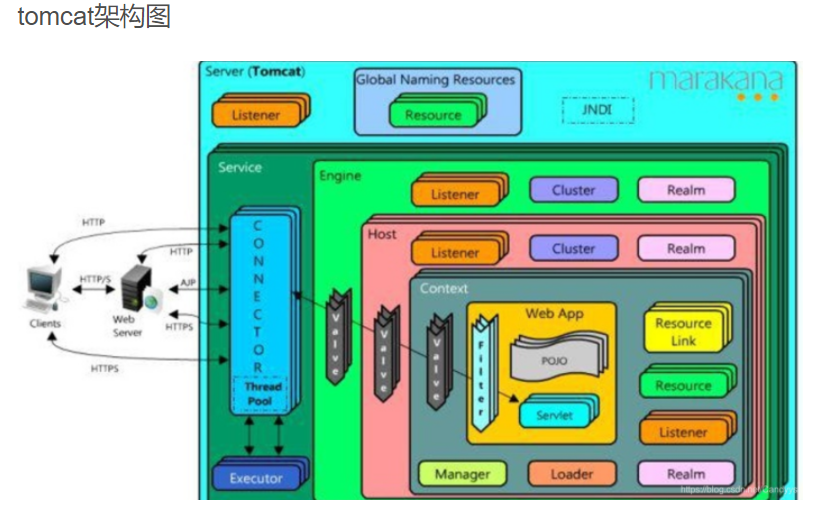
1. Servlet是在容器中运行的服务器端java程序，主要用于响应HTTP请求，和所有的Java程序一样，Servlet也运行在JVM中；
2. 引入Servlet容器是为了处理复杂的HTTP请求，由servlet容器来管理Servlet的创建、执行和销毁，常见Servlet容器程序有Tomcat、Jetty、WebLogic等等。

第二层：

Servlet是一套接口标准，Servlet容器通过Socket监听TCP连接，解析接收到的HTTP请求，然后调用开发者编写的Servlet，通过Servlet传递的参数，开发者可以相对更方便的获取到Http请求的相关信息，如参数，Cookies，Header等。SpringMVC框架中的控制器也是通过servlet实现的；

----------------扩展(无需背诵)：

Socket套接字是对网络中进行双向通信的端点的抽象，JAVA API中提供了Socket的实现类，支持TCP协议的Socket分为服务端、客户端Socket，支持UDP协议的Socket既可以是客户端也可以是服务端。



### 面试题6.11 servlet是单例吗？说一说Servlet的生命周期，执行过程?

**【技术难度： 出现频率： 】**

第一层：

1. Servlet是单例的，每个Servlet容器只会创建一个对象；
2. Servlet生命周期是这样：
3. 默认在第一次请求Servlet时，容器创建Servlet对象，并执行init初始化方法，init只调用一次，因为servlet是单例的，只创建一个对象；
4. 每次请求的时候都会执行service业务方法；
5. 容器关闭时，会调用destroy销毁方法，和init方法一样，只调用一次。

第二层：

HTTPServlet中已经重写service方法，会根据用户的请求方式分别调用对应的doGet、doPost等方法，但如果自己写的Servlet实现类同时重写了service方法和其它的doGet、doPost方法，则service方法优先被调用。

### 面试题6.12 实例化servlet有几种方式? load-on-startup的作用？

**【技术难度： 出现频率： 】**

1. Servlet实例化一般是由容器完成的，无非两种方式，一是在第一次被访问时实例化，二是通过配置让Servlet在容器启动时实例化，在web.xml中配置Servlet的时候可配置；
2. 在web.xml中给Servlet设置了load-on-startup为大于等于0的数，那么在容器启动的时候，容器会会按此值升序加载对应的Servlet并执行init方法。

----------------扩展(无需背诵)：

翻译：load on startup 在启动时加载

### 面试题6.13 servlet线程安全吗？

**【技术难度： 出现频率： 】**

第一层：

Servlet不是线程安全的，Servlet本身不存在安全与否的问题，如果Servlet中需要修改公共变量就会出现线程安全问题。

第二层：

详细说是这样，由于用户的每个请求，容器都会用一个线程来调用对应的Servlet，而Servlet是单例的，这导致一个Servlet的实例变量或静态变量可能有多个线程访问，那时就会出现线程不安全的问题。所以针对Servlet我们尽量不修改这些公共变量，可以考虑添加static final变为只读，如果非要做修改操作可以加同步锁，比如synchronized。

第三层：

如果确定要给修改公共变量的代码加线程同步锁，可以给公共变量前加volatile关键字，这样的话，读这个公共变量的代码可以不用加锁，提高读的效率。

### 面试题6.14 Filter的生命周期与执行过程

**【技术难度： 出现频率： 】**

第一层：

Filter过滤器的生命周期是这样：

1. 容器启动时创建Filter过滤器对象，并执行init初始化方法，只调用一次，Fileter也是单例的；
2. 每次请求时都会执行doFilter方法，过滤用户请求；
3. 当容器关闭时，执行destroy销毁方法，进行销毁，只调用一次。

Filter过滤器的执行过程是这样：

1. 浏览器发送一个请求，会到达servlet容器；
2. 容器会根据项目中web.xml中的Filter过滤路径配置，调用匹配的过滤器过滤请求；
3. 执行对应Filter过滤器中的doFilter方法；
4. 在doFilter方法中实现过滤业务逻辑，最后写上一句代码，调用FilterChain过滤器链的doFilter方法，将用户请求传递给下一个过滤器过滤；
5. 请求的URL路径匹配的所有过滤器执行完毕后，容器调用该路径匹配的资源。

第二层：

Filter 的执行顺序是这样：

1. 通过 web.xml 配置的 Filter 过滤器，执行顺序由 <filter-mapping> 标签的书写顺序决定，当请求路径匹配到多个Filter时，web.xml配置中靠前面的先执行；
2. 通过 @WebFilter 注解配置的 Filter 过滤器，无法进行排序，若需要对 Filter 过滤器进行排序，建议使用 web.xml 进行配置。

### 面试题6.15 过滤器有哪些作用？

**【技术难度： 出现频率： 】**

1. 可以设置请求、响应数据的字符编码；
2. 可以验证用户是否已登录；
3. 可以过滤敏感词；
4. 可以验证客户是否来自可信的网络；
5. 可以验证客户的浏览器是否支持当前的应用；
6. 可以记录系统的日志

等等……

### 面试题6.16 servlet三个作用域对象和每个域数据的适用范围？session 和 application的区别？

**【技术难度： 出现频率： 】**

第一层：

1. Request 请求域对象，数据在一次请求中有效；
2. Session 会话域对象，数据在一次会话中有效；
3. ServletContext 应用域对象，数据在整个web应用中有效，所有用户共享同一个。

第二层：

他们的生命周期是这样：

1. Request对象，容器不一定会每次请求都创建这个对象，这要看容器的优化策略，通常是多次请求使用同一个Reuqest对象，每次请求结束就重置刷新这个对象中的数据；
2. Session对象在用户首次访问项目时创建，用户离开项目之后被丢弃，有两种方式表示用户离开，一种是用户关闭浏览器窗口，另一种是用户无动作30分钟，这个等待时间可以调整；
3. ServletContext对象在Web服务器启动时创建，关闭Web服务器时销毁。

第三层：

1. JSP中还有个Page页面域对象，变量名是 pageContext， 数据只能在当前页面使用；
2. JSP中的application就是ServletContext，只是JSP中另外用了变量名application保存，后来大家都习惯将ServletContext用application名字的变量保存，便于理解；
3. 除了ServletContext是所有用户共享一个对象，其他作用域对象都是容器为各用户分别创建一个。

### 面试题6.17 request.getAttribute() 和 request.getParameter() 有何区别

**【技术难度： 出现频率： 】**

第一层：

1. request.getAttribute()是根据属性名从request域对象中获取对应的属性值；
2. request.getParameter()是从浏览器提交的请求数据取值

第二层：

1. URL参数传递的数据是字符串，因此request.getParameter()获取到的是字符串String类型；
2. request域对象中setAttribute()存的数据是任意对象，用Object类型变量接收保存，所以request.getAttribute()取出来取到的是Object类型。

----------------扩展(无需背诵)：

翻译：get获取，set设置，Attribute属性，Parameter参数

### 面试题6.18 request对象常用方法，以及方法的功能描述

**【技术难度： 出现频率： 】**

1. request.getMethod() 返回浏览器的请求方式 get或post等；
2. request.getContextPath() 获取项目访问路径；
3. request.getRequestURI() 获取请求行的资源名部分；
4. request.getRequestURL() 获取完整的请求路径；
5. request.getHeader(String key) 获取请求头中key名对应的value值；
6. request.getParameter(String key) 获取URL参数中key名对应的第一个value值；
7. request.getParameterValues(String key) 获取URL参数中key名对应的若干个value值, 返回一个String数组；
8. request.setCharacterEncoding("UTF-8") 将请求数据的字符编码设为 UTF-8；
9. req.getCookies() 获取所有Cookie，返回Cookie数组；
10. req.getRequestDispatcher(转发路径).forward(request, response) 转发其他路径。

### 面试题6.19 response对象常用的方法，以及方法的功能描述

**【技术难度： 出现频率： 】**

第一层：

1. response.setStatus(500) 设置响应行的状态码；
2. response.sendError(404,"访问的资源不存在") 设置状态的同时设置错误信息, 错误信息可以在页面中展示给用户，通常由服务器自动处理这两步；
3. response.setHeader("age","20") 用给定名称和值设置响应头, key和value都可以自定义；
4. response.addHeader("age","21") 用给定名称和值添加响应头，此方法允许响应头有多个值；
5. response.setContentType() 设置请求体数据的MIME类型，告诉浏览器如何解读数据；
6. response.getOutputStream() 获取字节输出流；
7. response.getWriter() 获取字符输出流；
8. response.setCharacterEncoding("UTF-8") 设置响应体数据的字符编码类型为UTF-8；
9. req.getRequestDispatcher("/WEB-INF/show.jsp").forward(req, resp)；
10. response.sendRedirect(URL地址) 让客户端重定向到指定的URL，可以重定向服务器外部资源。

第二层：

1. 通过response获取字节输出流或字符输出流，只能用其中一个流发送数据，否则引发异常；
2. response的setHead和setContentType方法都可以通过响应头告诉浏览器，响应体中的数据的MIME类型是HTML网页文件，字符编码是UTF-8，需要注意的是，这不能替代response.setCharacterEncoding("UTF-8")去放在Filter过滤器中做字符编码过滤，因为服务器上的其它资源有其它的MIME类型，比如图片、JS文件、CSS文件等，被过滤修改MIME类型浏览器就不认识了。

参考代码：

response.setHead("Context-type", "text/html;charset=UTF-8");

response.setContentType("text/html;charset=UTF-8");

----------------扩展(无需背诵)：

MIME类型是什么？

MIME (Multipurpose Internet Mail Extensions 多用途Internet邮件扩展) ，它设计的最初目的是为了在发送电子邮件时附加多媒体数据，让邮件客户程序能根据MIME类型进行处理，HTTP协议也支持了它，用来表示数据的格式、类型，让浏览器根据MIME类型用对应的方式解析响应体中数据。

### 面试题6.20 如何让浏览器从直接打开图片改成下载图片

**【技术难度： 出现频率： 】**

1. 设置Content-Type响应头, 告诉浏览器文件数据的MIME类型；
2. 设置Content-Disposition响应头, 通知浏览器以attachment附件下载方式打开, 并指定默认文件名。

----------------扩展(无需背诵)：

代码案例：

String fileName="black.jpg";

String mimeType = application.getMimeType(fileName);//根据文件名获取MIME类型

response.setHeader("Content-Type",mimeType);//设置Content-Type响应头

//设置Content-Disposition响应头，告诉浏览器用附件下载方式打开

response.setHeader("Content-Disposition","attachment;filename="+fileName);

翻译：

type类型，header头部，content内容，Disposition处理，attachment附件，filename文件名

### 面试题6.21 如何解决 get 和 post 乱码问题？字符编码怎么处理？

**【技术难度： 出现频率： 】**

第一层：

简单说的话要这样做：

1. 前端做两件事，一是需要注意代码编辑软件要用UTF-8字符编码写文件，并通过语法标识告诉浏览器这些页面、JS、CSS文件使用的字符编码是UTF-8，二是页面中对包含非ASCII码的URL都用JS内置的encodingURI()方法进行转码；
2. 后端做两件事，一是需要写一个过滤器，过滤所有请求，在doFilter方法中通过request和response的setCharacterEncoding("UTF-8")方法，分别将请求和响应数据都设置为UTF-8的字符编码，二是需要在web.xml文件中将这个<filter-mapping>标签前置，确保先执行它。

第二层：

详细说的话是这样的：

由于URL只支持ASCII码字符集，各浏览器会对页面发出请求的URL中非ASCII码字符进行URL编码，问题是地址栏的字符编码可能不同，有GBK、UTF-8等等，各浏览器URL编码算法可能也有不同。

一般不需要用户到地址栏输入请求的字符数据，所以前端只需要在JS脚本中处理好请求的URL字符，统一使用js自带的encodingURI()方法，避免各浏览器的差异问题，另外就是文件的字符编码统一用UTF-8，别忘了还要写对应的标识告诉浏览器用UTF-8字符编码解析这些文件。

而后端的request.getParameter()方法获取参数数据时会自动做URL解码，所以后端只需写一个过滤器，过滤所有请求数据和响应数据的字符编码设置为UTF-8，这样request.getParameter()方法就可以按UTF-8字符编码正确解读request中的数据，包括get和post请求的数据，并且客户端浏览器也可以根据页面标识的UTF-8编码正确解读UTF-8字符编码的响应数据。

----------------扩展(无需背诵)：

不同浏览器中字符编码对URL编码的影响测试：https://blog.csdn.net/u014785687/article/details/74078512

代码案例：https://www.cnblogs.com/lonlycoder/p/16983730.html

### 面试题6.22 servlet中怎么定义forward 和redirect？怎么使用转发和重定向？

**【技术难度： 出现频率： 】**

第一层：

1. 转发forword，调用request.getRequestDispatcher().forward()，需要将转发路径传入getRequestDispatcher()方法的参数中，另外还要将request对象和response对象传入forward()方法的参数中；
2. 重定向redirect，调用response.sendRedirect()方法，需要将重定向路径传入sendRedirect()方法的参数列表中。

第二层：

如果项目设置以webapp目录作为页面资源根目录，其中的WEB-INF文件夹就只能通过转发访问，重定向或直接请求都无法访问，这是容器的设置，能带来一定的安全性。

第三层：

转发和重定向中的的路径是这样：

1. 绝对路径，用 / 正斜线开头，表示从项目访问路径开始，比如当前请求路径是/demo/a/b，如果项目访问路径是/demo，此时写绝对路径/c，结果就是/demo/c；
2. 相对路径，不以正斜线开头则，表示从当前页面所属的路径开始，比如当前请求路径是/demo/a/b，表示从当前页面所属的路径是/demo/a，此时写相对路径c，结果就是/demo/a/c。

重定向有些许不同需要注意：

1. 绝对路径 / 在重定向中会回到容器的根目录，比如http://localhost:8080/，因为重定时，项目内外都可访问，服务器不知道这个请求是项目内部还是外部，所以如果如果还要重定向回到当前项目中，需要通过request.getContextPath()方法在前面加上项目访问路径，比如response.sendRedirect(request.getContextPath() + "/c");
2. 重定向还可以通过HTTP、HTTPS等协议开头访问外部路径。

----------------扩展(无需背诵)：

代码案例：

request.getRequestDispatcher("/WEB-INF/show.jsp").forward(request, response);

response.sendRedirect(request.getContextPath() + "/index.jsp");

翻译：get获取，Dispatcher调度器，send发送，context path项目路径

### 面试题6.23 forward和redirect的区别? 转发和重定向的区别？

**【技术难度： 出现频率： 】**

第一层：

1. 转发是服务器内部行为，只能转发当前项目内部资源，重定向是客户端行为，在这个过程中服务器只是告知客户端，实际是客户端访问新的地址，所以重定向可以访问任意资源；
2. 转发地址栏不会变，是一次请求，重定向地址栏会改变，是二次请求；
3. 转发性能优于重定向。

第二层：

转发由于地址栏不变，所以刷新会重复提交之前的表单数据。重定向是刷新跳转之后的地址。

### 面试题6.24 cookie有了解么？cookie本质上是什么？怎么从前端传输到后台

**【技术难度： 出现频率： 】**

cookie是一段不超过4KB的文本数据，由服务器生成发送给浏览器，浏览器把cookie保存在本地硬盘中，下一次请求同一网站时，会把该站点相关的cookie放在HTTP请求数据的请求头中，发送给服务器。

### 面试题6.25 cookie和session区别？

**【技术难度： 出现频率： 】**

第一层：

1. session数据存在服务器, cookie存在客户端浏览器
2. session数据安全, cookie数据相对不安全
3. session可以存任意大小的数据, cookie大小有限制, 一个cookie最大4KB

第二层：

1. session可以存任意类型对象，但非必要的数据尽量不要往里存，它生命周期长，用户多了数据也多，很容易占用过多内存；
2. cookie有很多限制：
3. 只能存字符串，而且只能放ASCII码字符集支持的字符；
4. 不支持空格、=等号、,逗号、;分号等字符，可以通过URL编码后再存入；
5. 每个域名或ip地址最多只能存50个左右到客户端浏览器，多了会被删。

### 面试题6.26 页面传递对象的方法？

**【技术难度： 出现频率： 】**

request、session、application、cookie、URL传参等。

其中cookie和URL只能传递字符串，而且只支持ASCII码中的字符，所以最好通过URL编码再存入。

### 面试题6.27 请描述下 cookie，sessionStorage 和 localStorage的区别？

**【技术难度： 出现频率： 】**

第一层：

相同之处是他们都只能存字符串。

主要的不同之处有这些：

1. cookie只支持ASCII字符集支持的字符，并且不支持空格、=等号、,逗号、;分号等字符，需要转URL编码存入，其它两个storage对象没有这些限制；
2. 浏览器每次请求服务器时都在请求数据中带上cookie, 而sessionStorage和localStorage都只存在浏览器本地保存；
3. localStorage 的数据是永久的，除非主动删除，sessionStorage 的数据只在当前会话中有效，浏览器窗口关闭即会话结束就失效，cookie是在设置的过期时间前有效，不加则在浏览器关闭前有效；
4. localStorage的数据在同源的所有页面共享，cookie的数据在相同域名或ip的请求和页面中共享，sessionStorage不能跨页面共享。
5. 在JS中，cookie是document对象的属性，其它两个storage对象是window对象的属性；
6. cookie的存储大小最大4KB，其它两个storage对象的存储大小一般5MB；
7. cookie删除数据需要通过设置max age过期时间为0s来删除，其它两个storage对象可以通过delete语法删除某个属性或clear()方法清空全部数据。

第二层：

1. JS要往这些地方存储数组或对象时，应先转换成字符串格式再进行存储，比如可使用JSON.stringify(obj)方法转字符串；
2. 同源和跨域是这样：域由协议、域名、端口号组成，两个请求的这些内容一致就属于同源，其中任何一个不一样就是跨域，比如 https://cn.bing.com:80 和 https://cn.bing.com:8080 属于跨域。

----------------扩展(无需背诵)：

翻译：

local本地，storage存档，document文档，cookie饼干（命名偶尔比较随性），delete删除，stringify字符串化、变成字符串，clear清除

### 面试题6.28 session的实现原理

**【技术难度： 出现频率： 】**

session是依赖Cookie实现的服务器端对象。

当用户第一次请求服务器，容器会创建session对象并随机分配唯一的session id会话编号给这个对象，并将这个session id通过cookie给到客户端，保存在客户端浏览器，如果禁用了cookie，就在URL中作为参数传递。

之后的每次请求，客户端请求中都会带上这个cookie，服务器根据cookie中的session id就能找到对应的session对象。

cookie默认是浏览器关闭前都存在，所以0只要浏览器不关，无论访问服务器多少次，使用的都是同一个session对象. 这样就可以让多个请求共享同一个session了。

### 面试题6.29 如何使用session进行身份验证?

**【技术难度： 出现频率： 】**

第一层：

1. 用户登录成功将用户信息保存在Session对象中；
2. 后续请求如果在Session对象中能查到说明已经登录，如果查不到说明未登录；
3. 若需退出登录就从Session中删除用户的信息；

第二层：

这个会话状态的保持是基于cookie实现的（详情参考上一题session实现原理来表达）

第三层：

1. session数据在服务器，所以这个方案是安全的。
2. 若需记住用户登录状态，就是关闭掉浏览器后，重新打开浏览器还能保持登录状态，就配合Cookie来用，将加密后的关键数据通过cookkie，发给客户端浏览器，这个cookie可以设置一定的过期时间比如7天，不过cookie会带来一定的安全性问题。

### 面试题6.30 单点登录中，如果 cookie 被禁用了怎么办？

**【技术难度： 出现频率： 】**

第一层：

单点登录就是在多个系统中，用户只需一次登录，各个系统即可感知该用户已经登录。

以下两个方案，生成的数据都可以在客户端任意位置保存，比如cookie中、localStorage中，cookie被禁用就用localSotrage来保存数据，这些方案可以做到不使用依赖cookie的session对象，在单点登录需求中比session的使用成本低很多。

方案一，发送Token令牌数据给客户端，服务端也保存一份，客户端的每次请求都带上它给服务器比对；

方案二，发送JWT对象数据给客户端，通过签名对象数据，客户端的每次请求都带上它给服务器，服务端只需要校验数据可用性，无需比对数据，需要更高安全性还可以加密。

第二层：

展开说是这样：

方案一，每次请求中都带上一个Token令牌数据：

Token是在服务端产生的。前端使用用户名/密码向服务端请求认证，如果服务端认证成功，就在服务端会返回Token 给前端。前端可以在每次请求的时候带上 Token 证明自己的合法地位。通常这个Token在服务端会存入内存数据库中，比如Redis，用于和用户的进行比对，效率更高而且可以设置Token的可用时长。如果这个Token在服务端持久化，比如存入磁盘中的MySQL数据库，那它就是一个永久的身份令牌。

方案二，使用JWT(JSON Web Token - JSON格式的用于WEB项目的令牌数据)

JWT方案是在服务器身份验证之后，生成一个JSON对象并将其发送回用户。在之后的用户请求中，客户端发回JSON对象。服务器仅依赖于这个JSON对象来标识用户。为了防止用户篡改数据，服务器将在生成对象时添加签名，当用户请求时通过数据中的签名校验数据是否被修改、是否可用，在整个过程中，服务器不保存任何会话数据，更容易扩展。

第三层：

对比两种方案的优缺点：

方案一：

使用传统Token服务端需要提供授权模块，用于验证客户端发来的token信息，要做数据的查询操作等，效率很低，优点是适用于分布式系统，安全系数高，Token不包含真实数据，而且可以限制它的有效时间等；

方案二：

使用JWT在服务端只需校验JWT数据的可用性并取出数据使用就可以，优点是适用于分布式系统，无需保存数据查询比对、使用方便等等，缺点一是无法吊销已生成的JWT、而且不知道生成了多少JWT，二是被拿到JWT就有了登录权限，服务器最好使用HTTPS协议加密传输数据更安全。

## 模块7：Spring 框架

### 面试题7.1 说说你对Spring的理解，Spring能帮我们做什么事情，什么是Spring

**【技术难度： 出现频率： 】**

第一层：

1. Spring是一个开源框架，它的创建目的是简化企业应用的开发，主要用它开发JavaWeb项目；
2. Spring是分层的框架，由七个模块组成，通过这些模块组成不同的层级，每个模块既可以相互独立存在于项目中，又可以被其他第三方框架兼容使用；
3. Spring的工作原理就是启动一个容器来管理项目中用到的各种对象，通过IOC控制反转，将创建对象的控制权交给容器，我们只需要通过xml配置文件来告诉容器需要创建哪些对象即可。

