# 面试题集合

# Java 基础

## 1. 概念

### 1). 什么是 JDK 什么是 JRE？

JDK：Java 开发工具包

JRE：Java 运行环境

### 2). 解释面向对象、面向对象的特征。

建议用例子去回答

抽封继多

### 3). 继承有何用处？

从代码利用上去回答

### 4). 谈谈你对接口的理解，你在你的项目中有使用到接口吗？如果有，为什么要使用接口？

从代码可维护性、规范代码、合作开发、可扩展性和灵活性上去回答

### 5). Java 语言的运行机制

Java 既不是编译型语言也不是解释型语言，它是编译型和解释型语言的结合体。首先采用通用的 java 编译器将 Java 源程序编译成为与平台无关的字节码文件（class 文件），然后由 Java 虚拟机对字节文件解释执行

### 6). 对象与类之间有何关系？

### 7). 什么是 JVM？有什么作用？工作机制如何？

JVM 是一个虚构出来的计算机，可在实际的计算机上模拟各种计算机功能，JVM 有自己完善的硬件架构，如：处理器、堆栈、寄存器等，还有相应的指令系统。

JVM 是由 Java 字节码执行的引擎，为 java 程序的执行提供必要的支持，它还能优化 java 字节码，使之转换成效率更高的机器指令。JVM 屏蔽了与具体操作系统平台相关的信息，从而实现了 Java 程序只需要生成在 JVM 上运行的字节码文件，就可以在多种平台上不加修改地运行。JVM 中类的装载是由类加载器（CLassLoader）和它的子类来实现的。CLassLoader 是 java 运行时一个重要的系统组件，负责在运行时查找和装入类文件的类。

操作系统装入 JVM 是通过 JDK 中的 java.exe 来实现，主要通过以下几个步骤完成：

a、 创建 JVM 装载环境和配置

b、装载 jvm.dll

c、 初始化 jvm.dll

d、调用 JNIEnv 实例装载并处理 class 类

e、 运行 Java 程序

## 2. 语法

**1). Object b = "4324"; b.getClass()输出的结果为？**

String

**2). 将一个时间写成 2012-4-4 的格式，用的是（ ）类的（ ）方法，参数为（ ）**

**3). int a=1,b=3,c=6,d=9; (a>b)?a:(c>d)?c:d 的输出结果是什么？**

**4). switch 能接受哪些数据类型？**

1.6：byte、short、int、char、Enum

新：增加 String

**5). 2<<8 的值为多少？**

**6). 一个".java"源文件中是否可以包括多个类（不是内部类）？有什么限制？**

可以，但是只能有一个类使用 public 修饰。

**7). finalize、final、finally 有何区别？**

finalize 是 Object 上的方法名，不是关键字，而后两者是关键字。

final 是修饰符，而 finally 是用于异常处理的语句

**8). 简述 this 与 supper 的异同**

相同点：都是调用当前类的实例的方法和字段，都可以用来调用构造方法

不同点：一、this 可以调用当前类的任意字段和方法，包括继承的字段和方法。但 super 只能调用由父类定义的字义的字段和方法；二、在调用构造方法时，this 是调用当前类定义的重载构造方法，而 super 是调用父类定义的构造方法。

**9). Java 当中有哪些访问修饰符？各是什么含义？**

private：只能在当前类中调用

package：包含 private，并可以在当前包中调用

protected：包含 package，并可以在异包的子类调用

public：任意位置都可以调用

### 10). String 可以被继承吗？为什么？

不可以，因为它被 final 修饰。

### 11). 简述 abstract class 和 interface 的异同。

相同点：它们都可以有抽象方法。

不同点：语法不同、使用方式不同、设计理念不同（is a ,like a）、使用关系

### 12). 用最快的编码方式计算 16 乘以 8 的结果。

移位运算符

### 13). 短路运算符与逻辑运算符有何区别？

短路运算符当前一个表达式已经可以决定全部运算结果，不会计算后一个表达式。

而逻辑运算符无论第一个表达式计算结果如何，都会计算全部表达式。

所以通常短路运算符效率更高，也更常使用。

### 14). 重载（Overload）与重写（Override）有何异同?

相同点：都是多态的体现，且都是一系列方法名相同的方法构成。

不同点：重载必须是在同一个类中（包括继承方法）发生，而重写必须是在有继承关系的两个类或有实现关系的类与接口中发生。

### 15). 重写可不可以改变返回值类型？

基本数据类型不可以。引用类型可以，但必须与被重写方法的类型兼容。兼容指引用数据类型能够自动转换的情况。

### 16). Overloaded 的方法是否可以改变返回值的类型?

可以。

### 17). 为什么要定义包？

从类组织上去回答

### 18). Anonymous Inner Class (匿名内部类) 是否可以 extends(继承)其它类，是否可以 implements(实现)interface(接口)

可以继承其它类，也可以实现一个接口。并且它必须实现一个接口或继承一个类。但是它不可以同时继承类和实现接口。

### 19). 是否可以从一个 static 方法内部发出对非 static 方法的调用？

不可以直接调用，但是可以使用实例变量调用。

### 20). 在 JAVA 中，如何跳出当前的多重嵌套循环？

```
loop:for(;;) {
    for(;;) {
        break loop;
    }
}
```

### 21). static 有哪些作用？

可修饰方法、属性、自由块、内部类。使用 static 修饰这些成员时，可以理解成这些成员与类相关，通过“类名.成”的形式调用；没有 static 修饰可以理解成这些成员与对象相关，需要通过“对象名.成员”的形式调用。

a、 static 不能修饰构造方法

b、在 static 修饰的方法中，不能调用没有 static 修饰的方法和属性，也不能使用 this 和 super 关键字。

c、 static 修饰属性时，这个静态属性还具有一个特性，那就是该属性衄多个当前类对象共享。

d、static 修饰自由块，只要类被加载，即使没有创建对象，也将被执行。此外静态自由块无论创建多少个对象，仅执行一次。

### 22). 成员变量和局部变量区别？

局部变量指方法体内部定义的变量，作用域只在方法块内部有效。局部变量在使用时，必须初始化。成员变量指在类中定义的变量，也就是属性，作用域是在整个类中有效。成员变量在定义时可以不指定初始值，系统可以按默认原则初始化。以下几个方面：

public protect private static 等修饰符可用于修饰成员变量，但不能修饰局部变量。两者都可以使用 final 修饰。

成员变量存储在堆内存中，局部变量存储在栈内存中。

### 23). this 有何用途？

用于解决变量的命名冲突和不确定性问题而引入的关键字。使用情况：

a、 返回调用当前方法的对象的引用

b、在构造方法中调用当前类中的其他构造方法

c、 当方法参数名和成员变量名（字段更专业，java 专有名词）相同时，用于区分参数名和成员变量名

### 24). super 有何用途？

代表父类的实例，在子类中，使用 super 可以调用其父类的方法、属性和构造方法。使用情况：

a、 调用父类中的构造方法；

b、调用父类中的方法和属性：super.xxx();

### 25). short s1 = 1; s1 = s1 + 1;与 short s1 = 1; s1 += 1;的编译运行结果是什么？

short s1 = 1; s1 = s1 + 1; （s1+1 运算结果是 int 型，需要强制转换类型）

short s1 = 1; s1 += 1;（可以正确编译）

### 26). Math.round(11.5)等於多少? Math.round(-11.5)等於多少？

round 方法返回与参数最接近的长整数，参数加 1/2 后求其 floor（四舍五入）.

所以：Math.round(11.5)==12 Math.round(-11.5)==-11

### 27). String s = new String("xyz");创建了几个 String Object?

两个

### 28). 接口是否可继承接口？抽象类是否可实现(implements)接口？抽象类是否可继承实现类(concrete class)？

接口可以继承接口。抽象类可以实现(implements)接口，抽象类是可继承实现类，但前提是实现类必须有子类可访问的构造函数。

### 29). 构造器 Constructor 是否可被 override？

构造器 Constructor 不能被继承，因此不能重写 Overriding，但可以被重载 Overloading。

### 30). 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？

是值传递。Java 编程语言只有值传递参数。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。对象的内容可以在被调用的方法中改变，但对象的引用是永远不会改变的。

### 31). 如下代码是否正确？为什么？

```
public abstract class A{
    private String name();
    public abstract void a(){}
}
```

### 32). 如何在 Java 中使用 MD5 或 SHA 加密信息？

### 33). 如下代码中，计算后，变量 i 的值为多少？

```
int i=3, j=10,a=10;
i *= j+a;
```

### 34). 写 clone()方法时，通常都有一行代码，是什么？

super.clone()

### 35).

## 3. 数据类型

### 1). 请列举 Java 中的所有数据类型？

引用类型：数组、接口、类、枚举

基本数据类型：long、int、short、byte、double、float、byte、boolean

### 2). 解释基本数据类型与引用数据类型的区别。

基本类型变量是直接在栈内存中开辟存储空间存储变量值。

引用类型变量是由引用空间和存储空间两部分构成，引用空间在栈内存中，存储空间在堆内存中，存储空间负责存储变量值，引用空间负责存放存储空间的首地址。引用变量中存放的

是地址值，通过地址值可以定义存储位置并修改存储信息。当变量与变量之间赋值时，引用类型变量和基本变量都属于值传递，不同的是基本变量传递的是内容本身，而引用变量传递的却是引用地址。

### 3). int 和 Integer 有什么区别?

int 是基本数据类型所以占用存储空间较小，Integer 是引用数据类型，所以占用空间较大。

### 4). 什么是自动装箱，什么是自动拆箱?

自动装箱：把基本数据转换为包装类或 Object
自动拆箱：把包装类转换为基本数据类型

### 5). 哪些情况下会产生自动类型转换?

基本数据类型从小类型往大类型转换时
子类型转换到父类型时
实现类转换到被实现接口时
子接口转换为父接口
自动装箱和拆箱时
任意类型与字符串使用“+”连接时

### 6). 列举所有数据类型作为类字段时的自动初始化值。

整数：0
实数：0.0
字符：‘\0’
引用：null
boolean：false

## 4. 字符串、数组与简单算法

### 1). String、StringBuffer、StringBuilder 有何区别?

String 是常量，不可变。后两者可变。

StringBuffer 与 StringBuilder 在 API 上兼容，但 StringBuffer 是线程安全的，StringBuilder 不是。

### 2). 将一句英语反序输出如：”I love you”输出后为”you love I”，写出代码

### 3). String str = “a”+”b”+”c”+”d”有几个对象，理由?

### 4). 列举你所知道的排序算法，并比较它们的优劣。

### 5). 有如下代码：

```
String s = "hello";
String s1 = new String("hello");
char a[] = {'h','e','l','l','o'};
```

那么：

```
/*1、*/ s.equals(s1);
```

```
/*2、*/ s1.equals(new String(s));
/*3、*/ =s==s1;
/*4、*/ s.equals(a);
```

的比较结果分别为什么？

### 6). 数组有没有 length()这个方法？String 有没有 length()这个方法？

数组没有 length()这个方法，有 length 的字段。

String 有有 length()这个方法。

### 7). char 型变量中能不能存贮一个中文汉字?为什么？

能够定义成为一个中文的，因为 java 中以 unicode 编码，一个 char 占 16 个字节，所以放一个中文是没问题的

### 8). 判断身份证：要么是 15 位，要么是 18 位，最后一位可以为字母，并写程序提出其中的年月日。

使用正则

### 9). 编写一个程序，将 a.txt 文件中的单词与 b.txt 文件中的单词交替合并到 c.txt 文件中，a.txt 文件中的单词用回车符分隔，b.txt 文件中用回车或空格进行分隔。

使用 IO 读入，使用正则分词，再用 IO 写出

### 10). 有一个字符串，其中包含中文字符、英文字符和数字字符，请统计和打印出各个字符的个数。

使用 Map 记录

## 5. 异常

### 1). Exception 与 Error 有何异同？

相同点：它们都是 Throwable 的子类，都可以被 throw 抛出和被 try……catch 捕获

不同点：它们各自代表含义不同，Exception 指合理的应用程序应当捕获和处理的条件。

Error 通常表示应用程序无法完整处理的条件，虽然它可以被捕获。

### 2). CheckedException 与 RuntimeException 有何区别？

语法上的区别：如果 CheckedException 被申明可能抛出，必须对其进行处理。而 RuntimeException 不用。

概念上的区别：CheckedException 代表无法在程序中避免，而必须捕捉处理的异常 。RuntimeException 代表应当在开发中避免的异常，所以可以不必捕捉。

但是因为在开发中过多使用 CheckedException 的重复异常处理代码太多，在很多时候都会使用 RuntimeException 代替 CheckedException 然后进行统一处理。所以这两者往往却没有了定义者原本赋予的概念。

