（一）Java 核心基础篇

1. 讲讲 Java 语言的三大特性。

Java 语言的最重要三大特征是：封装、继承、多态。

其中封装指的是把一个对象属性私有化，同时提供一些可以给外界访问该属性的方法。比如写一个类时会用 private 修饰属性，用 public 修饰的 set 和 get 方法提供外界访问和修改属性的方法。

继承是指使用已经存在的类作为基础建立新的类。子类拥有父类的所有属性和方法，但是父类中的私有属性和方法子类无法访问。子类可以拥有自己的属性和方法。

多态是指同一个行为具有多个不同表现形式或形态的能力，Java 中有两种多态的实现形式：继承和接口。比如 List list=new ArrayList()这种写法就是多态的一种实现，从代码上来看，就是父类的对象变量调用了子类或者调用了接口实现类。

2. String、StringBuilder、StringBuffer 的区别是什么?

String 类中使用 final 关键字修饰字符数组来保存字符串(Java9 以前)，因为使用了 final 修饰符，因此 String 是不可变的。每次改变 String 类型的值时，会生成一个新的 String 对象，然后将指针指向新的 String 对象。

StringBuilder 和 StringBuffer 都是继承 AbstractStringBuilder 类，在 AbstractStringBuilder 中也是使用字符数组保存字符串，但是没有用 final 关键字修饰，所以这两种对象都是可变的。StringBuffer 对字符串的一些操作方法加上了同步锁，因此是线程安全的。StringBuilder 没有对方法加锁，因此是线程不安全的。

3. 两个对象的 hashCode()相同，则 equals()也一定为 true，对吗?

不对，hashcode 其实就是对一个对象中的每个元素进行一次运算生成的结果值，两个不同的对象是有可能出现同一个 hash 值的。 Ma 和 NB 两个字符串不同，但是他们有相同的 hashcode 值。

4. this 和 super 在 Java 中的含义是什么。

this 关键字表示当前对象，在一个对象内部，可以通过 this.xxx 去访问该对象的属性或者方法。super 关键字可以理解为指向自己父类对象的指针，可以从子类访问父类的属性和方法。

5. 什么是重写？什么是重载?

重载指的是同样的方法名可以接受不同的参数类型和参数数量，以及给出不同的结果。重写指的是子类继承父类后，对同样的方法名有自己的处理逻辑，就可以重写父类的方法。

6. 简单讲讲你对 Java 中的异常的理解?

Java 中的异常都来自于 java.lang.Throwable 类，从异常类型上分，Java 中的异常可以分为 Exception 和 Error。Exception 异常可以被程序本身处理，Error 无法被程序处理。

Exception 异常又可以分为受检查异常和不受检查异常，所谓受检查异常是指那些在编程期间就需要把异常 try/catch 或 throws 出来的异常，不受检查异

常是指在编程期间不需要通过代码来显式地 catch 出来。

不受检查异常往往是程序员的代码逻辑疏忽导致，比如空指针异常，只需要在调用对象之前判断是否是空对象就可避免；下标越界异常只需要保证永远在下标范围内访问即可避免。

受检查异常必须在代码中通过 try/catch 或者 throws 才能通过编译。

7. 如果 for 循环中有一次循环被 try/catch 捕获异常，循环还会继续执行下去吗?

如果异常发生的位置在 try 代码块里，那么如果在 catch 里，没有主动抛出异常，并且 catch 的代码没有发生异常，循环继续。否则循环中止。循环是否继续执行下去取决于代码是否自己将异常处理了，如果程序处理了，不抛给 JVM，那么循环就不会终止。

8. 简单聊聊什么是深拷贝？什么是浅拷贝？

开发过程中，有时会遇到把现有的一个对象的所有成员属性拷贝给另一个对象的需求。如果只是单纯使用 clone 方法进行拷贝，只拷贝引用地址的动作就是浅拷贝。相反，如果拷贝一个对象时不是简单的将地址引用拷贝出来，而是新建了一个对象，这种方式就是深拷贝。

9. 什么是自动拆箱、什么是自动装箱

自动装箱是指将基本类型用对应的应用类型包装起来，自动拆箱是指将包装类型转化为对应的引用类型。以 Integer 为例，装箱的时候调用的是 Integer.valueOf(int)方法，拆箱的时候用的是 Integer 的 intValue 方法，算术运算会触发自动拆箱。

10. 抽象类和接口有什么区别

1、接口的方法默认是 public，所有的方法在接口中不能有实现（在 JDK8 时接口可以有默认方法和静态方法），抽象方法可以被 public、protected、defalut 修饰，但是不能被 private 和 final 修饰，抽象类可以有非抽象的方法。

2、接口中除了 static 和 final 变量以外不能有其他变量，抽象类中不一定。

3、一个类可以实现多个接口，但是只能实现一个抽象类，因为抽象类首先是一个类。

4、抽象类里的抽象方法必须全部被子类所实现，如果子类不能全部实现父类抽象方法，那么该子类只能是抽象类。同样，一个实现接口的时候，如不能全部实现接口方法，那么该类也只能为抽象类。