第二层：

1. Spring能非常简单的帮我们管理数据库事务；
2. Spring能与第三方框架无缝集成，也提供一些自带框架，比如数据库访问框架MyBatis，自己也提供了JDBC Template（JDBC访问模板），再比如第三方的Web框架Struts，它也提供了一个Spring Web MVC框架；
3. Spring能方便的与Java EE的技术整合，比如Java Mail、任务调度，还可以和更多技术整合，比如缓存框架。

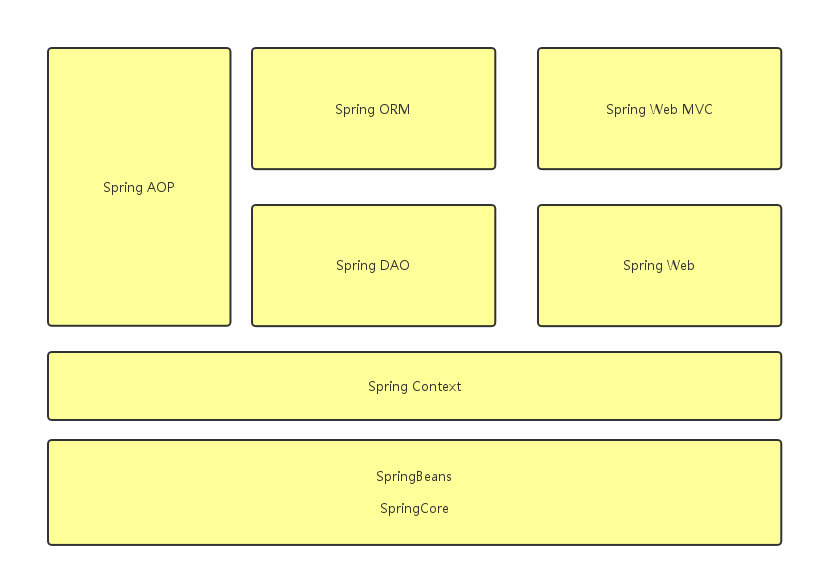
第三层：

Spring的7个模块有这些：Spring Core（Spring核心）、Spring Context（Spring应用上下文）、Spring AOP（Spring面向切面编程）、Spring ORM（Spring对象关系映射）、Spring DAO（Spring数据访问对象）、Spring Web（Spring网络编程）、Spring Web MVC（Spring基于MVC分层模型的网络编程）。

它们展开说是这样：

1. Spring Core（Spring核心）模块能帮我们根据配置文件管理对象的生命周期，以及组装对象之间的依赖关系；
2. Spring Context（Spring应用上下文）模块帮助Spring核心读取配置文件、装载bean的定义；
3. Spring AOP（Spring面向切面编程）模块通过生成代理对象来增强目标对象，它能帮助我们无耦合的实现日志记录，性能统计，安全控制；
4. Spring ORM（Spring对象关系映射）模块能帮我们处理对象和数据库记录的映射关系，Spring没有自己开发ORM框架，而是支持主流映射框架；
5. Spring DAO（Spring数据访问对象），通过传统DAO方式整合各个ORM映射框架的JDBC访问流程，抽象出统一的异常体系、数据访问模板、事务管理等，并且提供了一个JDBC Template（JDBC模板），不过现在这种整合方式基本不用了；
6. Spring Web模块提供WebApplicationContext（Web应用上下文信息），它建立在应用程序上下文模块之上，为基于Web的应用程序提供了上下文，因此 Spring 框架支持与 Struts 等Web框架的集成；
7. Spring Web MVC是Spring提供的一个基于MVC分层模型的Web框架，提供了对Controller控制器、Service服务、View视图等组件的管理，简称Spring MVC，它基于Spring Web模块，因此导入了它就导入了Srping Web，它可以帮我们实现最高效的web开发。

----------------扩展(无需背诵)：



### 面试题7.2 介绍一下 Spring Bean 的生命周期

**【技术难度： 出现频率： 】**

第一层：

简单概括就是四个阶段：1. 实例化；2. 初始化；3. 使用中；4. 销毁。

第二层：

详细的说是这样：

1. 实例化Bean对象，通过工厂模式或构造方法；
2. 设置Bean属性，通过DI依赖注入；
3. 检查Bean是否实现了相关Aware接口，如果实现了就设置相关依赖；
4. 如果有Bean实现了BeanPostProcessor接口，则Spring将调用该Bean的预初始化方法 postProcessBeforeInitialzation()对Bean进行加工操作，Spring 的 AOP 就是利用它实现的；
5. 调用初始化Bean的方法invokeInitMethods()，如果对象实现了InitializingBean接口，spring将调用它的afterPropertiesSet方法；
6. 如果Bean配置中使用了init-method属性声明了初始化方法，该方法也会被调用；
7. 如果有Bean实现了BeanPostProcessor接口，它的postProcessAfterInitialization方法将被调用，此时对象已经可以被应用系统使用了；
8. 如果在Bean配置中指定了该对象的作用范围是单例（scope="singleton"），则将该Bean放入 Spring IOC的缓存池中，触发Spring对该对象的生命周期管理；如果在Bean配置中指定了该Bean的作用范围为原型（scope="prototype"），则将该Bean交给调用者，调用者管理该Bean的生命周期，Spring不再管理；
9. 在Spring容器关闭前会调用容器管理的Bean的销毁方法，如果Bean实现了 DisposableBean 接口，则调用Bean的destory()方法；如果在配置文件中通过destory-method属性指定了Bean的销毁方法，则调用该方法销毁Bean。

### 面试题7.3 Spring中Bean的scope范围

**【技术难度： 出现频率： 】**

第一层：

在Spring 2.0之前，有singleton单例和prototype原型两种。

singleton单例，是指只创建一个实例，默认值。

prototype原型，是指在每次使用时都创建实例。

第二层：

在Spring 2.0之后推出另外三种：request，session 和 global session。

request请求，是指每次请求各创建一个实例。

session会话，是指每次会话各创建一个实例。

global session全局会话，在基于porlet的web应用中使用才有意义。

### 面试题7.4 Spring框架中的单例bean是线程安全的吗?

**【技术难度： 出现频率： 】**

第一层：

单例bean不是线程安全的。

第二层：

如果只是查询、读取，就不存在线程安全问题。

如果有对这个bean做修改操作，就产生线程安全问题。

### 面试题7.5 谈谈反射和框架的联系？

**【技术难度： 出现频率： 】**

反射允许java在运行时动态的加载任意的类，并获取任意类的属性，调用任意类的方法，而不用在编写代码时写死具体的类。这使得java编码变得更加灵活，很多框架都是以反射为基础开发出来的。

比如说Spring框架对象的创建，都是通过反射来完成，我们在配置文件中，定义类的依赖关系，并制定类的全路径，就是为反射创建对象提供信息。

### 面试题7.6 谈谈Spring的IOC和DI

**【技术难度： 出现频率： 】**

第一层：

1. IOC（Inversion of Control）控制反转，它是一种思想, 就是把实例化的控制权交给外部容器，而不是在类代码中直接实例化，这样降低了类与类之间的耦合性，在Spring框架中我们可以通过xml配置文件或注解让spring容器去控制类的实例化；
2. DI（Dependency Injection）依赖注入，它负责把类与类之间的依赖关系组合起来，使用它可以通过spring容器将所需的对象或属性注入到目标对象中；
3. 它们都是通过反射实现的。

第二层：

DI依赖注入常用三种方式，分别是set注入、构造注入、注解注入。注解注入有两种，@Resource和@Autowired。

### 面试题7.7 spring注入方式有几种

**【技术难度： 出现频率： 】**

第一层：

Spring有三种依赖注入方式，分别是set注入、构造注入、注解注入。

第二层：

注解注入有两种，@Resource和@Autowired。通过注解向属性注入bean时，容器会通过反射打开访问权限，因此可以无需set方法。

### 面试题7.8 IOC 的优点是什么？

**【技术难度： 出现频率： 】**

1. 实现对象的可配置和易管理；
2. 降低代码耦合度；
3. 支持饿汉式和懒汉式加载。

### 面试题7.9 请说说Spring中常用的注解

**【技术难度： 出现频率： 】**

1. @Component、@Repository、@Service、@Controller、@RestController，让Spring容器管理对象，其中@RestController等于两个注解，@Controller和@ResponseBody相加；
2. @Resource、@Autowired，注入对象；
3. @RequestMapping、@GetMapping、@PostMapping，为处理器类或方法映射请求URL和请求类型；
4. @RequestParam 将请求数据根据key名映射到目标参数，也可以给参数设置默认值；
5. @PathVariable 获取Restful风格URL中的数据给参数；
6. @RequestBody、@ResponseBody，接收、响应JSON格式字符串；
7. @Transaction 声明式事务注解；
8. @DateTimeFormat 让字符串日期入参时转换为Date；
9. @Scope 设置Bean的作用域；
10. @Configuration 告诉Spring容器，当前类是一个配置类。

### 面试题7.10 @RestController 和 @Controller的区别

**【技术难度： 出现频率： 】**

1. @RestController等于@Controller加上@ResponseBody；
2. @Controller注解是让spring容器管理当前类的对象, 并表示此类是Controller控制器；
3. @ResponseBody注解是将方法的返回值由JAVA对象转JSON字符串。

### 面试题7.11 @Autowired 注解有什么用？@Autowired和@Resource区别？知道自动注入的注解吗？

**【技术难度： 出现频率： 】**

第一层：

@Autowired和@Resource都是自动注入的注解，它们都可以用于属性、构造方法、set方法。

区别是这样：

1. @Autowired 默认按类型装配，当有多个同类型bean的时候效率低；
2. @Resource 先按属性名装配，找不到bean再按属性类型装配。它可以指定name属性，若指定了就只按名称进行装配。

第二层：

再往下展开说是这样：

1. @Autowired默认按类型装配，当有多个同类型的bean时，可以用@Qualifier注解指定需要注入的bean，或者用@Primary注解标识出首选Bean。
2. @Resource注解是JAVAEE的注解，可降低与Spring的耦合度。

### 面试题7.12 @Qualifier 注解有什么用？

**【技术难度： 出现频率： 】**

@Qualifier 注解是用于消除依赖冲突的。

当使用@Autowired注入bean时，当有多个相同类型的bean而只需要注入其中一个，就可以用@Qualifier注解指定要装配哪个名字的bean，让容器正确装配。

### 面试题7.13 什么是AOP？

**【技术难度： 出现频率： 】**

第一层：

AOP是面向切面编程, 通过AOP可以实现在不修改源代码的情况下给程序动态添加功能。

在开发过程中，如果需要加入日志管理、权限管理等非业务性功能的时候，容易产生大量冗余代码不方便管理，这时就该使用AOP。

第二层：

AOP底层是依赖JDK动态代理和CGLIB动态代理, 如果目标类实现了接口, 优先使用JDK动态代理, 否则切换成CGLIB, 就是基于目标类动态生成子类来代理。

### 面试题7.14 动态代理的两种实现方式，Spring的两种代理JDK和CGLIB的区别

**【技术难度： 出现频率： 】**

第一层：

Spring 是通过动态代理来实现AOP的，具体是这样切换：

1. 当Bean实现接口时，Spring就用JDK动态代理；
2. 当Bean没有实现接口时，Spring使用CGLIB；
3. 可以强制使用CGLIB（在spring配置中加入<aop:aspectj-autoproxy proxy-target-class="true"/>）。

第二层：

它们的原理是这样：

1. JDK动态代理是利用拦截器加上反射机制生成一个实现代理接口的匿名类，它只能对实现了接口的类生成代理；
2. CGLIB主要是通过修改指定类的字节码生成一个子类并覆盖父类方法生成代理；

它们的区别在于，JDK代理不需要第三方库支持，只需要JDK环境就可以进行代理，CGLIB依赖于第三方实现了CGLIB的类库。

----------------扩展(无需背诵)：

为什么JDK动态代理要求被代理对象必须实现一个接口？

因为JAVA是单继承多实现，而JDK动态代理类已经继承了Proxy这个类，所以只能通过接口来与被代理类建立联系。

### 面试题7.15 Spring的通知类型有哪些？

**【技术难度： 出现频率： 】**

有5种通知：

1. 前置通知，在目标方法之前运行；
2. 后置通知，在目标方法之后运行；
3. 异常通知，目标方法发生异常后运行；
4. 最终通知，在目标方法之后运行, 就算目标方法发生异常最终通知也会运行；
5. 环绕通知，等于以上四种通知。

### 面试题7.16 你用过AOP吗，它可以用来做什么？

**【技术难度： 出现频率： 】**

第一层：

AOP可以做日志，权限，事务等等，我在项目中用AOP做过日志：

比如记录谁在什么时间调用了哪个方法，这个方法执行的时长是多少，记录下来插入数据库。

具体是这样的，我通过aop的环绕通知，在目标方法执行之前记录用户的请求地址，请求方式，请求类名和方法，以及用户传递过来的请求数据，目标方法执行的开始时间。

当目标方法执行之后, 还记录了目标方法的返回结果, 目标方法的结束时间, 结束时间减去开始时间就是目标方法的执行时间, 也一起记录下来。

第二层：

AOP还可以做权限管理, 比如调用方法的时候, 用AOP技术在调用方法之前判断访问者有没有对应的权限, 有权限才放行；

AOP还可以做事务，Spring框架就是通过AOP去做的事务。

### 面试题7.17 在你的项目中是如何记录日志的？

**【技术难度： 出现频率： 】**

1. 日志框架log4j；
2. 使用Spring AOP，用环绕通知切入需要记录日志的controller层方法（具体参考上一题）

### 面试题7.18 有没有做过自定义注解？怎样实现自定义注解？

**【技术难度： 出现频率： 】**

第一层：

在Spring中使用自定义注解比较简单，只需要分为两步做：一、创建注解，二、编写切面编程处理注解。

第二层：

展开说是这样：

第一步，创建注解，这里分三小步来做：

1. 用@interface像写一个接口一样写一个简单的自定义注解；
2. 在自定义注解上用@Retention指定生命周期，比如运行时，并且用@Target指定使用范围比如在类和方法上有效；
3. 在自定义注解中写方法指定注解的属性名和属性类型，可以通过default设定默认值。

第二步，编写切面代码处理注解，这里也分三小步来做：

1. 写一个切面类，在类上用@Aspect、@Component向Spring容器注册；
2. 在类中写切面方法，在这些方法上用切面注解，比如@Before、@After、@Around，然后给这些切面注解的value属性指定@annotation(自定义注解名)，表示切入点是自定义的这个注解；
3. 在这些方法中写自己的业务代码，比如记录日志等等功能，通过方法参数就可以拿到自定义注解的代理对象。

第三层：

自定义注解是一个继承了Annotation接口的接口，我们通过@interface编写自定义注解，JDK编译代码会帮我们处理好细节，注解本身没有实际功能，需要通过反射获取目标对象上的注解，这会返回一个JDK动态代理对象，我们通过这个代理对象就可以拿到注解中的属性值执行所需操作了。

而通过AOP面向切面编程，以注解为切入点来编写代码对比反射去获取注解要方便很多。

### 面试题7.19 @Transactional(rollbackFor = Exception.class)注解了解吗？

**【技术难度： 出现频率： 】**

第一层：

@Transactional是声明式事务的注解，如果类或者⽅法加了@Transactional注解，那被声明事务方法中的若干条SQL语句就组成一个事务，如果这些⽅法抛出异常，事务就会回滚，让数据库回滚到事务执行前状态。

第二层：

异常Exception分为运⾏时异常和⾮运⾏时异常。在@Transactional 注解中如果不加 rollbackFor 属性，那事务只会在遇到运行时异常的时候才会回滚，而加上 rollbackFor=Exception.class ,可以让事务遇到非运行时异常也回滚。

### 面试题7.20 Spring 支持的事务管理类型

**【技术难度： 出现频率： 】**

1. 编程式事务，在代码中需要加入处理事务的逻辑，可能需要在代码中显式调用beginTransaction()、commit()、rollback()等事务管理相关的方法；
2. 声明式事务, 这是spring倡导的非侵入式的开发方式。声明式事务管理使业务代码不受污染，有两种常用的方式, 一是基于tx和aop命名空间的xml配置文件，二是基于@Transactional注解。

### 面试题7.21 Spring 事务中哪⼏种事务传播⾏为? spring事务传播特性有哪些?

**【技术难度： 出现频率： 】**

第一层：

有七种类型，分别是required、supports、mandatory、requires\_new、not\_supported、never、nested。它们被定义在枚举类Propagation中。

第二层：

它们分别是这样：

1. required表示支持当前事务，若不存在，新建一个事务，这是默认值；
2. supports表示支持当前事务，若不存在，不使用事务；
3. mandatory表示支持当前事务，若不存在，抛出异常；
4. requires\_new表示新建一个事务，若有事务存在，挂起当前事务；
5. not\_supported表示以非事务方式运行，若有事务存在，挂起当前事务；
6. never表示不使用事务，若有事务存在，抛出异常；
7. nested表示若当前事务存在，嵌套事务执行；

### 面试题7.22 Spring 事务中的隔离级别有哪⼏种?

**【技术难度： 出现频率： 】**

第一层：

TransactionDefinition 接⼝中定义了五个表示隔离级别的常量，分别表示默认的、读未提交、读已提交、可重复读、序列化。常量名分别是ISOLATION\_DEFAULT、ISOLATION\_READ\_UNCOMMITTED、ISOLATION\_READ\_COMMITTED、ISOLATION\_REPEATABLE\_READ、ISOLATION\_SERIALIZABLE。

第二层：

它们分别是这样：

ISOLATION\_DEFAULT表示默认的，使⽤数据库默认的隔离级别，MySQL默认是可重复读，Oracle 默认是读已提交；

ISOLATION\_READ\_UNCOMMITTED表示读未提交，这是最低的隔离级别，允许读取并发事务未提交的数据，可能会导致脏读、幻读或不可重复读；

ISOLATION\_READ\_COMMITTED表示读已提交，读取并发事务已提交的数据，可以避免脏读，但避免不了幻读、不可重复读；

ISOLATION\_REPEATABLE\_READ表示可重复读，一个事务中对同一条记录的多次读取结果都是⼀致的，除⾮是被自己这个事务所修改，它可以避免脏读和不可重复读，但避免不了幻读。

ISOLATION\_SERIALIZABLE表示序列化，它是最⾼的隔离级别，也是完全符合事务四个特性ACID的隔离级别。所有的事务依次逐个执⾏，这样事务之间就不会产⽣⼲扰，它可以防⽌脏读、不可重复读以及幻读，但是会严重影响程序的性能，一般不用这个级别。

## 模块8：MyBatis 框架

### 面试题8.1 Mybaits 的优点

**【技术难度： 出现频率： 】**

1. 能很好的与各种数据库兼容，并能很好的与Spring集成；
2. 直接编写原生态SQL，可以严格控制SQL执行性能，与JDBC相比减少了50%以上的代码量，不需要手动开关连接；
3. 基于SQL语句编程，SQL写在XML文件中，解除SQL与程序代码的耦合，便于统一管理并可重用，提供XML标签，支持编写动态SQL语句。

### 面试题8.2 MyBatis 框架的缺点

**【技术难度： 出现频率： 】**

1. SQL语句的编写工作量较大，尤其当字段多、关联表多时，对开发人员编写SQL语句的功底有一定要求；
2. SQL语句依赖于数据库，导致数据库移植性差，不能随意更换数据库。

### 面试题8.3 MyBatis的工作原理？MyBatis执行步骤?

**【技术难度： 出现频率： 】**

1. 加载配置文件mybatis-config.xml；
2. 创建会话工厂SqlSessionFactory；
3. 打开会话SqlSession；
4. 创建Mapper接口代理对象；
5. 执行SQL语句，将结果根据映射关系映射成所需对象；
6. 提交事务并关闭连接；
7. 返回执行结果对象；

### 面试题8.4 MyBatis动态mapper的实现原理? mapper接口中的方法能不能重载? MyBatis中通常一个Xml映射文件，都会写一个DAO接口与之对应，请问，这个DAO接口的工作原理是什么？？DAO接口里的方法，参数不同时，方法能重载吗？

**【技术难度： 出现频率： 】**

MyBatis的Mapper接口简称DAO接口。这个DAO接口的工作原理是JDK动态代理，MyBatis运行时会使用JDK动态代理为DAO接口生成代理对象，代理对象会拦截接口方法，转而执行对应的MapperedStatement所代表的SQL，然后将SQL执行结果返回。

DAO接口里的方法，是不能重载的，因为是全限定名加方法名的保存和寻找策略。

### 面试题8.5 MyBatis中$和#区别？#{} 和 ${} 区别？MyBatis 如何防止 sql 注入？

**【技术难度： 出现频率： 】**

第一层：

MyBatis使用#{}防止SQL注入。

${}和#{}的区别是这样：

1. ${}是直接取值替换，加入SQL的编译所以有SQL注入风险，#{}是占位符不加入SQL的编译，可以避免SQL注入；
2. ${}可以直接在SQL语句中的字符串内部使用，#{}不可以。

第二层：

展开说是这样：

MyBatis在处理${}时，直接把${}替换成变量的值做字符串拼接。

MyBatis在处理#{}时，会将SQL中的#{}替换为“?”号，然后使用PreparedStatement的set方法来赋值。

### 面试题8.6 MyBatis拦截器了解过吗，应用场景是什么？

**【技术难度： 出现频率： 】**

第一层：

MyBatis拦截器可以拦截某些方法的调用，我们可以选择在这些被拦截的方法执行前后加上某些逻辑，也可以在执行这些被拦截的方法时执行自己的逻辑而不再执行被拦截的方法。

它的实现方式是实现MyBatis的Interceptor接口，用@Intercepts注解来标注拦截的类和方法。

它的使用场景有很多，比如说这些：

1. 日志打印；
2. 分页处理；
3. 对查询结果的某个字段加密。

第二层：

拦截器的类型，总共有4种，按照执行的先后顺序分别是：

1. Executor，拦截执行器的方法；
2. ParameterHandler，拦截参数的处理；
3. ResultHandler，拦截结果集的处理；
4. StatementHandler，拦截SQL语法构建的处理。

MyBatis可以对这四个接口中所有的方法进行拦截。

### 面试题8.7 mapper.xml中有哪些标签

**【技术难度： 出现频率： 】**

1. 增删改查，insert、delete、update、select；
2. 动态SQL，where、set、trim、bind、if、choose、when、otherwise、foreach；
3. 映射管理，resultMap、id、result、association、collection；
4. SQL片段，sql、include；
5. 设置别名，typeAlias；
6. 开启二级缓存，cache​。

### 面试题8.8 MyBatis里XML怎么传数据

**【技术难度： 出现频率： 】**

第一层：

MyBatis接口向xml中传值是通过接口方法的参数传递，xml中接收对应的参数可以用#{}和${}来获取，并用在parameterType属性中定义参数的数据类型。

而数据从xml中返回则通过定义resultType，将SQL查询的结果映射成对应类型的对象返回给dao层接口作为接口返回值。

第二层：

当参数需要传递多个值有这四种方式可以处理：

1. 可以在dao接口中定义参数是加上@Param(name)给参数取个名称，xml中可用#{name}取值；
2. 可以用#{数字}来取值，如果传递三个参数，可以用#{0}，#{1}，#{2}来取值；
3. 将值放入一个map中，dao层接口参数传入一个map,xml中用#{key}取值。同时xml中parameterType声明为java.util.Map；
4. 传递一个对象：当参数传入的是一个对象的时候，对应的xml中应该指定parameterType为对应的对象。然后取值的话直接通过#{对象的属性}来获取。

当返回的字段名与对象属性名不一致时，可以通过resultMap标签处理结果映射关系。

### 面试题8.9 MyBatis 动态 sql 有什么用？执行原理？有哪些动态 sql？

**【技术难度： 出现频率： 】**

MyBatis动态SQL是根据表达式完成逻辑判断并动态拼接SQL的功能。具体是在mapper.xml映射文件中，以标签的形式编写。

MyBatis提供了9种动态SQL标签，where、set、trim、bind、if、choose、when、otherwise、foreach。

### 面试题8.10 trim标签 怎么用

**【技术难度： 出现频率： 】**

MyBatis的trim标签一般用于去除sql语句中多余的and关键字、逗号，或者给sql语句前拼接 “where”、“set”以及“values(” 等前缀，或者添加“)”等后缀。

