1.java 语言是强类型还是弱类型语言?为什么?

Java 是**强类语言**，在使用变量时有如下规则：

- 变量必须声明，并且初始化以后才能使用。
- 变量必须有明确的类型（type）。
- 变量不能重复定义。

javascript 是一种弱类型语言，即变量在声明时，不能明确声明其类型

变量的类型是在运行时确定的，并且可以随时改变

2.JAVA 的数据类型有哪些?

⑴基本数据类型（八种）：

整型： byte 1 字节, short 2 字节, int 4 字节, long 8 字节

浮点型： float 4 字节, double 8 字节

【float 类型共 32 位（不 int 相同），其中 1 位为符号位, 指数 8 位, 尾数 23 位。】

【double 类型能表示 64 位，其中 1 位符号位,11 位指数,52 位尾数】

【浮点数的字面量默认是 double】

字符型： char 2 字节【unicode 编码值】

boolean 型： boolean 1 字节 【值只有 true 和 false】

隐式类型转换：

正方向: char→

byte→short→ int→long→float→double

负方向:

⑵引用类型（自定义类型）:对象:比如 String

数组:int[]

接口：interface

3.JAVA 中成员变量和局部变量的区别?

⑴成员变量：是在类范围内定义的（也叫成员属性）

类属性：使用 static 修饰的就是类属性。

作用域: 类属性的作用域与这个类的生存范围相同，它作为类的一个成员，与类共存亡。只要类存在，程序就可以访问该类的类属性。

实例属性：不被 static 修饰的就是实例属性。

作用域： 实例属性则从这个类的实例（对象）被创建开始存在，直到系统完全销毁这个实例，它作为实例（对象）的一个成员，与实例（对象）共存亡。只要实例存在，程序就可以访问该实例的实例属性。

⑵局部变量：在一个方法内定义的变量。(包括方法的形式参数)

1.形参：作用域是整个方法体

2.方法局部变量：一个代码块中

3.代码块局部变量：一个代码块中

注意：局部变量除了形式参数外，都必须显示初使化(必须显示指定初使值)。否则不可以访问它们。

形式参数不须显示初使化，它在被调用时由系统完成。

4.前++和后++的区别?

i++, 后++, 先将 i 的值作为整个表达的值, 然后将 i 增加 1。

）　++i，　先++，　先将 i 增加 1，　然后将 i 的值作为整个表达的值。

5. 短路运算符和非短路运算符的区别?

短路运算符[条件 1 &&　条件 2]，如果条件 1 不成立，则条件 2 不执行；
非短路运算符[条件 1 &　条件 2]，两个条件都会执行。

6.怎样取得数组的长度?

数组:arr.length　集合:list.size()　字符串:str.length()

7. 实现数组拷贝（复制）的 2 种方法?

System.arraycopy(src , srcPos, dest , destPos , length) ;
- src - 源数组。
- srcPos - 源数组中的起始位置。
- dest - 目标数组。
- destPos - 目标数据中的起始位置。
- length - 要复制的数组元素的数量。

Arrays.copyOf(src , length)：是 JDK1.6 版本提供的方法，比起 System.arraycopy()使用更简便.

(注:当然，可以用新建数组用 for 循环的方式进行复制)

8.java 中的 4 种访问制权限有哪些? 分别作用范围是什么?

(1).public：最大访问控制权限，对所有的类都可见。
(2).protect：修饰的，在类内部、同一个包、子类中能访问

(3).default：包访问权限，即同一个包中的类可以可见。默认不显式指定访问控制权限时就是 default 包访问控制权限。

(4).private：最严格的访问控制权限，仅该类本身可见。

(注:访问控制修饰符可以修饰类，成员变量，方法，但是修饰类只用 public 和 default)

9.JAVA5 的新特性有哪些?

(1)循环(For-each 循环)

for (type variable : array){ body}

for (type variable : arrayList){body}

而 1.4 必须是：

for (int i = 0; i < array.length; i++){ type variable = array[i]; body}

for (int i = 0; i < arrayList.size(); i++){type variable = (type) arrayList.get(i); body}

(2)泛型

以 ArrayList 为例，包括创建一个容器对象和取得容器内对象操作：

1.5 ArrayList<Type> arrayList =new ArrayList<Type>(); arrayList.get(i)

1.4 ArrayList arrayList =new ArrayList(); (Type) arrayList.get(i)

(3) 自动装箱拆箱

在 JDK5.0 以前，在原始类型与相应的包装类之间的转化是不能自动完成的。要完成这种转化，需要手动调用包装类的构造函数，在 JDK5.0 环境中，可以自动转化：

1.5 Integer wrapper = n; int n = wrapper;

1.4 Integer wrapper = new Integer(n); int n = wrapper.intValue();

自动装包/拆包大大方便了基本类型数据和它们包装类地使用。

自动装包：基本类型自动转为包装类.(int >> Integer)；

自动拆包：包装类自动转为基本类型.(Integer >> int)；

(4)静态导入

静态导入功能对于 JDK 5.0 以前的版本是不支持的。

```
import static java.lang.Math;
import static java.lang.System;
...
1.5 out.println(sqrt(PI));
1.4 System.out.println(Math.sqrt(Math.PI));
```

(5) 可变参数(Varargs)

可变参数使程序员可以声明一个接受可变数目参数的方法。注意，可变参数必须是函数声明中的最后一个参数。在 JDK1.5 之前，可以用重载来实现，但是这样就需要写很多的重载函数。

```
line1 public void write(Object... objs) {
line2 for (Object obj: objs)
line3 System.out.println(obj);
line4 }
```

10.面向对象编程中几种对象组合方式——is-a /has-a/use-a：

(1).is-a 组合：一个类继承具有相似功能的另一个类，根据需要在所继承的类基础上进行扩展。

优点：具有共同属性和方法的类可以将共享信息抽象到父类中，增强代码复用性，同时也是多态的基础。

缺点：子类中扩展的部分对父类不可见，另外如果共性比较少的时候使用继承会增加冗余代码。

(2).has-a 组合：has-a 组合是在一个类中引用另一个类作为其成员变量。

优点：可扩展性和灵活性高。在对象组合关系中应优先考虑 has-a 组合关系。

缺点：具有共性的类之间看不到派生关系。

(3).use-a 组合：是一个类中使用到了另外一个类，依赖关系

11.构造方法（构造器）特点?

- Java 中的构造器（构造方法）声明在类内部。
- 方法名与类名一致的方法叫构造方法
- 构造方法不能声明返回值类型。
- 构造方法可以包含参数，参数一般是创建对象实例必须依赖的条件（前提条件）。

子类默认调用父类的无参构造器，如果父类没有无参构造器，那么子类必需显示的去调用父类的有参构造器

如果一个类没有提供无参构造器，那么编译器将会自动提供一个无参构造器。

12.JAVA 中属性和方法的静态绑定和动态绑定?

静态绑定：Java 根据引用变量类型查找属性

动态绑定：java 根据实际的对象查找方法

13. JavaBean 规范?

1) 必须有包（package）
2) Java 类，具有无参数构造器
3) 有用 getXxx() 和 setXxx() 声明的 Bean 属性

- 如：getName() 和 setName(String n) 声明的 Bean 属性为：name，不是否有实例变量 name 无关
- boolean 类型的 get 方法可以有两种形式：getMarried() 戒者 isMarried()

4) 必须实现序列化接口（注：在学习 IO 的时候具体学习）

14.static 关键字的特点?

static 静态关键字修饰：属性、方法、内部类、代码块

static 修饰的资源属于类级别，是全体对象实例共享的资源

static 变量在类加载期间初始化

静态代码块是在类加载期间运行的代码块，由于类只加载一次，所以静态代码块只执行一次!

15.final 关键字的特点?

final 可以修饰类，方法，变量
final 修饰的类，不能再被继承
final 修饰的方法，不能覆盖 final 方法
final 修饰的变量
final 的局部变量，只能初始化不能改
final 的方法参数，不能改
final 的引用，引用指向不能改，但是对象的属性可以改

16.常见的 final 类有哪些?

Java 的 String 就是 final 类，不能被继承!
Math 是 final 类，不能被继承!
Integer 、Long、Character 等包装类是 final 类，不能被继承!

17.抽象类和接口的区别?