(二) Java 集合篇

1. List、Set、Map 之间的区别是什么?

List 和 Set 都继承于 Collection 接口，List 中的元素有序且可重复，Set 中的元素无序不可重复。Map 保存的是 Key-Value 形式的键值对。

2. ArrayList 和 LinkedList 的区别?

ArrayList 底层是结构是可变数组，LinkedList 底层结构是双向链表。因此在效率上，ArrayList 增删慢，改查快；LinkedList 增删快，改查慢。在扩容情况上，ArrayList 采用的是 1.5 倍扩容的方式，LinkedList 由于采用双层链表的底层结构，因此无需扩容。

3. Vector 和 ArrayList 的区别?

Vector 和 ArrayList 很像，底层也是一个数组，和 ArrayList 不同的是，Vector 是线程安全的，它使用了 synchronized 关键词。在扩容倍数上，ArrayList 是 1.5 倍，而 Vector 是 2 倍。

4. ArrayList 的扩容机制是怎样的?

当初始化时，ArrayList 中会初始化一个 Object 数组，容量为 0；当要添加数据时，初始容量变为 10；当第 11 个元素要进来时，容量扩容为 15；当第 16 个元素要进来时，容量扩容为 22。当每次要扩容时，都会扩容成原来容量的 1.5 倍。

5. HashMap 和 HashTable 有什么区别?

线程是否安全：HashMap 是线程不安全的，HashTable 是线程安全的，如果要保证线程安全，更建议你去使用 ConcurrentHashMap。

运行效率：HashMap 因为线程不安全，效率比 HashTable 高。

键值是否可以为空：HashMap 允许 key 和 value 为 null，Hashtable 不允许

扩容机制：

如果不设置初始容量：HashMap 初始容量为 16，每次扩容两倍，HashTable 默认的初始大小为 11，之后每次扩充为原来的 2n+1；如果设置初始容量：HashTable 会直接使用这个初始容量，HashMap 会将其扩容为 2 的幂次方。

底层结构：HashMap 在 JDK8 后，当链表长度大于 8 且当前数组容量大于

64 的情况下，会将链表转化为红黑树，减少搜索时间。而 HashTable 还是数组+链表的形式。

6. HashMap 和 HashSet 的区别，以及 HashSet 去重原理

HashMap 存储的是键值对，HashSet 存储的是对象

HashSet 底层调用的大多数都是 HashMap 的代码，HashMap 使用 key 计算 hashcode，HashSet 使用成员对象计算 hashcode。

HashSet 去重原理：当往 HashSet 中添加元素时，HashSet 会先计算 hashcode 值来和已经加入对象的 hashcode 值进行比较，如果没有相同的 hashcode，说明一定没有重复元素，则直接将对象添加到 hashcode 指定位置。如果有相同的 hashcode，还需要再通过 equals 比较（可能存在 hashcode 相同，但是对象不同的情况），如果 equals 方法相同，则不会把对象添加进来。

7. 说一下 HashMap 的实现原理?

HashMap 在 JDK1.7 和 JDK1.8 中不同：

JDK1.7：

JDK1.7 时底层是数组和链表组成的链表散列，添加元素时首先计算出 key 的 hash 值，然后把 hash&(n-1)计算处元素放置的位置（n 是此时链表的长度），如果此位置元素存在，就判断该元素和要放入的元素的 hash 值和 key 是否相同，相同的话直接覆盖，不相同的话通过链表解决冲突。

JDK1.8

HashMap 中维护了 Node 类型的数组 table，当 HashMap 创建对象时，设

置负载因子为 0.75，table 还是 null。

当第一次添加元素时，将 table 的容量设置为 16，临界值设置为 12

每次添加元素调用 putVal 方法：

1.将 key 的 hash 值和 table 容量-1 进行与运算，得到索引值

2.判断该存放位置上是否有元素，如若没有元素则直接放上去；如果该索引位置已存在元素，则继续判断

3.如果该位置的元素和添加元素相等，则直接覆盖，如果不相等，则继续判断是链表结构还是树状结构，按照相对应的方式添加。

如果添加的数量大于临界值，执行 resize 方法对容量双倍扩容。并打乱顺序重新排列。

8. 简单讲一下 ConcurrentHashMap 实现线程安全的方式?

ConcurrentHashMap 在 JDK1.7 和 JDK1.8 中实现方式发生了很大改变：

JDK1.7：

底层采用分段数组+链表的方式实现，他对整个桶数组做个分割分段的操作，每一段都是一个 segment，没一把锁锁住每个 segment 中的数据，多线程访问容器里不同数据段的数据，就不会存在锁竞争，提高并发访问率。

JDK1.8：

在 JDK1.8 中，ConcurrentHashMap 的底层数据结构变成了 Node 数组+链表+红黑树的数据结构来实现，并发控制使用 synchronized 和 CAS 来操作。在 JDK1.8 中只会锁定当前链表或者红黑树的首节点，hash 不冲突就不会产生并发的问题，大大提高了并发效率。