### 3). throw 与 throws 有何区别？

throw 用于抛出异常，相当于 return 返回数据，只是它仅能抛出 Throwable 及其子类实例
throws 用于申明一个方法可能会抛出的异常，以便在调用都捕获处理。

### 4). try {}里有一个 return 语句，那么紧跟在这个 try 后的 finally {}里的 code 会不会被执行，什么时候被执行，在 return 前还是后？

会被执行，在 return 前。（因为单个线程不可能并发处理，所以将对没有 retun 中的概念。）

### 5). try 语句可以没有 catch 吗？

可以，但是如果没有 catch 的话，必须有 finally。

### 6). 有没有办法让 finally 里面儿的语句不执行？

有，在 finally 执行前调用 System.exit()、或者关闭程序。

### 7). 列举你在项目中的常见异常，至少 8 个。

一定要分散列举，基础一些，Servlet 一些，框架一些。

### 8). 什么情况下会产生 ClassCastException,?

### 9). 如下代码当产生异常时返回值是什么？如果是 zero-exception 呢？

```
public int fun() {
     try {
          // do something
          return 1;
     } catch (Exception e) {
          return 2;
     } finally {
          return 3;
     }
}
```

### 10). 什么时候用 assert。

assertion(断言)在软件开发中是一种常用的调试方式，很多开发语言中都支持这种机制。在实现中，assertion 就是在程序中的一条语句，它对一个 boolean 表达式进行检查，一个正确程序必须保证这个 boolean 表达式的值为 true；如果该值为 false，说明程序已经处于不正确的状态下，系统将给出警告或退出。一般来说，assertion 用于保证程序最基本、关键的正确性 。assertion 检查通常在开发和测试时开启。为了提高性能，在软件发布后，assertion 检查通常是关闭的。

# 6. 线程

### 1). 线程与进程有何区别？

### 2). 进程之间如何通信？线程间如何通信？

### 3). 简述阻塞、非阻塞、同步、异步。

### 4). 线程优先级是什么含义？最大优先级是多少？最小优先级是多少？默认是多少？

代表线程获得 CPU 处理的时间片更多。在 Java 中，最大为 10，最小为 1，默认为 5。

### 5). 在 Java 中怎么启动一个线程？

第一种：继承 Thread，并重写其 run()方法，然后生成其实例，并调用 start()方法。

第二种：将实现了 Runnable 接口的实例作为构造参数，构造一个 Thread 实例，并调用 Thread 实例的 start()方法。

### 6). 有哪些手段保证线程安全？

保证线程安全就是保证数据对各线程访问的一致性，通常有如下手段：

第一种：保证数据在同一时间只能被一个线程访问。如：使用 synchronism 锁定数据、使用 Lock 类锁定数据等。

第二种：保证数据为一个线程私有。如：将数据变量放入 ThreadLocal，并在其它任何其它线程可访问地方去除数据引用。

第三种：数据为只读。如：对数据变量使用 final 修饰或在编码中保证不对数据进行写操作。

### 7). 如何正确终止一个 Java 线程？

Java 本身并没有提供一个可以安全终止线程的方法。所以如果要正确终止一个线程，只能在编码中设置线程运行状态变量，并在线程中检查这个变量，以正确终止线程；

但是可能线程会处于阻塞状态而一直未检查状态变量，那么这个时候可以调用该线程的 interrupt ()方法，清除其阻塞状态。

### 8). sleep() 和 wait() 的异同？

相同点：它们都可以挂起线程。

不同点如下：

1. sleep()是 Thread 上定义的，而 wait()是 Object 上定义的方法；
2. sleep()是 static 修饰的，而 wait()是不是；
3. wait()具备无参重载，也就是说如果没有外部唤醒（使用 notify 或 notifyAll），可以一直处理阻塞状态；而 sleep()必须存在使线程挂起的时间参数，不能无限制挂起。
4. 使用 wait()挂起线程后会释放线程监视器，而 sleep()不会。

### 9). 怎么去判定、评估一个方法是否是线程安全？

首先看方法所在对象是否处于唯一线程所访问实例，如果是则安全；
否则再看其是否有访问字段，如果没有则安全；
如果访问了字段，则看其是否在访问字段的时，是否进行了写操作，没有写操作则安全；
如果有写操作，则看在写操作时是否只有唯一线程能访问该字段，如果是则安全；
其它情况为不安全。

### 10). 同步和异步有何不同，在什么情况下分别使用他们？举例说明。

同步：同一时间只允许一个线程访问数据，其它线程等待。数据安全，但通常效率较低。
异步：将等待操作让另外的线程操作，调用线程继续工作。

### 11). 设计 4 个线程，其中两个线程每次对 j 增加 1，另外两个线程对 j 每次减少 1。写出程序。

### 12). 简述 synchronized 和 java.util.concurrent.locks.Lock 的异同？

主要相同点：Lock 能完成 synchronized 所实现的所有功能
主要不同点：Lock 有比 synchronized 更精确的线程语义和更好的性能。synchronized 会自动释放锁，而 Lock 一定要求程序员手工释放，并且必须在 finally 从句中释放。

### 13). 简述 Daemon Thread（守护线程，精灵线程）。它有什么用处？

## 7. 集合框架

### 1). 什么是泛型？

在程序编码中一些包含类型参数的类型，又叫模板类型。
在 Java 中泛型又分类上的泛型和方法上的泛型。

### 2). Java 中都有哪些引用？各是什么含义？

虚引用：存进去后取不出来，但能在被垃圾回收时收到一个消息
弱引用：当存入的引用对象没有其它引用时将会被清除
软引用：当存入的引用对象没有其它引用并且虚拟机内存不足时清除。
强引用：不会被自动清除。

### 3). Collections 与 Collection 有何关系？各有什么用处？

Collections 是操作 Collection 的工具类

### 4). List、Set、Map 有何异同？

相同点：它们都是属于集合框架的容器类。
不同点：List 与 Set 是 Colleciton 的子接口，而 Map 不是；
List 定义了 get 方法、set 方法所以可以操作索引，可以获取取单个元素，也可以存入重复元素；
而 Set 代表数据上的 Set 集，不可以存入重复元素，并且无法对单个元素进行获取操作。

### 5). 谈谈你对 equals 与 hashCode 的理解。

hashCode 是用来确定 Hash 类容器中的存放位置，但两个不同的对象却有可能有相同的 hashCode，那么这个时候 Hash 类容器就需要使用 equals 来确定相同 hashCode 的两个对象是否是同一个对象。

### 6). ArrayList、Vector、LinkedList 有何异同？

相同点：它们都实现了 List 接口的集合框架的容器类。

不同点：ArrayList 与 Vector 的存储是数组实现，并且实现了 RandomAccess 接口，所以其支持快速随机访问，但中间插入、删除元素的时候效率较低，而 LinkedList 的存储是由链接列表实现，所以任意位置插入删除元素时效率较高，但任意位置访问效率较低；

ArrayList 与 Vector 的区别在于，ArrayList 非线程安全，而 Vector 线程安全，且它们的增长率不相同。

### 7). Set 的实现类 HashSet 与 TreeSet 分别是如果保证值唯一的？

HashSet：其内部是基于 HashMap 的，即是散列表实现，所以其是使用元素的 hashCode 与 equals 比较保证唯一性。

TreeSet：其内部是基于 TreeMap 的，也即是通过红黑树实现，所以其是通过自然排序或者排序器的比较结果遍历树节点实现。

### 8). 如果我想把一组数字按照从小到大，或从大到小输出，怎么实现？那如果是一组姓名，按我需要的顺序输出呢？

使用 Arrays、Collections 工具类、或者排序容器实现，也可以自己写排序算法实现

### 9). HashMap 可以存 NULL 键和 NULL 值吗？可以存多个吗？

只可以存一个 Null 键，但可以存任意个 Null 值。

### 10). HashMap 与 Hashtable 有何异同？

Hashtable 是线程安全的，不可以存空键或值。

HashMap 不是线程安全的，可以可以存一个 Null 键，但可以存任意个 Null 值

### 11). TreeSet 里面放对象，如果同时放入了父类和子类的实例对象，那比较时使用的是父类的 compareTo 方法，还是使用的子类的 compareTo 方法，还是抛异常？

取决于存入顺序和各自的 compareTo 方法的具体实现。

## 8. IO

### 1). Java 中都有哪些流分类？

按处理单位分：字符流与字节流

按流向分：输入流与输出流

按功能分：节点流与处理流

### 2). 如何判断 IO 包下的类是字节流类，还是字符流类？

根据后缀判断，后缀为 Stram 的是字节流，后缀为 Reader 为 Writer 的为字符流。

### 3). 什么是 java 序列化，如何实现 java 序列化？

序列化双称串行化，指把 Java 对象数据以一连串字节描述（转换为字节数据）的过程。

通常一个以 `Serializable` 标记一个类后，使用对象流的 writeObject()就可实现，也可以 `Serializable` 标记类，进行更细节的处理。

### 4). 将一个 GBK 编码的文本文件转存为一个 UTF-8 编码的文本文件。

### 5). 有一个 500G 的文件，将对当中的一部分数据进行处理，你会如何操作？

随机文件读写
NIO

## 9. 反射

### 1). 简述 ClassLoader 加载类的过程。

通常是从调用它的 loadClass 方法开始，加载过程如下：

1. 调用 findLoadedClass(String) 来检查是否已经加载类。
2. 在父类加载器上调用 loadClass 方法。如果父类加载器为 null，则使用虚拟机的内置类加载器。
3. 调用 findClass(String) 方法查找类。findClass 方法如果查找到类数据，那么将调用 defineClass 将数据转换为 Class 类的实例。

如果使用上述步骤找到类，并且 resolve 标志为真，则此方法将在得到的 Class 对象上调用 resolveClass(Class) 方法。

### 2). 有哪些方法判断一个对象是否是一个类的实例？

使用 instanceof 运算符判断、调用类上的 isInstance()判断

### 3). Class.forName 的作用?为什么要用？

根据类名加载类。

作用：类名是字符串，所以使得类加载可配置，增加程序灵活

### 4). 什么是动态代理？在 Java 中如何使用或实现？

动态代理是 java.lang.reflect 包的一部分，它允许程序创建 代理对象，代理对象能实现一个或多个已知接口，并用反射代替内置的虚方法分派，编程地分派对接口方法的调用。这个过程允许实现“截取”方法调用，重新路由它们或者动态地添加功能。

在 Java 中要实现动态代理的类必须实现了接口，然后使用 Proxy 生成动态代理类，并注册 InvocationHandler 监听。

### 5). java 中会存在内存泄漏吗，请简单描述。

会。

java 导致内存泄露的原因很明确：

其一：长生命周期的对象持有短生命周期对象的引用就很可能发生内存泄露，尽管短生命周期对象已经不再需要，但是因为长生命周期对象持有它的引用而导致不能被回收，这就是

java 中内存泄露的发生场景。

其二：当有较复杂的循环引用时。

其三：当产生不确定的递归调用时。

### 6). 列举生成对象的方式，至少 3 个。

使用 new、使用 class 实例的 newInstance()、使用构造器的 newInstance()、反序列化。

### 7). 简述对象的生命周期。

对象生命周期指对象的创建、使用到销毁的过程；主要讨论的是创建过程与销毁过程；

对象创建：首先会在堆中开辟存储对象数据的空间，然后调用构造方法对对象进行初始化，构造方法会对父类构造方法递归调用。

对象销毁：当没有任何引用指向对象时，垃圾回收机制会对对象进行垃圾回收，在回收前会调用 finallize 方法。

### 8). GC 是什么？为什么要有 GC？

GC 是垃圾收集的意思（Gabage Collection）。

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，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示操作方法。

### 9). 垃圾回收器的基本原理是什么？垃圾回收器可以马上回收内存吗？有什么办法主动通知虚拟机进行垃圾回收？

垃圾回收器是一个优先级较低的守护线程,它通过不定时监测程序使用的内存中被占用的动态分配的内存内的对象是否还存在它的引用来判断是否该回收那个内存单元,如果不存在则回收,否则相反

为并不是只要监测到就会回收的,垃圾回收器线程的优先级较低,所以当另一个优先级比它高的线程跟他同时竞争运行时间时,前者优先运行,我们通过 Thread 或者继承 Runnable 的线程都级别都比它高,所以你无法知道垃圾回收器何时回收。

可以调用 System.gc()通知，但 System.gc()仅只是通知垃圾回收器进行回收处理,调用它并不能保证它回立即回收

### 10). 简述 Java 中的动态绑定。

将一个方法调用同一个方法主体连接到一起称为“绑定”。如果在程序运行之前执行绑定，由编译器决定方法调用的程序，称为“早期绑定”或“静态绑定”。如果绑定过程在程序运行期间进行，以对象的类型为基础，则称为“后期绑定”或“动态绑定”。