抽象类--不具体的类

1 抽象方法，只有行为的概念，没有具体的行为实现。
使用：abstract 关键字修饰，并且没有方法体。
2 包含抽象方法的类，就一定是抽象类。
使用： abstract 关键字修饰，包含抽象方法。
如：平面图形一定可以计算面积。

```
public abstract class CRMSystem{
    public abstract Client addClient(
            String name, String qq);
}
```

3 抽象方法和抽象类非常适合作为系统的分析和设计的工具。
4 抽象类不能直接创建实例。可以定义引用变量。
5 抽象类只能被继承，一个具体类继承一个抽象类，必须实现所有抽象方法。

接口

1. 接口：全部的方法都是抽象方法，全部的属性都是常量。
接口用来表示纯抽象概念，没有任何具体的方法和属性。
2.不能实例化，可以定义变量。
3.接口变量可以引用具体实现类的实例。
4.接口只能被实现，一个具体类实现接口，必须使用全部的抽象方法。
5.接口之间可以继承。
6.一个具体类可以实现多个接口，实现多继承现象，表示：一个概念即是 XXX 也是 XXX.
7.接口中的属性，默认是常量 public static final
8.接中的方法一定是:public abstract
9.实现一个接口，使用关键字 implements, 实现实际上是一种继承关系。接口和实现类是父子类型的关系

18.重载和重写的区别?

重载：方法名相同，参数不同（参数类型或者长度）

重载和修饰符和返回类型无关。

一是方法的参数列表必须改变，包括参数的类型，参数的个数多少，参数顺序。

二是重载对返回类型，访问修饰符，异常声明没有任何限制，可以作任意的修改。

实质上，重载只是创建了一个方法而已，特殊的地方在于方法的名字。

重写：两同两小一大（规则）

两同：方法名相同 参数类型 相同

两小：返回值类型（基本数据类型要一致，引用类型可以是其子类）

抛出的异常要小（也可以抛出父类型的异常的部分异常，或者不抛出异常）

一大：访问控制修饰符大

⑴ 重写方法必须和被重写方法具有相同的参数列表，返回类型必须和被重写方法的返回类型相同或者是返回类型的子类型。

⑵ 重写方法的访问控制修饰符不能比被重写方法更严格（比如一个在父类中声明为 public 的方法重写成一个 protected 的方法）。

⑶ 只有实例方法才能被重写，超类中的 final 方法不能被重写。

⑷ 重写方法不能抛出新的检查异常，或者是抛出比被重写方法声明的检查异常更广泛的检查异常。

19.==和 equals()的区别?

“==”,比较引用值和基本数据类型是否相等。

∪ xxx.equals()方法比较对象的内容是否相等。默认的比较规则是：比较引用

20. 为什么要同时覆盖 hashcode()和 equals()?

hashCode()方法要不 equals 方法一同覆盖（Sun 公司规定）

∪ 当两个对象 equals 比较为 true 时，应具有相同的 hashCode()值

∪ 当两个对象 equals 比较为 false 时，应具有不相同的 hashCode()值

∪ hashCode() 值要稳定（一致性），一个对象创建以后就不应该再变化

默认的 hashCode()值是当前堆对象地址转换的一个整数，这个整数不是内存地址!

在 java 的中，判断两个对象是否相等的规则是：

首先，判断两个对象的 hashCode 是否相等

如果不相等，认为两个对象也不相等

如果相等，则判断两个对象用 equals 运算是否相等

如果不相等，认为两个对象也不相等

如果相等，认为两个对象相等

21.String 类有哪些常用的方法?

charAt()

length()

trim()

toLowerCase()

toUpperCase()

indexOf()

lastIndexOf()

endsWith()

startsWith()

substring(int start, int end)

substring(int start)

toCharArray()

22.String,StringBuilder,StringBuffer 的区别?

String = char[] + 操作(复制创建新对象) char[]不可变

StringBuilder = char[] + 对 char[]操作(处理当前数组内容) char[]可变

1) StringBuilder 是变长字符序列

2) StringBuilder 方法：append，insert ... 都返回当前 StringBuilder 对象本身的引用

StringBuffer 和 StringBuilder API 几乎一样!

StringBuffer 是 java 早期提供的（JDK1.0），速度稍慢，线程安全

StringBuilder 是 Java5 以后提供的（JDK5.0），速度快，非线程安全

23.谈谈集合框架的理解?

集合框架包括集合不映射（Collection and Map）

List 元素有先后次序的集合, 元素有 index 位置, 元素可以重复, 继承自 Collection 接口,实现类: ArrayList, Vector, LinkedList

List 表示有先后次序的对象集合

ArrayList 是使用变长数组算法实现的，ArrayList 实现自 List

1) ArrayList 和 Vector 的比较

v Vector(1.0 版本提供的)，线程安全的，效率稍低，也是使用变长数组算法实现的，继承自 List 接口

v ArrayList，线程不安全的，效率高速度快（现在较常用）

2) ArrayList 和 LinkedList 的比较

v LinkedList 是采用双向循环链表实现的 List

v ArrayList 是采用变长数组算法实现的的 List

在 ArrayList 的中间插入或删除一个元素意味着这个列表中剩余的元素都会被移动；而在 LinkedList 的中间插入或删除一个元素的开销是固定的。

ArrayList 查询速度快，而 LinkedList 增删速度快

Set 元素无续, 不能重复添加, 是数学意义上的集合, 继承自 Collection 接口

v 实现类: HashSet(是一个只有 Key 的 HashMap), 使用 Set 时要重写 hashCode,equals 方法

HashMap 以键-值对（关键字：值）的形式存储对象，关键字 key 是唯一的、不重复的

1) key 可以是任何对象，Value 可以任何对象

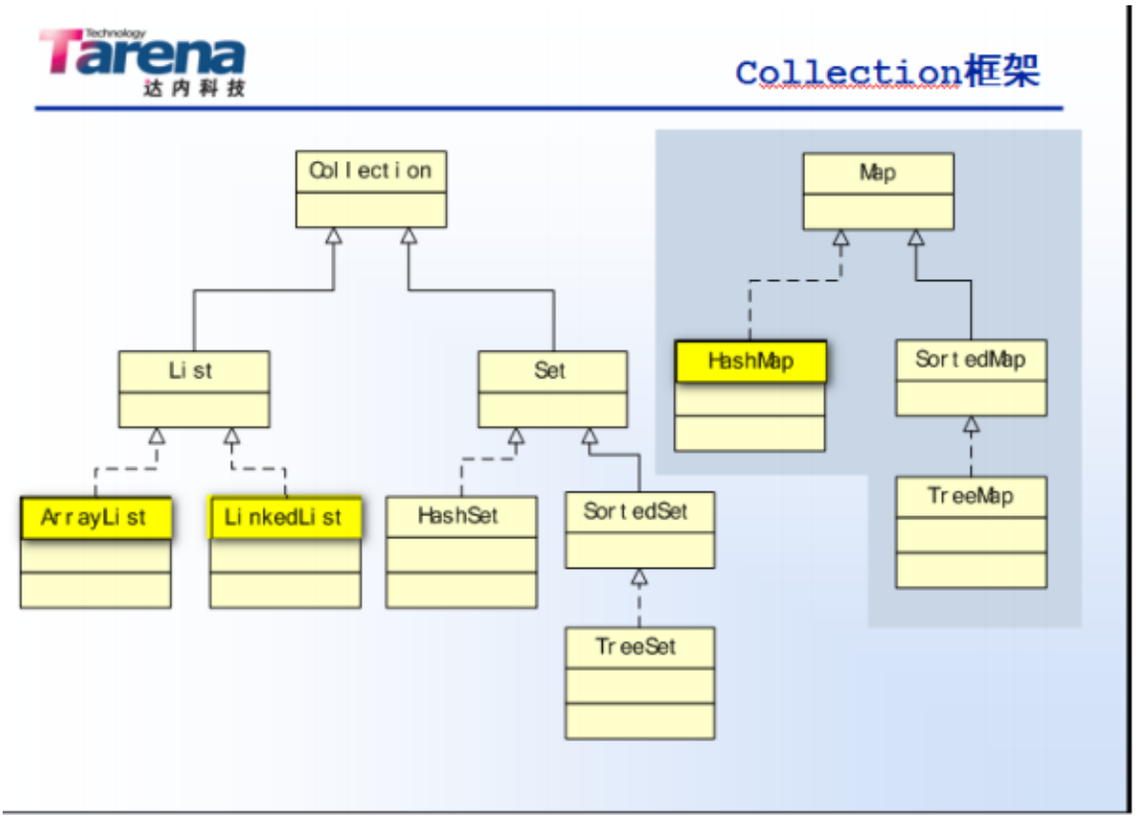
2) （key：value）成对放置在集合中

3) 重复的 key 算一个，重复添加是替换操作（会覆盖原来的元素）

4) HashMap 根据 key 检索查找 value 值

HashMap 新，非线程安全，不检查锁，快

Hashtable 旧 (JDK1.2 版本以前), 线程安全, 检查锁, 慢一点 (差的很小)