(三) Java 虚拟机

1. 介绍一下 JVM 的内存区域。

JVM 的内存区域可以分为方法区、虚拟机栈、本地方法栈、堆和程序计数器。

程序计数器的作用可以看成是当前线程所执行的字节码的行号指示器 Java 虚拟机的多线程是通过线程轮流切换来分配处理器执行时间的方式实现的，为了线程切换之后能恢复到正确的执行位置，每个线程就需要一个独立的程序计数器。

java 虚拟机栈线程私有，每个方法被执行的同时都会创建一个栈帧用于存放局部变量表、操作数栈、动态链接、方法出口等信息。

本地方法栈的功能和虚拟机栈类似，本地方法栈为虚拟机用到的 Native 方法服务，一个 Native Method 就是一个 Java 调用非 Java 代码的接口，本地方法栈也会抛出 StackOverFlow 和 OutOfMemoryError 异常

JVM 堆可以说是 Java 虚拟机中所管理的内存最大的一块，堆被所有线程共享，虚拟机启动的时候创建。堆中存放的对象实例的数组。几乎所有的对象实例以及数组都在堆上分配，堆也是垃圾回收器管理的主要区域，堆可以处于物理上不连续的内存空间中，只要逻辑上是连续的即可，这和磁盘空间很相似。当一个堆无法再扩展时，会抛出 OutOfMemoryError 异常。

方法区也是线程共享的内存区域，用于存放已经被虚拟机加载的类信息、常量、静态变量、即时编译器编译后的代码等数据，Java 虚拟机对方法区的限制十分宽松，“除了”和 Java 堆一样不需要连续的内存外，还可以选择不实现垃圾回收。

2. 方法区、永久代、元空间的区别是什么?

方法区是 JVM 规范中的一个概念，在不同的 Java 虚拟机以及不同版本的 Java 虚拟机中，方法区都有各自的实现。以最流行的 HotSpot 虚拟机为例，

在 JDK1.8 之前, 方法区的实现叫做永久代, 放在 JVM 内存之中。到了 JDK1.8, 方法区的实现变成了元空间，放在直接内存中。

3. 简单聊聊 JVM 内存分配。

为了更加高效的回收垃圾，JVM 将堆内存划分为了多个 generation（代）。新生代和老年代是垃圾回收最主要的区域，新生代又被划分为 Eden 空间，From Survivor 空间和 To Survivor 空间。如此划分的目的是为了更好地进行内存回收。

新生代和老年代都在堆内存中，新生代和老年代所占的默认比例为 1：2，其中一个新生代又由一个 Eden 区+两个 survivor 区组成，默认比例为 8：1：1。

4. JVM 如何判断一个对象是垃圾对象?

现在绝大部分 Java 虚拟机通过根搜索算法（GC Roots Tracing）来判断一个对象是否存活。

这个算法的思路就是通过一系列名为 GC Roots 的对象作为起始点，从这些节点向下搜索，当 GC Roots 到达不了这个某个对象时（或者说某个对象没有被任何其他对象所引用），就证明这个对象是不可用的，这些对象会被判定为需要回收的对象。

在 Java 语言中，可作为 GC Roots 的对象包括下面这些：

1. 虚拟机栈（栈帧中的本地变量表）中引用的对象
2. 方法区中的类静态属性引用的对象
3. 方法区中的常量引用的对象
4. 本地方法栈（Native 方法）引用的对象

5. 什么是类的双亲委派模型?

所谓双亲委派模型，就是指一个类接收到类加载请求后，会把这个请求依次传递给父类加载器（如果还有的话），如果顶层的父类加载器可以加载，就成功返回，如果无法加载，再依次给子加载器去加载。

6. 介绍几种常见的垃圾收集器。

Serial 是一个单线程的收集器，Serial 的特点是它在进行垃圾收集时，必须 Stop the World ，意思就是当这个垃圾收集器开始工作时，必须停止其他所有的工作线程。

ParNew 垃圾收集器是 Serial 的多线程版本，使用多条线程进行垃圾收集。除此之外，和 Serial 基本相同，ParNew 在多线程收集垃圾时依旧需要 Stop the World。

Parallel Scavenge 也是新生代收集器，也同样是多线程的收集器，但是和 ParNew 不同，Parallel Scavenge 收集器关注的是一个可控制的吞吐量（Throughput）。所谓吞吐量指的是 CPU 用于运行代码的时间和 CPU 总消耗的时间比例。

CMS（Concurrent Mark Sweep）收集器是一种以获取最短回收停顿时间为目标的收集器。CMS 是基于标记-清除算法的老年代垃圾回收器，CMS 是目前应用最广泛的老年代垃圾回收器。

7. 讲讲你对 CMS 垃圾收集器的理解

CMS（Concurrent Mark Sweep）收集器是一种以获取最短回收停顿时间为

目标的老年代收集器。CMS 是基于标记-清除算法的老年代垃圾回收器，CMS 是目前应用最广泛的老年代垃圾回收器。

CMS 的运行过程主要分为四个阶段：

初始标记：标记 GC Roots 可以直接关联到的对象，速度很快（Stop the World）。

并发标记：根搜索算法的过程。
重新标记：为了修正并发标记期间，因程序运行导致标记产生变动的对象。（Stop the World）。

并发清除：清除垃圾。

CMS 垃圾收集器的主要有点为并发收集、并发清除、低停顿。相比较前几代的垃圾收集器，CMS 垃圾收集器给用户的体验更好，因为它追求的是最短的回收停顿时间。

缺点是对 CPU 资源十分敏感、无法处理浮动垃圾、产生空间碎片。

8. 双亲委派模型是绝对的吗?