使用很简单，通过它的属性就可以实现添加或去除。添加前缀、后缀用prefix和suffix属性，去除前后缀用prefixOverrides和suffixOverrides属性。

### 面试题8.11 bind是什么 有什么用

**【技术难度： 出现频率： 】**

bind是MyBatis的一个标签。bind标签通常用于处理模糊查询的字符串拼接，然后用#{}传参来防止SQL注入。

具体是这样用，在bind标签的value属性中按需求给目标参数前后拼接“%”的字符串，并且给这个bind标签设置name属性，给value中的SQL片段绑定名字，然后就可以在SQL语句中通过#{}调用这个名字来传参了。

### 面试题8.12 MyBatis的多对一和一对多如何实现?

**【技术难度： 出现频率： 】**

第一层：

多对一使用association标签，一对多使用collection标签。

第二层：

他们都有分步和单步两种查询模式：

1. 分步查询是用两条以上的SQL语句查询出所需数据，需要在association或collection标签中通过select属性指定集合属性的查询语句，通过column属性指定要查询的条件值；
2. 单步查询则是用一条SQL语句查询出所需的数据，可以节省数据库性能，单步查询需要给查询结果的列取好别名，在resultMap标签中将各个列分别映射到多对一或一对多的关系中，这种SQL语句通常会写得比较复杂。

----------------扩展(无需背诵)：

参考网址：https://blog.csdn.net/qq\_45974648/article/details/120355072

### 面试题8.13 MyBatis批量插入怎么用？

**【技术难度： 出现频率： 】**

有两种方式：

1. 通过原生的foreach标签实现批量插入，原理是用一条SQL语句实现多条数据的插入，这最快，缺点是语句太长时需要额外代码进行处理；
2. 使用mybatis-plus提供的批量插入saveBatch()方法，原理是每1000条插入语句拼接成一条SQL并执行，最后一起提交事务，速度其次。

### 面试题8.14 MyBatis怎么用悲观锁和乐观锁？什么是悲观锁、乐观锁, for update用过吗

**【技术难度： 出现频率： 】**

悲观锁是这样：

悲观锁在操作数据时比较悲观，认为别人会同时修改数据，所以锁住数据不让别人动。

当事务隔离级别是读已提交以上时，一个事务操作时会用悲观锁自动锁住相关数据，当这个事务提交时才释放锁，才允许其他事务对这些数据做删改操作。

在查询的SQL语句最后手写for update是手动加悲观锁的方式，数据库鼓励调整事务隔离级别，而不是手动加锁。

乐观锁是这样：

乐观锁在操作数据时非常乐观，认为别人不会同时修改数据，只是在执行更新的时候判断一下在此期间别人是否修改了数据。

乐观锁可以通过版本号version实现，当查询数据时，将数据版本号version一起查询，当执行update语句时，在update语句的where部分将自己的version和数据库中的version做匹配，匹配成功, 数据库的version加1，匹配失败, 更新失败。

### 面试题8.15 MyBatis中一级缓存与二级缓存？

**【技术难度： 出现频率： 】**

第一层：

缓存是指将数据保存在内存中，读取时无需再从磁盘读入，因此比较快。

MyBatis中的缓存分为一级缓存和二级缓存，它们具体是这样：

1. 一级缓存是默认开启，底层是HashMap，作用域是会话中，当会话刷新或关闭时被清空；

2. 二级缓存需要手动开启，缓存机制与一级缓存一致，作用域是命名空间Namespace，被多个会话共享。

第二层：

启动二级缓存需在配置文件中设置cacheEnabled为true，并在需要开启二级缓存的mapper配置文件中写<cache />标签。在这个cache标签中还可以设置返回的缓存对象策略等等。

缓存数据更新机制是这样，当某个作用域进行了增删改操作后，默认该作用域下所有查询的缓存将被清空。

第三层：

建议统一使用第三方插件来做缓存，比如redis，避免读取到脏数据，也方便搭建分布式系统。

一级缓存无法关闭，不想使用可以在select标签中加上useCache="false"属性。

### 面试题8.16 MyBatis 是否支持延迟加载？如果支持，它的实现原理是什么？

**【技术难度： 出现频率： 】**

第一层：

MyBatis延迟加载的原理是，使用CGLIB创建目标对象的代理对象，当调用目标方法时，进入拦截器方法，拦截器通过反射调用关联对象的方法发现是null值，那此时就去查询数据库获取关联对象，然后再完成关联对象方法的调用。不光是 MyBatis，几乎所有的持久层框架支持延迟加载的原理都是一样的。

第二层：

MyBatis只支持association关联对象和collection关联集合对象的延迟加载，在 MyBatis 配置文件中，可以配置是否启用延迟加载lazyLoadingEnabled=true或false。

### 面试题8.17 MyBatis 中的事务管理器?

**【技术难度： 出现频率： 】**

在 MyBatis 中有两种事务管理器类型：

1. JDBC，使用JDBC的事务管理机制，就是用Connection对象完成对事务的提交；
2. MANAGED，托管，由外部容器管理事务，默认情况下它会关闭连接，然而一些容器并不希望这样, 因此可以将closeConnection属性设置为false。

### 面试题8.18 mybatis-plus用过吗？怎么用？

**【技术难度： 出现频率： 】**

MyBatis-plus只是在Mybatis的基础上增强而不改变，使用它需要以下几步：

1. Maven导入相关依赖，配置好数据库连接信息、mapper接口所在包等等；
2. 通过Mapper-plus逆向工程生成工具生成实体类、mapper接口、mapper.xml文件等，包括Controller、Service层代码都可以一起生成；
3. 在service层注入mapper接口的代理对象，就可以调用mapper代理对象实现的BaseMapper的基础的增删改查方法，其中SQL语句的条件通过创建QueryWrapper对象进行设置。

## 模块9：SSM 框架开发

### 面试题9.1 什么是MVC？

**【技术难度： 出现频率： 】**

第一层：

MVC是一种开发模式，M表示Model指业务模型，V表示View指用户界面，C表示Controller指控制器，使用MVC的目的是将业务模型和视图的实现代码分离，使程序可以用不同的表现形式。

第二层：

MVC三个部分细说是这样：

1. Model业务模型，由数据和业务逻辑组成，目的是可以被调用并返回所需数据；
2. View视图，是指用户能看到并与之交互的界面，通常用jsp、html实现；
3. Controller控制器，负责连接业务模型与视图，通常用servlet实现。

第三层：

MVC开发模式优点是代码的结构层次清晰、耦合度低、可重用性高，缺点是一个应用程序分成了三个部分开发，增加开发工作量。

### 面试题9.2 Spring和SpringMVC的区别

**【技术难度： 出现频率： 】**

1. Spring是一个轻量级的IOC控制反转和AOP面向切面编程的容器框架；
2. Spring MVC是一个基于MVC开发模式的web框架，它的出现让web应用开发变得更容易而且可以很好的支持各种视图技术；
3. Spring MVC是Spring中的一个模块，依赖于Spring框架。

### 面试题9.3 什么是SSM框架？

**【技术难度： 出现频率： 】**

SSM是指Spring、Spring MVC、MyBatis组合成的框架集，Spring MVC是Spring框架的一个模块，因此可以说SSM框架由Spring、MyBatis两个开源框架整合而成。

### 面试题9.4 SSM框架的数据是怎样流转的

**【技术难度： 出现频率： 】**

SSM项目从页面发起请求，到服务器响应请求，整个流程如下：

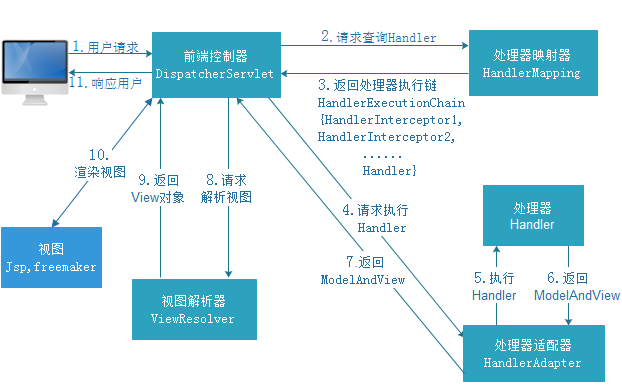
页面发起请求，Spring MVC根据请求路径匹配对应的Controller，Controller调用对应的Service业务模型层方法，接着Service调用MyBatis的DAO层接口方法，然后执行对应的MyBatis中的SQL语句，最后将结果逐级返回。

### 面试题9.5 SpringMVC的工作原理？SpringMVC架构流程、工作流程？

**【技术难度： 出现频率： 】**

1. 用户发送请求至前端控制器DispatcherServlet；
2. 前端控制器DispatcherServlet收到请求调用处理器映射器HandlerMapping；
3. 处理器映射器HandlerMapping根据请求URL找到具体的处理器Handler，生成处理器对象，如果有还会生成处理器拦截器HandlerInterceptor，一并返回给前端控制器DispatcherServlet；
4. 前端控制器DispatcherServlet调用处理器适配器HandlerAdapter；
5. 处理器适配器HandlerAdapter经过适配调用目标处理器Handler；
6. 处理器Handler执行完成返回ModelAndView；
7. 处理器适配器HandlerAdapter将执行结果ModelAndView返回给DispatcherServlet；
8. 前端控制器DispatcherServlet将ModelAndView传给视图解析器ViewReslover；
9. 视图解析器ViewReslover解析后返回具体View；
10. 前端控制器DispatcherServlet对View进行视图渲染，将模型数据填充至视图中；
11. DispatcherServlet响应用户。

----------------扩展(无需背诵)：



### 面试题9.6 Spring MVC拦截器的作用？拦截器和过滤器的区别？

**【技术难度： 出现频率： 】**

Spring MVC中的拦截器主要用于拦截用户请求并作相应的处理。例如通过拦截器可以进行权限验证、记录请求信息的日志、判断用户是否登录等。

过滤器和拦截器的区别是这样：

1. 过滤器Filter是Servlet规范中的一部分，任何Java Web项目都可以使用，可以对所有要访问的资源拦截；
2. 拦截器HandlerInterceptor是Spring MVC框架的，只有使用了Spring MVC框架的项目才能用，它只能拦截控制器Controller中的方法，如果访问的是jsp、html页面或其他静态资源是不会被拦截器拦截的。

### 面试题9.7 springmvc怎么做文件上传的

**【技术难度： 出现频率： 】**

页面中需要做这几步：

1. input标签设置为文件上传类型，type="file"；
2. form标签设为post请求方式，method = "post"；
3. form标签添加enctype属性，enctype="mutilpart/form-data"。

后台服务需要做这几步：

1. 控制器Controller的方法中声明MultipartFile类型参数接收上传的文件；
2. 获取一个UUID拼接上传文件的扩展名，作为新的文件名；
3. 准备上传路径, 如果路径不存在, 创建出来；
4. 文件上传；
5. 将新旧文件名存入数据库以便后续使用。

### 面试题9.8 什么是Restful

**【技术难度： 出现频率： 】**

restful就是一个资源定位及资源操作的风格, 在URL中只使用名词来定位资源，通过HTTP请求方式post、delete、put、get来实现资源的增删改查操作。

### 面试题9.9 权限模型, 用户权限怎么做，如何实现安全访问控制？你用过什么安全管理框架吗？请说说Spring Security？

**【技术难度： 出现频率： 】**

第一层：

可以用Spring Security来做，它是为Spring应用提供身份验证和授权等功能的安全管理框架，依赖于Spring容器，减少了为安全控制编写大量重复代码的工作。

第二层：

数据库用五张表做，用户表、用户和角色中间表、角色表、角色和权限中间表、权限表，其中用户表需要多加几个字段，比如locked表示是否锁定状态等等。

URL访问权限推荐通过XML或配置类配置，而不是用代码耦合度较高的注解配置。

第三层：

SpringSecurity的原理其实就是一个过滤器链，内部包含了提供各种功能的过滤器。它的身份、授权认证流程是这样：

1. 先是一个请求带着身份数据进来，其中的用户名和密码被身份认证过滤器UsernamePasswordAuthenticationFilter获取到，封装成Authentication对象表示当前用户；
2. AuthenticationManager身份管理器负责验证这个Authentication，简单说就是认证用户名和密码；
3. 后续会调用到UserDetailsService实现类的loadUserByUsername方法查询数据库，根据用户名从数据库获取用户数据进行比对，比对成功返回UserDetails接口的实现类对象，这一步可以把用户角色也获取出来存入其中；
4. 认证成功后，AuthenticationManager身份管理器返回一个填充用户数据的Authentication实例，里面填充了上一步返回的对象；
5. 接下来通过SecurityContextHolder安全容器上下文对象将这个Authentication保存到Security容器中，方便后面的其他过滤器使用；
6. 接下来鉴权过滤器Filtersecuritylnterceptor进行权限认证，通过SecurityContextHolder安全容器上下文对象获取Security容器中的Authentication，根据用户的角色信息判断是否有请求的URL的访问权限，如果有就放行，否则拦截。

### 面试题9.10 认证 (Authentication) 和授权 (Authorization)的区别是什么？

**【技术难度： 出现频率： 】**

Authentication认证是指身份认证，是验证身份凭据的，比如说⽤户名和密码，系统通过这个凭据认证用户身份，所以Authentication被称为身份认证或⽤户验证。

Authorization授权是指权限认证，发⽣在身份认证之后，它主要掌管我们访问系统的权限，⽐如一些特定资源只有Admin管理员角色才能访问。

### 面试题9.11 什么是AJAX？AJAX由什么组成？AJAX的优点？

**【技术难度： 出现频率： 】**

AJAX是异步的JavaScript和XML，用于在Web页面中实现异步数据交互，实现页面局部刷新。

优点是可以使得页面不重载全部内容的情况下加载局部内容，降低数据传输量，避免用户不断刷新或者跳转页面，提高用户体验。

### 面试题9.12 用什么发AJAX请求？

**【技术难度： 出现频率： 】**

1. JS原生AJAX（XMLhttpRequest）；
2. AXIOS；
3. JQuery。

### 面试题9.13 AJAX的核心对象, 有哪些重要属性

**【技术难度： 出现频率： 】**

1. AJAX的核心对象是XMLHttpRequest；
2. readyState==4 代表服务器响应结束, status==200 代表服务器响应成功。

### 面试题9.14 简述ajax的执行过程？

**【技术难度： 出现频率： 】**

1. 创建XMLHttpRequest对象；
2. 执行open()方法，设置请求方式，请求url，是否异步；
3. 执行send()方法发送请求；
4. 绑定事件，根据响应状态，4表示响应完成，和服务器响应状态，200表示响应成功，处理结果。

### 面试题9.15 AJAX请求时，如何将对象转成JSON字符串？如何解释JSON字符串数据?

**【技术难度： 出现频率： 】**

第一层：

1. 将对象转JSON字符串用JSON.stringify()；
2. 解释JSON字符串使用eval()或者JSON.parse()。

第二层：

通过第三方的AXIOS库发送AJAX请求会自动完成JSON对象与字符串之间的互相转换，非常方便。

### 面试题9.16 简述AJAX中JS脚本缓存问题该如何解决？

**【技术难度： 出现频率： 】**

这是比较常见的问题，当修改了js内容调试的时候并不能显示新写代码的结果，这是因为JS为了加速页面执行，当前页面会使用缓存保持当前调用相同的连接。

可以在连接地址后面增加一个随机函数来解决这个问题。

### 面试题9.17 同步和异步怎么理解？分别在什么情况下使用。

**【技术难度： 出现频率： 】**

同步和异步是这样：

1. 同步是指在提交请求、等待服务器处理、处理完毕返回结果这个过程中，客户端浏览器不能干任何事；
2. 异步是指请求通过事件触发、服务器处理，这时浏览器仍然可以作其他事情，然后某个时刻服务器处理完毕返回结果，触发回调函数执行。

当我们需要获取的内容只占整个页面很小一部分，或者需要不刷新整个页面的情况下更新一些数据时，可以考虑使用异步请求去做。

### 面试题9.18 什么是JSON

**【技术难度： 出现频率： 】**

JSON（JavaScript Object Notation）是指JS对象表示方法，是一种轻量级的数据交换格式。

优点是轻量级、人读写起来比较轻松、机器解析也很容易，支持复合数据类型，比如数组、对象、字符串、数字等。

### 面试题9.19 JSON与XML的区别

**【技术难度： 出现频率： 】**

1. XML可读性比JSON更好，但是编码、解码更复杂；
2. XML使用更广泛，JSON未来前景更好；
3. JSON比XML体积更小，传递速度更快，而且更容易被JS处理。

### 面试题9.20 什么是跨域，如何解决？

**【技术难度： 出现频率： 】**

第一层：

域由协议、域名、端口号组成，任何一个不一样都是不同的域，比如 https://cn.bing.com:443 和 https://cn.bing.com:80 是两个域。

发送跨域请求时，被请求的服务器能接收到请求，也能响应浏览器，但是浏览器会拦截数据，不会显示。

常用解决方案有JSONP和CORS两种，另外还有nginx反向代理技术。

第二层：

JSONP和CORS具体是这样的（讲下一题）

### 面试题9.21 什么是JSONP和CORS,它们的区别是什么？

**【技术难度： 出现频率： 】**

第一层：

JSONP和CORS是用来解决跨域访问的解决方案。

JSONP的原理是动态创建script标签，利用script标签不受同源限制的特性发送跨域请求，服务器端返回调用回调函数的字符串，通过参数列表实现数据传递。

CORS是跨域资源共享，需要在服务器端设置相关响应头，告诉浏览器不要拦截数据。

他们的区别是这样：

1. JSONP发送的不是ajax请求, CORS发送的是真正的ajax请求；
2. JSONP只支持get请求，CORS支持get、post等请求；
3. JSONP没有兼容问题, CORS有兼容问题，只有IE10和谷歌火狐等浏览器才支持。

## 模块10：Spring Boot 框架开发

### 面试题10.1 springboot是什么？请说说你对Spring boot的理解

**【技术难度： 出现频率： 】**

第一层：

Springboot设计目的是简化Spring项目的搭建和开发，整合了Spring和其他常用框架（Spring包含Spring MVC），导包后即完成自动化配置，我们只要写很少的配置即可运行。

第二层：

Springboot项目可选择内嵌tomcat等Servlet容器，默认使用tomcat，可选择打成jar包，因此可以独立的被JVM执行。

### 面试题10.2 springboot和ssm的区别

**【技术难度： 出现频率： 】**

一、使用SpringBoot的好处有这些：

1. 快速构建项目，对主流开发框架的无配置集成极大地提高了开发、部署效率；
2. 可选择内嵌Servlet容器独立运行，而无须依赖外部Servlet容器；
3. 其他还有提供运行时的应用监控、与Spring Cloud云服务天然集成等等；

二、如果不用SpringBoot，我们可以自主控制整合的各个框架版本，并可以根据自己的配置快速定位到问题，稳定版本的各个框架遇到问题时网上的解决方案更多。

### 面试题10.3 springboot如何定义多套不同的环境配置

**【技术难度： 出现频率： 】**

有两种方式：

1. 多个属性文件的方式，有.properties文件和.yml文件两种文件格式，它们都能通过文件名的命名形式区分；
2. yml文件多文档块的方式，在yml配置文件中，可以使用“---”（三个减号）把配置文件分割成多个文档块，在on-profile属性中指定配置名来区分。

### 面试题10.4 springboot自动装配的原理

**【技术难度： 出现频率： 】**

第一层：

自动装配是指由Spring自动把其他组件中的Bean装载到IOC容器中，不需要开发人员再去配置文件中添加大量的配置。开发人员只需导入相关依赖的jar包，并且在Spring Boot的启动类上添加 @SpringBootApplication注解开启自动装配接口就可以实现自动装配。

第二层：

具体是这样的，启动类注解SpringBootApplication中有三个注解，@Configuration标识配置类、@EnableAutoConfiguration开启自动装配、@ComponentScan扫描启动类所在包和子包中所有的类

重点看这个@EnableAutoConfiguration注解，开启自动装配：

它里面有个@Import注解，加载AutoConfigurationImportSelector类，该类实现了ImportSelector接口，这个接口有一个selectImports方法，该方法用来找到要自动加载的配置类对象，内部会再去调用一个getCandidateConfigurations方法用于获取要自动装配的对象的配置文件，这个文件是autoconfigure这个jar包的META-INF文件夹中的spring.factories文件。

最后加载spring.factories中的配置类，完成自动装配过程。

第三层：

并不是所有的配置类都会被加载，因为在这个jar包的META-INF目录中还添加了过滤配置文件（spring-autoconfigure-metadata.properties），此文件也会被读取以过滤无需加载的配置类。

具体是通过在过滤配置文件中写ConditionalOn开头的属性来添加过滤条件，比如ConditionalOnClass表示当目标类存在时才加载，也就是说当相关jar包导入到项目中时才会加载。另外还有在配置类上添加@ConditionalOn开头的注解的配置方式。

## 模块11：Git 管理项目

### 面试题10.1 用过什么版本管理工具？说下有哪些常用的操作？commit和pull有什么区别？

**【技术难度： 出现频率： 】**

用过git，常用这些命令：

1. git clone 克隆，从远程库拷贝项目到本地；
2. git add 添加文件到暂存区；
3. git checkout 创建或切换分支；
4. git commit 本地修改的文件提交到本地库；
5. git pull 拉取远程仓库最新版本到本地，并与本地合并；
6. git push 推送，上传代码到远程库并合并。

### 面试题10.2 请介绍一下如何添加远程仓库，那些字段是可以自定义的？

**【技术难度： 出现频率： 】**

第一层：

git remote add命令，后面跟两个参数，远程仓库名、远程仓库地址。

远程仓库名是自定义的，一般github用origin。

第二层：

如果执行了git clone命令，自动添加远程仓库，无需再git remote add。

### 面试题10.3 请介绍一下如何推送本地仓库到远程仓库？

**【技术难度： 出现频率： 】**

需要以下四步：

1. git remote 添加远程仓库地址；
2. git add 添加要文件到暂存盘；
3. git commit 提交到本地版本库；
4. git push 推送到远程仓库。

### 面试题10.4 push失败怎么办？push时版本冲突怎么解决

**【技术难度： 出现频率： 】**

第一层：

git push失败一般是版本冲突，此时需要手动解决，按以下三步走：

1. git pull 拉取最新的版本信息，拉取过程可能会产生冲突，手动处理冲突代码；
2. git commit 提交到本地版本库；
3. git push 推送到远程仓库。

到这时可能又会冲突，原因是这会有别人提交了，只能再重复这就几个步骤，直到push成功。

第二层：

还有一种可能性是本地和远程端的仓库各自独立，此前未产生关联，可以给git push添加参数--allow-unrelated-histories强制拉取合并。

### 面试题10.5 文件工作区被修改，没有添加到暂存盘，请问如何撤销？已添加到暂存盘又该如何撤销呢？

**【技术难度： 出现频率： 】**

1. 未添加或已添加到暂存盘都可用git checkout命令恢复目标文件；
2. 已添加到暂存盘还有git reset命令可用。

### 面试题10.6 如何恢复到上一个版本或指定版本

**【技术难度： 出现频率： 】**

第一层：

用git reset --hard命令。

第二层：

1. 恢复上一版本，后面再加个参数， HEAD^（Head后面有个上尖括号）；
2. 恢复指定版本，后面再加个参数，版本号。

### 面试题10.7 有些文件不能上传到远程仓库该怎么做？.gitignore是什么文件？

**【技术难度： 出现频率： 】**

第一层：

有时候我们不想把一些敏感文件或者缓存文件上传到远程仓库，这时可以在工作区根目录创建“.gitignore”文件，可以通过它配置git忽略上传的文件名单，ignore就是忽略的意思。

第二层：

在它里面可以配置按照具体的文件名、文件类型、目录这些方式来忽略目标文件。

### 面试题10.8 有个文件已经推送到远程仓库了，取消它的推送但本地不删除它，怎么做？

需要以下三步：

1. git rm 文件名 --cached，从缓存区删除文件，--cached表示在工作区保留文件；
2. git commit 提交到本地版本库；
3. git push 推送到远程仓库。