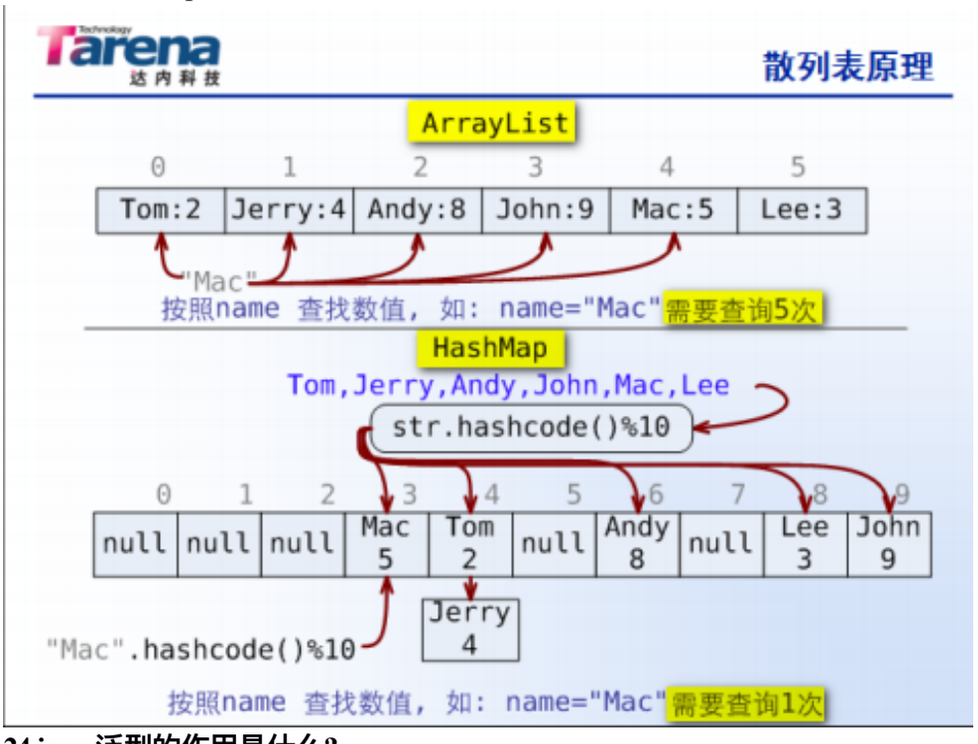
23.散列表的特点?

散列表中存放的对象是不连续的，所以称为“散列表”

散列表的优点：查找迅速

在 ArrayList 中查找 Mac，顺序查找，需要查找 5 次

在 HashMap 中（底层实现原理是散列表）查找 Mac，经过散列运算，仅需 1 次



24.java 泛型的作用是什么?

泛型是 Java5 以后提出的语法现象，作用是在编译期检查的类型约束（运行期不检查

泛型）,泛型可以用来约束类中元素的类型

25.Collection 和 Collections 的区别?

Collection 是集合接口，下面有子接口，List,Set

集合的工具类为 Collections，同数组的工具类 Arrays 相同，其中提供了许多的方法，诸如排序、二分查找、打乱、填充等操作。

26.内部类的分类? 各有什么特点?

1) 根据位置的不同，Java 中的内部类分为四种：

v 静态内部类

u 使用 static 修饰，声明在类体中

u 静态内部类中可以访问外部类的静态成员

v 成员内部类

u 声明在类体中，不使用 static，具有类的成员特征，也就是，必须有类的实例才能创建内部类实例

u 内部类实例可以访问共享外部类的成员变量（很常用）

u 如：链表的节点就可以定义为内部类

v 局部内部类 把类声明在方法中，就是局部内部类，作用域

u 类似局部变量（很少见）

v 匿名内部类

匿名类，非常常见，可以写在任何地方,就像一般的语句

语法更象是创建对象：Date d = new Date(){//...};

匿名类是对原类的一个继承，同时创建了实例，{} 就是继承以后的类体

类体中可使用所有类的语法

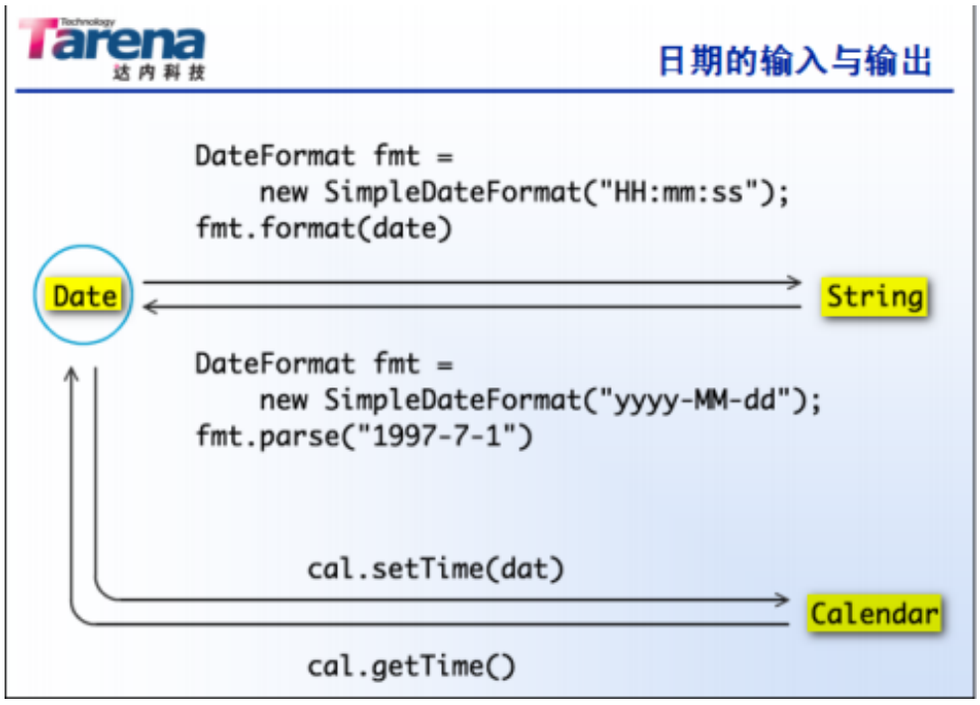
匿名类不能写构造器

匿名类可以从抽象类或者接口继承，必须提供抽象方法的实现

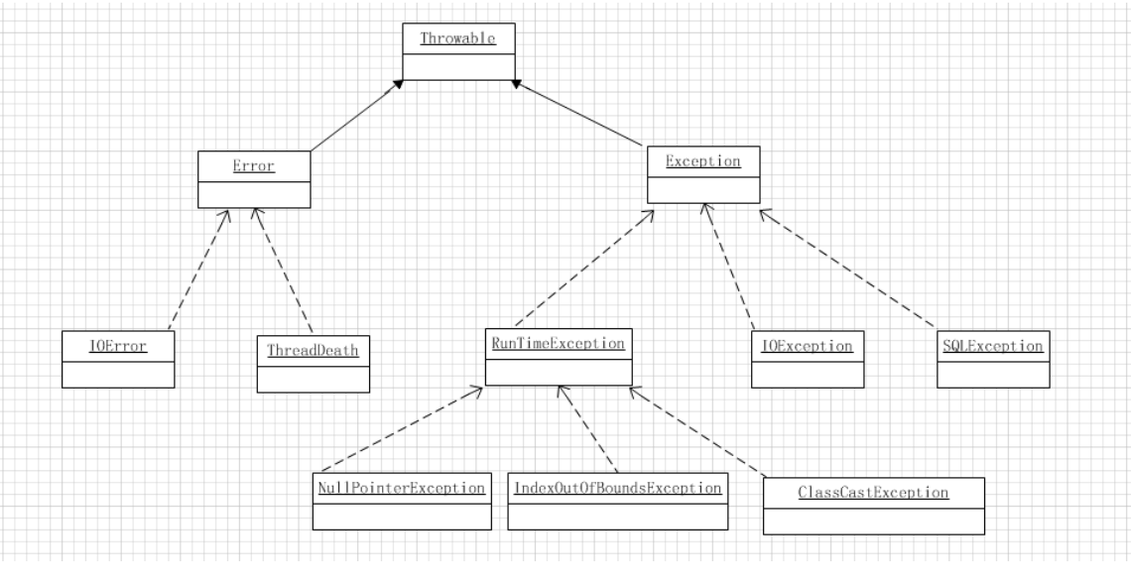
2) 任何内部类都编译成独立的 class 文件

3) 最大的作用：封装!

27.怎么将 Date 和 String 互相转换?