双亲委派模型并非是绝对的，spi 机制就可以打破双亲委派模型。最典型的就是 JDBC。

Java 提供了一个 Driver 接口用于驱动各个厂商的数据库连接，在 JDK1.8 中 Driver 接口位于 JAVA_HOME 中 jre/lib/rt.jar 中，应该由 Bootstrap 类加载器进行加载；在 JDK 1.8 之后，由于类加载器有调整，这个接口将由 PlatformClassLoader 平台类加载器进行加载。根据类加载机制，当被加载的类引用了另外一个类的时候，虚拟机就会使用加载该类的类加载器加载被引用的类，因此如果其他数据库厂商定制了 Driver 的实现类之后，按理说也得把这个实现类放到启动类加载器加载的目录下，这显然是很不合理的，类加载器加载的目录下放的是 JDK 自己的一些类。

于是 Java 提供了 spi 机制，即使 Driver 由启动类加载器去加载，但是他可以让线程上下文加载器（Thread Context ClassLoader）去请求子类加载器去完成加载，默认是应用程序类加载器。但是这确实破坏了类加载机制。

（四）并发编程

1. 进程和线程有什么区别?

进程是系统运行程序的基本单位，一个进程就是一个执行中的程序，进程是独立运行的个体。

线程是比进程更小的执行调度单位，一个进程在执行中可以产生多个线程，多个线程共享同一块内存空间和系统资源。

2. 并行和并发有什么区别?

并行是指两个或者多个事件在同一时刻发生，并行的关键是有同时处理多个任务的能力；

并发是指两个或多个事件在同一时间间隔发生，并发的关键是有处理多个任务的能力，不一定要同时。

并行和并发的本质区别在于能否“同时”处理任务。

3. 了解什么是守护线程吗?

守护线程是运行在后台的一种特殊进程。它独立于控制终端并且周期性地执行某种任务或等待处理某些发生的事件。在 Java 中垃圾回收线程就是特殊的守护线程。

专门用于服务其他的线程，如果其他的线程（即用户自定义线程）都执行完毕，连 main 线程也执行完毕，那么 jvm 就会退出（即停止运行），此时连 jvm 都停止运行了，守护线程当然也就停止执行了。

4. 什么是死锁？如何解除死锁？

死锁是指两个或两个以上的线程在执行过程中，由于竞争资源或者由于彼此通信而造成的一种阻塞的现象。

死锁的产生有四个条件：

互斥条件：该资源任意一个时刻只由一个线程占用。

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。

不可剥夺条件:线程已获得的资源在未使用完之前不能被其他线程强行剥夺，只有自己使用完毕后才释放资源。

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。

只要这四个条件中任何一个无法满足，死锁就解除了。

5. 创建线程的方式有哪些？

在 Java 中，可以通过四种方式创建线程：

继承 Thread 类 实现方法 run() ： 不可以抛异常，无返回值。

实现 Runnable 接口的 run() 方法 ： 不可以抛异常，无返回值。

实现 Callable<T> 接口,接口中要覆盖的方法是 public <T> call() 注意:此方法可以抛异常,而前两种不能 而且此方法可以有返回值。

第四种通过线程池创建线程来使用。

6. 讲一下线程池的工作流程。

当提交任务后，首先判断当前线程数是否超过核心线程数，如果没超过则创建新线程执行任务，否则判断工作队列是否已满，如果未满则将任务添加到队列中，否则判断线程数是否超过最大线程数，如果未超过则创建线程执行任务，否则执行拒绝策略。

7. 说说你对 Java 内存模型的理解。

Java 内存模型是定义了线程和主内存之间的抽象关系，Java 内存模型中规定了所有的变量都存储在主内存中，每条线程还有自己的工作内存，线程对变量的所有操作都必须在工作内存中进行，而不能直接读写主内存中的变量。

每个线程拥有一个自己的私有工作内存，需要变量时从主内存中拷贝一份到工作内存，如果更新过变量之后再将共享变量刷新到主内存。

8. 什么是指令重排序。

为了使处理器内部的运算单元能尽量被充分利用，处理器可能会对输入的代码进行乱序执行优化，处理器会在计算之后将乱序执行的结果重组，并确保这一结果和顺序执行结果是一致的，但是这个过程并不保证各个语句计算的先后顺序和输入代码中的顺序一致。这就是指令重排序。

简单来说，就是指在程序中写的代码，在执行时并不一定按照写的顺序。

9. volatile 有什么作用，你在哪里用到过它?

并发编程的三大特性：可见性、原子性和有序性。volatile 保证了可见性和有序性，但是不保证原子性。在写双重校验单例模式时，就用到了 volatile 。

10. volatile 使用时有要注意的地方吗?