### 面试题10.9 git分支是什么，怎么用？

**【技术难度： 出现频率： 】**

分支就是从主线上分离出来做其它操作，从而不影响主线，做完分支的修改再合并到主线上。

分支相关命令有这些：

1. git branch 创建、删除分支；
2. git switch 或 git checkout 切换分支；
3. git merge 合并分支。

## 模块12：VUE 框架开发

### 面试题12.1 jquery和vue的区别？

**【技术难度： 出现频率： 】**

jQuery主要思想还是对dom进行操作，要先获取到dom对象，然后对dom对象做值的修改等操作。

Vue基于MVVM思想，做到了数据和视图完全分离开，不再需要繁琐的dom操作，通过Vue对象实现数据和视图的相互绑定。

### 面试题12.2 谈谈你对vue的理解?

**【技术难度： 出现频率： 】**

第一层：

Vue是一款轻量级框架，对比React，Angular等重量级框架，可以快速上手。

它运行时版本压缩得很小，并且实现了超快虚拟dom，减少对真实dom树的操作。

它是一套构建前端界面的渐进式框架，可在一个库和一套完整框架之间自由选择使用程度。

第二层：

VUE最大的两个特点是双向数据绑定和组件化开发：

组件化开发是说VUE中对于一个完整的页面可以拆分成多个组件，就像积木一样，由多个组件组成一个整体。

双向数据绑定是基于MVVM思想实现的，由Model数据层、View视图层、ViewModel中间层组成，通过ViewModel中间层进行双向绑定，由ViewModel实时监控着视图层和数据层的变化，当视图层或数据层任意一边发生改变时，ViewModel都会将数据变化动态传递到另一边。

### 面试题12.3 mvvm和mvc区别？

**【技术难度： 出现频率： 】**

第一层：

MVC和MVVM思想都是将View视图层与Model数据层分离，达到代码解耦合的目的。

它们分别是这样：

1. MVC由Model数据层、View视图层、Controller控制层组成，通常是这样，当视图层改变时触发控制层，控制层调用数据层，数据层返回结果，控制层再将这个结果更新到视图层；
2. MVVM由Model数据层、View视图层、ViewModel中间层组成，通过ViewModel中间层进行双向绑定，由ViewModel实时监控着视图层和数据层的变化，当视图层或数据层任意一边发生改变时，ViewModel都会将数据变化动态传递到另一边。

第二层：

MVVM由MVC演化而来，它们的区别有以下几点：

1. MVC只能实现单向通信，MVVM实现了双向通信；
2. MVC代码多、流程复杂、会有大量dom操作，MVVM无需手动操作dom、代码简单易维护；
3. MVC适用于系统架构，MVVM适用于单个页面中。

### 面试题12.4 v-model是什么？怎么使用？ vue中标签怎么绑定事件？

**【技术难度： 出现频率： 】**

1

v-model可以实现将数据双向绑定到元素的value属性。

2

v-model的使用方式只需要以下三步：

1. vue对象绑定要挂载的外部标签；
2. vue对象data属性中写自定义的属性，一般用方法返回值的形式来写；
3. 给外部标签中的input、textarea、select、radio、checkbox这些能获取用户输入的表单标签，加上v-model属性，值为上一步写的自定义属性。

3

绑定事件用v-on指令，使用方式也是三步走：

1. vue对象绑定要挂载的外部标签；
2. vue对象methods属性中写自定义的函数属性；
3. 给外部标签中的目标标签加上v-on属性，并用冒号拼接不带on字的事件名，比如v-on:click，可以用@符号简写成@click，值为上一步写的自定义函数名，可以按需求传参或只写函数名表示调用。

### 面试题12.5 说出几种vue当中的指令和它的用法？

**【技术难度： 出现频率： 】**

1. v-model：将数据双向绑定到元素的value属性；
2. v-bind：将数据单向绑定到元素属性；
3. v-text：将数据单向绑定到标签中间，按普通文本处理；
4. v-html：将数据单向绑定到标签中间，会解析html代码；
5. v-if：条件渲染，增删元素，与它关联的指令还有v-else、v-else-if，类似if分支语句；
6. v-show：条件渲染，显示或隐藏元素；
7. v-for：循环遍历数据，根据数据多次渲染生成元素；
8. v-on：给元素绑定事件和回调函数；
9. v-once：只编译一次，显示内容之后不在具有响应式的功能；
10. v-cloak：给标签设置样式，在vue实例编译结束时移除，通常用于在渲染结束前隐藏标签；
11. v-pre：跳过目标元素和它子元素编译过程，加快编译。

### 面试题12.6 v-if与v-show的区别？

**【技术难度： 出现频率： 】**

它们都是根据绑定属性的真假值显示或隐藏元素，

区别有以下两点：

1. v-if通过增删节点实现，v-show通过切换display样式实现；
2. v-if需要时加载能减轻服务器压力，但切换开销高，v-show切换开销低，但初始渲染开销大；

所以，需要频繁切换用v-show，否则用v-if。

### 面试题12.7 v-text和v-html的区别？什么是xss攻击？

**【技术难度： 出现频率： 】**

v-text不能渲染html代码，会以字符串形式显示。

v-html可以渲染html代码，但是这种方式存在安全隐患，简称XSS。

XSS是跨域脚本攻击，它的本质是注入恶意代码来达到非法目的。当v-html用在用户提交的内容上就会引发这个问题，所以vue官网建议只在可信内容上使用它。

### 面试题12.8 vue如何获取dom?

**【技术难度： 出现频率： 】**

按以下三步走：

1. vue对象绑定要挂载的外部标签；
2. 给外部标签中的目标标签写ref属性，值为自定义名称，比如ref="abc"；
3. 接下来就可以通过vue对象的$refs属性点自定义名称获取目标标签了，比如在vue对象内部使用this.$refs.abc就可以获取。

### 面试题12.9 vue有那些技术栈？

**【技术难度： 出现频率： 】**

vue.js 基础库

vue-router 路由

vue-vuex 组件状态管理器

vue-cli 项目构建工具，简称脚手架

vue-devtools 调试工具

vue-press 静态网站生成器，基于markdown语法生成网页

### 面试题12.10 说出vue.cli项目中src目录每个文件夹和文件的用法？

**【技术难度： 出现频率： 】**

1. assets文件夹，放静态资源；

2. components文件夹，放全局组件；

3. router文件夹，放路由相关配置；

4. store文件夹，放vuex组件状态管理器相关文件；

5. views文件夹，放所有视图，路由级别的组件；

6. App.vue文件，入口视图，根组件；

7. main.js文件，入口文件，加载组件、初始化等。

### 面试题12.11 请详细说下你对vue生命周期的理解？

**【技术难度： 出现频率： 】**

第一层：

总共分为8个阶段，创建前后、载入前后、更新前后、销毁前后。它们对应vue实例中的8个函数，分别在不同的时间点执行，其中更新前后的函数在每次data中的数据变化时都会执行，而其它函数只执行一次。

第二层：

展开说是这样：

1. 创建前/后：

beforeCreated创建前：此时vue实例的挂载元素$el和数据对象data都为undefined未初始化。

created创建后：此时vue实例的数据对象data有了，$el还没有。

1. 载入前/后：

beforeMount载入前：此时vue实例的$el和data都初始化了，data数据还未挂载到dom中。

mounted载入后：此时vue实例挂载完成，data数据成功渲染到dom中。

1. 更新前后：

当data变化时，会触发beforeUpdate更新前和updated更新后方法。

1. 销毁前后：

执行Vue实例的$destroy()销毁方法后会触发beforeDestroy销毁前和destroyed销毁后方法。

第三层：

页面加载时会触发前面四个函数beforeCreate, created, beforeMount, mounted。

mounted函数执行时，dom已经渲染完成，用户能看到初始化的界面了，所以通常在这里发一些ajax请求做数据初始化。

### 面试题12.12 请介绍一下key属性

**【技术难度： 出现频率： 】**

vue会复用已渲染的元素只改变元素中数据，而不是从头开始渲染，从而达到高效渲染的目的。

那对于相对独立的两个元素，不能让他们被复用的情况怎么办？这就需要给它们添加不同的key属性帮助vue进行区分，key属性的值要求唯一，类似id属性不可重复。

### 面试题12.13 axios是什么？怎么使用？描述使用它实现登录功能的流程？

**【技术难度： 出现频率： 】**

第一层：

axios用于发AJAX请求，具有体积小、支持浏览器和node.js、自动转换JSON数据等优点，当只需局部刷新页面时就可以用它请求服务器数据。

用它实现登录功能需要这样做：

如果是普通页面中使用，先用script标签引入axios库，然后用axios对象.post发post请求，将URL和请求数据封装成JSON对象在参数中发送即可，回调.then函数表示服务器响应成功，回调.catch函数表示响应失败。

第二层：

如果是前后端分离的项目，先npm install axios -s装好，然后在vue项目的main.js中import导入并通过Vue.prototype挂载到全局，还可以设置统一的访问前缀，然后就可以通过vue对象调用它发请求了。

第三层：

发送的如果是跨域请求，前后端分离项目可以在前端配置文件config/index.js设置开启虚拟服务器代理访问获取数据。

或者在后端做CORS配置，让浏览器允许跨域数据，Spring中可以通过WebMvcConfigurer接口实现类快速配置CORS。

### 面试题12.14 计算属性和 watch 的区别？

**【技术难度： 出现频率： 】**

它们分别是这样：

1. computed计算属性会将返回的结果缓存起来，只要数据没有发生变化, 回调函数就只执行一次；
2. watch属性是专门用于监听数据变化的，只要数据发生变化就调用对应的回调方法，watch属性还能监听路由地址的变化。

那怎么使用呢？

数据要经过复杂逻辑处理就用计算属性。需要在数据改变后执行一些业务逻辑，才用watch。

### 面试题12.15 路由之间跳转？

**【技术难度： 出现频率： 】**

第一层：

1. 声明式，使用router-link标签跳转；
2. 编程式，使用js代码this.$router.push()方法跳转

第二层：

this.$router还有go()和replace()方法可用，用法类似于history对象的这些方法。

### 面试题12.16 vue-router的两种模式？

**【技术难度： 出现频率： 】**

第一层：

前端路由的核心是改变视图时不向后端发出请求，为达到这一目的，浏览器提供了hash和history两种模式。

第二层：

它们分别是这样：

1. hash模式，表示可读可写的字符串，是指地址栏URL中“#”符号开始的部分，用来指导浏览器动作而不发送HTTP请求；
2. history模式，通过改变URL实现，利用了window.history对象HTML5新增的pushState()和replaceState()方法完成跳转。

第三层：

hisotry模式虽然更美观，不过存在一个问题，当用户通过路由切换页面后再手动刷新浏览器会发送请求，这会请求到不存在的URL，这时需要在服务器做一些额外配置来处理这个问题。

### 面试题12.17 导航钩子有哪些？它们有哪些参数？

**【技术难度： 出现频率： 】**

第一层：

1

vue-router的导航钩子，主要用来作用是拦截导航，让他完成跳转或取消跳转。

导航钩子函数有beforeEach、afterEach、beforeEnter、afterEnter、beforeRouteEnter、beforeRouterUpdate、beforeRouteLeave。

2

参数有to、from、next。

第二层：

1

导航钩子分为三个级别，全局的、单个路由的、组件的，它们分别是这样：

1. 全局钩子有beforeEach切换路由前、afterEach切换路由后；
2. 单个路由的钩子有beforeEnter、afterEnter，功能与全局钩子一致，在全局钩子之后运行；
3. 组件的钩子有三个，beforeRouteEnter、beforeRouterUpdate、beforeRouteLeave，它们分别是这样：
4. beforeRouteEnter在进入当前组件对应的路由前调用；
5. beforeRouterUpdate 在当前路由改变，但是依然渲染该组件时调用；
6. beforeRouteLeave 在离开当前组件对应的路由前调用。

2

3个参数分别表示：

to：目标路由；

from：离开的路由；

next：必须要调用这个函数才能去到下一个路由，或拦截路由，由参数不同执行不同动作。

## 模块13：Redis 数据库

### 面试题13.1 能说一下你对redis的理解吗

**【技术难度： 出现频率： 】**

Redis是开源的、在内存中运行的非关系型数据库，以key-value键值对的方式存储数据。

它的特点是高性能、可持久化数据、操作具有原子性、支持高并发读写、支持分布式等等。

大部分公司用Redis做数据缓存，用来提高数据查询效率。

### 面试题13.2 redis的数据类型有哪些

**【技术难度： 出现频率： 】**

第二层：

基本类型有5种：字符串string、列表list、哈希hash、集合set、有序集合zset（sorted set）。

第二层：

它们分别是这样：

1. 字符串string存储二进制数据，所以各种类型数据都可以存，最大512MB；
2. 列表list按插入顺序存储若干string，值可重复，类似java的LinkedList；
3. 哈希hash以key-value键值对的方式无序存储若干对string，key的值不可重复，类似java的HashMap；
4. 集合set以无序方式存储若干string，值不可重复，类似java的HashSet；
5. 有序集合zset会对存入的string排序，每个值要有个对应的score分数作为排序依据，值不可重复，类似java的TreeSet。

第三层：

其他还有几个类型：

1. 字符串string有个扩展类型bitmap位图，可以按位读写二进制数据。
2. 在2.8.9版本添加了HyperLogLog超级日志类型，它是根据输入元素来计算基数（不重复元素），而不储存输入元素本身，优点是占用空间很低，缺点是不精确。
3. 在5.0版本添加了Stream，这是redis专门为消息队列设计的数据类型。

### 面试题13.3 redis的持久化方式有哪几种，有什么区别

**【技术难度： 出现频率： 】**

第一层；

有两种：

1. RDB快照方式，以二进制方式存储到磁盘文件中，是redis默认持久化方式；
2. AOF日志方式，记录每一条命令的操作到磁盘中日志文件。

第二层：

它们的优缺点是这样：

RDB快照方式，可配置多少秒内多少个键更新就持久化一次，优点是恢复更快，但容易丢数据；

AOF日志方式，默认每秒记录一次，可改为有写操作就记录，优点是数据更稳定，缺点是恢复起来更慢。

----------------扩展(无需背诵)：

RDB - Redis DataBase - redis数据库

AOF - Append Only File - 仅附加文件

### 面试题13.4 在项目里面到底是使用RDB还是AOF(Redis的持久化机制), 你是怎么决定的

**【技术难度： 出现频率： 】**

1. 如果项目能接受一段时间的缓存丢失，就使用RDB；
2. 如果项目中实时性的数据比较重要，就用AOF；
3. 如果项目足够庞大，性能需求需要进一步拆分，可以配置多个redis数据库，分别用不同的持久化策略，根据数据的特性分别使用符合的数据库，不过这会提升项目的复杂度。

### 面试题13.5 介绍下redis的淘汰策略

**【技术难度： 出现频率： 】**

第一层：

redis提供了maxmemory最大内存设置其最多可占用的内存空间。当redis使用的内存超出最大内存时，默认情况下新的数据将无法写入。

所以redis提供了数据淘汰策略，当内存不足时删除一些数据以继续提供服务，redis提供了8种策略以供选择，比如淘汰最少使用的数据。

第二层：

可以通过expire命令给key设置多少秒的过期时间。

redis还提供了两种方案来删除过期数据，一种是定期删除，需要修改配置，这几乎没人用，因为数据量大时检查数据非常吃性能，第二种是内置的惰性删除，用到数据的时候才检查key是否过期，过期则删除。

### 面试题13.6 Redis是单线程还是多线程，为什么效率这么高？

**【技术难度： 出现频率： 】**

1

Redis采用单线程的模型，保证了每个操作的原子性，还减少了线程切换的性能开销。

2

redis效率高有以下几点原因：

1. redis是内存数据库，所以读写速度非常快；
2. redis的数据结构设计也加快了读写速度；
3. redis使用的是IO多路复用模型，实现了单线程服务端使用不阻塞方式来处理多个客户端请求，效率高而且吞吐量比较大。

### 面试题13.7 什么是Redis缓存雪崩、缓存穿透和缓存击穿？如何解决

**【技术难度： 出现频率： 】**

它们分别是这样：

1. 缓存穿透是指查询数据库一定不存在的数据，那缓存中肯定也没有，就会一直请求数据库；
2. 缓存击穿是指缓存中没有但数据库中有的数据，当有大量并发集中访问某个key时，在这个key失效的瞬间，比如缓存时间到期，这些大并发会直接请求数据库，像是在屏障上凿开了一个洞；
3. 缓存雪崩是指缓存中大批量数据过期了，而查询数据量巨大，引起的数据库压力过大甚至崩溃。

它们对应的解决方案如下：

1. 解决缓存穿透可以将空对象也缓存起来，并设置一个过期时间，在这个过期时间内可以有效保护数据库；
2. 解决缓存击穿可以使用布隆过滤器，有两种使用方式，一是导入jar包用后端代码实现，二是在redis 4以上版本集成布隆过滤器插件；
3. 缓存雪崩的处理要复杂一些：
   1. 设置热点数据永远不过期；
   2. 将产品数据分类，缓存不同周期，热门比冷门的缓存久一点，同类的用随机数分散周期；
   3. 缓存数据库如果是分布式部署，将热点数据均匀分布在不同的缓存数据库中。

### 面试题13.8 redis的应用场景？后端这边redis缓存是怎么用的呢，你在项目中怎么使用redis？

**【技术难度： 出现频率： 】**

主要有这些：

1. 作为缓存用在数据查询、新闻内容、商品内容等，string、hash、set都可以用；
2. 各种排行榜比如热门商品、应用排行榜，用zset；
3. 点赞统计、网站访问统计，string的increby命令可以增长点赞数；
4. 聊天室的在线好友列表、最新新闻列表等，用list就可以；
5. 任务队列，秒杀、抢购等等，用list，或者可以用redis5.0的stream类型；
6. 分布式会话，可以序列化用户的session会话对象保存缓存起来，在用户请求数据中发送sessionid来获取对应的数据反序列化，用hash类型；
7. 登录校验，缓存用户的JWT或Token数据，作为登录校验数据缓存起来，用hash类型，用在分布式架构中也很好；
8. 避免客户端重复请求，用set缓存客户端的ip、请求的接口等，下次先查缓存看是否重复请求，用hash类型。

### 面试题13.9 springboot结合jwt+shiro+redis实现token无状态登录授权

**【技术难度： 出现频率： 】**

第一层：

用springboot + spring security + jwt + redis整合实现，spring security比shiro更强大。

流程大致是这样：

1. 首先客户端发送用户名、密码到服务端进行认证；
2. 服务端查询数据库中的用户数据进行认证，
3. 认证通过后，服务端将用户id编码成一个JWT；
4. 接着将用户数据保存到redis，以用户id为key；
5. 然后把JWT发送给客户端，在以后客户端每次发送请求都在请求头中携带这个JWT用于认证；
6. 在之后的请求中，服务端解析客户端发来的JWT，根据里面的用户id查询redis，该数据存在则表示已登录，放行请求，若数据不存在则表示未登录。

第二层：

通过Spring Sercurity具体怎么做呢，这里主要抓关键点说一下：

1. 在Service层写login登录校验方法，内部调用身份管理器的authentication方法进行认证，认证通过就把返回的用户数据存入redis，之后拿用户id生成一个JWT放入响应对象中返回，认证失败就在响应对象中放入登录失败信息返回；
2. 在Controller层写登录接口方法，接收用户名、密码，调用上一步写的service层方法，然后在Spring Sercurity配置类中放行此接口和静态资源，让用户能够不登录访问；
3. 写一个JWT认证过滤器继承OncePerRequestFilter，从请求头中获取JWT，若有则解析出用户id去获取redis中的用户数据，获取成功则将用户数据封装为身份认证成功的AuthenticationToken放到Security的安全上下文容器中，以便于后面的其它过滤器进行认证，若JWT数据有问题或redis中无相关数据，则认证失败抛异常，无需往后运行。

第三层：

（结合SSM模块中Spring Sercurity的题目自由发挥，流程可以说得更细致）

## 模块14：Nginx 服务

### 面试题14.1 什么是Nginx，为什么要用，有什么优点

**【技术难度： 出现频率： 】**

Nginx是一款高性能Web服务器，目前一直稳居热门服务器第一，它主要有以下功能：

1. 正向、反向代理；
2. 负载均衡、分流；
3. 虚拟主机。

它的优点是跨平台、配置简单成本低、占有内存少、并发能力强。

### 面试题14.2 什么是正向代理，什么是反向代理

**【技术难度： 出现频率： 】**

正向代理是指客户端主动通过代理服务器访问目标访问任意网络资源，这可以隐藏客户端，服务器不知道谁在访问。

反向代理是指客户端不知道访问的是代理服务器，代理服务器接收用户请求去访问真实服务器，这可以解决跨域问题，还可以实现负载均衡。

### 面试题14.3 tomcat怎么配负载均衡

**【技术难度： 出现频率： 】**

第一层：

通过Nginx服务器反向代理客户请求，将请求分发给不同的服务器，实现负载均衡。

具体这样操作：

1. 在若干台服务器上搭建好tomcat，并部署项目；
2. 在另外用一台有公网ip或者做了内网穿透的服务器搭建nginx，在配置文件nginx.conf中的upstream中添加这些tomcat服务器的ip和端口号，server中的servername配置nginx监听的域名或ip，linsten配置监听的端口号，然后proxy\_pass设置代理给这个upstream即可；
3. 配置完毕，启动各个服务器，就可以通过这个公网ip的niginx访问到各个服务了。

第二层：

在upstream中还可以配置各个服务器的最大错误次数、等待超时时间，设置备用服务等等，做进一步的优化。

### 面试题14.4 Nginx 有哪些负载均衡策略

**【技术难度： 出现频率： 】**

第一层：

Nginx的upstream设置中支持5种分配方式，其中轮询、权重、ip\_hash、least\_conn这四种为Nginx原生支持的分配方式，第三方策略fair和url\_hash。

第二层：

它们分别是这样：

1. 轮循，默认方式，按照请求先后顺序分配到各个服务；
2. weight权重，轮询的加强版，可以通过weight指定访问的概率；
3. ip\_hash，基于客户端IP的分配方式，确保相同的客户端固定访问一个后端服务器；
4. least\_conn，把请求转发给连接数较少的后端服务器；
5. fair，响应时间短的优先分配，需要安装第三方插件；
6. url\_hash，按照url的hash结果来分配请求，使得每个url定向到同一个后端服务器，需要安装第三方插件。

## 模块15：RabbitMQ 消息队列

### 面试题15.1 为什么需要消息队列？消息队列有什么优缺点？

**【技术难度： 出现频率： 】**

消息队列类似缓存池，生产者将消息往里放，消费者就可以从中获取到消息做对应处理。

为什么需要消息队列？因为它有以下优点：

1. 异步化，提高响应速度，生产者将消息往队列里放就可以响应用户，无需等待处理结果；
2. 解耦合，提高系统稳定性，消息队列将业务解耦，一个业务崩溃或网络故障不影响其他业务；
3. 流量削峰，消息队列可以作为缓冲节点，避免高并发直接冲击目标业务而引发业务崩溃。

基于这些优点，消息队列能显著提升应用系统处理并发请求的性能，而且很适合分布式系统。

缺点是提升了系统复杂度，难以保证数据一致性，需要搭建镜像集群来保证消息队列的高可用性。

### 面试题15.2 消息队列应用场景？

**【技术难度： 出现频率： 】**

第一层：

典型的应用场景有这些：

1. 异步处理，比如注册成功需要发邮件和短信给客户，将这两个任务放到队列中就可以响应用户，这两个任务分别会在某个时刻被消费者执行；
2. 解耦合，比如订单服务和库存服务，订单服务收到订单，将订单保存到数据库并将订单消息发送到队列中就告知用户下单成功，库存服务监听到队列中的订单消息就根据订单操作库存，这个过程中两个服务通过消息队列实现了解耦合；
3. 流量削峰，比如秒杀活动，通过消息队列接收高并发的用户请求，对高并发做一定规则限制，可以设定超出限定量就不再接收更多请求，或跳转错误页面