28. Java 中的异常理解?



异常是程序运行过程过程出现的错误, 在 Java 中用类来描述, 用对象来表示具体的异常。Java 将其区分为 Error 与 Exception，Error 是程序无力处理的错误，Exception 是程序可以处理的错误。

1)Error 与 Exception

Error 是程序无法处理的错误，比如 OutOfMemoryError、ThreadDeath 等。这些异常发生时，Java 虚拟机（JVM）一般会选择线程终止。Error:一般指虚拟机相关问题，如虚拟机崩溃，虚拟机出错等这种错误无法恢复或不可捕获，将导致应用程序中断。对于 Error 一般不编写针对性代码对齐进行处理。

Exception 是程序本身可以处理的异常，这种异常分两大类运行时异常和非运行时异常。程序中应当尽可能去处理这些异常。

2)运行时异常和非运行时异常

检查异常:当代码中抛出了一个检查异常，那么编译器在编译代码时会检查代码是否有处理该异常的代码片段，没有则编译不通过。

非检查异常:编译器不检查该类异常抛出是否有代码处理。

(ClassNotFoundException 不是 RuntimeException 的子类)

运行时异常都是 RuntimeException 类及其子类异常，如 NullPointerException、IndexOutOfBoundsException，ArrayIndexOutOfBoundsException, ClassCastException

等，这些异常是不检查异常，程序中可以选择捕获处理，也可以不处理。这些异常一般是由程序逻辑错误引起的，程序应该从逻辑角度尽可能避免这类异常的发生。

非运行时异常是 RuntimeException 以外的异常，类型上都属于 Exception 类及其子类。从程序语法角度讲是必须进行处理的异常，如果不处理，程序就不能编译通过。如 IOException、SQLException 等以及用户自定义的 Exception 异常，一般情况下不自定义检查异常。

自定义异常（自己定义异常表达错误)

MyException extends Exception　检测(check)异常

MyException extends RuntimException 运行时(runtime)异常

29.JAVA 中异常处理的方式有哪些?

1)try…catch…finally

```
try{
  //（尝试运行的）程序代码
}catch(异常类型 异常的变量名){
   //异常处理代码
}finally{
      //
}
```

注:子类异常的处理块必须在父类异常处理块的前面，否则会发生编译错误。

finally 块中一定会执行吗?

2)throws,throw

throw 关键字是用于方法体内部，用来抛出一个 Throwable 类型的异常。如果抛出了检查异常，则还应该在方法头部声明方法可能抛出的异常类型。

throws 关键字用于方法体外部的方法声明部分，用来声明方法可能会抛出某些异常。仅当抛出了检查异常，该方法的调用者才必须处理或者重新抛出该异常。

30.实现序列化的作用？（implements Serializable）

序列化的作用是，将数据分解成字节流，以便存储在文件中或在网络上传输。

31.IO 流的分类? 以及常用流的写法?

分为：字节流和字符流　或者　输入流和输出流

```
InputStream is = new FileInputStream("gbk.txt");
Reader in = new InputStreamReader(is);
   BufferedReader reader = new BufferedReader(in);

PrintWriter out = new PrintWtirer(
```

```
        new OutputStreamWriter(
                        new FileOutputStream(filename)));
```

32.创建线程的两种方式?

继承 Thread 类（extends Thread）或者实现 Runnable 接口（implements Runnable）

1) 继承 Thread 类

v 实现步骤：

⎫ 继承 Thread 类，覆盖 run()方法，提供并发运程的过程

⎫ 创建这个类的实例

⎫ 使用 start() 方法启动线程

2) 实现 Runnable 接口

v 实现步骤：

⎫ 实现 Runnable 接口，实现 run()方法，提供并发运程的过程

⎫ 创建这个类的实例，用这个实例作为 Thread 构造器参数，创建 Thread 类

⎫ 使用 start() 方法启动线程

33.线程的 5 中状态

1) New 新建状态

v 当程序使用 new 关键字创建了一个线程后，该线程就处于新建状态，此时线程还未启动，当线程对象调用 start()方法时，线程启动，进入 Runnable 状态

2) Runnable 可运行（就绪）状态

v 当线程处于 Runnable 状态时，表示线程准备就绪，等待获取 CPU

3) Running 运行（正在运行）状态

v 假如该线程获取了 CPU，则进入 Running 状态，开始执行线程体，即 run()方法中的内容

v 注意：

如果系统另有 1 个 CPU，那么在伕意时间点则另有 1 条线程处于 Running 状态；

如果是双核系统，那么同一时间点会有 2 条线程处于 Running 状态

但是，当线程数大于处理器数时，依然会是多条线程在同一个 CPU 上轮换执行

当一条线程开始运行时，如果它不是一瞬间完成，那么它不可能一直处于 Running 状态，线程在执行过程中会被中断，目的是让其它线程获得执行的机会，像这样线程调度的策略取决于底层平台。对于抢占式策略的平台而言，系统系统会给每个可执行的线程一小段时间来处理伕务，当该时间段（时间片）用完，系统会剥夺该线程所占资源（CPU），让其他线程获得运行机会。

v 调用 yield()方法，可以使线程由 Running 状态进入 Runnable 状态

4) Block 阻塞（挂起）状态

v 当如下情冴下，线程会进入阻塞状态：

⎫ 线程调用了 sleep()方法主动放弃所占 CPU 资源

⎫ 线程调用了一个阻塞式 IO 方法（比如控制台输入方法），在该方法返回前，该线程被阻塞

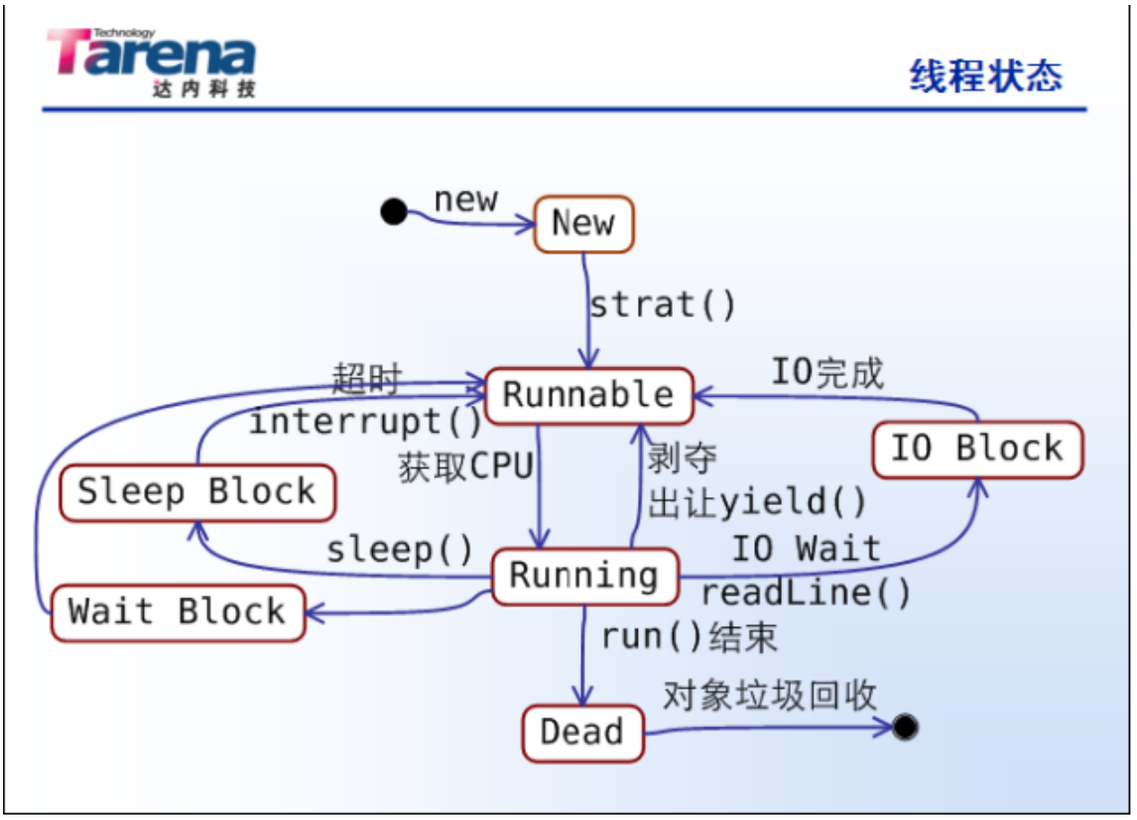
⎫ .....

v 当正在执行的线程被阻塞时，其它线程就获得执行机会了。需要注意的是，当阻塞结束时，该线程将进入 Runnable 状态，而非直接进入 Running 状态

5) Dead 死亡状态

v 当线程的 run()方法执行结束，线程进入 Dead 状态

v 需要注意的是，不要试图对一个已经死亡的线程调用 start()方法，线程死亡后将不能再次作为线程执行，系统会抛出 IllegalThreadStateException 异常



注：

1) new 运算创建线程后，线程进入 New 状态（初始状态）

2) 调用 start()方法后，线程从 New 状态进入 Runnable 状态（就绪状态）

v start()方法是在 main()方法（Running 状态）中调用的

3) 线程结束后，进入 Dead 状态（死亡状态），被对象垃圾回收

4) main()方法结束后，其它线程，比如上例中 p1 和 p2 开始抢着进入 Running 状态

v 由谁抢到是底层操作系统决定（操作系统分配时间片）

v 单核处理器：在一个时间点上另有一个线程在 Running 状态；双核处理器：2 个

v 如果 p1 进入 Running 状态，当操作系统分配给它的时间片到期时，p1 进入 Runnable 状态，p2 进入 Running 状态

v 在期间有可能其它的进程的线程获得时间片，那么 p1 和 p2 同时进入 Runnable 状态，等待操作系统分配时间片

5) 线程进入 Dead 状态后，另能被垃圾回收，不能再开始

6) 如果线程在运行过程中，自己调用了 yield()方法，则主动由 Running 状态进入 Runnable 状态

34. 异步与同步的区别?