如果一个程序中用了大量的 volatile，就有可能会导致总线风暴，所谓总线风暴，就是指当 volatile 修饰的变量发生改变时，总线嗅探机制会将其他内存中的这个变量失效掉，如果 volatile 很多，无效交互会导致总线带宽达到峰值。

11. 知道多线程中的 CAS 操作吗?

CAS：compare and swap，比较且交换。使用 CAS 操作可以在没有锁的情况下完成多线程对一个值的更新。CAS 的具体操作如下：

当要更新一个值时，先获取当前值 E，计算更新后的结果值 V（先不更新），当要去更新这个值时，比较此时这个值是否还是等于 E，如果相等，则将 E 更新为 V，如果不相等，则重新进行上面的操作。

12. CAS 操作会不会遇到问题，如果有问题如何解决?

CAS 操作可能会出现 ABA 问题，ABA 问题即要去比较的这个值 E，经过多个线程的操作后从 0 变成 1 又变成了 0。此时虽然 E 值和更新前相等，但是还是已经被更新了。

解决办法是对 E 值增加一个版本号，每次要获取数据时将版本号也获取，每次更新完数据之后将版本号递增，这样就算值相等通过版本号也能知道是否经过修改。

13. 谈谈你对 synchronized 的认识

synchronized 可以保证在同一时刻，只有一个线程可以执行某个方法或某个代码块,synchronized 把锁信息存放在对象头的 MarkWord 中。

synchronized 作用在非静态方法上是对方法的加锁，synchronized 作用在静态方法上是对当前的类加锁。

14. synchronized 的锁升级机制有了解吗?

在早期的 jdk 版本中，synchronized 是一个重量级锁，保证线程的安全但是效率很低。后来对 synchronized 进行了优化，有了一个锁升级的过程：无锁态（new）-->偏向锁-->轻量级锁（自旋锁）-->重量级锁。

当没有任何线程去访问这个对象的时候，没有任何线程对资源进行锁定，这个时候处于无锁状态。

当有一个线程去请求时，就把这个对象 MarkWord 的 ID 改为当前线程指针 ID（JavaThread），只允许这一个线程去请求对象。锁状态也就变成了偏向锁。

当有其他线程也去请求时，就把锁升级为轻量级锁。每个线程在自己的线程栈中生成 LockRecord，用 CAS 自旋操作将请求对象 MarkWordID 改为自己的 LockRecord，成功的线程请求到了该对象，未成功的对象继续自旋。

如果竞争加剧，当有线程自旋超过一定次数时（在 JDK1.6 之后，这个自旋次数由 JVM 自己控制），就将轻量级锁升级为重量级锁，线程挂起，进入等待队列，等待操作系统的调度。

15. 除了 synchronized 外，还有什么锁?

ReentrantLock 和 synchronized 一样都可以保证同一时刻只有一个线程可以访问临界资源。ReentrantLock 通过 lock() 方法和 unlock() 方法控制临界资源。

16. 可重入、公平锁分别表示什么意思?

可重入锁的意思就是对一个对象可以实现多次加锁。

如果是公平锁，当有新的线程来的时候，他会先去看看等待队列中有没有等待的线程，如果有，则乖乖跑到最后去排队。如果是非公平锁，当有新的线程来的时候，不管等待队列有没有等待的线程，直接去和被唤醒的锁竞争，如果竞争

失败了，则乖乖跑到队列最后去排队，否则就直接占有锁。

17. 你还了解 JUC 包中的哪些类?

资源的分配方式有两种，一种是独占，比如 ReentrantLock，另外一种是共享，比如 Semaphore、CyclicBarrier 以及 CountDownLatch，这些都是 JUC 包下的类。

18. 谈谈你对 ThreadLocal 的理解。

ThreadLocal 是保存在每个线程本地的数据，ThreadLocal 提供了线程局部变量，即每个线程可以有属于自己的变量，其他线程无法访问。如果创建了一个 ThreadLocal 变量，那么访问这个变量的每个线程都会有这个变量的本地副本。每个线程可以通过 set() 和 get() 方法去访问 ThreadLocal 变量。

19. ThreadLocal 有什么问题吗? 为什么?

Thread 存在内存泄露的问题，内存泄漏是指程序中已经动态分配的堆内存由于各种原因未释放或者无法释放，导致内存的浪费。

ThreadLocal 内存泄漏的根源是因为 ThreadLocalMap 拥有和 Thread 相同的生命周期，因此如果不手动 remove 掉 entry 对象，只要线程还在运行，就会发生内存泄漏。

（五）Spring 核心知识点

1. 你知道 Spring 框架中有哪些重要的模块吗?