如果一种语言实现了后期绑定，同时必须提供一些机制，可以在运行期间判断对象的实际类型，并分别调用适当的方法，即编译器此时依然不知道对象的类型，但方法调用机制能够自己去调查，找到正确的方法主体。Java 方法的执行主要采用动态绑定技术，在程序运行时，虚拟机将调用对象实际类型所限定的方法。

Java 方法在调用过程中主要经历了以下过程。

编译器查看对象变量的声明类型和方法名，通过声明类型找到方法列表。

编译器查看调用方法时提供的参数类型。

如果方法由 private,static,final 修饰或者是构造器，编译器就可以确定调用哪一种方法，即采用静态绑定技术。如果不是上述情况，就使用动态绑定技术，执行后续过程。

虚拟机提取对象的实际类型的方法表。
虚拟机搜索方法签名。
调用方法。

### 11). 创建类的对象时，类中各成员的执行顺序：

属性、方法、构造方法和自由块都是类中的成员，在创建对象时，各成员的执行顺序如下：
父类静态成员和静态初始化块，按在代码中出现的顺序依次执行。
子类静态成员和静态初始化块，按在代码中出现的顺序依次执行。
父类实例成员和实例初始化块，按在代码中出现的顺序依次执行。
执行父类构造方法
子类实例成员和实例初始化块，按在代码中出现的顺序依次执行。
执行子类构造方法

### 12). 简述静态初始化块和非静态初始化块。

静态初始化块比非静态初始化块执行要早，而且静态初始化块只执行一次，
非静态的初始化块可执行多次。静态初始化块的的执行时机需要注意，它是在类加载器第一次加载该类时调用，不一定非要创建对象才触发，如果使用“类.静态方法”也会执行静态方法。

## 10. JDBC

### 1). 描述使用 JDBC 连接数据库的过程。

1. 加载驱动
2. 使用 DriverManager 创建 Connection 对象
3. 创建 Statement 或其子接口实例，并执行 Sql 语句
4. 如果执行的是 DQL 则取得 ReultSet 实例并处理
5. 以创建的 JDBC 对象的反序依次关闭全部对象

### 2). 简述 JDBC 连接池的实现方法。

当一个线程需要用 JDBC 对某一个数据库操作时，它从池中请求一个连接。当这个线程使用完了这个连接，将它返回到连接池中，这样相同的连接就可以被利用。

### 3). 如何在编程中使用事务？

首先调用 Connection 的 setAutoCommit()方法参数 false，关闭自动提交；
然后根据情况调用 Connection 的 commit()或 rollback()控制事务提交或回滚。

### 4). 如何使用 JDBC 调用存储过程？

使用 CallableStatement 调用。

### 5). 说出数据连接池的工作机制是什么？

连接池在启动时会建立一定数量的池连接，并一直维持不少于此数目的 池连接。客户端程序需要连接时，池驱动程序会返回一个未使用的池连接并将其表记为忙。
如果当前没有空闲连接，池驱动程序就新建一定数量的连接，新建连接的数量有配置参数决定。当使用的池连接调用完成后，池驱动程序将此连接标记为空闲，其他调用就可以使用这个连接。

# ♦JavaWeb

## 1. Servlet

### 1). 能不能自己实现一个 Session？可以的话，请简述思路。

### 2). 什么是 Servlet?

Servlet 是 java 语言类定义于 Java 服务器端的小程序接口，用来拓展通过请求响应模式的服务端的能力。

尽管 servlets 可以响应任何类型的响应，它们通常用于拓展基于 Web 的应用程序。在这些应用程序中，Java servlet 技术定义了特定的 HTTP servlet 类。

### 3). 简述 HttpServletRequest 的生命周期。

### 4). 说出 Servlet 的生命周期，并说出 Servlet 和 CGI 的区别?

部署了的 Servlet 的生命周期是由容器控制的。当一个请求映射到相应的 servlet 时，容器产生下面的步骤：

1）.根据配置，将在服务器启动，或者第一次请求时加载 Servlet 并创建一个唯一实例和调用其的 init 方法初始化实例

2）.当每次请求到达时调用其 service,如果是 HttpServlet,将被分发到对应的 doXXX 方法

3）.在卸载前，调用 destroy，然后销毁

与 cgi 的区别在于 servlet 处于服务器进程中，它通过多线程方式运行其 service 方法，一个实例可以服务于多个请求，并且其实例一般不会销毁，而 CGI 对每个请求都产生新的进程，服务完成后就销毁，所以效率上低于 servlet。并且 CGI 是针对平台编译，无法跨平台运行。

### 5). 你都使用过哪些应用服务器（Servlet 容器）？你能演示布署一个 Web 应用到容器吗？

使用过 Tomcat、jetty、jBoss、WebLocgic 等。

JavaWeb 应用通常是一个目录，该目录包含配置信息和 Servlet、jsp 等，只需要将该目录，放到服务器特定目录便可；也可以将这个目录打成 War 包进行布署

### 6). 如何布署一个 Servlet 到容器？

首先需要编写一个 Servlet，然后将它拷贝到应用的 WEB-INF/classes 目录，然后在 web.xml 中使用<servlet>、<servlet-name>、<servlet-class>、<url-pattern>标签进行正确配置便完成布署。

Servlet 标准 3.0 后可以使用 @WebServlet 布署

### 7). Servlet、GenericServlet、HttpServlet 有何关系？

Servlet 是定义了服务器小程序的接口。

GenericServlet 实现了 Servlet 接口，并利用 init 方法帮助实现了 ServletConfig 接口，以及对 destory 的空实现。

HttpServlet 继承于 GenericServlet，并针对于 HTTP 协议进行了一些特定实现：

1. 参数为 HttpServletRequest、HttpServletResponse 的 serivce()方法
2. 与 HTTP 协议请求方式对应的一系列 doXXX 方法，如：doGet()、doPost()

### 8). 为何 GenericServlet 上有两个 init 方法？它们有何关系？

因为 GenericServlet 利用 init 方法帮助实现了 ServletConfig 接口，为避免开发人员错误重写带参数的原始 init 方法，而重新定义了一个无参的 init 方法。并在原 init 方法中进行调用。

### 9). 为何 HttpServlet 上有两个 service 方法？它们有何关系？

因为 HttpServlet 是针对于 HTTP 协议进行的特定实现，而原 Servlet 接口定义的 service 方法并非特定于 HTTP 协议的，所以 HttpServlet 定义了参数为 HttpServletRequest、HttpServletResponse 的 serivce()方法。

并且在原 service 中对参数转型后该用了该 service 方法。该 service 方法负责根据请求方式调用对应的 doXXX 方法。

### 10). HttpServletRequest 上为何有两个 getSession 方法？它们各有何用途？及它们的关系是什么？

是重载的两个方法，一个无参，一个有一个 boolean 参数。

有参的如果参数为 true 时，如果不存在 session 便会创建一个 session；如果值为 false，那么在没有 Session 存在的时候便不会创建，这对在服务器内存使用敏感情况下特别有用。

无参的相当于调用 getSession(true)。

### 11). 在什么情况下调用 HttpServlet 的 doGet()和 doPost()？

doGet 和 doPost 对应了 Http 请求方式的 GET 与 POST，也就是说当当前 HttpServlet 接受到 GET 请求时，将调用 doGet，同理接收到 POST 请求时将调用 doPost()Jsp 页面中的 form 标签里的 method 属性为 get 时调用 doGet()，为 post 时调用 doPost()。

### 12). HttpServletRequest 的 forward 和 HttpServletResponse 的 redirect 有何区别?

forward: 请求转发，或叫服务器内部转发， 只在在服务器内部转发请求，相当于特殊的方法调用，所以只产生一次请求，且只能转发到本服务器。所以转发前与转发后有相同的 request 对象；对浏览器不可见，所以浏览器地址栏不会发生变化. 在下一个页面中,request 保留上一个页面中的 request 的所有值

redirect: 跳转，不传递 request 对象。重定向，服务器会根据 HTTP 协议向客户端发送 302 状态码和 Location 字段，客户端会根据 Location 字段发起第二次请求；所以重定向前后有不同的 request 对象，并可以重定向到任意 url，浏览器地址栏会显示为最终请求的地址

### 13). Cookie 是对象吗？它与 session 有何区别？

Session 和 Cookie 都是用于跟踪同一会话，存储和标记同一会话的数据标记，Session 通常情况下是 Cookie 实现的

使用上的区别：Session 是在服务器端标记，Cookie 是在客户端标记

实现上的区别：Cookie 是基于 Http 协议实现的，Session 是一个逻辑上的概念

### 14). request.getAttribute()和 request.getParameter() 有何区别?

getAttribute()：获取 request 作用域变量。

getParameter()：获取请求数据。

### 15). RequestDispatcher 上的 forward 与 include 有何区别?

forword：请求转发；会清空输出缓存，所以如果 原来页面有输出内容会被的全部东西都不清空会不会发送到浏览再显示（被清空），如果转发前有输出到浏览器的操作，那么就无法转发。

include：包含。include 会把原来页面的东西显示出来不会清空输出缓存，会前 include 前后页面的东西组合在一起输出到浏览器显示

### 16). 如何让一个 Session 失效?

有很多方法，但最常用的是调用该 session 对象的 invalidate()方法。

### 17). 如何在不支持 Cookie 的浏览器中保持会话(session)?

使用 HttpServletResponse 上面的 encodeURL 对所有链接进行编码，使用 url 传值的方式来保持会话。

### 18). 如何从 form 表单中得取 checkbox 的值?

用 getParamterValues("name")能取到 checkbox 的一组值。

### 19). Filter 的作用是什么？主要实现什么方法？request 和 response 全称是什么?

Filter 过滤器，在每次请求资源前都将访问 Filter，所以可以对请求或响应作预处理，如设置编码，准备资源；也可以用来作访问权限验证。

主要实现它的 doFilter 方法。

request：HttpServletRequest

response：HttpServletResponse

## 2. JSP

### 1). 列举 JSP 内置对象，至少 8 个。

request：表示 HttpServletRequest 对象。取客户端表单域信息及 cookie, header, 和 session

response：表示 HttpServletResponse 对象，对客户端的响应返回文本、写 cookies。

out：向客户端打印 html 文本.

pageContext :当前 jsp 页面的上下文环境，可以得到 session、request、application 等内置对象，在自定义标签中使用的很多。

session：表示一个请求的 javax.servlet.http.HttpSession 对象。Session 一个用户多个页面共享同一变量。

application：表示一个 javax.servle.ServletContext 对象。存放容器级的变量。

config：表示一个 javax.servlet.ServletConfig 对象。该对象用于存取 servlet 实例的初始化参数。

page：表示从该页面产生的一个 servlet 实例

exception:异常，当 iserrorpage=true 时方可使用

### 2). 简述 JSP 的执行过程。

1，JSP 文件先要翻译成 Java 文件（Servlet），在 tomcat 中翻译后的 java 文件在 tomcat 下的 work\Catalina\localhost 中相应名字的应用目录里。

2，编译 Java 文件

3，加载运行.class 文件

### 3). <%! %> 、 <%%> 、<%=%>各有何用途？

<%! %>：scripts 申明，用于定义 jsp 转换后类代码的字段或方法。

<%%>：scripts，用于定义 jsp 转换后_jspService 方法内的代码，所以可以使用内置对象。

<%=%>：<% out.print() %>的简写。

### 4). page 指令的 import、session、contentType、pageEncoding 有何用途？

import：用于导入包

session：用于定义是否可以使用 session 内置对象

contentType：定义该页面响应的内容类型

pageEncoding：定义该页面响应的字符编码。

### 5). 列举 jsp 的所有作用域，及解释各个作用域的作用范围。

page 否是代表与一个页面相关的对象和属性。一个页面由一个编译好的 Java servlet 类（可以带有任何的 include 指令，但是没有 include 动作）表示。这既包括 servlet 又包括被编译成 servlet 的 JSP 页面

request 是是代表与 Web 客户机发出的一个请求相关的对象和属性。一个请求可能跨越多个页面，涉及多个 Web 组件（由于 forward 指令和 include 动作的关系）

session 是是代表与用于某个 Web 客户机的一个用户体验相关的对象和属性。一个 Web 会话可以也经常会跨越多个客户机请求

application 是是代表与整个 Web 应用程序相关的对象和属性。这实质上是跨越整个 Web 应用程序，包括多个页面、请求和会话的一个全局作用域

### 6). jsp 动作:useBean 有何用途？它与 setProperty 及 getProperty 有何关系？

useBean：在 jsp 中定义 javaBean

setProperty：对已经定义的 javaBean 设置属性值。

getProperty：获取已经定义的 javaBean 属性值显示页面。

### 7). JSP 的静态包含与动态包含有何区别？

jsp:include:在运行时调用另一个页面，变量是可以重复的。

<%@include file=””%>:在转译时合在一起，会成为同一个类，变量不可以重复。

### 8). 请求转发与重定向有何区别？

请求转发：或叫服务器内部转发， 只在在服务器内部转发请求，相当于特殊的方法调用，所以只产生一次请求，且只能转发到本服务器。所以转发前与转发后有相同的 request 对象；对浏览器不可见，所以浏览器地址栏不会发生变化. 在下一个页面中,request 保留上一个页面中的 request 的所有值