1) 异步

并发，各干自己的。如：一群人上卡车

2) 同步

步调一致的处理。 如: 一群人上公交车

35.数据库中，char 和 varchar 类型的区别?

char(n) 表示定长字符串(方便查询) 最长放入 n 个字符 ，放入的数据如果不够 n 个字符则补空格 ，无论如何都占 n 个字符长度。

varchar(n) 表示变长字符串(节省空间) 最长放入 n 个字符 ，放入的数据是几个长度就占多大空间。

```
select decode(a2,'A','AAAA','BBBB') FROM table ;
select decode(a1,'A ','AAAA','BBBB') FROM TT ; --此处是两个空格
```

36.在数据库中怎么做表的复制? (一条语句完成)?

create table emp_xxx as select * from emp_yyy ;

(注:如何将一个表中的数据复制到另外一个表中?

insert into table(select * from table2);两个表结构一样

insert into table(name,age) (select name,age from table2);复制指定的字段

)

37.分别简述一下 DDL,DML,DQL,TCL,DCL?

1)数据定义语言 DDL(Data Definition Language) ，是 SQL 语言集中负责数据结构定义不数据库对象定义的语言 ，主要有 create、alter、drop 和 truncate 四种常用语句。

DDL 对数据结构起作用。

- create 数据库对象的创建
- alter 修改数据库对象
- drop 删除数据库对象
- truncate 清空表数据

2) 数据操纵语言 DML(Data Manipulation Language) ，用户通过它可以实现对数据表的基本操作, 即对表中数据的增、删、改。

DML 对数据起作用。

- insert 插入操作
- update 更新操作
- delete 删除操作

3) 数据查询语言 DQL(Data Query Language) ，用户主要通过它实现对数据的查询操作。

- select 查询操作

4) TCL 事务控制语句是用来对 DML 操作进行确认的。

commit 提交数据

rollback 数据回滚

savepoint 保存点

5) 数据控制语言（Data Control Language，DCL）用于对用户授权或撤销其权限，也可使用角色实现对用户的批量授权或撤销权限，在应用开发层面较少用到。

grant(授予权限)/revoke(回收权限)

38.Oracle 常用的单行函数有哪些?

round(数字 ，小数点后的位数)用于数字的四舍五入

trunc(数字 , 小数点后的位数)用于截取，如果没有第二个参数 ，默认是 0

to_date()和 to_char()是时间处理的函数

- to_date 将字符串数据 按指定格式 转换为 日期数据
- to_char 将日期数据 按指定格式 转换为 字符串数据

coalesce(参数列表)函数的作用：返回参数列表中第一个非空参数 ，参数列表中最后一个值通常为常量

decode()函数是 Oracle 中等价于 case when 语句的函数 ，作用同 case 语句相同。

decode 函数语法如下：

decode(判断条件 ，匹配 1，值 1，匹配 2，值 2，… ，默认值)

nvl(bonus,0) 空值转换函数

39.常用的组函数有哪些?

AVG 求平均数

COUNT 求数量

MAX 求最大值

MIN 求最小值

SUM 求和

注意：

avg/sum 操作数字

max/min 可以操作各种数据类型

组函数默认忽略空值

40.判断语句是否正确?

select ename,count(*) from emp where ename='KING' group by ename;

select count(*),sum(sal) from emp group by ename;

在 SELECT 列表中所有未包含在组函数中的列都应该包含在 GROUP BY 子句中。

包含在 GROUP BY 子句中的列不必包含在 SELECT 列表中。

41. 研发部有哪些职位?

```
select distinct job from emp_xxx
      where deptno = ( select deptno
                              from dept_xxx
                              where dname = 'developer' ) ;
```

42.ALL,Any,In 的用法?

ALL --大于最大值

Any--大于最小值

子查询的条件是单列还是多列没关系 ，关键是要分清返回的是单行还是多行。

如果是单行 ，用单行比较运算符 , = , >, < 这些

如果是多行 ，用 in, >all, >any, <all, <any 这些

43. 哪个部门的平均薪水比部门 20 的平均薪水高?

```
select deptno , avg( nvl(salary , 0) ) avg_s
      from emp_xxx
      group by deptno
      having avg(nvl(salary,0))   > ( select avg(nvl(salary,0))
                                                from emp_xxx
                                               where deptno = 20 ) ;
```

44.什么叫关联子查询?

子查询中不再是独立的 Sql 语句 ，需要依赖主查询传来的参数 ，这种方式叫关联子查询

哪些员工的薪水比本部门的平均薪水低？不再和整个部门的平均薪水比较。

```
select ename, salary, deptno
        from emp_xxx a
        where salary < ( select avg(nvl(salary,0))
                                from emp_xxx
                                where deptno = a.deptno ) ;

        --子查询不再是独立的 Sql 语句 ，需要依赖主查询传来的参数 a.deptno
```

45. 哪些人不是别人的经理?

```
select ename from emp_xxx a
        where not exists (select 1 from emp_xxx
                                where mgr = a.empno) ;
```

46. union 和 union all 的区别?

union 去掉重复记录 , union all 不去重

v union 排序 , union all 不排序

（当列的个数、列的顺序、列的数据类型一致时,我们称这两个结果集结构相同

） 只有结构相同的结果集才能做集合操作）

47.Oracle 中表连接的方式有哪些?

内连接(自然连接)

等值连接、自然连接和不等值连接

{内连接是 join 关键字连接两个表，语法为 table1 join table2 on 条件。

根据使用的比较方式不同，内连接又分为等值连接、自然连接和不等值连接。

等值连接：所谓等值连接，是指在连接条件中使用等于（=）运算符比较被连接的值，也就是通过相等的列值连接起来的查询。

例 子 ： Select empno,ename,sal,emp.deptno,dname from emp,dept where emp.deptno=dept.deptno;

非等值连接： 所谓不等连接，就是在连接条件中使用除等号（=）外的其他比较运算符，构成非等值连接查询。可以使用的比较运算符包括：>（大于）、<（小于）、>=（大于等于）、<=（小于等于）、<>（不等于）、!=（不等于）、LIKE、IN 和 BETWEEN 等。

例子：select e.ename,e.sal,s.grade from emp e join salgrade s on e.sal between s.losal and s.hisal; 查询所有员工的薪水等级

自然连接：自然连接是在两张表中寻找那些数据类型和列名都相同的字段，然后自动地将他们连接起来，并返回所有符合条件的结果。

例子：select * from emp natural join dept;

}

外连接

外连接的结果集 = 内连接的结果集 + 驱动表在匹配表中找不到匹配记录的数据和空值

{

使用一张表中的所有记录去和另一张表中的记录按条件匹配(空值也会匹配)，这个

表中的所有记录都会显示。

左外连接，右外连接，全外连接

LEFT/RIGHT/FULL OUTER JOIN

}

⑴ 左外连接 (左边的表不加限制)

{

【例】t1 left outer join t2 －－其中 t1 是驱动表，t2 是匹配表

等价于：t2 right outer join t1

【例】查询每个员工的经理的名字?

select worker.ename,manager.ename from emp worker left outer join emp manager on worker.mgr=manager.empno;

}

⑵ 右外连接(右边的表不加限制)

{

t1 right outer join t2 －－t2 是驱动表，t1 是匹配表

等价于: t2 left outer join t1

【例】－哪些员工没有下属（不是别人的领导）？外连接＋匹配表 PK is null 表示否定问题，不是，不包括，等等。

select manager.ename from emp worker

right outer join emp manager on worker.mgr = manager.empno

（首先找到所有经理下面的员工是哪些）

where worker.empno is null;

(然后将员工为空的过滤出来)

}

⑶ 全外连接(左右两表都不加限制)full outer join

{

左表和右表都不做限制，所有的记录都显示，两表不足的地方用 null 填充。

【例】

select e.ename, d.loc from emp e full outer join dept d on (e.deptno = d.deptno);

}

自连接（同一张表内的连接）自连接是等值连接的一种。表中的列外键关联自己表的主键列。

{

自连接(self join)是 SQL 语句中经常要用的连接方式，使用自连接可以将自身表的一个镜像当作另一个表来对待，从而能够得到一些特殊的数据。

【例】查询每个员工的经理的名字?

select worker.ename,manager.ename from emp worker join emp manager on worker.mgr=manager.empno;

}

48.什么是事务（Transaction）?什么是事务控制?