常用的模块有以下几个：

Spring Core：提供了 Spring 框架的基本工程，主要提供了依赖注入的能力。

Spring Test：提供了对 JUnit 测试的支持。

Spring AOP：提供面向切面的能力。

Spring Aspects：提供了 AspectJ 的集成支持。

Spring JDBC：提供了一个简单有效的 JDBC 连接支持。

Spring ORM：提供了对其他 ORM 框架的支持，如 Hibernate，都可以和 Spring 进行良好的结合。

Spring Web：提供创建 Web 应用程序的能力。

2. 谈谈你对 IOC 的认识

IOC 并不是一种技术，它是一种思想，即控制权反转的思想，DI（依赖注入）则是 Spring 实现 IOC 的方法。Spring 容器在初始化时根据配置文件或元数据创建和组织对象存入容器中，需要使用时再从 IOC 容器中获取。

IOC 从本质上讲，就是把原本由程序员创建的对象的这个动作交给 Spring 去实现，程序员无需再去管理对象的创建，这种方式可以大大减少系统的耦合性。没有 IOC 之前，对象的创建和对象间的依赖关系都完全编码在程序中，使用 IOC 之后，对象的创建由程序自己控制，使得程序解耦合。

3. 谈谈你对 AOP 的认识

AOP(Aspect-Oriented Programming:面向切面编程)翻译为面向切面编程，能够将那些与业务无关的逻辑封装起来，从而减少系统的重复代码，降低模块之间的耦合度，同时也具备着更强的可扩展能力。

Spring AOP 基于动态代理实现，如果被代理的类实现了某个接口，Spring AOP 会用 JDK Proxy 去创建代理对象，如果被代理对象没有实现接口，Spring AOP 会使用 Cglib 创建代理对象。除此之外，Spring AOP 集成了 AspectJ，AspectJ 是目前 Java 领域应用最广泛的 AOP 框架。

4. 在实际写代码时，有没有用到过 AOP？用过的话是在什么地方或者什么场景下？

我曾经写过一个自定义日志的功能是通过 AOP 实现，通过 AOP 做到日志与业务逻辑分离，通过 AOP 在业务逻辑执行前写日志，也可以在业务逻辑执行后写日志，而不用改动已经写好的业务逻辑代码。

5. 说说你对 Spring 中的 Bean 的理解

Spring 中的 Bean 简单来讲就是一个个被 Spring 容器管理的 Java 对象，我们写了一个类之后，这个类只是一个单纯的 Java 类，可以通过 new() 的方式去创建它。当我们把这个类添加到 Spring 的容器里之后，这个类就变成了 Bean，由 Spring 容器管理，可以通过自动注入的方式去使用。

6. 有哪些方法往 Spring 容器中添加 Bean。

常用的添加 Bean 的方式主要有以下几种方法。

@Bean： 写一个普通的类时最常用的添加 Bean 的方式

@ComponentScan + @Controller @Service @Component @Repository：@ComponentScan 用来扫描后面的四个注解，被后面四个注解修饰的类如果被@ComponentScan 扫描到，就会注册到 Spring 容器中。

@Import：通过导入的方式注入 Bean。

@ImportBeanDefinitionRegister：和 Import 类似，可以指定 Bean 的名称。

7. Spring 中常用的注解有哪些?

@Required

此注解用于 bean 的 setter 方法上。表示此属性是必须的，必须在配置阶段注入。

@Autowired

此注解用于 bean 的 field、setter 方法以及构造方法上，显式地声明依赖。根据 type 来注入。

@Configuration

此注解用在 class 上来定义 bean。其作用和 xml 配置文件相同，表示此 bean 是一个 Spring 配置。

@Controller

此注解使用在 class 上声明此类是一个 Spring controller，是@Component 注解的一种具体形式。

@RequestBody

此注解用在请求 handler 方法的参数上，用于将 http 请求的 Body 映射绑定到此参数上。HttpMessageConverter 负责将对象转换为 http 请求。

8. 什么是事务?

所谓事务就是指这个方法中的所有持久化操作要么都进行，要么都不进行。比如订单系统中生成订单和库存删减两个动作要么同时发生，要么都不发生，这就是事务操作，如果生成订单失败就不能进行接下来的库存删减操作。

9. Spring 如何对事务进行管控?

在 Spring 中通过 Transactionl 注解让被修饰的方法以事务的方式运行，Transactionl 是 AOP 的应用之一。Spring 的事务有五种隔离级别和七种传播属性（能引导面试官继续问下去）。

10. Transaction 在哪些情况下可能会失效?

当用 Transactional 修饰非 public 方法时，Transactional 注解是不会生效的。

如果在一个类的内部调用了事务方法，Transactional 也不会生效。

默认情况下只会在捕获了 RuntimeException 后 Transactional 注解才会生效，如果在代码中捕获了异常后未抛出，则 Transactional 失效。

（六）MyBatis 核心知识点

1. 谈谈你对 MyBatis 的认识?

Mybatis 是一个半 ORM（对象关系映射）框架，它内部封装了 JDBC，开发时 只需要关注 SQL 语句本身，不需要花费精力去处理加载驱动、创建连接、

创建 statement 等繁杂的过程。程序员直接编写原生态 sql，可以严格控制 sql 执行性 能，灵活度高。

MyBatis 可以使用 XML 或注解来配置和映射原生信息，将 POJO 映射成数据库中的记录，避免了几乎所有的 JDBC 代码和手动设置参数以及获取结果集。

通过 xml 文件或注解的方式将要执行的各种 statement 配置起来，并通过 java 对象和 statement 中 sql 的动态参数进行映射生成最终执行的 sql 语句，最 后由 mybatis 框架执行 sql 并将结果映射为 java 对象并返回。

简单来说，MyBatis 是一个能将传统 JDBC 变得很方便和易于维护的框架。我们只需要提供 SQL 语句，而建立连接、创建 Statement、处理 JDBC 异常等工作都可以交给 MyBatis 去做。

2. `#{}`和`${}`有什么区别?