重定向：不传递 request 对象。重定向，服务器会根据 HTTP 协议向客户端发送 302 状态码

和 Location 字段，客户端会根据 Location 字段发起第二次请求；所以重定向前后有不同的 request 对象，并可以重定向到任意 url，浏览器地址栏会显示为最终请求的地址

### 9). 简述你对 WebMvc 的理解及你所知道的对 WebMvc 的实现方式。

WebMvc 就是将著名的 Mvc 模式应用在 Web 应用程序开发中。

一般采用 jsp 实现视图、Servlet 实现控制器、JavaBean 实现模型。

但更多的时候我们会采用前端 MVC 框架实现，如：Struts、SpringMvc 等。

### 10). 在 servlets 和 JSP 之间能共享 session 对象吗？

当然可以。

### 11). JavaScript 的变量能复制到 JSP 的 SESSION 中吗？

可以，但必须使用 javascript 和服务器进行通信。不能直接由 java 代码调用。

### 12). Jsp 的异常处理

首先应该使用<%@page errorPage="xxx.jsp"%> 指定本页面出现异常后要转到的页面

<%@page iserrorPage="true"%> 见本页面指定为异常处理页面，也就是其他出异常的页面可以指定本页面为要转到的异常处理页面。定义处理异常的 JSP 页面

然后使用<%@page errorPage="xxx.jsp"%> 异常处理页面，或者在 web.xml 中指定特定异常，或特定错误编码的异常处理页面。处理异常

### 13). 描述 JSP 和 Servlet 的区别、共同点、各自应用的范围

共同点：jsp 最终会实现 HttpJspPage 而 HttpJspPage 继承于 JspPage，JspPage 继承于 Servlet 接口；也就是 jsp 就是一个 Servlet。

区别和各自使用范围：jsp 不需要在 web.xml 中配置就可以使用，并且能较方便的混合书写 html 和 java 代码，所以特别适合用于展现数据，但是因为混合了 html 代码，其 java 代码就不是如纯 java 源程序那样易读易写。

Servlet 需要在 web.xml 中配置才能使用，且输出的 html 是以字符串的方式存在，所以不太适合书写 html 代码；但本身就是 java 源程序，所以特别适合书写 java 代码。

所以通常在项目中都是使用 jsp、Servlet 组合 JavaBean，jsp 实现视图逻辑、Servlet 实现控制逻辑、JavaBean 实现业务逻辑组成 WebMvc 进行开发。

### 14). web.xml 有何作用？

用于配置 web 应用的信息；如 listener、filter 及 servlet、JSP 异常处理等的配置信息等。

### 15). 简述 jsp 的动作指令。

JSP 共有以下 6 种基本动作

jsp:include：在页面被请求的时候引入一个文件。

jsp:useBean：寻找或者实例化一个 JavaBean。

jsp:setProperty：设置 JavaBean 的属性。

jsp:getProperty：输出某个 JavaBean 的属性。

jsp:forward：把请求转到一个新的页面。

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

**16). 列举会话跟踪的方式。**

1) session
2) request
3) Cookie
4) 表单隐藏域
5) 通过 Url 参数传值

# ♦Java 应用框架

## 1. Struts

### 1). struts1.2 和 struts2.0 的区别？如何控制两种框架中的单例模式？

struts1.2 和 struts2.0 的对比

a、Action 类：

struts1.2 要求 Action 类继承一个基类。struts2.0 Action 要求继承 ActionSupport 基类

b、线程模式

struts1.2 Action 是单例模式的并且必须是线程安全的,因为仅有一个 Action 的实例来处理所有的请求。

单例策略限制了 Struts1.2 Action 能做的事情,并且开发时特别小心。Action 资源必须是线程安全的或同步的。

struts2.0 Action 为每一个请求产生一个实例,因此没有线程安全问题。

c、Servlet 依赖

struts1.2 Action 依赖于 Servlet API,因为当一个 Action 被调用时 HttpServletRequest 和 HttpServletResponse 被传递给 execut 方法。

struts2.0 Action 不依赖于容器,允许 Action 脱离容器单独测试。如果需要,Struts2 Action 仍然可以访问初始的 Request 和 Response。

但是,其他的元素减少或者消除了直接访问 HttpServletRequest 和 HttpServletResponse 的必要性。

d、可测性

测试 struts1.2 Action 的一个主要问题是 execute 方法暴露了 Servlet API(这使得测试要依赖于容器)。一个第三方扩展：struts TestCase

提供了一套 struts1.2 的模拟对象来进行测试。

Struts2.0 Action 可以通过初始化、设置属性、调用方法来测试,“依赖注入”也使得测试更容易。

### 2). 在项目中为什么要使用 Struts？

规范开发风格，方便搭建 Mvc，使代码易读易维护

便于和 Spring 集成、提供了校验框架

集成视图技术、国际化支持，简化开发

### 3). struts2.0 的常用标签

1. 往 action 里传值：<input name="userName" type="text" class="input6" size="15">

2. 显示标签 property 用于输出指定值：<s:property value="userName "/>

3. 用于从页面往 action 中(user)的对象内传值：<s:text name="user.userName " id="username"/>

4. 判断<s:if> </s:if> 用于在页面中判断指定数据 <s:if test="userName == admin">…. </s:if> <s:else>…. </s:else>

5. 迭代<s:iterator>用于将 List、Map、ArrayList 等集合进行循环遍历

```
<s:iterator value="userList" id="user" status="u">
        <s:property value="userName"/></a>
</s:iterator>
```

6. URL 地址标签，<s:url>用于生成一个 URL 地址，可以通过 URL 标签指定的<s:param>子元素向 URL 地址发送请求参数

```
<s:url action=" ">
        <s:param name=" " value=""></s:param>
</s:url>
```

7. 超链接 <a href >一般和<s:url>标签一起使用，用于带多个参数。

```
<a href="
        <s:url action=" ">
            <s:param name=" " value=""></s:param>
            <s:param name=" " value=""></s:param>
            <s:param name=" " value=""></s:param>
        </s:url>
">超链接</a>
```

8. set 标签，用于将某个值放入指定的范围内。例如 application,session 等。

```
<s:set name="user" value="userName" scope=”request”/>
```

### 4). struts2.0 的 mvc 模式？与 struts1.0 的区别？

struts2 的 mvc 模式：当用户在页面提交用户请求时,该请求需要提交给 struts2 的控制器处理。struts2 的控制器根据处理结果,

决定将哪个页面呈现给客户端。

与 struts1 最大的不同是：struts2 的控制器。struts2 的控制器不再像 struts1 的控制器,需要继承一个 Action 父类,甚至可以无需实现

任何接口,struts2 的 Action 就是一个普通的 POJO。实际上，Struts2 的 Action 就是一个包含 execute 方法的普通 Java 类

该类里包含的多个属性用于封装用户的请求参数。

### 5). 简述 Struts 的工作原理。

### 6). 简述 Struts 的处理流程。

### 7). 简述 Struts2 的处理流程。

## 2. Spring

### 1). 什么是 Spring？

Spring 是一个开源的 Java EE 开发框架。Spring 框架的核心功能可以应用在任何 Java 应用程序中，但对 Java EE 平台上的 Web 应用程序有更好的扩展性。Spring 框架的目标是使得 Java EE 应用程序的开发更加简捷，通过使用 POJO 为基础的编程模型促进良好的编程风格。

## 2). Spring 有哪些优点？

轻量级：Spring 在大小和透明性方面绝对属于轻量级的，基础版本的 Spring 框架大约只有 2MB。

控制反转(IOC)：Spring 使用控制反转技术实现了松耦合。依赖被注入到对象，而不是创建或寻找依赖对象。

面向切面编程(AOP)： Spring 支持面向切面编程，同时把应用的业务逻辑与系统的服务分离开来。

容器：Spring 包含并管理应用程序对象的配置及生命周期。

MVC 框架：Spring 的 web 框架是一个设计优良的 web MVC 框架，很好的取代了一些 web 框架。

事务管理：Spring 对下至本地业务上至全局业务(JAT)提供了统一的事务管理接口。

异常处理：Spring 提供一个方便的 API 将特定技术的异常(由 JDBC, Hibernate, 或 JDO 抛出)转化为一致的、Unchecked 异常。

## 3). Spring 框架有哪些模块？

Spring 框架的基本模块如下所示：

Core module

Bean module

Context module

Expression Language module

JDBC module

ORM module

OXM module

Java Messaging Service(JMS) module

Transaction module

Web module

Web-Servlet module

Web-Struts module

Web-Portlet module

## 4). 解释核心容器(应用上下文)模块

这是 Spring 的基本模块，它提供了 Spring 框架的基本功能。BeanFactory 是所有 Spring 应用的核心。Spring 框架是建立在这个模块之上的，这也使得 Spring 成为一个容器。

## 5). BeanFactory – BeanFactory 实例

BeanFactory 是工厂模式的一种实现，它使用控制反转将应用的配置和依赖与实际的应用代码分离开来。

最常用的 BeanFactory 实现是 XmlBeanFactory 类。

## 6). XmlBeanFactory

最常用的就是 org.springframework.beans.factory.xml.XmlBeanFactory，它根据 XML 文件中定义的内容加载 beans。该容器从 XML 文件中读取配置元数据，并用它来创建一个完备的系统或应用。

### 7). 解释 AOP 模块

AOP 模块用来开发 Spring 应用程序中具有切面性质的部分。该模块的大部分服务由 AOP Aliance 提供，这就保证了 Spring 框架和其他 AOP 框架之间的互操作性。另外，该模块将元数据编程引入到了 Spring。

### 8). 解释抽象 JDBC 和 DAO 模块

通过使用抽象 JDBC 和 DAO 模块保证了与数据库连接代码的整洁与简单，同时避免了由于未能关闭数据库资源引起的问题。它在多种数据库服务器的错误信息之上提供了一个很重要的异常层。它还利用 Spring 的 AOP 模块为 Spring 应用程序中的对象提供事务管理服务。

### 9). 解释对象/关系映射集成模块

Spring 通过提供 ORM 模块在 JDBC 的基础上支持对象关系映射工具。这样的支持使得 Spring 可以集成主流的 ORM 框架，包括 Hibernate, JDO, 及 iBATIS SQL Maps。Spring 的事务管理可以同时支持以上某种框架和 JDBC。

### 10). 解释 web 模块

Spring 的 web 模块建立在应用上下文(application context)模块之上，提供了一个适合基于 web 应用程序的上下文环境。该模块还支持了几个面向 web 的任务，如透明的处理多文件上传请求及将请求参数同业务对象绑定起来。

### 11). .解释 Spring MVC 模块

Spring 提供 MVC 框架构建 web 应用程序。Spring 可以很轻松的同其他 MVC 框架结合，但 Spring 的 MVC 是个更好的选择，因为它通过控制反转将控制逻辑和业务对象完全分离开来。

### 12). Spring 的配置文件

Spring 的配置文件是一个 XML 文件，文件包含了类信息并描述了这些类是如何配置和互相调用的。

### 13). Spring IoC 容器是什么？

Spring IOC 负责创建对象、管理对象(通过依赖注入)、整合对象、配置对象以及管理这些对象的生命周期。

### 14). IOC 有什么优点？

IOC 或依赖注入减少了应用程序的代码量。它使得应用程序的测试很简单，因为在单元测试中不再需要单例或 JNDI 查找机制。简单的实现以及较少的干扰机制使得松耦合得以实现 。IOC 容器支持勤性单例及延迟加载服务。

### 15). 应用上下文是如何实现的？

FileSystemXmlApplicationContext 容器加载 XML 文件中 beans 的定义。XML Bean 配置文件的完整路径必须传递给构造器。

FileSystemXmlApplicationContext 容器也加载 XML 文件中 beans 的定义。注意，你需要正确的设置 CLASSPATH，因为该容器会在 CLASSPATH 中查看 bean 的 XML 配置文件。