事务（Transaction）是指组成单个逻辑工作单元的一系列操作

事务控制（Transaction Control）则是指通过将一组相关操作组合为一个要么全部成功、要么全部失败的逻辑工作单元（即事务），以简化错误恢复、提高应用程序的可靠性。

49. 约束条件(constraint)有哪些?

主键(Primary key, 简称 PK)

1) 主键约束(primary key) = 不能重复 + 不能为 null
2) 主键约束可以用两种方式定义：列级约束和表级约束

非空约束(not null , 简称 NN)

注意：非空约束只能定义在列级

唯一约束(Unique , 简称 UK)

可以定义在表级和列级

检查约束(Check , 简称 CK)

可以定义在表级和列级

因为约束条件建立在列级时可读性不好 ，而丐不方便定义约束条件名字 ，一般建议定义在表级。

外键(Foreign key, 简称 FK)

之前讲的几个约束条件都是用来约束单个表中的列 ，而外键约束定义在两个表的两个字段上(或者一个表的两个字段上) , 用于保证相关两个字段的关系

constraint fk_customerid__cardinfo2 foreign key(customerID) references userInfo(customerID)

注意：一张表中，只能有一个 PK,但是可以有多个 FK

50. 视图(View), 索引 (Index), 序列(Sequence)?

视图(View)：

视图的使用和表相同

视图的好处：简化查询 ;屏蔽数据库表结构、限制对数据的访问

视图不包含任何数据。是基表数据的投影。

视图本身并不在物理上保存数据, 在视图上进行的查询或更新操作实际上都是针对其基表来完成的。

当基表变化时，视图也随着变化。

索引 (Index)：

index ：索引（Index）一种用于提升查询效率的数据库对象，使用索引可以快速定位数据、减少磁盘 I/O 操作次数。,注意：对于数据变更频繁(DML 操作频繁)的表 ，索引会影响查询性能。

如果数据表有 PK/Unique 两种约束 ，索引自动创建 ，除此以外 ，索引必须手动创建

自定义索引语法： create index 索引名 on 表名(列名) ;

为什么索引查询快?

(Oracle server 通过 rowid 快速定位要找的行

通过 rowid 定位数据能有效的降低读取数据块的数量

索引的使用和维护是自动的，一般情况下不需要用户干预)

序列(Sequence)：

序列的特性：产生连续的不同的数字值用来作为数据表的主键。序列这种对象在 Oracle、db2 等数据库中有 ，在 mysql、sql server 中没有。(在 mysql 通过 auto_increment 自动增长列来实现同样的功能)

51. TRUNCATE 与前述 DELETE 操作的区别?

DELETE 为 DML 操作，可以回滚，而 TRUNCATE 为 DDL 操作，一经执行不可撤销，故其效率要高一些；DELETE 操作可以选择删除表中全部或部分数据，而 TRUNCATE 操作只能删除表中全部数据。

如果不再用到表中数据、但又需要保留表的结构，则可该使用 TRUNCATE TABLE 操作；如果连表的结构也需要了，则可使用 DROP TABLE 操作将表彻底删除。

52. 第 n 条到第 n 条记录的获取方式?

Oracle 获取方式:

```
select ename,sal,rn from (select ename,sal,rownum rn from emp) where rn>=5 and rn<=10;
select ename,sal,rn from (select ename,sal,rownum rn from emp) where rn between 5 and 10;
select id,name,jop,rn from(select id,name,jop,rownum rn from dept10 where rownum<10) where rn>5;
select t2.* from (select t1.*,rownum rn from emp t1 where rownum<=5) t2 where rn>3;
```

MySql 获取方式:

```
SELECT * FROM table LIMIT 5,10; // 检索记录行 6-15
```

53.JDBC 编程步骤?

1)注册 Driver

```
Oracle: Class.forName("oracle.jdbc.driver.OracleDriver");
Mysql: Class.forName("com.mysql.jdbc.Driver");
```

2)建立连接

```
Mysql:    String url = "jdbc:mysql://localhost:3306/tarena";
Oracle:   String url = "jdbc:oracle:thin:@localhost:1521:tarena";
          String name="root";
          String pwd ="root";
Connection conn = DriverManager.getConnection(url,name,pwd);
```

3)获得一个 Statement 对象(两种方式)

```
Statement sta = conn.createStatement();
PreparedStatement pstm = conn.prepareStatement();
```

4) 通过 Statement 执行 Sql 语句

```
ResultSet rs = sta.executeQuery(String sql);返回一个查询结果集。用于 select 语句
int I = sta.executeUpdate(String sql);返回值为 int 型，表示影响记录的条数。用于 insert,update,delete 语句。
```

5) 处理结果集

```
while(rs.next()){
System.out.println(rs.getString("name"));
```

//------------------

}

6) 关闭数据库连接（释放资源） 调用.close()

rs.close(); sta.close(); con.close();

ResultSet Statement Connection 是依次依赖的。

54. Statement 和 PreparedStatement 区别?

1） PreparedStatement 代码的可读性和可维护性

2) PreparedStatement 尽最大可能提高性能.

数据库会对 PreparedStatement 语句进行预编译，下次执行相同的 sql 语句时，数据库端不会再进行预编译了，而直接用数据库的缓冲区，提高数据访问的效率。

3）最重要的一点是极大地提高了安全性.

如果是 Statement 构建的语句：select * from tb_name = '随意' and passwd = '' or '1' = '1'; 很容易造成 Sql 注入。而如果你使用预编译语句.你传入的任何内容就不会和原来的语句发生任何匹配的关系。

55.xml 和 html 的区别?

XML 被设计用来传输和存储数据。

HTML 被设计用来 XML 不是 HTML 的替代。

XML 和 HTML 为不同的目的而设计：

XML 被设计为传输和存储数据，其焦点是数据的内容。

HTML 被设计用来显示数据，其焦点是数据的外观。

HTML 旨在显示信息，而 XML 旨在传输信息。显示数据。

1) 超文本标记语言 HTML（Hyper Text Markup Language

- 写法格式： link
- 关注数据的展示不用户体验
- 标记是固定的，不可扩展（如 <a>表示超链接）

2) 可扩展的标记语言 XML（eXtensible Markup Language

- 写法格式： 同 html 样式 <a>link
- 仅关注数据本身
- 标记可扩展，可自定义

56.XML 的解析方式?

1)DOM（Document Object Model 文档对象模型）

- 关键字：树(Document)
- 优点： 把 xml 文件在内存中构造树形结构，可以遍历和修改节点
- 缺点： 如果文件比较大，内存有压力，解析的时间会比较长

2)SAX（Simple API for Xml 基于 XML 的简单 API）

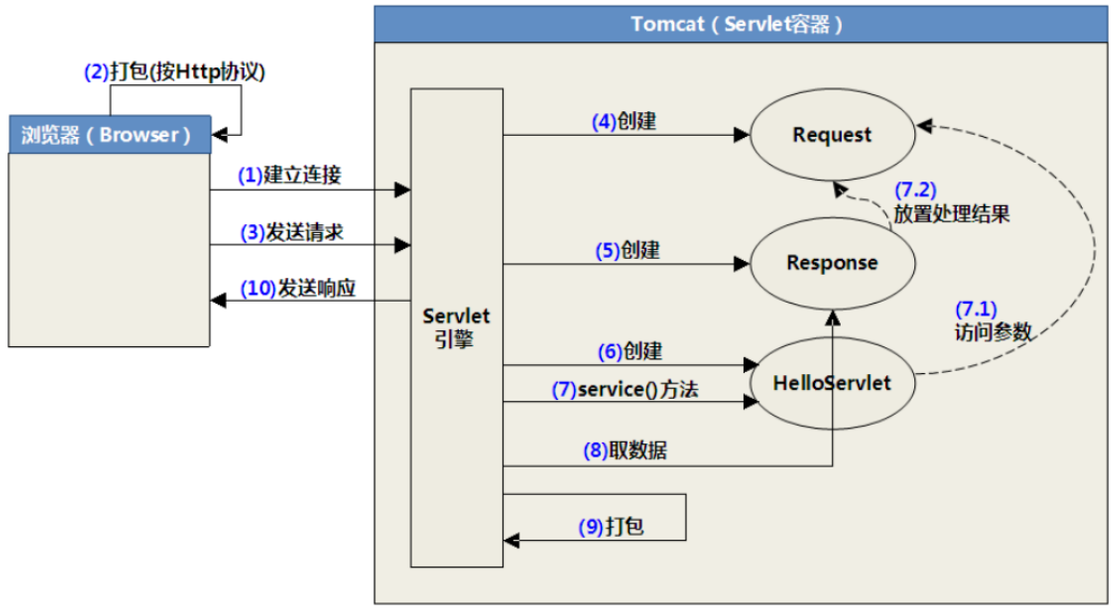
- 关键字：流(Stream)
- 把 xml 文件作为输入流，触发标记开始，内容开始，标记结束等动作
- 优点： 解析可以立即开始，速度快，没有内存压力
- 缺点： 不能对节点做修改

57. servlet 是如何运行的?