第二层：

除了典型的这几种，其它还有很多使用场景：

1. 比如日志处理，日志服务监听队列中有日志消息就读取并保存；
2. 比如聊天室，多个客户端监听相同主题的消息，有就读取，需要就发送消息。

### 面试题15.3 如何保证消息的可靠性？

**【技术难度： 出现频率： 】**

第一层：

假设用的是RabbitMQ，可以这样做：

RabbitMQ服务端创建队列时设置队列支持持久化，并且生产者发消息时也指定消息是持久化的，这样消息一发到队列就持久化到MQ服务端磁盘中，即使MQ服务端崩溃也可以从磁盘中恢复消息出来。

生产者、消费者一般是通过RabbitMQ的confirm确认机制来处理，这是异步处理，性能高，分为自动确认、手动确认两种，需手动配置，默认是无应答。

第二层：

自动确认容易引发问题，是这样，当消费者方法正常执行完毕后，会对MQ发出ACK，MQ就删除消息，若方法出现异常，会对MQ发出nack，消息会重回队列。

第三层：

而手动确认是这样：

1. 生产者设置开启回调函数，并实现ConfirmCallBack、ReturnCallback两个接口的回调方法，当MQ服务器接收消息会触发这些方法表示消息是否到达目标队列；
2. 消费者这边如果消费如果消费成功，手动调用ACK方法返回成功信息，如果消费失败调用Reject或Nack方法拒绝此消息，通过参数选择让消息回到队列或抛弃。

在第2点的回到队列这里需注意，当MQ服务端配置了死信队列后，这时去的就是死信队列。

### 面试题15.4 什么是死信队列，应用场景？

**【技术难度： 出现频率： 】**

第一层：

在使用MQ的时候通常会设计两个队列：一个是核心业务队列，一个是死信队列。

这个死信队列专门用来处理异常情况，重要的业务，如果消费者处理消息失败了，并且是无法恢复的问题，就可以把这些消息放在死信队列中保存。

第二层：

这肯定是由于消费者这边的服务遇到了什么故障，消费者中可以监听死信队列，发出警报提醒后台人员及时处理，并监控相关服务，一旦服务恢复了，就从死信队列中取出消息放回核心业务队列中让消费者正常处理。

第三层：

设置死信队列需要在MQ服务端给队列配置参数名x-dead-letter-routing-key注册死信队列，这样当死信消息需要回到队列时就会替换它们的key名让它们来到死信队列。

### 面试题15.5 什么是延迟队列，应用场景？

**【技术难度： 出现频率： 】**

第一层：

延迟队列中的消息希望在指定时间得到取出和处理，比如考试系统，100分钟的考试时间，如果用户开始考试了一直未提交，在100分钟后需要自动提交，就可以使用延时队列来处理。

第二层：

RabbitMQ中是这样做的：

1. 给延迟队列绑定key名为x-message-ttl，以及对应的延迟时间，当队列中消息到达这个延迟时间还未处理就转为死信，转到死信队列；
2. 消费者中监控死信队列，监控目标Tag标签，当发现目标延迟消息出现，就做对应的处理。

所以，像刚才提到的定时提交答卷功能可以通过RabbitMQ很好的实现。

## 模块16：Spring Cloud 微服务

### 面试题16.1 微服务个人理解，为什么要用微服务架构？

**【技术难度： 出现频率： 】**

微服务架构风格就是将一个应用程序拆分成若干个小服务，服务之间通过轻量通信机制交互。

对比传统的单体应用来看：

1. 单体应用部署成本高、改动影响大，一次要部署整个项目，轻微改动都要重新部署，所以改动风险高、难以快速迭代更新。
2. 而微服务可以分开发布特定服务，因此部署成本、改动风险都很低，实现了快速交付需求、快速迭代更新、弹性伸缩系统功能等优点。

因此，微服务特别适合需要服务快速上线运行的市场，能及时把握住盈利时机。

### 面试题16.2 微服务cap是什么

**【技术难度： 出现频率： 】**

第一层：

CAP理论是说一个分布式系统最多只能同时满足一致性、可用性、容忍网络分区这三项中的两项。

第二层：

CAP分别是这样：

1. C表示一致性，指各服务数据一致；
2. A表示可用性，指服务可用度高；
3. P表示容忍网络分区，允许集群中节点形成不同的子集，这可以允许网络故障引发节点无法通信的问题。

第三层：

为什么说它们三个只能同时满足两个？

比如支付宝在光缆被挖断，网络出现故障的时候，分布式系统容忍网络分区，允许网络故障。这时支付宝工程师选择了保证数据一致性，那服务就无法再保证高可用，只能阻塞或停止服务，将各服务的数据保持一致了再继续提供服务。

所以说对于分布式系统只能实现CP、AP两种，满足CA的只会是单体应用。如何在CAP中取舍呢？没有定论，只能根据实际场景定夺，适合的才最好，就像分布式系统涉及钱财的只能选择舍弃服务高可用来保证另外两项。

----------------扩展(无需背诵)：

CAP：C - Consistency一致性，A - Availability可用性，P - Partition Tolerance分区容错性

### 面试题16.3 springboot和springcloud关系?

**【技术难度： 出现频率： 】**

SpringBoot是一个快速开发框架，用于开发单体应用。

SpringCloud通过SpringBoot开发一个个微服务，并整合管理这些微服务。

### 面试题16.4 Spring Cloud Alibaba有哪些功能？Springcloud核心组件有哪些? springcloud常用组件

**【技术难度： 出现频率： 】**

第一层：

核心组件有5个：

1. 服务注册中心，Nacos、Eureka、Zookeeper、Consul；
2. 配置中心，Nacos、Config；
3. 网关，Gateway、Zuul；
4. 负载均衡，Feign、OpenFeign、Ribbon；
5. 服务熔断、降级，Sentinel、Hystrix。

第二层：

还有这些常用组件：

1. 消息中间件，RocketMQ、RabbitMQ、Kafka；
2. 分布式事务，Seeta、TCC-Transaction、TX-LCN；
3. 分布式调用追踪，Sleuth与Zipkin搭配使用。

### 面试题16.5 什么是服务雪崩效应?

**【技术难度： 出现频率： 】**

第一层：

在分布式应用中存在服务雪崩效应，它是一种由于“服务提供者的不可用”导致“服务调用者不可用”，并将不可用逐渐放大的现象。

比如：A是服务提供者，B调用A，而C和D又调用B，如果A出现故障会导致B出现问题，并且将问题扩大到C和D，这就形成了服务雪崩。

第二层：

服务不可用的原因有很多，比如硬件故障、程序bug、缓存击穿、用户大量请求等等。

### 面试题16.6 什么是服务熔断？什么是服务降级？

**【技术难度： 出现频率： 】**

服务熔断、降级可以应对服务雪崩：

1. 服务熔断的作用类似保险丝，当某服务一定时间内的请求次数和错误率达到了熔断设定值，为了防止整个系统出现雪崩，就暂时拒绝该服务的用户请求，进行服务降级，当熔断后一定时间检测到该服务响应正常就恢复该服务的调用链路；
2. 服务降级是在某服务出现故障时返回一个备用信息，这样虽然提供的是有损的服务，但保证了系统的完整性。

Sentinel或Hystrix都可以做服务熔断、降级。

### 面试题16.7 什么是Spring Cloud Sentinel？

**【技术难度： 出现频率： 】**

Sentinel是面向微服务的熔断降级、流量监控组件，能够监控并保护微服务，是阿里开源的项目。

它承接了阿里巴巴近10年促销活动的核心场景，比如秒杀场景将突然流量控制在系统容量可以承受的范围等等。

### 面试题16.8 @LoadBalanced注解的作用

**【技术难度： 出现频率： 】**

它用来开启客户端负载均衡。

当使用RestTemplate进行远程服务调用时，只需要在返回RestTemplate的配置类方法上添加一个@LoadBalanced注解就可以开启负载均衡。

### 面试题16.9 Nginx与Ribbon的区别

**【技术难度： 出现频率： 】**

Nginx属于服务器端负载均衡，客户端所有请求统一交给Nginx服务器，由Nginx服务器实现负载均衡转发请求。

Ribbon属于客户端负载均衡，从eureka注册中心服务器端上获取服务注册信息列表，缓存到客户端，让后在客户端实现轮循负载均衡策略。

### 面试题16.10 Ribbon负载均衡算法有哪些

**【技术难度： 出现频率： 】**

有七种，分别是这样：

1. RoundRobinRule，轮询，默认方式；
2. RandomRule，随机；
3. WeightedResponseTimeRule，按响应时间分配权重，响应越快，选择概率越大；
4. BestAvailableRule，最佳可用的，选择并发量最小的服务；
5. RetryRule，重试，间隔一定时间重试，默认用轮循策略；
6. ZoneAvoidanceRule，区域回避，根据服务所在区域性能和服务可用性来选择；
7. AvailabilityFilteringRule，可用性筛选，过滤当前熔断的、用户连接数超过设定值的服务。

### 面试题16.11 分布式事务用过吗，怎么实现（seata）

**【技术难度： 出现频率： 】**

第一层：

用Seata组件实现，Seata为用户提供了AT、TCC、SAGA、XA四种解决方案：

1. AT模式更通用，数据弱一致，不侵入业务代码；
2. TCC模式高性能，数据弱一致，侵入业务代码；
3. SAGA解决长事务，数据最终一致，侵入业务代码；
4. XA模式性能差，数据强一致，不侵入业务代码；

第二层：

Seata有三个组成部分，协调方TC、发起方TM、参与方RM，TC以Seata服务器单独部署，TM、RM以Seata客户端形式集成在其他服务中运行。

以AT模式为例，具体实现步骤是这样：

1. 先搭建Seata服务器，做好数据库配置，创建三张表，以及注册中心、配置中心配置，然后启动Seata服务器，会根据配置连接到注册中心；
2. 各RM服务的数据库都添加undo\_log表，都配置使用DataSourceProxy数据源代理；
3. TM业务层方法使用@GlobalTransaction注解开启全局事务；
4. TM业务方法中RPC调用远程服务，RM的业务层方法添加@Transactional注解就能参与事务；
5. 各个RM数据库都添加undo\_log表，并通过DataSourceProxy数据源代理实现自动生成undo\_log记录，以及向TC注册分支事务；
6. TM根据是否发生异常或超时，告知TC提交或回滚全局事务；
7. TC通知RM各自执行提交或回滚操作，同时清除undo\_log。

第三层：

Seata分布式事务的执行流程是这样：

1. TM向TC申请开启一个全局事务，全局事务创建成功并生成一个全局唯一的XID，
2. TM开始RPC调用远程服务，XID在微服务调用链路中通过请求头传播；
3. RM接收到调用参与事务，并向TC注册分支事务，将其纳入XID对应全局事务的管辖，事务一阶段结束；
4. TM根据是否发生异常或超时的情况，向TC发起针对XID的全局提交或回滚决议；
5. TC调度XID下管辖的全部分支事务完成提交或回滚请求，事务二阶段结束。

----------------扩展(无需背诵)：

单词解释：

1.协调方，TC（Transaction Coordinator），事务协调器；

2.发起方，TM（Transaction Manager）,事务管理器；

3.参与方，RM（Resource Manager）,资源管理器。

四种模式对比参考网址：https://blog.csdn.net/weixin\_45525272/article/details/126668782

AT模式参考网址：https://www.cnblogs.com/chengxy-nds/p/14046856.html

### 面试题16.12 常见的分布式事务解决方案？

**【技术难度： 出现频率： 】**

第一层：

现成可用的有这些：

1. 阿里提供的Seata组件，开源，实现了几种模式应对不同业务场景；
2. 基于消息队列MQ实现，解耦合，数据最终一致；
3. TCC-Transaction，基于TCC事务解决方案的开源框架，数据弱一致；
4. TX-LCN框架，已停止维护。

第二层：

其它还有2PC、3PC（二阶段、三阶段提交协议），以及Paxos算法等等解决方案，可以找实现了它们的开源框架。

## 模块17：Linux 管理

### 面试题17.1 linux 常用命令

**【技术难度： 出现频率： 】**

方向上键可以翻看历史执行过的命令

tab键可以在输入开头字符后自动完成后面部分，比如输入文件名

CTRL + C 结束当前程序

ll 查看当前目录中所有文件，展示详细信息

​ls 查看当前目录中所有文件，只显示文件名

touch 创建空文件

find 查找文件

mv 移动或重命名文件

cp -r 拷贝文件或目录及其子文件到目标位置

pwd 查看当前路径

cd 切换目录

mkdir -p 递归创建多级目录

rm -rf 删除文件或递归删除目录中所有文件，慎用

less 以浏览模式打开文件, 可以上下移动调整显示内容

vi或vim 用文本编辑器打开文件

shutdown 关机或重启, 可定时

reboot 重启

### 面试题17.2 LINUX 文本编辑器怎么用？

**【技术难度： 出现频率： 】**

第一层：

先通过 vi 或 vim 命令用文本编辑器打开文件。

基本使用是 i 键可以进入编辑模式，修改完毕ESC退出编辑模式，输入 :wq 保存退出，或者 :q! 退出不保存。

第二层：

其它常用命令有：

yy 复制光标所在行，左边加数字可以复制多行，比如3yy复制3行

dd 剪切光标所在行，左边加数字可以剪切多行，比如3dd剪切3行

dw 剪切一个词

p 粘贴

u 撤销上次修改

第三层：

还有几个好用的命令或快捷键：

:set nu 显示行号

?或/ 向下或向上查找，后面放查找单词

CTRL + U 或 CTRL + B 前后翻半页

PageUp或PageDown 前后翻页

如果是用XShell软件，还可以用鼠标选中多个字符，Ctrl + Insert复制，Shift + Insert粘贴

### 面试题17.2 LINUX 查找用什么工具呢？

**【技术难度： 出现频率： 】**

1. find，实时查找文件，可以指定文件名、文件大小等条件；
2. grep，可以查找包含指定字符串的文件。

### 面试题17.3 Linux修改某个文件权限用什么命令?

**【技术难度： 出现频率： 】**

第一层：

chmod 修改文件权限，可以同时设置文件当前用户、所属组、其它用户的权限。

第二层：

其它还有修改所属组、所有者的命令chgrp、chown：

chgrp 修改文件所属组

chown 修改文件所有者和所属组

### 面试题17.4 Linux复制文件命令?复制文件夹以及子文件命令?

**【技术难度： 出现频率： 】**

cp -r 拷贝文件或目录及其子文件到目标位置，-r表示递归处理子文件。

### 面试题17.5 用springboot开发完一个程序，在LINUX怎么让它跑起来

**【技术难度： 出现频率： 】**

第一层：

假设是由maven管理的、默认内置tomcat的springboot项目，打jar包的方式这样做：

在maven的pom文件中指定用jar包方式打包，并用maven的package命令打包，然后就可以将jar包发送到Linux系统，通过java -jar xxx.jar命令运行，还可以用nohup命令在后台运行。

当然，linux中至少需要先安装并配置好和项目设置版本匹配的JRE或JDK。

第二层：

打war包的方式，可以在pom文件中设置取消导入自带的tomcat依赖，打成war包，发布到Linux的tomcat容器中，tomcat有冷部署、热部署两种方式。

# 第二部分：项目经验面试问题

## 样本一：

### 1.1 项目背景

针对公安的感知平台，核心内容不能讲太多，主要是提供扫描车辆信息进行查询的功能。

### 1.2 功能模块

比如说高速系统一些车辆，先抓拍，然后查询匹配，数据资源从第三方厂商拿过来，比如说这边的什么商汤的视频库还有什么大华的库里面拿，拿了之后，然后我们对抓拍的数据进行比对，然后根据这边本地公安系统然后做协查。

### 1.3 技术架构

用的SSM框架，前端有的项目用VUE、有的用到JSP，比较杂。

### 1.4 项目预算

这个我不太清楚。

### 1.5 开发团队介绍

当时的话我们有六七个人，有2个前端，然后有4个后端，另外还有一个项目经理。测试的话是我们提交过去，是另一个部门了，不在我们这边，当然我们自己也会写单元测试先自测。

### 1.6 开发周期

这个项目是2022的3月份开始，然后一直到年底。现在的话主要是在维护，可能现在需求比较少，然后一边维护一边等需求吧。

### 1.7 具体负责工作

负责的话，因为我刚开始是进来实习的，然后主要看他们怎么做怎么写的，后来慢慢的就是写一些小的功能、小的需求嘛。

就是组长分配给我，你比如说一些页面，当时因为我不知道他这个项目，是因为当时接触的项目有一个是前后端不分离的，前端用的是JSP，然后他让我写一个前端的JSP页面和后端进行交互，然后我当时是边学边写这样完成的。

然后数据表都是接好的，我们有专门的人负责，后期还有的数据是从别的地方调过来的，因为它有些数据我们这边是接触不到的。

### 1.8 开发中遇到哪些问题、如何解决、有没有更好的解决方案

印象比较深的话，当时是请求后台的数据，然后拿到页面的时候它格式是乱码的格式，开始我以为是字符编码的问题，当时改了4次编码，然后也改了数据请求的格式嘛，当时都改了个遍，没用，然后就赶紧请教同事，我同事讲，你把它都统一转成JSON格式，传过来试一下，那我就这样试，然后通过了。

### 1.9 会去优化或者说重构自己的代码吗

这个会的，如果时间充足的话，我挺感兴趣的，这也是对思维的另一种开发模式嘛。就是写过之后，感觉可以换一种逻辑方式，同样可以实现此功能，就优化一下。会回头看看自己写的代码啊，再改改这样。

### 1.10 面试录音文字版

**张先生，27岁，南京，3年经验**

说话人1 00:01

喂？

说话人2 00:02

你好。

说话人1 00:03

你好，请问是张先生吗？

说话人2 00:05

对。

说话人1 00:06

你好，我是思诚科技公司的，我们公司在招Java开发，不知道你有没有兴趣。

说话人2 00:09

哪一个公司？

说话人1 00:11

思诚科技。

说话人2 00:35

哦哦哦，好的。

说话人1 00:37

好的，您现在方便吗？

说话人2 00:40

我现在方便的。

说话人1 00:41

嗯，好的，那我问你几个问题噢。

说话人2 00:43

好的。

说话人1 00:45

好的，您做过哪些项目啊？

说话人2 00:47

项目，一些学习的项目算吗？还有一些在公司里面用的项目。

说话人1 01:04

都可以的。你讲一下，您印象比较深的吧。

说话人1 01:06

在网上学的话，有一个学堂就是一个尚硅谷的学堂，然后学堂网站，还有一个是黑马的一个外卖系统，然后在公司的话，然后做过数据处理，然后做过感知平台。

说话人1 01:29

那您印象比较深的是哪个呢。

说话人2 01:32

印象比较深的？就是之前的那家公司做的一个感知平台，然后其实因为我是上一年毕业的，然后没有接触到什么太深奥的东西，然后做一些简单的增删改查吧，也是一些数据的增删改查转换。

说话人1 01:56

不好意思。我没听清楚，刚刚是什么平台来着。

说话人2 01:59

感知平台。

说话人1 02:00

哦，这个项目它是有哪些模块呢，它是做什么呢？

说话人2 02:10

比如说高速系统一些车辆的就是拍照嘛，然后查询，就是那种。

说话人1 02:24

嗯，拍照查询，然后是为了什么功能呢？

说话人2 02:26

因为这个是针对于公安的，然后核心的内容不能讲太多。

说话人1 02:33

好的，那我知道了。它里面有哪些模块呢，功能模块？

说话人2 02:40

功能模块就是可以查询然后匹配，然后数据，它的数据资源我们是从第三方厂商拿过来的，比如说这边的什么商汤的视频库还有什么大华的库里面拿的，拿了之后，然后我们对抓拍的数据进行比对嘛，然后根据这边本地公安系统然后做协查吧。

说话人1 03:13

那这些模块是怎么交互？

说话人2 03:16

有前端和后端，前端的话，我们有专门的就是前端VUE人员，然后做前端开发，然后话有后端人员做数据处理。

说话人1 03:26

比如说它是一个设备去扫描到了一个车辆信息，然后再去查询这样的调用关系是吗？对，还有更更深一点的关系吗？

说话人2 03:41

更深的话我暂时没有接触到。

说话人1 03:46

好的。那这个项目它的预算是多少？

说话人2 03:50

这个我不太清楚。

说话人1 03:54

好的，你没有管这块事。好的，那项目中它用到了哪些技术呢？

说话人2 04:01

它这个项目用的比较原始，他用的ssm框架，然后前端用了一些vue，然后还有一些他用了原始的，因为这里面所有的项目都是一样的，都是一样写的，有的项目用vue，然后有的用到jsp，比较杂嘛。

说话人1 04:28

嗯，好的，那有多少人做这个项目呢？

说话人2 04:32

当时的话我们有六七个人。

说话人1 04:36

嗯，人员是怎么分配的？

说话人2 04:39

有2个前端，然后有4个后端。

说话人1 04:43

还有呢？

说话人2 04:46

项目经理的话是有一个项目经理。

说话人1 04:50

好的。没有测试人员吗这样？

说话人2 04:54

测试的话是我们提交过去，然后测试是另一个部门了，就不在我们这边。

说话人2 05:01

有另外一个测试部门专门负责测试。

说话人2 05:03

对对对，它不和我们在一起。

说话人1 05:07

那您会做自测这些是吗？

说话人2 05:10

会啊，就是写单元测试是吧？

说话人1 05:14

嗯嗯，对。

说话人2 05:16

这个会啊。

说话人1 05:17

都会噢，那您这个项目开发了多久啊？

说话人2 05:21

嗯……这个项目是去年的3月份，然后一直到年底。对，现在的话还是在，可能现在需求比较少，然后一边维护一边等需求吧。

说话人1 05:38

就主要开发周期大概是7个月左右，然后后面一直在维护

说话人2 05:42

对对对。

说话人1 05:43

好的，那您在这个项目里面是负责哪些工作呢？

说话人2 05:48

负责的话，因为我刚开始是进来实习的，然后主要听取然后看他们怎么做怎么写的嘛，然后后来慢慢的就是写一些小的功能、小的需求嘛。

说话人1 06:01

就是组长分配给你？

说话人2 06:03

对对对，你比如说一些页面，当时因为我不知道他这个项目，是因为当时接触的项目有一个是前后端不分离的，还有前端用的是JSP，然后他让我写一个前端的JSP页面和后端进行交互，然后我当时也是边学边写，然后就这样的。

说话人1 06:29

包括建表什么的，也会让您自己去操作吗？

说话人2 06:33

数据库建表吗？

说话人1 06:35

对。

说话人2 06:36

那个不会，那个数据都是接好的，对，那个数据的话是我们有专门的人，后期还有的数据是从别的地方调过来的，因为它有些数据我们这边是接触不到的。

说话人1 06:49

就是说您目前负责的还是一些小的功能点噢，好的，那您开发中遇到印象比较深的问题有吗？是怎么解决的？

说话人2 06:59

印象比较深的话？当时就是字段转换嘛，可能刚来，然后不太清楚他用的自动转换那个模式是怎么转过去的，然后和同事一起交流的，然后请教的嘛，然后最后它是转成了一个以JSON的格式然后转出来的。

说话人1 07:29

字段转换转换成了一个JSON格式，您是说您去请求后台的数据，然后拿到页面的时候格式存在一些问题。

说话人2 07:46

对它格式是乱码的格式，然后当时的话是一般来说我感觉我以为他会做一个统一的处理，就是处理好之后的嘛，然后我就直接调嘛，调了之后然后转到前端的时候一看页面的话，它是一个乱码的格式。

说话人1 08:04

哦，是字符编码的问题是吧？

说话人2 08:10

嗯，对。当时改了4次编码，然后也改了数据请求的格式嘛，然后都改了遍，然后没用，没用之后，然后我同事讲，你把它都统一转成JSON格式，然后转过来试一下，然后就这样试，然后通过了。