#{}是预编译处理，MyBatis 在处理#{}时，它会将 sql 中的#{}替换为?，然后调用 PreparedStatement 的 set 方法来赋值;

${}是字符串替换，MyBatis 在处理${}时,它会将 sql 中的${}替换为变量的值。

3. xml 映射文件中，除了 select、insert、update、delete 这些，还用过什么标签?

还用过<resultMap>、<parameterMap>、<sql>、<include>等标签。另外还有动态 Sql 中的 trim、where、set、foreach、if、choose、when、otherwise 等标签。

4. 如果数据库字段和 java 实体类不一样，如何解决这个问题?

如果数据库字段和 java 实体类不一样，可以通过 resultMap 标签进行字段和属性之间的映射。

5. association 和 collection 分别在哪些场景中会遇到?

对象的关联（多对一）使用 association，集合的关联（一对多）使用 collection。

6. 什么是 MyBatis 中的动态 Sql，你用过哪些?

动态 Sql 中的标签有 trim、where、set、foreach、if、choose、when、otherwise 等等，平常用的比较多的是 where、if、foreach 这三个，比如使用 if 来判断当对象中某个属性不为空时才加上 if 中的 sql 片段。

7. MyBatis 中的缓存有哪几种，分别介绍一下。

MyBatis 系统中默认定义了两级缓存：一级缓存和二级缓存：MyBatis 一级缓存是一个 SqlSession 级别，Sqlsession 只能访问自己的一级缓存的数据。二级缓存是跨 sqlSession，是 mapper 级别的缓存，对于 mapper 级别的缓存不同的 sqlsession 是可以共享的。

8. MyBatis 二级缓存的清除策略有了解吗，介绍几个。

MyBatis 中默认的清除策略时 LRU，即最近最少使用，会移除最长时间不被使用的对象。另外还有三种：

FIFO – 先进先出：按对象进入缓存的顺序来移除它们。

SOFT – 软引用：基于垃圾回收器状态和软引用规则移除对象。

WEAK – 弱引用：更积极地基于垃圾收集器状态和弱引用规则移除对象。

(七) Spring Boot 核心知识点

1. 什么是 SpringBoot 框架?

SpringBoot 是一个可以轻松创建独立的、生产级的基于 Spring 的应用程序的框架，SpringBoot 框架将 Spring、MyBatis、SpringMVC 中的一系列配置进行了简化，只需要使用最简单的配置和注解就能快速开发应用程序。

2. SpringBoot 框架和 Spring 相比有哪些优势?

SpringBoot 相比于 Spring 的一个很大进步点在于不用再手动配置一系列配置文件，SpringBoot 提供了配置应用程序和框架所需要的基本配置，这就是自动装配。同时 SpringBoot 还简化了编码，简化了部署。

3. 什么是 SpringBoot Starter?

SpringBootStarter 是 Spring 官方提供的一系列依赖包，只需要引入一个 starter 依赖，就能实现一系列的功能。比如当引入了一个

spring-boot-starter-web 依赖。SpringBoot 框架就会自动引入 web 环境所有的依赖，并且自动配置，SpringBoot 将所有的功能场景，都变成一个个启动器，想要用什么功能，只需要找到对应的启动器就可以了。

4. SpringBoot 中的配置文件有哪些?

SpringBoot 提供了三种格式的配置文件，yml、yaml 和 properties，当三个文件都存在的时候，三个都会执行，执行顺序是 yml>yaml>properties，如果三者存放的配置有重合的地方，则后执行的覆盖前面执行的。假若三者都存放 port，最终访问 properties 中的 port。

5. 讲讲 SpringBoot 的自动装配原理。

SpringBoot 的自动装配原理可以分为下面七步：

1、SpringBoot 在启动的时候，从类路径下/META-INF/spring.factories 获取指定的值；

2、将这些自动配置的类导入容器，自动配置就会生效，进行自动配置!

3、以前我们需要自动配置的东西，现在 SpringBoot 帮我们做了

4、自动配置的东西都在 spring-boot-autoconfigure.jar 这个包下

5、它会把所有需要导入的组件，以类名的方式返回，这些组件就会被添加到容器；

6、容器中也会存在非常多的 xxxAutoConfiguration 的类（@Bean），就是这些类给容器中导入了这个场景需要的所有组件；并通过@Configuration 自动配置。

7、有了自动配置类，免去了我们手动编写配置文件的步骤。

6. SpringBoot 读取配置文件的方式有哪些?

第一种读取配置的方式是通过 @Value 实现, 这种方式是直接将注解加在变量上的。

第二种读取配置的方式是通过 @ConfigurationProperties 实现，该注解可以使得配置文件中的变量转换成一个类。

7. SpringBoot 可以通过什么方式检测请求的入参?

可以通过添加拦截器或者过滤器，在请求到达时立刻做处理。

8. 讲讲拦截器和过滤器的区别?