当用户向浏览器地址栏输入 http://ip:port/helloweb/sayHello?name=zs

1) 浏览器使用 ip：port（端口号）连接服务器

2) 浏览器将请求数据按照 http 协议打成一个数据包（请求数据包）发送给服务器
请求数据包的内容包含了请求资源路径(/helloweb/sayHello?name=zs)，
另外，在请求数据包当中，还会包含浏览器自动生成的一些信息。
3) 服务器创建两个对象：请求对象（Request）和响应对象（Response）
服务器解析请求数据包，将解析之后的数据存放到请求对象里面，方便 servlet 读取请求 数据(因为 servlet 不用解析请求数据包，如果要解析，需要理解 http 协议)。
请求对象是 HttpServletRequest 接口的一个实现。
响应对象是 HttpServletResponse 接口的一个实现，响应对象由于存放 servlet 处理的结果。
4) 依据请求资源路径找到相应的 servlet 配置，通过反射创建 servlet 实例。然后调用其 service()
方法。
在调用 service()方法时，会将事先创建好的请求对象(request)和响应对象(response)作为参数进行传递。在 servlet 内部，可以通过 request 获得请求数据，戒者通过 response 设置响应数据。
5) 服务器从 response 中获取数据，按照 http 协议打成一个数据包(响应数据包),发送给浏览器。
6) 浏览器会解析响应数据包，取出相应的数据，生成相应的界面。



58.get 和 post 请求的区别?

1)哪一些是 get 请求方式

a, 直接在浏览器地址栏输入某个地址。

b,点击链接地址

c, 表单默认的提交方式

2)get 请求方式的特点

a,请求参数会添加到请求资源路径后面。

请求资源路径后面添加的参数数据量大小是有限制的，

b,会将请求参数直接显示在浏览器地址栏，不安全。

c, get 方式只适合于向服务器请求资源或者是

向服务器提交少量的数据。

3)哪一些是 post 请求方式?

a,设置表单的 method="post"

4)post 方式的特点

a, post 方式会将请求参数及值添加到实体内容里面，可以放置大量的数据

b,因为不会将参数直接显示在浏览器地址栏，所以，相对安全。

get 方式会将请求参数及参数值放在请求资源路径里面，携带的数据大小有限制，不适合提交大量的数据； post 方式会将请求参数及参数值放在实体内容里面，理论上没有限制，适合大量数据的提交。

59.重定向和转发的区别?

重定向：

1)什么是重定向?

服务器向浏览器发送一个状态码 302 及一个消息头 location(location 的值是一个地址)，浏览器会立即向 location 所指定的地址发送一个新的请求。我们把这样一种机制叫重定向。

2)编程：

response.sendRedirect(String url);

3)需要注意的问题

在重定向之前，不能够有任何的输出；如果 response 缓存当中有数据，在重定向之前，会自动清空。

4)重定向的特点：

a,地址任意

b,浏览器地址栏地址会变化（即变化为跳转之后的地址）。

转发

1）什么是转发?

一个 web 组件（servlet/jsp）将未完成的处理交给另外一个 web 组件继续完成。转发所涉及的各个 web 组件可以共享 request 和 response 对象。

2）编程：

step1 绑订数据到 request 对象上。

request.setAttribute(String name,Object obj);

request.removeAttribute(String name);

Object request.getAttribute(String name);

//如果绑订名不存在，则返回 null。

step2 获得转发器

RequestDispatcher rd = request.getRequestDispatcher(String url);

step3 转发

rd.forward(request,response);

servlet:负责业务逻辑处理（包括数据访问） 。

jsp：负责生成界面。

3）需要注意的问题：

在转发之前，response 缓存的数据会被清空。

主要区别如下：

1) 地址

⎫ 转发的地址必须是同一个应用内部的某个组件（不能跨应用，不能跨服务器）

比如：

地址 1 http://localhost:8080/web06/aaa.jsp

地址 2 http://localhost:8080/web06/bbb.jsp

地址 3 http://localhost:8080/web07/ccc.jsp

地址 4 http://www.tarena.com.cn

在应用 web06 内部的组件 aaa.jsp 可以将信息转发到地址 2（同一应用），但是不可以转发到地址 3（跨应用）和地址 4（跨服务器）

⎫ 重定向的地址没有限制

2) 能否共享 request

⎫ 转发可以

⎫ 重定向不行

原因是转发是一次请求，重定向为两次请求，Request 的生命周期另能在一次请求内，请求结束，Request 被删除

3) 浏览器地址栏的地址是否变化

⎫ 转发不变

⎫ 重定向会变

4) 事件是否处理完毕

⎫ 转发是一件事未做完

⎫ 重定向是一件事已经做完

60.get 请求和 post 请求中文乱码问题的处理?

1)、Get 请求：

Tomcat 这个容器会用ＩＳＯ－８８５９－１这种编码去解析 URL 里面的值;

1. 解决方法：new String(request.getParameter("username").getBytes("ISO-8859-1"),"UTF-8");

2.直接改 Tomcat 配置：conf/server.xml

```
<Connector port="8080" protocol="HTTP/1.1"
           connectionTimeout="20000"
           redirectPort="8443" URIEncoding="utf-8"/>
```

2)、Post 请求：

在 html 文件中，添加

```
<meta http-equiv="content-type" content="text/html;charset=utf-8">
```

另外，表单的提交方式必须是 post。

```
request.setCharacterEncoding("utf-8");
```

get 和 post 请求的中文乱码问题处理?

当表单提交时，浏览器会对表单中的中文参数值进行编码(会使用打开表单所在的页面时的编码格式来进行编码)。服务器在默认情况下，会使用 iso-8859-1 这种编码格式进行解码。所以会出现乱码。

1)get 请求方式的处理:

第一种方式:

要确保表单所在的页面，使用指定的编码格式打开。对于 html 文件，可以添加:<meta http-equiv="content-type" content="text/html;charset=utf-8">,确保表单中的编码为 utf8,能够存储中文。

在 Servlet 中使用 name = new String(name.getBytes("iso-8859-1"),"utf-8");来处理编码

第二种方式:

找到 tomcat---->conf----->server.xml----><Connector port="8080" protocol="HTTP/1.1" connectionTimeout="20000" redirectPort="8443" URIEncoding="UTF-8"/> 后面加上 URIEncoding="UTF-8"

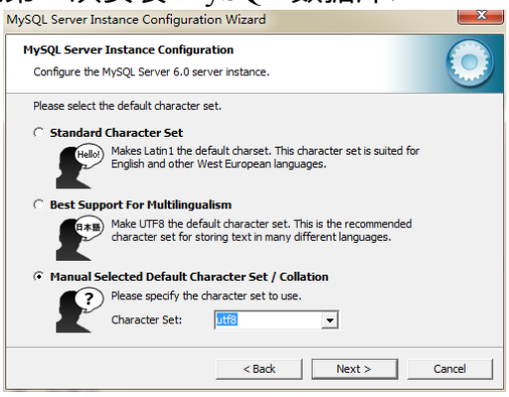
2)post 请求方式的处理:

2.1 首先确保 html 的编码为 utf-8

2.1 在服务器端，取数据之前,添加 request.setCharacterEncoding("utf-8");

MySQL 的乱码解决方案:

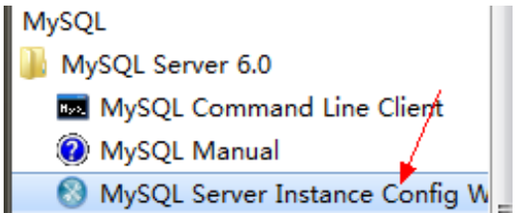
1. 如果第一次安装 MySQL 数据库:



在选择编码时，选择最后一项，并指定编码为 utf8 或者 gbk（这两种编码支持中文,是我们比较常用的）

2. 如果已经按照完成了 MySQL 数据库，那么可以进行重新配置修改:

在开始菜单中找到:



然后重复第一个步骤即可。(注意：这样修改完成后，那么数据库中原来保存的数据如果有中文，就会乱码，因为当前的编码已经变为了 utf8，而原来保存的可能是其它编码。)

3. 重启 MySql 的服务.右键管理员身份打开命令提示符窗口,

```
C:\Windows\system32>net stop MySQL
MySQL 服务正在停止..
MySQL 服务已成功停止。


C:\Windows\system32>net start MySQL
MySQL 服务正在启动 .
MySQL 服务已经启动成功。
```