说话人1 08:25

您没有用JSON之前您是用了什么？

说话人2 08:33

它自带的直接调的那种，我也不知道它原始的是什么。

说话人1 08:35

哦，就之前的数据你也不太清楚。

说话人2 08:39

对。

说话人1 08:41

就把数据转换成JSON格式，再发到前端页面是吧？

说话人2 08:45

嗯，对。

说话人1 08:47

好的，就是您这个问题是请教同事然后来解决的。

说话人2 08:50

对对对。

说话人1 08:51

嗯嗯，还有没有更复杂一点的问题呢？

说话人2 08:53

复杂的，暂时还没有诶。

说话人1 08:58

哦，好的。好，那您会去优化自己的代码吗，比如说有些代码我觉得它性能可以做得更好啊之类的。

说话人2 09:06

这个会的，如果时间充足的话，这个是会的。

说话人1 09:15

您是自己驱动自己去做，还是组长有要求之类的？

说话人2 09:19

自己会自己看吧，然后一般写过之后，然后感觉可以换一种逻辑方式，然后同样可以实现此功能，优化一下，有时候自己也会看自己写的代码啊，也会改啊。

说话人1 09:35

就是您也觉得挺有兴趣的。

说话人2 09:38

对啊。这也是对思维的另一种开发模式嘛。

说话人1 09:44

好的，您在这个公司是做了多久？

说话人2 09:49

从毕业到现在？

说话人1 09:51

那也挺长的，大概有多久？

说话人2 09:57

半年吧

说话人1 09:59

半年吗？哦，那您之前您写的这几个项目是学习项目噢。

说话人2 10:02

对，学习的项目。

说话人1 10:06

好的，看到了。好的，我这边就先了解到这里。

说话人2 10:09

哦好。

说话人2 10:10

如果后续有消息的话，会有其他同事联系您。

说话人2 10:25

好的。

说话人1 10:26

那打扰您了，谢谢，再见。

## 样本二：

### 2.1 项目背景

趣动，它是一个主打运动、健身的一个项目。

### 2.2 功能模块

里面有卖健身课程，然后卖一些体育器材啊，还有一些记录你每天的步数之类的，大概就是这些，还有一些支付之类的。

### 2.3 技术架构

用到的， 像比较常见的 Spring Boot，Spring Cloud 还有 MyBatis，还有 RabbitMQ吧，还有一些Spring Cloud里面的一些中间件，模块之间的调用一般都是用 feign 或者用 rabbitmq。数据库用的是MySQL。还有Redis也在用。

### 2.4 项目预算

预算啊？这个项目不是我接的，我怎么知道预算是多少。

### 2.5 开发团队介绍

我们那个组的话好像是有十几个的样子吧，Java开发的话大概是有四五个的样子，然后前端是一两个、两三个的样子。然后的话，有比如说安卓的，还有一个组长吧，大概就这些吧，然后有的时候会有实习的来。其他像测试的话，一两个的样子吧，然后UI的话，也就是一两个的样子。

### 2.6 开发周期

开发周期四五个月，然后后续还一直在维护。

### 2.7 具体负责工作

等我进去的时候那个项目已经做完了，就是说我一般在里面就是搞搞维护，增加增加功能，做一些后端的开发。

### 2.8 开发中遇到哪些问题、如何解决、有没有更好的解决方案

遇到印象比较深的问题……这个项目去年的我也记得不是很清楚了。我想一下啊，就是我做那个抢红包的模块的时候说，一开始我用的是那个分布式锁嘛，然后的话发现抢的时候抢的效果不是很理想……比如说抢红包的话抢的话，如果很多人抢的话就是说很慢吧。后来我把它就是用一个缓存嘛，就是用redis，然后搭配它的Lua脚本进行一个无锁化的抢红包，这样就可以提升它的一个性能。就是在网络上摸索查阅资料想办法嘛。

### 2.9 会去优化或者说重构自己的代码吗

公司要求的话肯定是会去优化的。一般的话，如果说性能足够这个项目使用的话，一般就不会去动。

### 2.10 面试录音文字版

**李先生，24岁，赣州，4年经验**

说话人1 00:00

你好，请问是李先生吗？

说话人2 00:02

嗯，你哪位？

说话人1 00:05

你好，我是思诚科技公司的，我们公司在招Java开发，看到你有投递简历，不知道你有没有兴趣。

说话人2 00:25

行，方便。

说话人1 00:26

好的，那我问你几个问题噢。

说话人1 00:29

好。

说话人2 00:30

就是你做过哪些项目？

说话人2 00:38

做过哪些项目？你有看我简历吗？

说话人1 00:39

有看一下。

说话人2 00:42

简历上面的那些项目我都有做过，然后还有兼职做过一些项目。

说话人1 00:56

那你印象比较深的是哪个呢？

说话人2 00:58

印象比较深的？就是之前在那个公司最后做的趣动的项目。

说话人1 01:15

它这里面有哪些功能模块呢？

说话人2 01:19

里面的话主要就是一些它本身是一个运动，就是说健身的这样一个项目，然后的话里面有这些，比如说卖健身教程，然后一些体育器材啊，还有一些就是说你每天的步数之类的，大概就是这些，还有一些支付的之类的。

说话人1 01:56

嗯，那这些模块他们是怎么交互的。

说话人2 02:00

怎么交互的，对。

说话人1 02:07

比如说谁调用谁之类的。

说话人2 02:08

模块之间的调用一般都是用比如说用feign或者用 rabbitmq啊。

说话人1 02:22

嗯，就这些模块之间的关联关系能说一下吗？

说话人2 02:30

关联关系？我不是很懂啊。

说话人1 02:38

就是您负责的模块可能也不太多的样子吗？

说话人2 02:40

是这样，我不是很懂你这个问题问的是啥。

说话人1 02:44

好吧，那算了，那下一个。这个项目它的预算是多少？

说话人2 02:49

预算啊？这个项目不是我接的，我怎么知道预算是多少。

说话人1 03:10

就是您没有管这块事，好的，那这个项目中用到了哪些技术呢？

说话人2 03:16

用到的，就是说比较常见的 Spring Boot，Spring Cloud 还有 MyBatis，还有 RabbitMQ吧，还有一些Spring Cloud里面的一些中间件，还有Redis，数据库用的是MySQL。

说话人1 03:49

好的，有多少人做这个项目？

说话人2 03:55

多少人做这个项目？

我们那个组的话好像是有十几个的样子吧，但是有的时候会有实习的来。

说话人1 04:10

人员配比大概是怎么样？

说话人2 04:11

Java开发的话大概是有四五个的样子，然后前端是一两个、两三个的样子。

说话人3 04:22

然后的话，有比如说安卓的，还有就是说，还有一个组长吧，大概就这些吧，然后的话还有比如说那个实习的啊。

说话人1 04:45

那有测试人员吗？

说话人2 04:49

有有有。

说话人1 04:51

也有一部分，那测试人员占比多少？

说话人2 04:57

测试的话，一两个的样子吧。

说话人1 04:59

还有UI吗？

说话人2 05:11

UI的话也就是一两个的样子。

说话人1 05:20

好的，那这个项目它开发了多久？

说话人2 05:23

开发的话，现在就是说还有在维护中，然后的开发的话大概是四五个月的样子吧。

说话人1 05:38

开发周期四五个月，然后后续还一直在维护。

说话人2 05:43

对

说话人1 05:44

好的，那您在这个项目中是负责哪些工作呢？

说话人2 05:51

等我进去的时候那个项目已经做完了，就是说我一般在里面就是搞搞维护，增加增加功能。

说话人1 05:59

嗯，就是根据新增的需求去做一些开发，做一些后端的开发噢？

说话人2 06:05

对对对。

说话人1 06:06

好的，那你开发中遇到印象比较深的问题，可以说一下吗？你遇到了什么问题怎么解决的？

说话人2 06:15

遇到印象比较深的问题……这个项目去年的我也记得不是很清楚了。

说话人3 06:32

我想一下啊，就是我做那个抢红包的模块的时候说，一开始我用的是那个分布式锁嘛，然后的话发现抢的时候抢的效果不是很理想，然后的话……

说话人1 07:05

嗯，是存在什么情况呢？

说话人2 07:07

比如说抢红包的话抢的话，如果很多人抢的话就是说很慢吧。

说话人1 07:24

嗯嗯，那您是怎么解决这个问题呢？

说话人2 07:26

后来我把它就是用一个缓存嘛，就是用redis，然后搭配它的Lua脚本进行一个无锁化的抢红包，然后的话提升它的一个性能。

说话人1 07:42

嗯，那这个性能得到了改善？

说话人2 07:45

对啊

说话人1 07:46

好的，您是怎么想到这个解决方案的？

说话人2 07:54

嗯……这个，就是说，有在网络上摸索嘛，然后的话……

说话人1 08:04

就主要还是自己查阅资料？

说话人2 08:05

对啊。

说话人1 08:06

好的，您会去优化自己的代码吗？

说话人2 08:20

这个，公司要求的话肯定是会去优化的。

说话人1 08:25

您自己会驱动自己去做吗？比如说有些代码我觉得它性能可以做得更好啊什么的。

说话人2 08:32

这个的话，一般的话，如果说性能足够这个项目使用的话，一般就是不会去动。

说话人1 08:40

就没有这个需求，没这个需要了噢？

说话人2 08:43

对

说话人1 08:44

好的，我这边就先了解到这里哈，后续有消息的话可能会有其他同事联系您。

说话人2 08:49

嗯，好好。

说话人1 08:51

好的，打扰您了，谢谢，再见。

说话人2 08:55

好，谢谢。

## 样本三：

### 3.1 项目背景

这个项目叫房抵贷，意思就是说把房屋抵押给银行，然后来拿到资金，但是有一些房屋银行是不会要的，比如那种很老的房子银行它是不会要的，所以让我们公司来给他做担保，然后拿凭证，意思就是我们实际上是处于一个担保方，然后大概就是说从一开始业务人员去接收到去填写房屋的信息，然后经过那种什么预审、复审、终审，然后进入到了尽调，就是尽职调查，然后就会业务人员就会到房子那里去实地考察，实地考察，然后把信息再反馈到系统里面去，然后这个系统就再经过一些审批，然后再跟银行联调，跟资金方联调，然后开始汇款打款，然后生成那种清分计划，就是每个月还多少钱之类的，什么到款，然后再清分收益，大概是这么个流程。

### 3.2 功能模块

首先一个项目最基本的那种什么权限模块，这个是肯定有的了，最基本的RABC那种吧，就是那种目录，就是那种角色对应的什么人员，对应组织，对应的目录对吧？这个最简单的，这个是最底层的权限控制管理。然后再往上面就有一些那种业务系统，主要是负责业务数据录入数据的，然后跟它对应的还有一个是OA系统是用来负责审批的，它们之间可以通过接口或者通过或者通过IMQ进行交互。然后这大概是两个，然后还有一部分就是跟银行对接的系统，跟银行对接方面的接口，就大概分成了这么几块。

### 3.3 技术架构

Spring cloud alibaba那一套，mybatis那一套，然后注册中心那种什么nacos喽，mysql喽，然后rabbitmq、redis反正基本上都用到了吧，那一套，然后还有一些那种外部的，还有nginx，什么skywalking、什么elk分布式日志收集这些都是有的。

### 3.4 项目预算

预算是多少？嗯……那我写代码的，我怎么知道预算是多少。

我肯定不关心哪，我加班还来不及了。

### 3.5 开发团队介绍

人员一直都在变，多的时候有三四个，少的时候就一两个。

人员配比的话大概是一个前端对两个后端，然后对一个测试。反正比例是这个比例哈，可能有2个前端4个开发2个测试的也有，大概就这个比例。

### 3.6 开发周期

有半年哦，迭代很快的，一直在迭代。

### 3.7 具体负责工作

主要是负责后端，然后大概像那种尽职调查那一块，我就做的比较多，然后还有跟OA审批流交互那一块我也做的比较多吧。审批流是有他们人去做的，反正我也是，我大概知道一点吧，做我也能搞两下子。

审批流他们用的是folowable，什么流程预测那些，反正我只要把那个节点里面的信息人，还有资金，反正必要的参数传给他，然后他们就会流程预测出来一条线，然后自然会把那些人给我，然后后面该怎么流转怎么流转，但是我负责的是业务数据那一边。你像比如他相互尽调了以后，那些什么视频照片，那些流处理那些乱七八糟的，还能够下载上传到mirror那些。

### 3.8 开发中遇到哪些问题、如何解决、有没有更好的解决方案

印象比较深，怎么说呢，你像那种正常的，正常的那些问题，反正你也知道吧，该有都会有的正常的一些问题，我只能说一个不太或者说比较坑的，不过也不是开发的时候，是那种情况，有一次大概应该是说他们想做那种，他们想用es，就是想做那种快速查询嘛，想用es，然后就想要数据异构，然后反正当时我就想让mysql嘛，然后canal，然后异构到es里面去，然后让他们直接在es里面查嘛，对吧。然后有一个问题就是说它一直拿不到binlog，就一直拿不到binlog同步嘛，然后就很奇怪，我一直拿不到binlog，因为当时在线上部署的，然后后面我就到线下到github上面一跑了一个canal的源码，然后我在本地debug，然后我才发现线上的mysql的数据库的那个binlog存放位置是被改过的，就不是默认的那种，这就很坑，就是说，用一个第三方控件，我还要把第三方控件的源码跑下来，在本地跑一遍，我才能判断出来是什么。

更复杂一点的就是一些那种包不兼容嘛，就是那种，因为有些时候他们升级，然后升级到了一些包不兼容的这种好坑的地方嘛，也有。这种只能看官网啊，就是拿着他们pom文件，然后点进去啊，然后看一下他们里面有什么依赖啊，然后是不是依赖跟依赖之间有冲突啊，然后找不出来就只能到官网上面看啊。你像那种spring cloud跟spring foot，还有spring cloud alibaba，你像这三个他们大版本实际上是有严格匹配规律的吧，你要是没匹配好，实际上你项目是起不来的，对吧？是经常有的，你像什么spring cloud的格林威治版本，跟alibaba的什么一点多少多少的版本，反正经常有那种冲突，奇奇怪怪的。另外有一些依赖，那种依赖怎么说，反正半看半猜嘛就是。

### 3.9 会去优化或者说重构自己的代码吗

重构是会去重构的哈，但是是要有足够的时间去重构的就是。你像我记得应该是有一个那种下载功能，其实下载的很慢嘛，然后当时就开了一个多线程去下嘛，然后还有那种比如调外部接口，比如我自己的业务系统调 OA系统，然后这个时候它OA系统它返回的数据它很慢，但是我业务要走下去，然后我就只能开一个异步嘛，就也是开异步，那我异步去调你的那个数据，但是我主线程还在往下走，然后搞一个那种内存屏障，然后等我要用到的时候我再去取，然后在主线程放开，就是有做过这种异步优化。

### 3.10 面试录音文字版

**盛先生，26岁，南昌，5年经验**

说话人1 00:00

你好，请问是盛先生吗？你好，我是思诚科技公司的，我们公司在招Java开发，看到你有投递简历，不知道你有没有兴趣？

说话人2 00:13

可以。

说话人1 00:15

好的，你现在方便吗？我问你几个问题。

说话人2 00:17

你说。

说话人1 00:19

好的，就是你做过哪些项目？

说话人2 00:23

做过哪些项目？

说话人1 00:24

对。

说话人2 00:25

主要是两个方面，政府的项目一方面，还有一方面就是金融业务，这是大的方向，然后至于你说政府里面有那种什么各种各样的类似于那种审批系统喽，然后是管理系统喽，然后金融业务里面也有各种那种什么保函，那种房抵贷呀这种项目。

说话人1 00:53

那你印象比较深的是哪一个呢？

说话人2 00:57

那肯定是最近一份工作的印象最深了，就是金融业务的比较深。因为之前政府业务的说句实话已经是两年前的事了。

说话人1 01:10

对。这个项目它是做什么的具体？

说话人2 01:15

我举一个例子吧，就是房屋贷款的可以吧？他那个项目叫房抵贷，意思就是说把房屋抵押给银行，然后来拿到资金，但是有一些房屋银行是不会要的，比如那种很老的房子银行它是不会要的，所以让我们公司来给他做担保。

然后拿凭证就意思就是我们实际上是处于一个担保方，然后大概就是说从一开始业务人员去接收到去填写房屋的信息，然后经过那种什么预审、复审、终审，然后进入到了尽调，就是尽职调查，然后就会业务人员就会到房子那里去实地考察，实地考察，然后把信息再反馈到系统里面去，然后这个系统就再经过一些审批，然后再跟银行联调，跟资金方联调，然后开始汇款打款，然后生成那种清分计划，就是每个月还多少钱之类的，什么到款，然后再清分收益，大概是这么个流程。

说话人1 02:32

嗯，那它是怎么划分功能模块，这些模块他们是怎么交互的？你可以说一下吗？

说话人2 02:41

首先一个项目最基本的那种什么权限模块，这个是肯定有的了，最基本的RABC那种吧，就是那种目录，就是那种角色对应的什么人员，对应组织，对应的目录对吧？这个最简单的，这个是最底层的权限控制管理。然后再往上面就有一些那种业务系统，主要是负责业务数据录入数据的，然后跟它对应的还有一个是OA系统是用来负责审批的，它们之间可以通过接口或者通过或者通过IMQ进行交互。然后这大概是两个，然后还有一部分就是跟银行对接的系统，跟银行对接方面的接口，就大概分成了这么几块。

说话人1 03:27

那它里面用到了哪些技术呢？

说话人2 03:32

 Spring cloud alibaba那一套，mybatis那一套，然后注册中心那种什么nacos喽，mysql喽，然后rabbitmq、redis反正基本上都用到了吧，那一套，然后还有一些那种外部的，还有nginx，什么skywalking什么elk分布式日志收集这些都是有的。

说话人1 03:57

嗯，那这个项目它的预算是多少。

说话人2 04:01

预算是多少？嗯……那我写代码的，我怎么知道预算是多少。

说话人1 04:08

等于说你没有管这块事情。

说话人2 04:12

我肯定不关心哪，对啊，我加班还来不及了。

说话人1 04:14

你们有多少个人在做这个项目啊？人员是怎么分的。

说话人2 04:18

人员一直在变动？就是说一直都在变，多的时候有三四个，少的时候就一两个，就这样。

说话人1 04:27

嗯，那人员配比呢？

说话人2 04:28

呃，一个前端对两个后端，然后对一个测试。反正比例是这个比例哈，可能有2个前端4个开发2个测试的也有，大概就这个比例。

说话人1 04:40

等于说这个项目周期还是比较长的，它做了多久啊。

说话人2 04:43

有半年哦，迭代很快的，一直在迭代。

说话人1 04:48

你在这里面具体是负责哪些工作？

说话人2 04:52

主要是负责后端，然后大概像那种尽职调查那一块，我就做的比较多，然后还有跟OA审批流交互那一块我也做的比较多吧。审批流是有他们人去做的，反正我也是，我大概知道一点吧，做我也能搞两下子。

说话人1 05:15

就审批流这边你们是用什么做的？

说话人2 05:17

他们用的是folowable，什么流程预测那些，反正我只要把那个节点里面的信息人，还有资金，反正必要的参数传给他，然后他们就会流程预测出来一条线，然后自然会把那些人给我，然后后面该怎么流转怎么流转，但是我负责的是业务数据那一边。你像比如他相互尽调了以后，那些什么视频照片，那些流处理那些乱七八糟的，还能够下载上传到mirror那些。

说话人1 05:47

那你在这个项目里面，你有没有遇到什么印象比较深的问题，你是怎么解决的？

说话人2 05:57

印象比较深，怎么说呢，你像那种正常的，正常的那些问题，反正你也知道吧，该有都会有的正常的一些问题，我只能说一个不太或者说比较坑的，不过也不是开发的时候。

是那种情况，有一次大概应该是说他们想做那种，他们想用es，就是想做那种快速查询嘛，想用es，然后就想要数据异构，然后反正当时我就想让mysql嘛，然后canal，然后异构到es里面去，然后让他们直接在es里面查嘛，对吧。

然后有一个问题就是说它一直拿不到binlog，就一直拿不到binlog同步嘛，然后就很奇怪，我一直拿不到binlog，因为当时在线上部署的，然后后面我就到线下到github上面一跑了一个canal的源码，然后我在本地debug，然后我才发现线上的mysql的数据库的那个binlog存放位置是被改过的，就不是默认的那种，这就很坑，就是说，用一个第三方控件，我还要把第三方控件的源码跑下来，在本地跑一遍，我才能判断出来是什么。

说话人1 07:15

嗯，就它那控件有问题哦？

说话人2 07:18

啊是，我也不知道运维为什么要把binlog调位一次，是真的很坑。

说话人1 07:23

那你自己再跑了一遍，在另外一个环境跑一遍，然后对比一下。

说话人2 07:28

对，我还是把canal从github上跑了一个源码，然后再debug它。

说话人1 07:39

好的，有没有更复杂一点的问题呢？

说话人2 07:41

更复杂一点的啊，那就是一些那种，一些那种包不兼容嘛，就是那种，因为有些时候他们升级，然后升级到了一些包不兼容的这种好坑的地方嘛，也有。

说话人1 07:59

这种情况你是怎么解决怎么处理？

说话人2 08:02

这种啊，这种只能看官网啊，就是拿着他们pom文件，然后点进去啊，然后看一下他们里面有什么依赖啊，然后是不是依赖跟依赖之间有冲突啊，然后找不出来就只能到官网上面看啊。

说话人1 08:15

官网看他们的文档？

说话人2 08:17

嗯，对啊，因为本来就有嘛，你像那种spring cloud跟spring foot，还有spring cloud alibaba，你像这三个他们大版本实际上是有严格匹配规律的吧，你要是没匹配好，实际上你项目是起不来的，对吧？

说话人1 08:32

嗯，是的

说话人2 08:33

是经常有的，你像什么spring cloud的格林威治版本，跟alibaba的什么一点多少多少的版本，反正经常有那种冲突，奇奇怪怪的。

说话人1 08:43

是的是的，你是直接看英文文档吗？

说话人2 08:45

嗯……这种一半一半吧，反正这种英文文档啊、找博客啊，反正你像spring cloud alibaba，反正他们官网有嘛，这个简单，只要到alibaba它们官网上面看嘛。另外有一些依赖，那种依赖怎么说，反正有个半看半猜嘛，也就是。

说话人1 09:12

有没有，你有没有去优化一些性能之类的，或者说你觉得你的代码可以做得更好，你有没有去做一些这样的重构的事情？

说话人2 09:25

重构啊，有有有，重构是会去重构的哈，但是是要有足够的时间去重构的就是。重构会，你像我记得有那种下载功能，我记得应该是有一个那种下载功能，其实下载的很慢嘛，然后当时就开了一个多线程去下嘛，就开了一个多线程，这种东西嘛，然后还有那种比如调外部接口，比如我自己的业务系统调 OA系统，然后这个时候它OA系统它返回的数据它很慢，但是我业务要走下去，然后我就只能开一个异步嘛，就也是开异步，那我异步去调你的那个数据，但是我主线程还在往下走，然后搞一个那种内存屏障，然后等我要用到的时候我再去取，然后在主线程放开，就是有做过这种异步优化。

说话人1 10:17

这个是你自己驱动自己去做呢，还是上面公司有要求？

说话人2 10:24

嗯……有空的时候。在忙的时候，我肯定没心思去管这个嘛，我还敢动这个东西，万一爆了怎么办？就是有空的时候，对吧，然后就会去处理一下。

说话人1 10:35

就自己也挺有兴趣的噢。

说话人2 10:37

对啊，用各种嘛，你像还有那种比如mysql查询喽，那种，就是统一批量去查了，对吧，就不用for去查了，你知道吧，因为当时写的时候其实心里是知道的，心里是知道的，但是没办法赶工期啊。

说话人1 10:52

对，那肯定要先做出来嘛。

说话人2 10:53

对对对，赶紧迭代，你特别是我们金融行业，对吧，分分钟多少钱。

说话人1 11:05

嗯，好，我就这边先做个简单的了解，等后面我们有消息的话，可能会有其他同事再联系你哈。

说话人2 11:11

好，没问题。

说话人1 11:14

好，打扰您了，谢谢，再见。

## 样本四：

### 4.1 项目背景

天印电子印刷平台，一个v6版，一个v5.3版，主要是做电子合同这一方面。因为我那个公司它是e签宝，e签宝其实它是在中国电子合同这一方面，它是领头羊嘛，然后它总部是在杭州，然后我们金格公司是前年被它收购了，收购了之后就是从去年1月份开始弄它那个天印5.3，这是他们自研的项目，天印5.3他以前就已经做好了的嘛，然后我们这边做的话从事一些二开，后面还有新的一些基线的需求。