WebXmlApplicationContext：该容器加载 xml 文件，这些文件定义了 web 应用中所有的 beans。

### 16). Bean Factory 和 ApplicationContext 有什么区别？

ApplicationContext 提供了一种解决文档信息的方法，一种加载文件资源的方式(如图片)，他们可以向监听他们的 beans 发送消息。另外，容器或者容器中 beans 的操作，这些必须以 bean 工厂的编程方式处理的操作可以在应用上下文中以声明的方式处理。应用上下文实现了 MessageSource，该接口用于获取本地消息，实际的实现是可选的。

### 17). Spring 应用程序看起来像什么？

一个定义功能的接口
实现包括属性，setter 和 getter 方法，功能等
Spring AOP
Spring 的 XML 配置文件
使用该功能的客户端编程
依赖注入

### 18). Spring 中的依赖注入是什么？

依赖注入作为控制反转(IOC)的一个层面，可以有多种解释方式。在这个概念中，你不用创建对象而只需要描述如何创建它们。你不必通过代码直接的将组件和服务连接在一起，而是通过配置文件说明哪些组件需要什么服务。之后 IOC 容器负责衔接。

### 19). 有哪些不同类型的 IOC(依赖注入)？

构造器依赖注入：构造器依赖注入在容器触发构造器的时候完成，该构造器有一系列的参数，每个参数代表注入的对象。

Setter 方法依赖注入：首先容器会触发一个无参构造函数或无参静态工厂方法实例化对象，之后容器调用 bean 中的 setter 方法完成 Setter 方法依赖注入。

### 20). 你推荐哪种依赖注入？构造器依赖注入还是 Setter 方法依赖注入？

你可以同时使用两种方式的依赖注入，最好的选择是使用构造器参数实现强制依赖注入，使用 setter 方法实现可选的依赖关系。

### 21). 什么是 Spring Beans？

Spring Beans 是构成 Spring 应用核心的 Java 对象。这些对象由 Spring IOC 容器实例化、组装、管理。这些对象通过容器中配置的元数据创建，例如，使用 XML 文件中定义的创建。

在 Spring 中创建的 beans 都是单例的 beans。在 bean 标签中有一个属性为”singleton”,如果设为 true，该 bean 是单例的，如果设为 false，该 bean 是原型 bean。Singleton 属性默认设置为 true。因此，spring 框架中所有的 bean 都默认为单例 bean。

### 22). Spring Bean 中定义了什么内容？

Spring Bean 中定义了所有的配置元数据，这些配置信息告知容器如何创建它，它的生命周期是什么以及它的依赖关系。

### 23). 如何向 Spring 容器提供配置元数据？

有三种方式向 Spring 容器提供元数据:
XML 配置文件
基于注解配置

基于 Java 的配置

### 24). 你如何定义 bean 的作用域？

在 Spring 中创建一个 bean 的时候，我们可以声明它的作用域。只需要在 bean 定义的时候通过’scope’属性定义即可。例如，当 Spring 需要产生每次一个新的 bean 实例时，应该声明 bean 的 scope 属性为 prototype。如果每次你希望 Spring 返回一个实例，应该声明 bean 的 scope 属性为 singleton。

### 25). 说一下 Spring 中支持的 bean 作用域

Spring 框架支持如下五种不同的作用域：

singleton：在 Spring IOC 容器中仅存在一个 Bean 实例，Bean 以单实例的方式存在。

prototype：一个 bean 可以定义多个实例。

request：每次 HTTP 请求都会创建一个新的 Bean。该作用域仅适用于 WebApplicationContext 环境。

session：一个 HTTP Session 定义一个 Bean。该作用域仅适用于 WebApplicationContext 环境.

globalSession：同一个全局 HTTP Session 定义一个 Bean。该作用域同样仅适用于 WebApplicationContext 环境.

bean 默认的 scope 属性是’singleton‘。

### 26). Spring 框架中单例 beans 是线程安全的吗？

不是，Spring 框架中的单例 beans 不是线程安全的。

### 27). 解释 Spring 框架中 bean 的生命周期

Spring 容器读取 XML 文件中 bean 的定义并实例化 bean。

Spring 根据 bean 的定义设置属性值。

如果该 Bean 实现了 BeanNameAware 接口，Spring 将 bean 的 id 传递给 setBeanName()方法。

如果该 Bean 实现了 BeanFactoryAware 接口，Spring 将 beanfactory 传递给 setBeanFactory()方法。

如果任何 bean BeanPostProcessors 和该 bean 相关，Spring 调用 postProcessBeforeInitialization()方法。

如果该 Bean 实现了 InitializingBean 接口，调用 Bean 中的 afterPropertiesSet 方法。如果 bean 有初始化函数声明，调用相应的初始化方法。

如果任何 bean BeanPostProcessors 和该 bean 相关，调用 postProcessAfterInitialization()方法。

如果该 bean 实现了 DisposableBean，调用 destroy()方法。

### 28). 哪些是最重要的 bean 生命周期方法？能重写它们吗？

有两个重要的 bean 生命周期方法。第一个是 setup 方法，该方法在容器加载 bean 的时候被调用。第二个是 teardown 方法，该方法在 bean 从容器中移除的时候调用。

bean 标签有两个重要的属性(init-method 和 destroy-method)，你可以通过这两个属性定义自己的初始化方法和析构方法。Spring 也有相应的注解：@PostConstruct 和 @PreDestroy。

### 29). 什么是 Spring 的内部 bean？

当一个 bean 被用作另一个 bean 的属性时，这个 bean 可以被声明为内部 bean。在基于 XML 的配置元数据中，可以通过把元素定义在 或元素内部实现定义内部 bean。内部 bean 总是匿名的并且它们的 scope 总是 prototype。

### 30). 如何在 Spring 中注入 Java 集合类？

Spring 提供如下几种类型的集合配置元素：

list 元素用来注入一系列的值，允许有相同的值。

set 元素用来注入一些列的值，不允许有相同的值。

map 用来注入一组”键-值”对，键、值可以是任何类型的。

props 也可以用来注入一组”键-值”对，这里的键、值都字符串类型。

### 31). 什么是 bean wiring？

Wiring，或者说 bean Wiring 是指 beans 在 Spring 容器中结合在一起的情况。当装配 bean 的时候，Spring 容器需要知道需要哪些 beans 以及如何使用依赖注入将它们结合起来。

### 32). 什么是 bean 自动装配？

Spring 容器可以自动配置相互协作 beans 之间的关联关系。这意味着 Spring 可以自动配置一个 bean 和其他协作 bean 之间的关系，通过检查 BeanFactory 的内容里没有使用和< property>元素。

### 33). 解释自动装配的各种模式？

自动装配提供五种不同的模式供 Spring 容器用来自动装配 beans 之间的依赖注入:

no：默认的方式是不进行自动装配，通过手工设置 ref 属性来进行装配 bean。

byName：通过参数名自动装配，Spring 容器查找 beans 的属性，这些 beans 在 XML 配置文件中被设置为 byName。之后容器试图匹配、装配和该 bean 的属性具有相同名字的 bean。

byType：通过参数的数据类型自动自动装配，Spring 容器查找 beans 的属性，这些 beans 在 XML 配置文件中被设置为 byType。之后容器试图匹配和装配和该 bean 的属性类型一样的 bean。如果有多个 bean 符合条件，则抛出错误。

constructor：这个同 byType 类似，不过是应用于构造函数的参数。如果在 BeanFactory 中不是恰好有一个 bean 与构造函数参数相同类型，则抛出一个严重的错误。

autodetect：如果有默认的构造方法，通过 construct 的方式自动装配，否则使用 byType 的方式自动装配。

### 34). 自动装配有哪些局限性？

自动装配有如下局限性：

重写：你仍然需要使用 和< property>设置指明依赖，这意味着总要重写自动装配。

原生数据类型:你不能自动装配简单的属性，如原生类型、字符串和类。

模糊特性：自动装配总是没有自定义装配精确，因此，如果可能尽量使用自定义装配。

### 35). 你可以在 Spring 中注入 null 或空字符串吗？

完全可以。

Spring 注解

### 36). 什么是 Spring 基于 Java 的配置？给出一些注解的例子

基于 Java 的配置允许你使用 Java 的注解进行 Spring 的大部分配置而非通过传统的 XML 文件配置。

以注解@Configuration 为例，它用来标记类，说明作为 beans 的定义，可以被 Spring IOC 容器使用。另一个例子是@Bean 注解，它表示该方法定义的 Bean 要被注册进 Spring 应用上下

文中。

### 37). 什么是基于注解的容器配置？

另外一种替代 XML 配置的方式为基于注解的配置，这种方式通过字节元数据装配组件而非使用尖括号声明。开发人员将直接在类中进行配置，通过注解标记相关的类、方法或字段声明，而不再使用 XML 描述 bean 之间的连线关系。

### 38). 如何开启注解装配？

注解装配默认情况下在 Spring 容器中是不开启的。如果想要开启基于注解的装配只需在 Spring 配置文件中配置元素即可。

### 39). @Required 注解

@Required 表明 bean 的属性必须在配置时设置，可以在 bean 的定义中明确指定也可通过自动装配设置。如果 bean 的属性未设置，则抛出 BeanInitializationException 异常。

### 40). @Autowired 注解

@Autowired 注解提供更加精细的控制，包括自动装配在何处完成以及如何完成。它可以像 @Required 一样自动装配 setter 方法、构造器、属性或者具有任意名称和/或多个参数的 PN 方法。

### 41). @Qualifier 注解

当有多个相同类型的 bean 而只有其中的一个需要自动装配时，将 @Qualifier 注解和 @Autowire 注解结合使用消除这种混淆，指明需要装配的 bean。

### 42). 在 Spring 框架中如何更有效的使用 JDBC？

使用 Spring JDBC 框架，资源管理以及错误处理的代价都会减轻。开发人员只需通过 statements 和 queries 语句从数据库中存取数据。Spring 框架中通过使用模板类能更有效的使用 JDBC，也就是所谓的 JdbcTemplate(例子)。

### 43). JdbcTemplate

JdbcTemplate 类提供了许多方法，为我们与数据库的交互提供了便利。例如，它可以将数据库的数据转化为原生类型或对象，执行写好的或可调用的数据库操作语句，提供自定义的数据库错误处理功能。

### 44). Spring 对 DAO 的支持

Spring 对数据访问对象(DAO)的支持旨在使它可以与数据访问技术(如 JDBC, Hibernate 及 JDO)方便的结合起来工作。这使得我们可以很容易在的不同的持久层技术间切换，编码时也无需担心会抛出特定技术的异常。

### 45). 使用 Spring 可以通过什么方式访问 Hibernate？

使用 Spring 有两种方式访问 Hibernate：

使用 Hibernate Template 的反转控制以及回调方法

继承 HibernateDAOSupport，并申请一个 AOP 拦截器节点

### 46). Spring 支持的 ORM

Spring 支持一下 ORM：

Hibernate

iBatis

JPA (Java -Persistence API)

TopLink

JDO (Java Data Objects)

OJB

### 47). 如何通过 HibernateDaoSupport 将 Spring 和 Hibernate 结合起来？

使用 Spring 的 SessionFactory 调用 LocalSessionFactory。结合过程分为以下三步：

配置 Hibernate SessionFactory

继承 HibernateDaoSupport 实现一个 DAO

使用 AOP 装载事务支持

### 48). Spring 支持的事务管理类型

Spring 支持如下两种方式的事务管理：

编程式事务管理：这意味着你可以通过编程的方式管理事务，这种方式带来了很大的灵活性，但很难维护。

声明式事务管理：这种方式意味着你可以将事务管理和业务代码分离。你只需要通过注解或者 XML 配置管理事务。

### 49). Spring 框架的事务管理有哪些优点？

它为不同的事务 API(如 JTA, JDBC, Hibernate, JPA, 和 JDO)提供了统一的编程模型。

它为编程式事务管理提供了一个简单的 API 而非一系列复杂的事务 API(如 JTA).