两者的相同点是拦截器与过滤器都是体现了 AOP 的思想，对方法实现增强，都可以拦截请求方法。

区别在于过滤器属于 Servlet 级别，拦截器属于 Spring 级别；

并且两者的执行顺序不同：首先当一个请求进入 Servlet 之前，过滤器的 doFilter 方法进行过滤，进入 Servlet 容器之后，执行 Controller 方法之前，拦截器的 preHandle 方法进行拦截, 执行 Controller 方法之后, 视图渲染之前，拦截器的 postHandle 方法进行拦截，请求结束之后，执行拦截器的 postHandle 方法；

过滤器基于函数回调方式实现，拦截器基于 Java 反射（动态代理）机制实现。

9. 用过注解吗？知不知道如何自定义一个注解

注解是一种能被添加到 Java 代码中的元数据，类、方法、变量、参数和包

都可以用注解来修饰。注解对于它所修饰的代码并没有直接的影响。

自定义注解主要有两步：

声明一个注解：通过@interface 声明一个注解，并且选择性地用四个元注解修饰该注解，分别是 @Target、@Retention、@Documented、@Inherited。

编写具体的切面实现类，也就是想要注解实现的功能。在定义切点时，让切点指向自定义注解即可。

(八) MySQL 核心知识点

1. 数据库的三大范式有了解吗?

设计关系数据库时，遵从不同的规范要求，设计出合理的关系型数据库，这些不同的规范要求被称为不同的范式，越高的范式数据冗余度越低。实际开发中涉及到的范式一般有三种：第一范式、第二范式、第三范式。

如果数据库中的每一列属性都是不可分解的原子值，那么说明这个数据库满足第一范式。

第二范式在第一范式的基础上，消除了非主属性对主属性的部分函数依赖。简单来讲，就是表中的每一列都要和主键有关，而不能只与主键的某一部分相关。

第三范式在第二范式的基础上，消除了非主属性对主属性的传递函数依赖，简单来讲，就是每一列数据都和主键直接相关，而不能间接相关。

2. MySQL 中主流的存储引擎有哪些，你最常用的是哪一个?

MySQL 中目前主流的存储引擎有 MyISAM、Innodb 等，我使用 MySQL 时最常用的是 Innodb。

3. 讲讲 InnoDB 和 MyISAM 之间的差异。

MyISAM 强调的是性能，每次查询具有原子性,其执行速度比 InnoDB 类型更快，但是不提供事务支持。但是 InnoDB 提供事务支持事务，外部键等高级数据库功能。 具有事务(commit)、回滚(rollback)和崩溃修复能力(crash recovery

capabilities)的事务安全(transaction-safe (ACID compliant))型表。

MyISAM 只支持表级锁、而 Innodb 即支持表级锁、又支持行级锁，默认是行级锁。

MyISAM 不支持外键，Innodb 支持外键

4. SQL 的事务隔离级别有了解吗，MySQL 默认的事务隔离级别是哪个?

SQL 标准定义的事务隔离级别分为四种：读未提交（read-uncommitted）、读已提交（read- committed）、可重复读（repeatable- read）、可串行化（Serializable）。MySQL 默认的事务隔离级别是可重复读。

5. 针对一条慢 SQL，通常会怎样去优化它?

如果遇到一条慢 SQL，我首先会用 explain 关键字去分析该 SQL，大部分情况下慢 SQL 的原因都是没有走索引导致，然后根据结果去优化 SQL 语句。

6. 简单讲讲你对 explain 的认识。

通过 explain 关键字可以看到 SQL 语句的执行过程中的一系列问题，只需要在执行的 SQL 前加上 explain 即可。explain 关键字的输出结果可以看出该条 SQL 语句的执行情况，比如通过 type 字段的结果就可以知道达到的索引级别，索引的优化一般到 ref 为止。

7. 在哪些情况下索引可能会失效?

复合索引不要跨列或无序使用（最佳左前缀），否则索引失效。
不要在索引上进行任何操作（计算、函数、类型转换），否则索引失效。

like 尽量以“常量”开头，不要以%开头，否则索引失效。

尽量不要使用类型转换（显示、隐式），否则索引失效。
两个表或者字段编码格式不同导致索引失效。

8. InnoDB 的索引数据结构有了解吗？介绍一下。

Innodb 默认使用 B+ 树作为索引的数据结构，它具有这样的几个特点：

1、数据都存储在叶子节点中、非叶子节点不存储数据，只存储索引。

2、叶子节点中包含所有的索引。

3、每个小节点的范围都在大节点之间。

4、叶子节点用指针相连，提高访问性能，比如条件是>或者 < （MySQL 中的 B+ 树叶子节点中的指针是双向指针）。

9. InnoDB 的主键索引和其他索引结构是一样的吗?

InnoDB 属于聚簇索引，即叶子节点包含了完整的数据记录。非主键索引和主键索引略有不同，非主键索引的叶子节点存储的是主键的 key 值。