### 4.2 功能模块

它主要就是有三个模块嘛，就是面向客户有三个端，一个就是统一管理平台，还有一个外部用章子系统，一个内部用章子系统。

内部用章子系统它就是内部员工嘛，外部用章子系统它就是一些，比如我们这些不是在一个公司里的外部的一些客户之类的，他们去使用的。因为签电子合同一般都是两方对吧？你要不就企业跟个人签，要不就是你内部，就两个个人签，一般都是这样子嘛。

然后他在功能模块其实就是围绕整个一个，因为以前我们传统的签一个合同都是在线下，对吧，然后最近疫情这三年嘛，然后国家都是倡导尽量线上化，线上办公嘛，对吧，减少一些线下的流程啊之类的，然后包括我们可能签一些劳动合同也是都可以直接用我们这个平台去签了，然后它主要就是，因为你现场签的话，你就需要有一系列的安全措施，你要得国家得承认，对吧，有法律保护嘛，这一系列。

你就从最开始的话，你就需要有用户，用户的话他需要注册进来嘛，注册进来的话你首先你需要他有身份证、手机号这样子的信息，他有这种用户之后，包括你也有企业，企业的话，你也要提供一些企业的资质，比如它的统一社会信用代码之类的，一些法律信息之类的，这样的话你企业也算是认证了，然后用户也算是实名认证了，对吧。

然后这一块的话因为双方都是已经实名过了的话，这个就可以保证一些安全性，然后再是有了这有了用户的基础，然后你再要进行操作合同了嘛，操纵了合同你要盖章嘛，对吧。然后你盖章的话它就需要有绑定证书，因为那个印章不是随随便便我往上抠个图就行了，对吧，这是要有专业的ca证书，然后和你的印章绑定起来，绑定成一个印章结构体，然后再加盖到我们的pdf上。

然后你盖章的过程中你需要有一些意愿认证嘛，对吧。你得确保当时那个盖章的人是你本人，对吧，你可能可以通过手机号验证码验证啊，包括一些人脸啊或者是密码呀，反正有很多验证的方式嘛，然后就是同意签署之后，我们系统就会自动把那个章盖在 pdf上，pdf它是一个流式的文档对吧？然后他是不能被篡改了，不能被修改嘛，对吧。

然后有这个文档之后，后面还有一些一系列的比如归档啊之类的一些操作嘛，后面并且在加盖，就是一系列这些操作的过程中，我们都会记录它的一些所有的包括意愿认证啊，还有他签署的一些信息之类的，我们都会记录到数据库，包括我们还对接了蚂蚁的区块链，然后会有一个上链的过程，然后后面如果这些加盖的一些合同，他有一些纠纷的话，然后还可以向我们公司开出公证，开一些认证报告之类的，公证书之类的，公证说明这样子的东西。

它整个嵌入过程它大概是这样嘛，然后你拆出微服务的话，是吧，你有用户的微服务嘛，然后再是有印章的微服务嘛，然后再是有模板啊，还有合同啊，包括后面一些认证啊，这些都是有拆成一些比较细块的微服务。

### 4.3 技术架构

它整体一个架构就是一套微服务，微服务整个就是spring cloud alibaba这一套嘛。

然后注册中心就是nacos嘛，配置中心也是nacos，然后链路啊，调用，服务之间调用就是用dubbo嘛，然后再是一些可能要用的那个熔断啊，熔断可能就是sentinel，然后还有一些事务，事务的话我暂时还没接触，但是我看了一些最开始的设计的技术是用seata嘛。

然后再是包括一些，因为我们公司分的比较，人员也比较多嘛，然后有很多测试，很多运维，包括还有交付。针对运维、交付这一块呢，我们又有一些，比如可视化部署，然后包括一些一键部署脚本，就是Linux写的一些一键部署脚本，当然这些不是我们去写的，就是专业的运维人员去写嘛。然后还有一个可视化部署这块也是我们这边研发去做的。然后刚才说的这些技术栈嘛。主要就是一个spring cloud、一个alibaba现在比较火的整个一套嘛。

### 4.4 项目预算

预算？这个，它一年都挣个5个亿的。对，您都可以去百度一下杭州天谷。他都是，我刚才不是吹牛的，他的蚂蚁都融资了10个多亿，他现在是一轮嘛，他就是准备上市了。

### 4.5 开发团队介绍

其实他这个有一个公有云，我们是分po线的，它有公有云线，还有一个混合云线，我们整个公司的研发团队的话大概有1000多人，然后我们混合云的话是有大概之前好像是500多人吧，然后刚才我也说了，因为我们……他做6.0的话，它是微服务架构嘛，微服务架构其实一个好处就是，你可以把每个微服务架构，就是每个小的微服务都提供出去给各个团队去做，对吧。

不像你做单体，单体的话，你可能整个团队都是做一个东西嘛。然后我刚才说的，可能就是，比如第一个组嘛，他就可以做那个文档中心，第二个组他可以做签署中心，第三个组可以做认证中心。

然后包括我这个组，我这边主要负责就是一个集成中心，集成中心的话，它主要就是对接一些第三方的一些平台嘛，然后就是比如它的一个泛微啊，资源啊，包括一些钉钉啊，企业微信这样子的对接他们一些异构系统的那些接口嘛，第三方提供的一些服务，然后接到我们这个平台上来。

我这个小组的人员的话主要就是5个人嘛，然后有一个前端，然后我上面还有一个算是整个项目经理吧，然后我算是一个组长吧，然后我下面还有两个实习的，然后还有1年左右工作经验的。

测试我们也是有专门的测试po线，反正都是同一个组嘛，在整个部门来说又是一个大的组，大的组里面测试资源是共享的，就不会像开发分的这么详细。

### 4.6 开发周期

它很久了，它以前是……我写的就是v5.3嘛，其实它在……我这公司它好像是02年就创立的，它从02年其实就已经开始了，它那个时候已经有v一点几了，不过那个时候是桌面应用嘛，然后慢慢的10年之后就开始转型，转型就是发展一些线上的那些软件，B/S这种软件了嘛。

### 4.7 具体负责工作

其实我们这个，也不仅仅只是做后端代码开发嘛，因为我们也是……我刚才说的我们分的很详细，他也有很多产品嘛，产品基本上每个需求都是要过产品，然后先是评审，评审完之后，我们再是看可行不可行，然后再是写一些开发的设计文档，设计文档之后然后就是写代码了嘛，然后还会有一个时间，就是你在哪个时间节点之类的，评的工作量之类的嘛，然后在这个时间范围，在这个工作量之内把这个项目做完，做完之后就交付。

然后测试……提个测试嘛，测试的话可能还需要一些时间，然后配合他进行一些测试嘛。测完之后大概差不多就是上线嘛，就是交付出去给运维啊，包括那个交付人员，他们就部署到线上环境去。

我们做需求，整个团队，就是整个组都得参与。大公司的话，它那个体系，包括一个流程就比较要详细嘛。

### 4.8 开发中遇到哪些问题、如何解决、有没有更好的解决方案

印象比较深的，其实也就是一开始说的对接很多那种……因为我们那个系统它要对接很多那个，第三方的那些异构系统，它异构系统的话它就很多就没有，不会有一套……就是不能设计出一套很标准的东西来嘛，因为你可能在钉钉，我就说一个最简单的就是组织用户同步嘛，对吧，你从钉钉获得过来，它的这边叫手机号就叫phone，然后你在微信那边它就要叫mobile，之类的，然后并且他这边提供的信息可能会有一些，就是……可能它这边邮箱不提供，然后钉钉那边或者是微信那边又提供。

反正就是每个异构系统，它能提供的一些接口，包括它的一些字段之类的，整合过来，那是比较复杂的，因为当时这个的话也是考虑了很多，因为我们做平台做产品嘛，肯定是想做的统一嘛，对吧，统一做一个标准嘛，然后当时做这个东西也是，唉……开了很久的会，都是一直都没有一个很好的解决方案嘛。

后面，也就是申请了一些，级别更高的一些……因为我们有架构师嘛，还有资深的java程序吧，然后就是在他们的配合下就有一套那个异构，就是……也是算一个异构表，和我们本地那些用户表，嗯……怎么说，其实就是一个以我们这边一套用户体系，组织用户体系，然后和和异构系统，这些所有异构系统，我们也提取出它的一些共性来，然后建一个就是中间表的概念，然后有了中间表，因为对于我们整个，就刚才说天印平台嘛，对吧，它的天印平台它的用户表，它其实就需要一些我刚才说的那些数据，就是比如一个普通的用户，他就需要手机号，然后再是邮箱啊，包括它的真实姓名啊，可能这几个字段就行了，但是怎么把异构系统的信息拉取过来，拉取到我们天印这边，那我刚才就说了，因为你用异构系统直接去对接的，就是对接我们天印的话，我们天印直接就对接那个异构系统的话，那就很复杂嘛，你来一个我就得写一套代码，来一个我就得写一套代码嘛，对吧，所以后面就说，就是加了一个我刚才说的那个集成中心，它主要是起中间，就是两方协调的一个作用嘛。

然后它就是，这里面就刚才说了设计一个中间表，一个用户的中间表，一个组织的中间表，然后会配置一个映射字段，就比如它那边，钉钉那边，首先是还有一个异构系统的表嘛，然后异构系统表它下面会关异构系统的字段，组织用户同步的一个字段表，然后比如他那边手机号是叫mobile，然后对应我们这边，我们这边可能就叫phone，然后包括email啊，他那边可能叫user\_email，我们这边就叫email，反正就是起这种一一对应的关系嘛，其实这一套弄完之后，我们刚才也说了，我有前端配置嘛，如果某一天哪个异构系统，哪个异构系统它那个字段，因为它一般现在都是json反回过来的嘛，对吧，他如果哪一天它那个字段增加了，或者是修改了，他那个名称之类的，然后我们也是可以直接改那个映射表，就在后台改那个映射表，然后我们这边也不需要改代码，就可以还是保证业务的通畅嘛。

其实还有一个东西，这只是一个问题，还有一个问题最深的，我们其实写代码，说实话这一块的工作量，其实也就顶多占一半嘛，还有另外一半，其实更多的是跟第三方异构系统的一些用户啊，嗯……就是厂家，我们需要去对接。

他们提供的文档千奇百怪，有可能有的提供 http的一个文档，有的有可能提供webservice的接口，接口文档之类的，反正各种各样的五花八门，然后并且每个厂商可能它能提供测试环境是吧，那有的又不能提供测试环境。然后这样的话我们可能又需要我们有一些，比如我们懂Linux啊，是吧，能够在他们那边配合他们把这个东西，就是我们，可能测试的包发到他们一些模拟环境或者是有的他没测试环境，直接怼到生产环境这样子的也是有。然后这样子的，因为我们这边也没有……他们不提供外网之类的，我们测试也测不了，也就只有在生产环境这样子的，然后也是，反正就一直要配合嘛。期包括他提供这个文档过来，我们肯定也要认真看，符不符合嘛，这些对接，很多，包括后面部署，这两块其实也有很大的工作量。说实话这两块工作量也不会比我们开发的工作量低。

### 4.9 会去优化或者说重构自己的代码吗

会啊。我们这个，每周都有code review的，就代码审查，然后代码评审都是有的。

我们这个组的话，上面还有更高的一些领导，我刚才说的可能更高级别的一些资深的或者是一些开发专家之类的，他们会组织，比如每周啊，可能你这个组来一下，就是来看一下你们这个组最近做的这个需求，然后看你们代码，怎么……刚才说了，我们还要从设计开始嘛，先从你的那个需求的设计文档看有没有问题，然后看完那个之后，你再大概讲一下你代码是怎么实现的，然后看一下代码里面有没有一些比较常见的一些错误，或者是有没有一些隐患之类的。

然后在这个review的过程，是吧，可能当时自己一个人的话可能是发现不了嘛，很多东西其实，然后有些经验更足的人可能站在更高的角度，是吧，可能可以想到一些更深一点的问题，然后就是这样做了，然后就及时去修改嘛，包括后面我们自己也会修改，因为刚才说了对接了很多，很多那个异构系统，一开始的话我们也没想那么多嘛，需求也是比较着急要上线嘛，那个代码一开始就是说写的能用就行了嘛，基本上都是这样嘛。怼上去了，怼上之后发现越写越大，可能以前来一个厂商，我们就写个if else，来一个又写一个if else，就是这样子，一直加上去嘛，不同的类型就加一个不同的实现嘛，就这样，后来发现那个代码越来越臃肿，然后发现这样肯定不行的，然后就，后面又把一些共性给提出来了。

其实这一方面我印象比较深的就是它对接那些不同厂商，它那个单点登录系统，单点登录系统其实它主要也就是一个OAuth2协议，还有一个 CAS协议嘛，这两个，然后一开始的话我们就是，刚才说的没考虑那么多，反正就来一个就这么实现，后来就是发现代码越来越臃肿，然后发现有的协议它是共用的，然后我们可能可以提出它一些共性，它只是在最后，可能实现会有一些区别，我比如对接钉钉啊，包括企业微信，这两个单点登录，它背后都是使用OAuth2那个协议，我们就提出了它共性嘛，其实它整个OAuth2协议的流程就是，你去请求它获得一个授权的code，你再去获得他的一个 access\_token，access\_token之后再去调它用户接口，获取那个用户信息。其实它主要就是这几步嘛，然后我们就把这几步共用的，然后提取出来，然后后面就是写一个OAuth2的一个工厂嘛。然后后面再加一些，你比如后面又对接一个新的，新的比如可能是QQ啊，他可能也是一个OAuth2的一个协议的话，那我们可能前面一些共性的我们已经提取出了，包括用的流程了，就刚才说的那几步共用的，可能我们写出来，它唯一可能后面不同的就是那个获取用户嘛，对吧。

我们做这个，就是要对这个产品负责，然后它从头到尾，因为交付也要我们管，我们自己把代码写好一点，说实话也是减少自己的工作量嘛，对吧。

### 4.10 面试录音文字版

**刘先生，26岁，南昌，6年经验**

说话人1 00:00

你好，请问是刘先生吗？

说话人2 00:03

你是？

说话人1 00:04

你好，我是思诚科技公司的，我们公司在招Java开发，不知道你有没有兴趣。

说话人2 00:40

可以啊。

说话人1 00:42

好的。您做过哪些项目呢？

说话人2 00:45

我最近的项目就是一个，因为我之前的公司主要是做电子合同这一方面的吧，然后做的一个天印电子印刷平台，一个v6版，一个v5.3版，还有一个就是金格信签。我就上了一家，因为我那个公司它是只做自己公司的产品嘛，然后主要就是围绕这三个产品，然后也在那边待了一年半多吧，主要就是做这三个产品的二开，包括一些基线的需求的后台开发。

说话人1 01:22

对，你印象最深的就是这个？

说话人2 01:25

这三个都是最近的，所以印象肯定是最深的嘛。

说话人1 01:30

好的。您说一个其中印象最深的，然后说一下它里面的一些功能模块吧，是怎么划分的。

说话人2 01:41

那我就说天印5.3吧，因为我那个公司它是e签宝，e签宝其实它是在中国电子合同这一方面，它是领头羊嘛，然后它总部是在杭州，然后我们金格（公司）是前年被它收购了，收购了之后就是从去年1月份开始弄它那个天印5.3，天印5.3他以前就已经做好了的嘛，然后我们这边做的话从事一些二开，后面还有新的一些基线的需求，然后它主要就是有三个模块嘛，就是面向客户有三个端，一个就是统一管理平台，还有一个外部用章子系统，一个内部用章子系统。

内部用章子系统它就是内部员工嘛，外部用章子系统它就是一些，比如我们这些不是在一个公司里的外部的一些客户之类的，他们去使用的。因为签电子合同一般都是两方对吧？你要不就企业跟个人签，要不就是你内部，就两个个人签，一般都是这样子嘛。

然后他在功能模块其实就是围绕整个一个，因为以前我们传统的签一个合同都是在线下，对吧，然后最近疫情这三年嘛，然后国家都是倡导尽量线上化，线上办公嘛，对吧，减少一些线下的流程啊之类的，然后包括我们可能签一些劳动合同也是都可以直接用我们这个平台去签了，然后它主要就是，因为你现场签的话，你就需要有一系列的安全措施，你要得国家得承认，对吧，有法律保护嘛，这一系列。

你就从最开始的话，你就需要有用户，用户的话他需要注册进来嘛，注册进来的话你首先你需要他有身份证、手机号这样子的信息，他有这种用户之后，包括你也有企业，企业的话，你也要提供一些企业的资质，比如它的统一社会信用代码之类的，一些法律信息之类的，这样的话你企业也算是认证了，然后用户也算是实名认证了，对吧。

然后这一块的话因为双方都是已经实名过了的话，这个就可以保证一些安全性，然后再是有了这有了用户的基础，然后你再要进行操作合同了嘛，操纵了合同你要盖章嘛，对吧。然后你盖章的话它就需要有绑定证书，因为那个印章不是随随便便我往上抠个图就行了，对吧，这是要有专业的ca证书，然后和你的印章绑定起来，绑定成一个印章结构体，然后再加盖到我们的pdf上。

然后你盖章的过程中你需要有一些意愿认证嘛，对吧。你得确保当时那个盖章的人是你本人，对吧，你可能可以通过手机号验证码验证啊，包括一些人脸啊或者是密码呀，反正有很多验证的方式嘛，然后就是同意签署之后，我们系统就会自动把那个章盖在 pdf上，pdf它是一个流式的文档对吧？然后他是不能被篡改了，不能被修改嘛，对吧。

然后有这个文档之后，后面还有一些一系列的比如归档啊之类的一些操作嘛，后面并且在加盖，就是一系列这些操作的过程中，我们都会记录它的一些所有的包括意愿认证啊，还有他签署的一些信息之类的，我们都会记录到数据库，包括我们还对接了蚂蚁的区块链，然后会有一个上链的过程，然后后面如果这些加盖的一些合同，他有一些纠纷的话，然后还可以向我们公司开出公证，开一些认证报告之类的，公证书之类的，公证说明这样子的东西。

然后整个系统，包括整个我刚才说的三个平台嘛，三个产品，对吧，其实它都是做这一块。然后刚才说了它细分的话，它整个嵌入过程它大概是这样嘛，然后你拆出微服务的话，是吧，你有用户的微服务嘛，然后再是有印章的微服务嘛，然后再是有模板啊，还有合同啊，包括后面一些认证啊，这些都是有拆成一些比较细块的微服务。

说话人1 06:45

嗯嗯，好的，那这个项目它的预算是多少呢？

说话人2 06:50

预算？这个，它一年都挣个5个亿的。

说话人2 06:56

1年5个亿？

说话人1 06:57

对，您都可以去百度一下杭州天谷。他都是，我刚才不是吹牛的，他的蚂蚁都融资了10个多亿，他现在是一轮嘛，他就是准备上市了。

说话人2 07:11

嗯，好的，那这个项目里面用到了哪些技术呢？

说话人1 07:16

我刚才说的一个如果是针对6.0版本嘛，6.0版本，其实我现在已经离职了嘛，对吧，但是它整体一个架构是有的，就是一套微服务嘛，微服务整个就是spring cloud alibaba这一套嘛。

然后注册中心就是nacos嘛，配置中心也是nacos，然后链路啊，调用，服务之间调用就是用dubbo嘛，然后再是一些可能要用的那个熔断啊，熔断可能就是sentinel，然后还有一些事务，事务的话我暂时还没接触，但是我看了一些最开始的设计的技术是用seata嘛。

然后再是包括一些，因为我们公司分的比较，人员也比较多嘛，然后有很多测试，很多运维，包括还有交付。针对运维、交付这一块呢，我们又有一些，比如可视化部署，然后包括一些一键部署脚本，就是Linux写的一些一键部署脚本，当然这些不是我们去写的，就是专业的运维人员去写嘛。然后还有一个可视化部署这块也是我们这边研发去做的。然后刚才说的这些技术栈嘛。主要就是一个spring cloud、一个alibaba现在比较火的整个一套嘛。

说话人2 08:46

嗯，是的。那有多少人做这个项目呢？

说话人1 08:51

有多少人，其实他这个有一个公有云，我们是分po线的，它有公有云线，还有一个混合云线，我们整个公司的研发团队的话大概有1000多人，然后我们混合云的话是有大概之前好像是500多人吧，然后刚才我也说了，因为我们……他做6.0的话，它是微服务架构嘛，微服务架构其实一个好处就是，你可以把每个微服务架构，就是每个小的微服务都提供出去给各个团队去做，对吧。

不像你做单体，单体的话，你可能整个团队都是做一个东西嘛。然后我刚才说的，可能就是，比如第一个组嘛，他就可以做那个文档中心，第二个组他可以做签署中心，第三个组可以做认证中心。

然后包括我这个组，我这边主要负责就是一个集成中心，集成中心的话，它主要就是对接一些第三方的一些平台嘛，然后就是比如它的一个泛微啊，资源啊，包括一些钉钉啊，企业微信这样子的对接他们一些异构系统的那些接口嘛，第三方提供的一些服务，然后接到我们这个平台上来。

说话人1 10:08

嗯，那您这个小组的人员是怎么分配呢？

说话人2 10:10

我这个小组的人员的话主要就是5个人嘛，然后有一个前端，然后我上面还有一个算是整个项目经理吧，然后我算是一个组长吧，然后我下面还有两个实习的，然后还有1年左右工作经验的。

说话人1 10:39

就您这边主要就是，您的小组主要是负责后端的一些开发？

说话人2 10:43

都是我刚才说了也有个前端，因为我们那个项目也有……你不可能说每个微服务它可能都是有一个相对来说，比如我刚才说集成中心对吧？它很多配置，配置当然也需要有前端的页面对吧？所以是有一个前端，然后我们再是做一些后端的开发。

说话人1 10:09

嗯，好的。那您这边测试的话是交给其他测试部门去测试噢？

说话人2 11:14

测试我们也是有专门的测试po线，反正都是同一个组嘛，在整个部门来说又是一个大的组，大的组里面测试资源是共享的，就不会像开发分的这么详细。

说话人1 11:35

嗯，好的，那这个项目是从什么时候开始的？

说话人2 11:40

这个项目，它很久了，它以前是……我写的就是v5.3嘛，其实它在……我这公司它好像是02年就创立的，它从02年其实就已经开始了，它那个时候已经有v一点几了，不过那个时候是桌面应用嘛，然后慢慢的10年之后就开始转型，转型就是发展一些线上的那些软件，B/S这种软件了嘛。

说话人1 12:08

等于说您是最近两年才参与进来？

说话人2 12:11

对。

说话人1 12:12

好的，那您主要就是负责后端的开发，在这个项目里面，对吧？

说话人2 12:17

对，其实我们这个，也不仅仅只是做后端代码开发嘛，因为我们也是……我刚才说的我们分的很详细，他也有很多产品嘛，产品基本上每个需求都是要过产品，然后先是评审，评审完之后，我们再是看可行不可行，然后再是写一些开发的设计文档，设计文档之后然后就是写代码了嘛，然后还会有一个时间，就是你在哪个时间节点之类的，评的工作量之类的嘛，然后在这个时间范围，在这个工作量之内把这个项目做完，做完之后就交付。

就是测试……提个测试嘛，测试的话可能还需要一些时间，然后配合他进行一些测试嘛。测完之后大概差不多就是上线嘛，就是交付出去给运维啊，包括那个交付人员，他们就部署到线上环境去。

说话人1 13:15

嗯，就是评审阶段开始，您就……哦，您是组长，您就一直在参与这个事，一直在部署这个事。

说话人2 13:22

没有，我们做需求，整个团队，就是整个组都得参与。

说话人1 13:28

哦，全体都要参与。

说话人2 13:31

对对对。

说话人1 13:22

哦，好的。

说话人2 13:34

大公司的话，它那个体系，包括一个流程就比较要详细嘛。

说话人1 13:40

那您在开发中有没有遇到什么印象比较深的问题，然后是怎么解决的？你可以说一下吗？

说话人2 13:48

印象比较深的，其实也就是一开始说的对接很多那种……因为我们那个系统它要对接很多那个，第三方的那些异构系统，它异构系统的话它就很多就没有，不会有一套……就是不能设计出一套很标准的东西来嘛，因为你可能在钉钉，我就说一个最简单的就是组织用户同步嘛，对吧，你从钉钉获得过来，它的这边叫手机号就叫phone，然后你在微信那边它就要叫mobile，之类的，然后并且他这边提供的信息可能会有一些，就是……可能它这边邮箱不提供，然后钉钉那边或者是微信那边又提供。