它支持声明式事务管理。

它可以和 Spring 的多种数据访问技术很好的融合。

### 50). 你更推荐那种类型的事务管理？

许多 Spring 框架的用户选择声明式事务管理，因为这种方式和应用程序的关联较少，因此更加符合轻量级容器的概念。声明式事务管理要优于编程式事务管理，尽管在灵活性方面它弱于编程式事务管理(这种方式允许你通过代码控制业务)。

### 51). 解释 AOP

面向切面编程,或 AOP 允许程序员模块化横向业务逻辑，或定义核心部分的功能，例如日志管理和事务管理。

### 52). 切面(Aspect)

AOP 的核心就是切面，它将多个类的通用行为封装为可重用的模块。该模块含有一组 API 提供 cross-cutting 功能。例如,日志模块称为日志的 AOP 切面。根据需求的不同，一个应用程序可以有若干切面。在 Spring AOP 中，切面通过带有@Aspect 注解的类实现。

### 53). 在 Spring AOP 中 concern 和 cross-cutting concern 的区别是什么？

Concern(核心逻辑)：表示在应用程序中一个模块的行为。Concern 可以定义为我们想要实现的功能。

Cross-cutting concern(横向的通用逻辑)：指的是整个应用程序都会用到的功能，它影响整个应用程序。例如，日志管理（Logging）、安全管理（Security）以及数据交互是应用程序的每

个模块都要涉及到的，因此这些都属于 Cross-cutting concern。

### 54). 连接点(Join point)

连接点代表应用程序中插入 AOP 切面的地点。它实际上是 Spring AOP 框架在应用程序中执行动作的地点。

### 55). 通知(Advice)

通知表示在方法执行前后需要执行的动作。实际上它是 Spring AOP 框架在程序执行过程中触发的一些代码。

Spring 切面可以执行一下五种类型的通知:

before(前置通知)：在一个方法之前执行的通知。

after(最终通知)：当某连接点退出的时候执行的通知（不论是正常返回还是异常退出）。

after-returning(后置通知)：在某连接点正常完成后执行的通知。

after-throwing(异常通知)：在方法抛出异常退出时执行的通知。

around(环绕通知)：在方法调用前后触发的通知。

### 56). 切入点(Pointcut)

切入点是一个或一组连接点，通知将在这些位置执行。可以通过表达式或匹配的方式指明切入点。

### 57). 什么是引入？

引入允许我们在已有的类上添加新的方法或属性。

### 58). 什么是目标对象？

被一个或者多个切面所通知的对象。它通常是一个代理对象。也被称做被通知（advised）对象。

### 59). 什么是代理？

代理是将通知应用到目标对象后创建的对象。从客户端的角度看，代理对象和目标对象是一样的。

### 60). 有几种不同类型的自动代理？

BeanNameAutoProxyCreator：bean 名称自动代理创建器

DefaultAdvisorAutoProxyCreator：默认通知者自动代理创建器

Metadata autoproxying：元数据自动代理

### 61). 什么是织入？什么是织入应用的不同点？

织入是将切面和其他应用类型或对象连接起来创建一个通知对象的过程。织入可以在编译、加载或运行时完成。

### 62). 解释基于 XML Schema 方式的切面实现

在这种情况下，切面由使用 XML 文件配置的类实现。

### 63). 解释基于注解方式(基于@AspectJ)的切面实现

在这种情况下(基于@AspectJ 的实现)，指的是切面的对应的类使用 Java 5 注解的声明方式。

### 64). 如何获取 Spring 管理的 Bean 实例？

在受管 Bean 中，可以直接采用各种方式的注入获取。

在非受管 Bean 中，可以采用 BeanFactroy 的 getBean()方法获取。

### 65). 为什么要在项目中使用 Spring？谈谈你对 Spring 的理解。

Ioc 与 Di 特性，依赖可配置，降低偶合度，增加灵活度，易维护

易于和其它框架集成，提供各种帮助类

申明式事务管理，aop、模块化、计划任务……

### 66). 列举 Spring 的注入方式，至少 3 种以上。

构造方法注入、字段注入、seters 方法注入。

### 67). Spring 有哪些事务管理方式？如何使用。

有编程式事务管理和申明式事务管理。我们通常不需要特别细粒度的事务控制，所以更多时候是采用申明式事务管理。

编程式事务管理利用 DefaultTransactionDefinition 实例的 rollback()和 commit()方法，并配合 PlatformTransactionManager 使用；也可以可选的配合 TransactionTemplate 更为方便。

申明式事务管理，首先需要配置申明 TransactionManager。然后可以申明为注解式事务管理配合@Transactional 注解完成，或者使用 Spring 的 AOP 语法申明事务管理。

### 68). Ioc 与 DI 是何关系？

相对于 IoC 而言，依赖注入(DI)更加准确地描述了 IoC 的设计理念。所谓依赖注入，即组件之间的依赖关系由容器在应用系统运行期来决定，

也就是由容器动态地将某种依赖关系的目标对象实例注入到应用系统中的各个关联的组件之中。

### 69). Spring 中 bean 有哪些使用域？

有三个 singleton，在每个 Spring IoC 容器中一个 bean 定义对应一个对象实例；prototype，一个 bean 定义对应多个对象实例和自定义作用域。

如果是在 WebApplicationContext 下，还有 3 个 Web 作用域：request，在一次 HTTP 请求中，一个 bean 定义对应一个实例；即每次 HTTP 请求将会有各自的 bean 实例， 它们依据某个 bean 定义创建而成；session，在一个 HTTP Session 中，一个 bean 定义对应一个实例；global session，在一个全局的 HTTP Session 中，一个 bean 定义对应一个实例。典型情况下，仅在使用 portlet context 的时候有效。

所以在 WebApplicationContext 下共 6 个作用域。

### 70). 如何注入 Properties？

```
<property name="Bean属性名">
    <props>
            <prop key="键名">键值</prop>
    </props>
</property>
```

### 71). 什么是 AOP，在 Spring 中有哪些实现方式？分别如何实现？

### 72). 你在项目中是如何使用 AOP 的？

### 73). 列举 Spring 的常用特性，至少 3 个以上。

IOC、DI、AOP、JdbcTemplate、事务管理

### 74). 描述 Spring 中 Bean 的生命周期。

## 3. SpringMvc

### 1). 什么是 Spring 的 MVC 框架？

Spring 提供了一个功能齐全的 MVC 框架用于构建 Web 应用程序。Spring 框架可以很容易的和其他的 MVC 框架融合(如 Struts)，该框架使用控制反转(IOC)将控制器逻辑和业务对象分离开来。它也允许以声明的方式绑定请求参数到业务对象上。

### 2). DispatcherServlet

Spring 的 MVC 框架围绕 DispatcherServlet 来设计的，它用来处理所有的 HTTP 请求和响应。

### 3). WebApplicationContext

WebApplicationContext 继承了 ApplicationContext，并添加了一些 web 应用程序需要的功能。和普通的 ApplicationContext 不同，WebApplicationContext 可以用来处理主题样式，它也知道如何找到相应的 servlet。

### 4). 什么是 Spring MVC 框架的控制器？

控制器提供对应用程序行为的访问，通常通过服务接口实现。控制器解析用户的输入，并将其转换为一个由视图呈现给用户的模型。Spring 通过一种极其抽象的方式实现控制器，它允许用户创建多种类型的控制器。

### 5). @Controller annotation

@Controller 注解表示该类扮演控制器的角色。Spring 不需要继承任何控制器基类或应用 Servlet API。

### 6). @RequestMapping annotation

@RequestMapping 注解用于将 URL 映射到任何一个类或者一个特定的处理方法上。

### 7). 简述 SpringMvc 的处理流程。

### 8). 为什么要在项目中使用 SpringMvc？

**9). 你怎么理解 SpringMvc 与 WebMvc(或 Mvc)的关系？**

**10). SpringMvc 中,bean 具备哪些作用域？**

**11). Controller 是线程安全的吗？你如何处理线程安全？**

**12). SpringMvc 中如何使用 Ajax？**

**13). 在 SpringMvc 中如何校验表单数据？**

**14). 在 SpringMvc 中如何处理异常？**

## 4. Hibernate

**1). 持久对象（PO）有哪些状态？它们各自特点是什么？**

**2). hibernate 和 jdbc 有什么联系？**

**3). Session 上的 get()与 load()有何区别？**

**4). 简述 Session 上的 save()、marge()、saveOrUpdate()、update()方法的联系及区别。**

**5). 乐观锁和悲观锁有何区别？在 Hibernate 中是如何实现的？**

**6). 为什么要在项目中使用 Hibernate？它的工作原理是什么？**

**7). set 的 fatch 有何用处？如何使用？**

有两个可选值:join 和 select 这个属性决定了你在查询的时候，是否先查主表记录

**8). Hibernate 支持哪些继承映射方式？**

**9). 如果要使用 Hibenate 插入 10 万条数据，该如何操作？**

**10). 关联映射的 cascade 有何用处？如何使用？**

all: 所有情况下均进行关联操作，即 save-update 和 delete。
none: 所有情况下均不进行关联操作。这是默认值。
save-update: 在执行 save/update/saveOrUpdate 时进行关联操作。
delete: 在执行 delete 时进行关联操作。

all-delete-orphan: 当一个节点在对象图中成为孤儿节点时，删除该节点。比如在一个一对多的关系中，Student 包含多个 book，当在对象关系中删除一个 book 时，此 book 即成为孤儿节点。

### 11). Hibernate 一级缓存与二级缓存有什么区别？

第一级是 Session 的缓存。由于 Session 对象的生命周期通常对应一个数据库事务或者一个应用事务，因此它的缓存是事务范围的缓存。第一级缓存是必需的，不允许而且事实上也无法比卸除。在第一级缓存中，持久化类的每个实例都具有唯一的 OID。当一次会话结束(session 关闭)时就会清除，缓存意义不大；

第二级缓存是一个可插拔的的缓存插件，它是由 SessionFactory 负责管理。由于 SessionFactory 对象的生命周期和应用程序的整个过程对应，因此第二级缓存是进程范围或者集群范围的缓存。这个缓存中存放的对象的松散数据。第二级对象有可能出现并发问题，因此需要采用适当的并发访问策略，该策略为被缓存的数据提供了事务隔离级别。缓存适配器用于把具体的缓存实现软件与 Hibernate 集成。第二级缓存是可选的，可以在每个类或每个集合的粒度上配置第二级缓存。即使会话结束也不会被清除。在程序中可以多个用户共用这个缓存，缓存意义大。（如：大量用户去查询同一条数据时，会现在缓存中找，找不到再在数据库中找，效率高）

### 12). 什么是 Hibernate 的并发机制？怎么去处理并发问题？

### 13). Hibernate 是如何进行延迟加载的？

### 14). Hibernate 的查询方式有哪些？

### 15). 如何优化 Hibernate？

### 16). Hibernate 自带的分页机制是什么？如果不使用 Hibernate 自带的分页，还有哪些分页方案？

### 17). hibernate 的三种状态之间如何转换？

当对象由瞬时状态(Transient)一 save()时，就变成了持久化状态。

当我们在 Session 里存储对象的时候，实际是在 Session 的 Map 里存了一份，也就是它的缓存里放了一份，然后，又到数据库里存了一份，在缓存里这一份叫持久对象(Persistent)。

Session 一 Close()了，它的缓存也都关闭了，整个 Session 也就失效了，这个时候，这个对象变成了游离状态(Detached)，但数据库中还是存在的。

当游离状态(Detached)update()时，又变为了持久状态(Persistent)。

当持久状态(Persistent)delete()时，又变为了瞬时状态(Transient),

### 18). 使用 Hibernate 时如何处理 n+1 问题？

### 19). hibernate 进行多表查询每个表中各取几个字段，也就是说查询出来的结果集没有一个实体类与之对应如何解决？

解决方案一：使用 Object[]或取出数据，然后自己组 bean

解决方案二：获取 Map 类型数据

解决方案三：每个表的 bean 写构造函数，比如表一要查出 field1,field2 两个字段，那么有一个构造函数就是 Bean(type1 filed1,type2 field2)，然后在 hql 里面就可以直接生成这个 bean 了。

### 20). hibernate 的核心配置文件是什么及其作用？

Hibernate.cfg.xml:

1)数据库连接、

2)数据库连接时相关属性例如 show_sql

3)指定相关的映射文件*.hbm.xml:具体的 o/r mapping 说明

### 21). 写 Hibernate 的一对多和多对一双向关联的 orm 配置。

### 22). hibernate 的 inverse 属性有何作用？

### 23). java 当有哪些事务管理方式（类型）？

## 5. 综合

### 1). 对比 Struts 与 SpringMvc，各自有何优缺点？

### 2). 你在 SSH 中有没有使用到事务？是如何处理的？

### 3). 对比 MyBatis(iBatis)与 Hibernate，各自有何优缺点？

### 4). MyBatis(iBatis)与 Hibernate 有何异同？

相同点：屏蔽 jdbc api 的底层访问细节，使用我们不用与 jdbc api 打交道，就可以访问数据。

ibatis 的好处：屏蔽 jdbc api 的底层访问细节；将 sql 语句与 java 代码进行分离;提供了将结果集自动封装称为实体对象和对象的集合的功能，queryForList 返回对象集合，用 queryForObject 返回单个对象；提供了自动将实体对象的属性传递给 sql 语句的参数。