反正就是每个异构系统，它能提供的一些接口，包括它的一些字段之类的，整合过来，那是比较复杂的，因为当时这个的话也是考虑了很多，因为我们做平台做产品嘛，肯定是想做的统一嘛，对吧，统一做一个标准嘛，然后当时做这个东西也是，唉……开了很久的会，都是一直都没有一个很好的解决方案嘛。

说话人1 15:08

那后来？

说话人2 15:10

后面，也就是申请了一些，级别更高的一些……因为我们有架构师嘛，还有资深的java程序吧，然后就是在他们的配合下就有一套那个异构，就是……也是算一个异构表，和我们本地那些用户表，嗯……怎么说，其实就是一个以我们这边一套用户体系，组织用户体系，然后和和异构系统，这些所有异构系统，我们也提取出它的一些共性来，然后建一个就是中间表的概念，然后有了中间表，因为对于我们整个，就刚才说天印平台嘛，对吧，它的天印平台它的用户表，它其实就需要一些我刚才说的那些数据，就是比如一个普通的用户，他就需要手机号，然后再是邮箱啊，包括它的真实姓名啊，可能这几个字段就行了，但是怎么把异构系统的信息拉取过来，拉取到我们天印这边，那我刚才就说了，因为你用异构系统直接去对接的，就是对接我们天印的话，我们天印直接就对接那个异构系统的话，那就很复杂嘛，你来一个我就得写一套代码，来一个我就得写一套代码嘛，对吧，所以后面就说，就是加了一个我刚才说的那个集成中心，它主要是起中间，就是两方协调的一个作用嘛。

然后它就是，这里面就刚才说了设计一个中间表，一个用户的中间表，一个组织的中间表，然后会配置一个映射字段，就比如它那边，钉钉那边，首先是还有一个异构系统的表嘛，然后异构系统表它下面会关异构系统的字段，组织用户同步的一个字段表，然后比如他那边手机号是叫mobile，然后对应我们这边，我们这边可能就叫phone，然后包括email啊，他那边可能叫user\_email，我们这边就叫email，反正就是起这种一一对应的关系嘛，其实这一套弄完之后，我们刚才也说了，我有前端配置嘛，如果某一天哪个异构系统，哪个异构系统它那个字段，因为它一般现在都是json反回过来的嘛，对吧，他如果哪一天它那个字段增加了，或者是修改了，他那个名称之类的，然后我们也是可以直接改那个映射表，就在后台改那个映射表，然后我们这边也不需要改代码，就可以还是保证业务的通畅嘛。

说话人1 18:17

嗯，是的是的。这个事情是您牵头发起的，还是您的项目经理去牵头的？

说话人2 18:28

牵头的话肯定是他牵头嘛，但是我一直都是跟他一起想解决方案嘛，想思路嘛之类的。

说话人1 18:40

就是您一直在积极参与这个事情。

说话人2 18:44

对对对，其实还有一个东西，这只是一个问题，还有一个问题最深的，我们其实写代码，说实话这一块的工作量，其实也就顶多占一半嘛，还有另外一半，其实更多的是跟第三方异构系统的一些用户啊，嗯……就是厂家，我们需要去对接。

他们提供的文档千奇百怪，有可能有的提供 http的一个文档，有的有可能提供webservice的接口，接口文档之类的，反正各种各样的五花八门，然后并且每个厂商可能它能提供测试环境是吧，那有的又不能提供测试环境。然后这样的话我们可能又需要我们有一些，比如我们懂Linux啊，是吧，能够在他们那边配合他们把这个东西，就是我们，可能测试的包发到他们一些模拟环境或者是有的他没测试环境，直接怼到生产环境这样子的也是有。然后这样子的，因为我们这边也没有……他们不提供外网之类的，我们测试也测不了，也就只有在生产环境这样子的，然后也是，反正就一直要配合嘛。

说话人1 20:08

嗯，就这部分沟通工作是花了很多时间的。

说话人2 20:10

对对对，前期包括他提供这个文档过来，我们肯定也要认真看，符不符合嘛，这些对接，很多，包括后面部署，这两块其实也有很大的工作量。说实话这两块工作量也不会比我们开发的工作量低。

说话人1 20:28

确实确实，嗯。大概就是这些是吗？

说话人2 20:32

对。

说话人1 20:33

嗯，好的，那您会去优化自己的代码吗？

说话人2 20:38

会啊。我们这个，每周都有code review的，就代码审查，然后代码评审都是有的。

说话人1 20:43

是小组一起？

说话人2 20:45

也不……上面，我们这个组的话，上面还有更高的一些领导，我刚才说的可能更高级别的一些资深的或者是一些开发专家之类的，他们会组织，比如每周啊，可能你这个组来一下，就是来看一下你们这个组最近做的这个需求，然后看你们代码，怎么……刚才说了，我们还要从设计开始嘛，先从你的那个需求的设计文档看有没有问题，然后看完那个之后，你再大概讲一下你代码是怎么实现的，然后看一下代码里面有没有一些比较常见的一些错误，或者是有没有一些隐患之类的。

说话人1 21:29

嗯，就是让您自己去说这些，您自己的代码？

说话人2 21:35

对，然后在这个review的过程，是吧，可能当时自己一个人的话可能是发现不了嘛，很多东西其实，然后有些经验更足的人可能站在更高的角度，是吧，可能可以想到一些更深一点的问题，然后就是这样做了，然后就及时去修改嘛，包括后面我们自己也会修改，因为刚才说了对接了很多，很多那个异构系统，一开始的话我们也没想那么多嘛，需求也是比较着急要上线嘛，那个代码一开始就是说写的能用就行了嘛，基本上都是这样嘛。怼上去了，怼上之后发现越写越大，可能以前来一个厂商，我们就写个if else，来一个又写一个if else，就是这样子，一直加上去嘛，不同的类型就加一个不同的实现嘛，就这样，后来发现那个代码越来越臃肿，然后发现这样肯定不行的，然后就，后面又把一些共性给提出来了。

其实这一方面我印象比较深的就是它对接那些不同厂商，它那个单点登录系统，单点登录系统其实它主要也就是一个OAuth2协议，还有一个 CAS协议嘛，这两个，然后一开始的话我们就是，刚才说的没考虑那么多，反正就来一个就这么实现，后来就是发现代码越来越臃肿，然后发现有的协议它是共用的，然后我们可能可以提出它一些共性，它只是在最后，可能实现会有一些区别，我比如对接钉钉啊，包括企业微信，这两个单点登录，它背后都是使用OAuth2那个协议，我们就提出了它共性嘛，其实它整个OAuth2协议的流程就是，你去请求它获得一个授权的code，你再去获得他的一个 access\_token，access\_token之后再去调它用户接口，获取那个用户信息。其实它主要就是这几步嘛，然后我们就把这几步共用的，然后提取出来，然后后面就是写一个OAuth2的一个工厂嘛。然后后面再加一些，你比如后面又对接一个新的，新的比如可能是QQ啊，他可能也是一个OAuth2的一个协议的话，那我们可能前面一些共性的我们已经提取出了，包括用的流程了，就刚才说的那几步共用的，可能我们写出来，它唯一可能后面不同的就是那个获取用户嘛，对吧。

说话人1 24:34

嗯，等于说公司有这个机制，然后您自己也是挺有兴趣的，去驱动自己去做优化的这个事情。

说话人2 24:44

对，因为刚才说了你code review的话，首先是自己需求嘛，然后毕竟，我刚才也说了，我们做这个，就是要对这个产品负责，然后它从头到尾，因为交付也要我们管，我们自己把代码写好一点，说实话也是减少自己的工作量嘛，对吧。

说话人1 25:08

交付也是需要您自己负责？

说话人2 25:10

也不自己负责，就是你总得协调告诉他们怎么部署嘛，是吧。我刚才说了那些那是配置项之类的东西，他们也不是很清楚嘛，这样的你都得指点他们怎么弄，然后包括有些可能交付啊，他那个命令啊，对Java这些应用启动的命令不是很熟嘛，是吧，我们都协助，包括出现了新的问题啊，是把，你都得自己去看嘛，他们可能顶多就帮你把那个代码，把那个日志给你下下来，对吧，最后你看那个问题还是得你去看嘛。

说话人1 25:53

所以尽量把代码优化到足够好。

说话人2 25:54

对啊。

说话人1 25:55

嗯，好的，那我这边就先了解到这里。

说话人2 25:59

嗯。

说话人1 26:00

嗯好。如果后续有消息的话，可能会有其他同事再联系您。

说话人2 25:59

嗯，好的。

说话人1 26:03

好的，打扰您了，不好意思。

说话人2 26:10

嗯好，没事。

说话人1 26:12

好，谢谢，再见。

说话人2 26:13

拜拜。

## 样本五：

### 5.1 项目背景

网络货运的，就是车货匹配的这种，它对接了银行的电子合同啊，还有中交轨迹啊，等等，是公司自研自用的项目。

### 5.2 功能模块

模块有好多，一个有运单模块了，然后有第三方服务的模块了，系统采用的就是微服务嘛，就是一个模块一个服务。基本上就是做业务，核心代码都是在业务上，像第三方的华夏银行、平安银行、中交轨迹，包括存储的那种识别啊。

具体功能逻辑我举个例子，首先是系统登录喽，我们登录储存的token，然后用token进行验证的，有网关的，然后网关通过openfeign调用你其他的业务模块，然后走到业务模块有可能需要第三方支持的，像证件识别或者查询轨迹或者生成合同，也是用openfeign去调用其他的一些功能模块。我们系统里面主要是一个功能模块、一个业务模块。

### 5.3 技术架构

存储的用redis或者平台mysql数据库了，我们redis主要用的是做缓存的嘛。然后mq了，然后一些微信支付了，然后阿里的OCR认证识别的，然后OSS存储的，基本上像微服务都有用到了，什么mybatis，还有一些 spring clooud啊，spring mvc啊，然后我们监听的，服务注册用的eureka了，然后网关用的gateway，基本上就这些吧。

### 5.4 项目预算

项目没有什么预算，我们都是自研的，公司自己用的。他自研项目不是说不是跟别的接的外包的或者是干嘛的。

### 5.5 开发团队介绍

我们后端的比较多，多的时候12个，再包括前端、后端，然后UI、产品、两个测试，总共加起来16个。真正写代码的参与代码的就是12个人，然后去年因为疫情导致大货车在路上不让过，就裁了一部分裁了6个人，后面剩下6个人了。

### 5.6 开发周期

他是一直在完善了，有什么新的需求都会往里加。

他之前是属于那种开票系统，补录的，把那个运单补录进来，后面又做了一个车货匹配的，然后车货匹配的，因为它步骤啊，装卸货这种操作比较繁琐嘛，最后用了二维码，然后用电子围栏基于中交的那种装卸货出发的嘛。就一直在维护，是长期在用的自研项目。

### 5.7 具体负责工作

我是这边的技术经理，要管理整个项目进度，还包括审查组员的代码，然后自己也要写代码，一些重点难点的事情主要是我去解决，这些难点我不解决，就没人解决了。

### 5.8 开发中遇到哪些问题、如何解决、有没有更好的解决方案

嗯……最难的就是对接中交轨迹嘛，因为中交轨迹拉取的是十几兆的那种轨迹，经纬度嘛，一个文本，然后我们存储的时候，因为我们拉中交轨迹是需要付费的嘛。

中交轨迹就是说北斗星，我们中国的北斗，北斗就是那种定位系统嘛，那我们需要知道大货车有没有去拉货啊，有没有到达卸货地点，都是通过北斗去定位的。就是获取一个车辆的轨迹信息。

然后当时这个是拉取是需要收费的嘛，我们需要存储这个，当时定了好多方案，就是把它当成文本存储，但是文本存储检索的时候会比较慢嘛。然后就是一辆一张表的这种方式，然后每次拉取的时候会覆盖之前的或者是连接之前的看搜索时间，然后再加生成另外一张表，当索引的这种去搜索，就是分库分表这种方式。然后第二个就是平安银行，平安银行对接也比较麻烦，因为它版本比较多，每一次版本都有一个编号嘛，如果用错了版本的话就导致付费，或者说直接平台他那边返回就是报异常嘛。我们当时看这个文档也看得够呛的。

这个是我接手对接的嘛，然后我们以前我过来的时候是对接华夏银行，华夏银行就比较简单一点，他是直接转账，然后平安银行它是有个电子虚拟账户，而且实现公对私转账，那我们财务上面公对私转账是不允许的嘛。对，他就是规避了一个公对私转账的，就是电子虚拟账号，因为它电子虚拟账号是在我挂在我们公司名下的嘛，那这样转账就不属于是，不属于那种公对私的意义上面转账。

平安银行的跟轨迹分解的是我这边，然后我们公司其实系统里面还有一个最困难的，就是一个风控组件，你一辆车子有没有跑，然后车速是多少，然后你装卸货是否超吨，然后到到达卸货地点是否有亏吨这种模式，我们有专门的组件在进行拦截。

### 5.9 会去优化或者说重构自己的代码吗

有，我不但优化自己的代码，别人的代码也会看一下，因为我们这边，当时这边人员技术有参差不齐的嘛，有些人写的代码比较绕。

我会看，他们提交git代码的时候，我们这边钉钉上面会有那个提醒嘛，我会主动看一下他们提交的代码，有问题就会让他们去优化修改，时不时抽查一下他们代码，特别是做久了，你知道哪些人写的代码一定会有问题的，你就是着重注意一下，有些时候写的对接第三方的那种token失效的这种都是够呛的，连接了又连接的这种。

### 5.10 面试录音文字版

**高先生，31岁，南昌，7年经验**

说话人1 00:00

你好，请问是高先生吗？

说话人2 00:05

啊。

说话人1 00:06

你好，我是思诚科技公司的，我们公司在招Java开发，不知道你有没有兴趣。

说话人2 00:12

嗯，最近是在找工作。

说话人1 00:16

好的，你现在方便吗？

说话人2 00:17

方便的。

说话人1 00:19

好的，那我问你几个问题，你做过哪些项目？

说话人2 00:26

项目我一般都是去的自研公司，一直去的是自研公司，没有跟外外包公司那样的时候，一个公司两个项目。我在南昌的莲塘那边做的是网络货运，就是那个车货匹配的这种，然后还有一个是保险的，再有一个就是那种南昌的叫什么维修资金的，就是房管局的那个系统。就这三个吧，然后景德镇的那一家是网上购物的。

说话人1 00:59

好的。那您印象比较深的是哪一个呢？

说话人2 01:11

最近的嘛，网络货运的，因为它对接了银行的电子合同啊，还有中交轨迹啊，等等。

说话人1 01:20

那它里面划分了哪些功能模块呢？

说话人2 01:25

模块有好多诶，一个有运单模块了，然后有第三方服务的模块了，系统采用的就是微服务嘛，就是一个模块一个服务。

说话人1 01:37

大概就这些吗？

说话人2 01:45

嗯，基本上就是做业务，核心代码都是在业务上，像第三方的华夏银行、平安银行、中交轨迹，包括存储的那种识别啊。

说话人1 01:59

那这些模块他们之间是怎么交互呢？

说话人2 02:01

openfeign。

说话人1 02:06

比如说逻辑关系上。

说话人2 02:09

逻辑关系啊？

说话人1 02:10

对，比如说我一个用户进来，他需要怎么去操作，怎么去调用这些模块。

说话人2 02:18

我没明白你的问题，点是在哪里，你说一个用户从登陆一直走到后面是吧？

说话人1 02:25

对对对，比如说哪一条线您可以大概描述一下吗？

说话人2 02:30

系统登录，我们登录储存的token，然后用token进行验证的，有网关的，然后网关通过openfeign调用你其他的业务模块，然后走到业务模块有可能需要第三方支持的，像证件识别或者查询轨迹或者生成合同，也是用openfeign去调用其他的一些功能模块。我们系统里面主要是一个功能模块、一个业务模块。

说话人1 03:02

大概就是这样是吗？

说话人2 03:06

对对，然后存储的用redis或者平台mysql数据库了，看我们redis主要用的是做缓存的嘛。

说话人1 03:14

既然说到这里，您说一下这个项目中用到了哪些技术？

说话人2 03:21

redis了，然后mq了，然后一些微信支付了，然后阿里的OCR认证识别的，然后OSS存储的，基本上像微服务都有用到了，什么mybatis，还有一些 spring clooud啊，spring mvc啊，然后我们监听的，服务注册用的eureka了，然后网关用的gateway，基本上就这些吧。

说话人1 03:58

好的，这个项目它的预算是多少？

说话人2 04:01

项目没有什么预算，我们都是自研的，公司自己用的。

说话人1 04:05

公司自己用，哦，好的好的。

说话人2 04:08

他自研项目不是说不是跟别的接的外包的或者是干嘛的。

说话人1 04:13

明白了，好的。那这个项目有多少人来做呢？

说话人2 04:19

多的时候13个，然后去年因为疫情导致大货车在路上不让过，然后就裁了一部分裁了6个人，后面剩下6个人了。

说话人1 04:31

哦，多的时候13个左右？

说话人2 04:32

对，包括前端、后端，然后UI产品测试，总共加起来16个，我们

后端的比较多，真正写代码的参与代码的就是12个人吧，那其他边缘的测试跟 UI就没算进来。

说话人1 04:54

嗯，测试人员这些相对来说少一些。

说话人2 04:56

只有两个测试。

说话人1 05:00

好的，那这个项目它开发了多久呢？

说话人2 05:05

开发，他是一直在完善了，他之前是属于那种开票系统，补录的，把那个运单补录进来，后面又做了一个车货匹配的，然后车货匹配的，因为它步骤啊，装卸货这种操作比较繁琐嘛，最后用了二维码，然后用电子围栏基于中交的那种装卸货出发的嘛。

说话人1 05:34

等于说这个是公司需要的一个大的项目，然后它里面需要什么新的需求都会往里加。

说话人2 05:41

对对对，一直会往里面加，属于自研的项目。

说话人1 05:45

好的，那我明白了，就是一个长期在用的东西。

说话人2 05:47

对对对，现在还在维护的。

说话人1 05:51

嗯嗯，好的。那您开发中遇到印象比较深的问题，您是怎么解决的可以说一下吗？

说话人2 05:56

嗯……最难的就是对接中交轨迹嘛，因为中交轨迹拉取的是十几兆的那种轨迹，经纬度嘛，一个文本，然后我们存储的时候，因为我们拉中交轨迹是需要付费的嘛。

说话人1 06:13

嗯，这个中交是指什么呢？

说话人2 06:15

北斗星，我们中国的北斗，北斗就是那种定位系统嘛，那我们需要知道大货车有没有去拉货啊，有没有到达卸货地点，都是通过北斗去定位的。

说话人1 06:29

就是获取一个车辆的轨迹信息。

说话人2 06:32

对对对，然后当时这个是拉取是需要收费的嘛，我们需要存储这个，当时定了好多方案，就是把它当成文本存储，但是文本存储检索的时候会比较慢嘛。然后就是一辆一张表的这种方式，然后每次拉取的时候会覆盖之前的或者是连接之前的看搜索时间，然后再加生成另外一张表，当索引的这种去搜索，就是分库分表这种方式。然后第二个就是平安银行，平安银行对接也比较麻烦，因为它版本比较多，每一次版本都有一个编号嘛，如果用错了版本的话就导致付费，或者说直接平台他那边返回就是报异常嘛。我们当时看这个文档也看得够呛的。

说话人1 07:29

是您自己去查阅这些资料呢，还是？

说话人2 07:35

这个是我接手对接的嘛，我们以前我过来的时候是对接华夏银行，华夏银行就比较简单一点，他是直接转账，然后平安银行它是有个电子虚拟账户，而且实现公对私转账，那我们财务上面公对私转账是不允许的嘛。对，他就是规避了一个公对私转账的，就是电子虚拟账号，因为它电子虚拟账号是在我挂在我们公司名下的嘛，那这样转账就不属于是，不属于那种公对私的意义上面转账。

说话人1 08:10

等于说您一直是您在主动的去处理这些问题。

说话人2 08:17

嗯，平安银行的跟轨迹分解的是我这边，然后我们公司其实系统里面还有一个最困难的，就是一个风控组件，你一辆车子有没有跑，然后车速是多少，然后你装卸货是否超吨，然后到到达卸货地点是否有亏吨这种模式，我们有专门的组件在进行拦截。

说话人1 08:43

就是您遇到问题的时候都是您自己去解决？

说话人2 08:50

肯定的，我是这边的技术经理啊，我不解决，没人解决了呀。

说话人1 08:54

哦，您是经理，这样是吧，好的，那您会去优化自己的代码吗？

说话人2 09:00

有，我不但优化自己的代码，别人的代码也会看一下，因为我们这边，当时这边人员技术有参差不齐的嘛，有些人写的代码比较绕。

说话人1 09:14

就是您的组员都要经过您审核。

说话人2 09:20

对。

说话人1 09:21

然后您是自己一个人去完成这个重构代码的事情，还是让手下的人去做？

说话人2 09:28

我会看，他们提交git代码的时候，我们这边钉钉上面会有那个提醒嘛，我会主动看一下他们提交的代码。

说话人1 09:38

嗯，就是您觉得他们可以搞定的就会去让他们去做。

说话人2 09:42

嗯，大概是这样，但是我会时不时抽查一下他们代码，特别是做久了，你知道哪些人写的代码一定会有问题的，你就是着重注意一下，有些时候写的对接第三方的那种token失效的这种都是够呛的，连接了又连接的这种。

说话人1 10:01

是的是的。好的，那我这边就先了解到这里噢。

说话人2 10:10

嗯。

说话人1 10:11

好的，如果后续有消息的话，可能会有其他同事联系您。

说话人2 10:13

好。你这边是人事还是技术部啊？

说话人1 10:15

我这边是人事。

说话人2 10:18

噢，人事也会问这种技术相关的问题吗？

说话人1 10:20

问一些，大概了解一下。

说话人2 10:25

好好。

说话人1 10:27

好的，打扰您了，谢谢。

# 第三部分：面试准备进度自测表

说明：统一设置若干个时间节点，学员对面试准备进度进行自测。

自测分为三档：

A 档：能理解面试官提问的重点，使用自己的语言准确回答面试问题，并能在文档基础上拓展发挥 ；

B 档：能理解面试官提问的重点，比较准确的回答面试问题 ；

C 档：基础理解面试提问的重点，比较准确的回答面试问题，有少部分错误 ；

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **面试模块** | | **进度自测1** | **进度自测2** | **进度自测3** | **进度自测4** |
| 合格上岗 | 模块1：JavaSE基础编程 |  |  |  |  |
| 模块2：Java 面向对象编程 |  |  |  |  |
| 模块3：JavaSE 高级编程 |  |  |  |  |
| 模块4：Java 常见算法 |  |  |  |  |
| 模块5：Oracle / MySQL 数据库 |  |  |  |  |
| 模块6：Servlet 编程 |  |  |  |  |
| 模块7：Spring 框架 |  |  |  |  |
| 模块8：Mybatis 框架 |  |  |  |  |
| 模块9：SSM 框架开发 |  |  |  |  |
| 模块10：Spring Boot 框架开发 |  |  |  |  |
| 模块11：Git 管理项目 |  |  |  |  |
| 项目经验1 |  |  |  |  |
| 项目经验2 |  |  |  |  |
| 挑战高薪 | 模块12：VUE 框架开发 |  |  |  |  |
| 模块13：Redis 数据库 |  |  |  |  |
| 模块14：Nginx 服务 |  |  |  |  |
| 模块15：RabbitMQ 消息队列 |  |  |  |  |
| 模块16：Spring Cloud 微服务 |  |  |  |  |
| 模块17：Linux 管理 |  |  |  |  |