Hibernate 是一个全自动的 orm 映射工具，它可以自动生成 sql 语句,ibatis 需要我们自己在 xml 配置文件中写 sql 语句，hibernate 要比 ibatis 功能负责和强大很多。因为 hibernate 自动生成 sql 语句，我们无法控制该语句，我们就无法去写特定的高效率的 sql。对于一些不太复杂的 sql 查询，hibernate 可以很好帮我们完成，但是，对于特别复杂的查询，hibernate 就很难适应了，这时候用 ibatis 就是不错的选择，因为 ibatis 还是由我们自己写 sql 语句。

**5). 对比 Struts+Spring+Hibernate 与 SpringMvc+SpringCore+MyBatis 架构优缺点？**

# ♦前端

## 1. Ajax

### 1). 如何使用 Ajax 从服务器获取复杂数据？

### 2). 什么是 Ajax？

Ajax（Asynchronous JavaScript + XML），即异步 JavaScript + XML 的缩写，主要用来页面异步刷新，也是构建 RIA 的一种基础技术。

### 3). 解释 XMLHttpRequest 是什么？

XMLHttpRequest，是我们得以实现异步通讯的根本。最早在 IE 5 中以 ActiveX 组件实现；

最近，Mozilla 1.0 和 Safari 1.2 中实现为本地对象。XMLHttpRequest 虽然不是 W3C 标准，但却得到了 FireFox、Safari、Opera、Konqueror、IE 等绝大多数浏览器的支持。

### 4). 谈谈你对 Ajax 的理解。你在项目中如何使用 Ajax？手写一个简单的 Ajax 操作。

Ajax（Asynchronous JavaScript + XML），即异步 JavaScript + XML 的缩写，主要用来页面异步刷新，也是构建 RIA 的一种基础技术。

因为它涉及浏览器兼容、跨域等问题，在项目中一般会使用一些基础类库辅助实现，如 jQuery 等。

一个简单的 Ajax 操作如下。

```
var xhr = new XMLHttpRequest();
//在环境中需要做浏览器兼容，这里省略了。。

xhr.onreadystatechange = function() {
//这里注册当xhr状态发生改变后调用事件
    if( xhr.readyState == 4 ) {
//通常在读取状态为4的时候才能获取到部分数据，所以一般状态在4的时候才进
//行处理
        if(status==200) {
            //当正常请求到资源时的处理,可以调用xhr.responseText或/
        /xhr.responseXml获取数据
        }
        else {
            //当请求资源失败时的处理
        }
    }
}

xhr.open( "GET", url);//设置xhr的请求方式和url,这里使用的是GET方式,
                      //如果有参数，则连接在url后面
 /*
```

```
  如果是POST请求，还当设置请求的Content-Type
  数据使用send作为参数发送
  */
xhr.send();
```

### 5). 谈谈你对 JSON 的理解。

JSON(JavaScript Object Notation) 是一种轻量级的数据交换格式。 易于人阅读和编写。同时也易于机器解析和生成。 它基于 JavaScript Programming Language, Standard ECMA-262 3rd Edition - December 1999 的一个子集。 JSON 采用完全独立于语言的文本格式，但是也使用了类似于 C 语言家族的习惯（包括 C, C++, C#, Java, JavaScript, Perl, Python 等）。这些特性使 JSON 成为理想的数据交换语言。

所以它往往在 AJAX 中替代 XML，交换数据。

### 6). 你的项目中有使用到跨域吗？你在项目中是如何处理 JS 跨域问题的？

有。

一个网站，主要是使用其它网站提供的 javascrip api 如 QQ。使用 script 的 src 可以直接读取跨域资源。

当然跨域还有其它处理方式：如代理服务器、改变 domain、JSONP 等。

### 7). 你在项目中有使用到网页到服务器的即时通信吗？说说你都采用什么手段处理以及你所知道的处理办法？

没有用到，但我知道 html 的 websockets、flash 的 socket、ajax 长轮询等都可以实现。

### 8). 你在 AJAX 中有遇到乱码吗？如果遇到，你是如何解决的？

遇到过。

一般我首先统一页面和服务器编码，对请求和响应的 Content-Type 设置正确编码；对请求参数进行编码处理。

# ♦Sql

## 1. 基础

**1). 谈谈什么事务处理？**

**2). 请写出数据类型(int char varchar datetime text)的意思；请详述 varchar 和 char 的区别？**

**3). 有如下 User 表，请按要求写出 SQL。**

| Name | Tel | Content | Date |
|---|---|---|---|
| 张三 | 13333663366 | 大专毕业 | 2006-10-11 |
| 张三 | 13612312331 | 本科毕业 | 2006-10-15 |
| 张四 | 021-55665566 | 中专毕业 | 2006-10-15 |

(a) 有一新记录(小王 13254748547 高中毕业 2007-05-06)请用 sql 语句新增至表中

insert into user('name','tel','content','date') values('小王','13254748547','高中毕业','2007-05-06')

(b) 请用 sql 语句把张三的时间更新成为当前系统时间

update user set date=date_format(now(),'%y-%m-%d') where name='张三'

(c) 请写出删除名为张四的全部记录

delete from user where name='张四'

**4). 有一个留言板,分别有 2 个表,1 是用户表 包括用户名 用户信息 email 属性 ,2 是留言表表,包括用户 Id,标题,留言内容,发表时间：**

**(1).根据需求设计表的结构；**

**(2).查询发表留言超过 10 个的用户的用户名 并按留言个数降序排列**

**5). 两张表中，比如 login,user 表，如果 login 里面的数据删了，但是 user 表里面的数据还在，怎么处理?**

**6). 有表如下**

| id | pid | firstname | lastname |
|---|---|---|---|
| 1 | 0 | 张 | 三 |
| 2 | 1 | 李 | 四 |

求查询结果：

| id | name | parname | |
|---|---|---|---|
| | 2 | 李四 | 张三 |

### 7). union 和 union all 有什么不同？通常谁的效率更高？

它们都用来作查询结果集的连接，不同之处在于 union 会去除两个结果集当中的重复值，而 union all 不会。

因为 union all 没有去掉重复值的额外开销，所以通常效率更高。

### 8). 在子查询中使用 in 与 exists 有何不同？通常谁的效率更高？

in 需要和子查询的结果行进行逐行匹配，而 exists 只需要确定子查询有没有查询结果，没有逐行匹配的开销。

所以通常 exists 在子查询中的效率比 in 高。

### 9). join 与子查询在联表查询数据情况下，谁的效率更高？

join 只进行一次查询，就直接返回全部查询结果；而子查询每一行都会作一次匹配查询。

所以通常 join 比子查询效率更高；

但是，如果子查询表的数据相当少或者所有子查询都是相同结果时，那么每次子查询的开销相当小，就有可能子查询效率更高。

### 10). 数据库三范式是什么？

第一范式（1NF）

在任何一个关系数据库中，第一范式（1NF）是对关系模式的基本要求，不满足第一范式（1NF）的数据库就不是关系数据库。

所谓第一范式（1NF）是指数据库表的每一列都是不可分割的基本数据项，同一列中不能有多个值，即实体中的某个属性不能有多个值或者不能有重复的属性。如果出现重复的属性，就可能需要定义一个新的实体，新的实体由重复的属性构成，新实体与原实体之间为一对多关系。在第一范式（1NF）中表的每一行只包含一个实例的信息。例如，对于图 3-2 中的员工信息表，不能将员工信息都放在一列中显示，也不能将其中的两列或多列在一列中显示；员工信息表的每一行只表示一个员工的信息，一个员工的信息在表中只出现一次。简而言之，第一范式就是无重复的列。

第二范式（2NF）

第二范式（2NF）是在第一范式（1NF）的基础上建立起来的，即满足第二范式（2NF）必须先满足第一范式（1NF）。第二范式（2NF）要求数据库表中的每个实例或行必须可以被唯一地区分。为实现区分通常需要为表加上一个列，以存储各个实例的唯一标识。如图 3-2 员工信息表中加上了员工编号（emp_id）列，因为每个员工的员工编号是唯一的，因此每个员工可以被唯一区分。这个唯一属性列被称为主关键字或主键、主码。

第二范式（2NF）要求实体的属性完全依赖于主关键字。所谓完全依赖是指不能存在仅依赖主关键字一部分的属性，如果存在，那么这个属性和主关键字的这一部分应该分离出来形成一个新的实体，新实体与原实体之间是一对多的关系。为实现区分通常需要为表加上一个列，以存储各个实例的唯一标识。简而言之，第二范式就是非主属性非部分依赖于主关键字。

第三范式（3NF）

满足第三范式（3NF）必须先满足第二范式（2NF）。简而言之，第三范式（3NF）要求一个数据库表中不包含已在其它表中已包含的非主关键字信息。例如，存在一个

部门信息表，其中每个部门有部门编号（dept_id）、部门名称、部门简介等信息。那么在图 3-2 的员工信息表中列出部门编号后就不能再将部门名称、部门简介等与部门有关的信息再加入员工信息表中。如果不存在部门信息表，则根据第三范式（3NF）也应该构建它，否则就会有大量的数据冗余。简而言之，第三范式就是属性不依赖于其它非主属性。

**11). 表中有 A B C 三列,用 SQL 语句实现：当 A 列大于 B 列时选择 A 列否则选择 B 列，当 B 列大于 C 列时选择 B 列否则选择 C 列。**

Oracle：

```
select
    case
        when A>B then A
    else
        case
            when B>C then B
        else
            C
        end
    end
from
    tableName;
```

MySql：

```
select
    if(A>B,A,
        if(B>C,B,C)
    )
from
    tableName;
```

**12). 请取出 tb_send 表中日期(SendTime 字段)为当天的所有记录?(SendTime 字段为 datetime 型，包含日期与时间)。**

**13). 有如下表和数据，用一条 SQL 语句 查询出每门课都大于 80 分的学生姓名**

| name | kecheng | fenshu |
|---|---|---|
| 张三 | 语文 | 81 |
| 张三 | 数学 | 75 |
| 李四 | 语文 | 76 |
| 李四 | 数学 | 90 |
| 王五 | 语文 | 81 |
| 王五 | 数学 | 100 |
| 王五 | 英语 | 90 |

**14). 如何复制表结构和数据到新表？用一条 Sql 完成。**

```
create table
    newTableName
as
    select
        *
    from
        oldTableName
```

**15). 如何只复制结构，而不复制数据到新表？用一条 Sql 完成。**

```
create table    --未验证
    newTableName
as
    select
        *
    from
        oldTableName
    where
        rownum>1
```

**16). 有如下表**

| courseid | coursename | score |
|---|---|---|
| 1 | java | 70 |
| 2 | oracle | 90 |
| 3 | xml | 40 |
| 4 | jsp | 30 |
| 5 | servlet | 80 |

**为了便于阅读,使用一条 Sql 查询此表后的结果显式如下(及格分数为 60):**

| courseid | coursename | score | mark |
|---|---|---|---|
| 1 | java | 70 | pass |
| 2 | oracle | 90 | pass |
| 3 | xml | 40 | fail |
| 4 | jsp | 30 | fail |
| 5 | servlet | 80 | pass |

**请写出此查询语句：**

**17). 有如下表及数据**

| year | month | amount |
|---|---|---|
| 1991 | 1 | 1.1 |
| 1991 | 2 | 1.2 |
| 1991 | 3 | 1.3 |
| 1991 | 4 | 1.4 |

| | | |
|---|---|---|
| 1992 | 1 | 2.1 |
| 1992 | 2 | 2.2 |
| 1992 | 3 | 2.3 |
| 1992 | 4 | 2.4 |

**为了便于阅读,使用一条 Sql 查询此表后的结果显式如下**

| year | m1 | m2 | m3 | m4 |
|---|---|---|---|---|
| 1991 | 1.1 | 1.2 | 1.3 | 1.4 |
| 1992 | 2.1 | 2.2 | 2.3 | 2.4 |

**请写出此查询语句：**

```
select  --未验证语法
    year,
    max(case
        when month==1 then amount
        else null
    end) m1,
    (case
        when month==2 then amount
        else null
    end) m2,
    max(case
        when month==3 then amount
        else null
    end) m3,
    max(case
        when month==4 then amount
        else null
    end) m4,
from
    tableName
group by
    year;
```

### 18). 描述你所知道的分页方式。

通常有逻辑分页（在程序中对结果集分页）和物理分页（使用 Sql 分页）。

通常我们采用后者，通常在 Oracle 中使用 rownum 进行分页，在 mysql 中采用 limit 进行分页，在 sqlserver 中采用 top 进行分页。

### 19). 什么是“事务”？

事务(Transaction)是访问并可能更新数据库中各种数据项的一个程序执行单元(unit)。事务通常由高级数据库操纵语言或编程语言（如 SQL，C++或 Java）书写的用户程序的执行所引起，并用形如 begin transaction 和 end transaction 语句（或函数调用）来界定。事务由事务开始(begin transaction)和事务结束(end transaction)之间执行的全体操作组成。

例如：在关系数据库中，一个事务可以是一条 SQL 语句，一组 SQL 语句或整个程序。

特性

事务是恢复和并发控制的基本单位。

事务应该具有 4 个属性：原子性、一致性、隔离性、持续性。这四个属性通常称为 ACID 特性。

- 原子性（atomicity）。一个事务是一个不可分割的工作单位，事务中包括的操作要么都做，要么都不做。
- 一致性（consistency）。事务必须是使数据库从一个一致性状态变到另一个一致性状态。一致性与原子性是密切相关的。
- 隔离性（isolation）。一个事务的执行不能被其他事务干扰。即一个事务内部的操作及使用的数据对并发的其他事务是隔离的，并发执行的各个事务之间不能互相干扰。
- 持久性（durability）。持续性也称永久性（permanence），指一个事务一旦提交，它对数据库中数据的改变就应该是永久性的。接下来的其他操作或故障不应该对其有任何影响。

**20). 一个用户表中有一个积分字段，假如数据库中有 100 多万个用户，若要在每年第一天凌晨将积分清零，你将考虑什么，你将想什么办法解决？**

通常不会这样设计表，会有单独的表存储每年的积分，到第二年时，由于日期不同，查询结果当然积分为 0。

**21). 提升 SQL 语句的查询性能：**

数据库设计与规划：

Primary Key 字段的长度尽量小，能用 small integer 就不要用 integer

字符字段如果长度固定，就不要用 varchar、nvarchar 类型

设计字段时，如果其值可有可无，最好给一个默认值，并设成“不允许 NULL”

适当地创建索引：

a、 Primary Key 字段可以自动创建索引，而 Foreign Key 字段不可以。

b、为经常被查询或排序的字段创建索引

c、 创建索引字段的长度不宜过长，不要用超过 20 个字符。

d、不要为内容重复性高的字段创建索引

e、 不要为使用率低的字段建立索引

f、 不宜为过多字段建立索引，否则影响到 insert update delete 语句的性能

g、如果说数据表存放的数据很少，就不必刻意使用权索引。

使用索引功能：

在查询数据表时，使用索引查询可以极大提升查询速度，但是如果 where 子句书写不当。即使某些列存在索引，也不能使用该索引查询，而同样会使用全表扫描，这就造成了查询速度的降低。在 where 语句中避免使用以下关键词：NOT、！=、<>、!>、!<、Exists、In、Like、||。使用 LIKE 关键字做模糊查询时，即使已经为某个字段建立索引，但需要以常量字符开头才会使用到索引，如果以“%”开头则不会使用索引。例如“name Like ‘%To’”不启用 name 字段上的索引；而“name LIKE ‘TO %’”会启用 name 字段上的索引。

避免在 where 子句中对字段使用函数：

对字段使用函数，也等于对字段做运算或连接的动作，调用函数的次数与数据表的记录成正比。如果数据表内记录很多时，会严重影响查询性能。

在 AND 与 OR 的使用：

在 AND 运算中，只要有一个条件使用到索引，即可大幅提升查询速度。但在 OR 运算中，则要所有的条件都有使用到索引才能提升查询速度，因此使用 OR 运算符时需要特别小心

JOIN 与子查询：

相对于子查询，如果能使用 JOIN 完成的查询，一般建议使用后者。原因除了 JOIN 的语法较容易理解外，在多数的情况下，JOIN 的性能也会比子查询高。

其他查询技巧：

DISTINCT、ORDER BY 语法，会让数据库做额外的计算。如果没有要过滤重复记录的需求，使用 Union All 会比 Union 更好，因为后者会加入类似 DIStinct 的算法。

尽可能使用存储过程(Store Procedure)：

Store Procedure 除了经过事先编译、性能较好以外，也可减少 SQL 语句在网络中的传递，方便商业逻辑的重复使用。

尽可能在数据源过滤数据

使用 Select 语法时，尽量先用 SQL 条件或 Store Procedure 过滤所要的信息，避免将大量冗余数据返回给程序，然后由程序处理

## 22). 列举各种表连接方式。

| 连接类型 | 定义 | 例子 |
|---|---|---|
| 内连接 | 只连接匹配的行 | select A.c1,B.c2 from A join B on A.c3 = B.c3; |
| 左外连接 | 包含左边表的全部行（不管右边的表中是否存在与它们匹配的行）以及右边表中全部匹配的行 | select A.c1,B.c2 from A left join B on A.c3 = B.c3; |
| 右外连接 | 包含右边表的全部行（不管左边的表中是否存在与它们匹配的行）以及左边表中全部匹配的行 | select A.c1,B.c2 from A right join B on A.c3 = B.c3; |
| 全外连接 | 包含左、右两个表的全部行，不管在另一边的表中是否存在与它们匹配的行 | select A.c1,B.c2 from A full join B on A.c3 = B.c3; |
| （ theta ） 连接 | 使用等值以外的条件来匹配左、右两个表中的行 | select A.c1,B.c2 from A join B on A.c3 != B.c3; |
| 交叉连接 | 生成笛卡尔积——它不使用任何匹配或者选取条件，而是直接将一个数据源中的每个行与另一个数据源的每个行一一匹配 | select A.c1,B.c2 from A,B; |

### 23). 存储过程和函数的区别

存储过程是用户定义的一系列 sql 语句的集合，涉及特定表或其它对象的任务，用户可以调用存储过程，而函数通常是数据库已定义的方法，它接收参数并返回某种类型的值并且不涉及特定用户表。

### 24). 事务是什么？

事务是作为一个逻辑单元执行的一系列操作，一个逻辑工作单元必须有四个属性，称为 ACID(原子性、一致性、隔离性和持久性)属性，只有这样才能成为一个事务:

原子性：事务必须是原子工作单元;对于其数据修改，要么全都执行，要么全都不执行。

一致性：事务在完成时，必须使所有的数据都保持一致状态。在相关数据库中，所有规则都必须应用于事务的修改，以保持所有数据的完整性。事务结束时，所有的内部数据结构(如 B 树索引或双向链表)都必须是正确的。

隔离性：由并发事务所作的修改必须与任何其它并发事务所作的修改隔离。事务查看数据时数据所处的状态，要么是另一并发事务修改它之前的状态，要么是另一事务修改它之后的状态，事务不会查看中间状态的数据。这称为可串行性，因为它能够重新装载起始数据，并且重播一系列事务，以使数据结束时的状态与原始事务执行的状态相同。

持久性：事务完成之后，它对于系统的影响是永久性的。该修改即使出现系统故障也将一直保持。

### 25). 游标的作用?如何知道游标已经到了最后？

游标用于定位结果集的行，通过判断全局变量@@FETCH_STATUS 可以判断是否到了最后，通常此变量不等于 0 表示出错或到了最后。

### 26). 触发器分为事前触发和事后触发，这两种触发有何区别?语句级触发和行级触发有何区别？

事前触发器运行于触发事件发生之前，而事后触发器运行于触发事件发生之后。通常事前触发器可以获取事件之前和新的字段值。

语句级触发器可以在语句执行前或后执行，而行级触发在触发器所影响的每一行触发一次。

### 27). 简述聚族索引、非聚族索引、组合索引？

### 28). 简述临时表的生命周期

### 29). crate table T(num  integer)，不用 max(),min()求最大值；

## 2. MySql

### 1). MySQ 自增类型(通常为表 ID 字段),必需将其设为什么字段(自增关键字)？

### 2). MYSQL 取得当前时间的函数是？格式化日期的函数是？

### 3). mysIq 数据库  InnoDB 和  MyISAM 的区别在哪？

MyISAM 是非事务安全型的，而 InnoDB 是事务安全型的。

MyISAM 锁的粒度是表级，而 InnoDB 支持行级锁定。

MyISAM 支持全文类型索引，而 InnoDB 不支持全文索引。

MyISAM 表是保存成文件的形式，在跨平台的数据转移中使用 MyISAM 存储会省去不少的麻烦。

InnoDB 表比 MyISAM 表更安全，可以在保证数据不会丢失的情况下，切换非事务表到事务表（alter table tablename type=innodb）。

综上所述，如果一个 table 查询更多而没有严格的数据完整性要求，适合选用 MyISAM 引擎；反之，如果有严格的数据完整性要求或者增删改操作比较多面查询比较少，适合选用 InnoDB 引擎。

# 3. 性能与安全。

### 1). 对于大流量的网站,您采用什么样的方法来解决访问量问题？

首先，确认服务器硬件是否足够支持当前的流量

其次，优化数据库访问。

第三，禁止外部的盗链。

第四，控制大文件的下载。

第五，使用不同主机分流主要流量

第六，使用流量分析统计软件。

### 2). 如何实现 session 的多台 web 共享?

Session 复制、数据库共享、memcache 共享

# 4. 网络

### 1). 写出、你能想到的所有 HTTP 返回状态值，并说明用途。

### 2). 在 HTTP 1.0 中，状态码 301、302、304、401、404、500、503、504 表示什么意思？

301:服务器已经搬迁，永久转移

302:URL 被临时替换；

304:页面未更改

401:未授权访问受保护页面；

404:页面未找到

500:服务器出错，无法与客户端产生会话请求；

503:服务器过载而未响应

504:服务器代理状态，出现非法服务响应；

### 3). http,ssh,ftp,https，telnet..的默认端口是什么？

http:80 ftp:21 ssh:22 telnet:23 https:439

### 4). TCP 协议与 UDP 协议各有何优缺点？

TCP：安全，但需要保持连接，资源消耗大，效率低。

UDP：不需要保持连接，效率高，但可能会丢包，不安全。

### 5). Http 协议中的 Content-Type 是何含义？

指请求或响应体的 mime 类型。

### 6). HTTP 协议的默认端口号是多少？

80

### 7). 谈谈你对 TCP/IP 的理解。

TCP/IP 分层模型（TCP/IP Layening Model）被称作因特网分层模型(Internet Layering Model)、因特网参考模型(Internet Reference Model)。TCP/IP 分层模型的四层。

| 层 | 作用 | 协议 |
| --- | --- | --- |
| 第四层，应用层 | 负责对软件提供接口以使程序能使用网络服务。 | FTP<br>HTTP<br>SMTP<br>其它…… |
| 第三层，传输层 | 提供端到端的通信服务。 | TCP<br>UDP |
| 第二层，网间层 | 负责数据的包装、寻址和路由。同时还包含网间控制报文协议(Internet Control Message Protocol,ICMP)用来提供网络诊断信息。 | ICMP<br>IP |
| 第一层，网络接口 | 提供数据结构和实际物理硬件之间的接口。 | ARP<br>RARP<br>其它…… |

## 5. XML

### 1). XML 文档定义有几种形式？它们之间有何本质区别？

有 dtd 与 schema 两种形式；

其本质区别在于 dtd 是单独的语法，而 schema 本身也是 xml

### 2). xml 有哪些解析技术?区别是什么？

DOM:会将全部数据装入内存，生成 Dom 对象结构，处理大型文件时其性能下降的非常厉害。这个问题是由 DOM 的树结构所造成的，这种结构占用的内存较多，而且 DOM 必须在解析文件之前把整个文档装入内存,适合对 XML 的随机访问 SAX:不现于 DOM,

SAX:是事件驱动型的 XML 解析方式。它顺序读取 XML 文件，不需要一次全部装载整个文件。 当遇到像文件开头，文档结束，或者标签开头与标签结束时，它会触发一个事件，用户通过在其回调事件中写入处理代码来处理 XML 文件，适合对 XML 的顺序访问

TAX:Streaming API for XML (StAX) 如其名称所暗示的那样，StAX 把重点放在流上。实际上，StAX 与其他方法的区别就在于应用程序能够把 XML 作为一个事件流来处理。将 XML 作为一组事件来处理的想法并不新颖（事实上 SAX 已经提出来了），但不同之处在于 StAX 允许应用程序代码把这些事件逐个拉出来，而不用提供在解析器方便时从解析器中接收事件的处理程序。

# 6. 项目管理

### 1). 您是否用过版本控制软件？如果有您用的版本控制软件的名字是？

Svn、cvs、git

### 2). 一个软件的生命周期

软件生命周期(SDLC，Systems Development Life Cycle,SDLC)是软件的产生直到报废的生命周期，

周期内有问题定义、可行性分析、总体描述、系统设计、编码、调试和测试、验收与运行、维护升级到废弃等阶段，

这种按时间分程的思想方法是软件工程中的一种思想原则，即按部就班、逐步推进，每个阶段都要有定义、工作、审查、

形成文档以供交流或备查，以提高软件的质量。但随着新的面向对象的设计方法和技术的成熟，

软件生命周期设计方法的指导意义正在逐步减少。