4. 进入 MySQL 中，如果通过一下命令查看编码如下图所示，那么表示配置正确:

```
mysql> show variables like 'char%';
+--------------------------+----------------------------------------------------
-----------+
| Variable_name            | Value
          |
+--------------------------+----------------------------------------------------
-----------+
| character_set_client     | utf8
          |
| character_set_connection | utf8
          |
| character_set_database   | utf8
          |
| character_set_filesystem | binary
          |
| character_set_results    | utf8
          |
| character_set_server     | utf8
          |
| character_set_system     | utf8
          |
| character_sets_dir       | D:\Program Files (x86)\MySQL\MySQL Server 6.0\share
\charsets\ |
+--------------------------+----------------------------------------------------
-----------+
```

但是这个时候如果向表中插入数据，会报编码错误，如下图:

```
mysql> insert into users values(3,'张三','123123',1,'1998-12-11','2012-04-21',0,
'乱码啊');
ERROR 1366 (HY000): Incorrect string value: '\xD5\xC5\xC8\xFD' for column 'usern
ame' at row 1
```

这个时候，我们就可以使用 set names gbk;命令来设定字符集为 gbk。就可以正常插入数据了。(注意：这个只是让显示的时候正确，并不会改变表中的数据的编码。此命令只在当前会话中有效，也就是如果新开一个窗口打开 MySql，用 select 查询时会发现还是乱码，这个时候如果要正确显示，那么还得用此语句。)

Servlet 中的数据插入 MySql 乱码:（我们都用 UTF8 这样的编码来存储）

1. 首先在 Servlet 这个 java 文件得是 utf8 编码，在其中打印要插入 MySql 中的数据，查看是否能够正确显示，如果能够正确显示那么说明 Servlet 中的数据就是 utf8 编码。
2. 如果上一步不能通过检查，那么插入数据就会出现问题，直到检查第一步完成为止。
3. 建数据库之前首先保证 MySql 按照上面的安装正确:
4. 创建数据库: create database wuxi default character set 'utf8';
5. 可以使用 show create database wuxi;来查看这个数据库的 SQL 语句.
6. 建表时加上: create table t_pic(id bigint primary key auto_increment,picName varchar(100), userId bigint) ENGINE=InnoDB DEFAULT CHARSET=utf8;
7. JDBC 连接数据库时: conn = DriverManager.getConnection(

"jdbc:mysql://localhost:3306/jd1105db?useUnicode=true&characterEncoding=utf8","root","1234");

1.乱码的解决方案:

1.1、html/jsp

1.1.1、html 中：模拟一个消息头(content-type),告诉浏览器正在处理的数据的类型和字符集

<meta http-equiv="content-type" content="text/html;charset=utf-8">,这个编码应该和 html 文件保存的编码一致(

如果不写这句话，可能浏览器在读取 html 时使用的编码有些步一致,而且同样会导致提交数据时的编码不一致)

1.1.2 、 jsp 中 ： <%@ page language="java" contentType="text/html; charset=utf-8" pageEncoding=”utf-8”%>

contentType 属性:设置 response.setContentType 方法的参数值。

pageEncoding 属性：告诉容器 jsp 文件的字符集，容器会按照该属性指定的字符集去解码(容器需要读取 jsp 文件的内容，有些容器默认情况下会按照 iso-8859-1 去解码)。

jsp 中不需要加<meta charset=”utf-8” />因为 contentType 已经解决了告诉浏览器的编码问题

1.2、从 html 提交数据到 Servlet

从 html/jsp 中的 form 表单将数据提交到 Servlet,使用的编码是浏览器打开此页面的编码也就是和 1.1 中的设置有关系

1.3、Servlet

1.2.1、获取数据:分两种，get 和 post 请求的处理方式不同

GET 请求方式：

1.2.1.1、 TOMCAT 默认 ISO-8859-1 来处理 get 请求，因此可以设置默认编码为 UTF-8 解决，在 tomcat 安装的 conf\server.xml 文件中设置如下

<Connector port="8080" protocol="HTTP/1.1"
connectionTimeout="20000"
redirectPort="8443" URIEncoding=”UTF-8”/>

1.2.1.2、在所写到的 Servlet 中直接进行处理

String username = request.getParameter(“username”);
username = new String(username.getBytes("ISO-8859-1“), "UTF-8");

POST 请求方式：

必须在获取参数之前，调用如下方法先解码
request.setCharacterEncoding(“UTF-8”);
String username = request.getParameter(“username”);

1.2.2、输出数据:response.setContentType("text/html;charset=utf-8");指定 HTTP 响

应的编码,同时指定了浏览器显示的编码.

1.4、连接数据库时的编码:

"jdbc:mysql://localhost:3306/jsd1402db?useUnicode=true&characterEncoding=utf8","root","1234")
;

1.5、DB 数据库的编码:

1.5.1、修改整个数据库保存数据的编码: MySQL 的乱码解决方案

1.5.2、创建数据库时设置的编码:

1. 创建数据库: create database wuxi default character set 'utf8';
2. 可以使用 show create database wuxi;来查看这个数据库的 SQL 语句
3. create table t_pic(id bigint primary key auto_increment,picName varchar(100), userId bigint) ENGINE=InnoDB DEFAULT CHARSET=utf8;

61. servlet 的生命周期?

所谓生命周期，指的是 servlet 容器如何创建 servlet 实例、分配其资源、调用其方法、并销毁其实
例的整个过程。

阶段一: 实例化（就是创建 servlet 对象,调用构造器）

在如下两种情况下会迆行对象实例化。

第一种情况:

当请求到达容器时，容器查找该 servlet 对象是否存在，如果不存在，才会创建实例。

第二种情况:

容器在启动时，或者新部署了某个应用时，会检查 web.xml 当中，servlet 是否有 load-on-starup 配置。如果有，则会创建该 servlet 实例。 load-on-starup 参数值越小，优先级越高（最小值为 0，优先级最高）。

阶段二: 初始化

为 servlet 分配资源, 调用 init(ServletConfig config);方法 config 对象可以用来访问 servlet 的初始化参数。

初始化参数是使用 init-param 配置的参数。

init 可以 override。

阶段三: 就绪/调用

有请求到达容器，容器调用 servlet 对象的 service()方法。

HttpServlet 的 service()方法，会依据请求方式来调用 doGet()或者 doPost()方法。

但是，这两个 do 方法默认情况下，会抛出异常，需要子类去 override。

阶段四: 销毁

容器依据自身的算法，将不再需要的 servlet 对象删除掉。

在删除之前，会调用 servlet 对象的 destroy()方法。

destroy()方法用于释放资源。

在 servlet 的整个生命周期当中，init,destroy 只会执行一次，而 service 方法会执行多次。

62.JSP 和 Servlet 的区别?

Servlet 是 sun 公司制订的一种用于扩展 web 服务器功能的组件规范。

java server page(java 服务器端页面技术)，是 sun 公司制订的一种服务器端动态页面生

成技术的规范。 因为直接使用 servlet 生成页面，如果页面比较复杂，则代码过于繁琐，并且难以维护，所以对于比较复杂的页面，使用 jsp 来编写，更容易编写和维护。

当客户端请求访问某个.jsp 文件，则服务器会自动将.jsp 文件转换成一个.java 文件(该.java 文件其实是一个 servlet)。

63.JSP 的隐含对象有哪些? (共 9 个)

out

request

response

session

application 隐含对象其实就是 ServletContext（Servlet 上下文：容器在启动的时候，会为每一个应用创建唯一的一个 Servlet 上下文对象,该对象会一直存在，除非容器关闭。）

exception

config:ServletConfig 实例。

page:相当于 this,代表当前 jsp 实例。

pageContext:PageContext 实例。

容器会为每一个 jsp 实例创建唯一的一个 pageContext 对象。该对象一直存在，除非 jsp 实例被容器销毁。

64.Cookie 和 Session 的区别?

1、cookie 数据存放在客户的浏览器上，session 数据放在服务器上。

cookie 存放在内存或者硬盘上，cookie.setMaxAge(int seconds); cookie 是一小段文本信息，伴随着用户请求和页面在 Web 服务器和浏览器之间传递。用户每次访问站点时，Web 应用程序都可以读取 cookie 包含的信息。

session 失效时间的设置，这里要分两方面来看：浏览器端和服务端。对于浏览器端而言，session 与访问进程直接相关，当浏览器被关闭时，session 也随之消失；而服务器端的 session 失效时间一般是人为设置的，目的是能定期地释放内存空间，减小服务器压力，一般的设置为当会话处于非活动状态达 20 或 30 分钟时清除该 session。

2、cookie 不是很安全，别人可以分析存放在本地的 COOKIE 并进行 COOKIE 欺骗

考虑到安全应当使用 session

3、session 会在一定时间内保存在服务器上。当访问增多，会比较占用你服务器的性能

考虑到减轻服务器性能方面，应当使用 COOKIE

4、单个 cookie 在客户端的限制是 3K，就是说一个站点在客户端存放的 COOKIE 不能超过 3K。

5、所以个人建议：

将登陆信息等重要信息存放为 SESSION

其他信息如果需要保留，可以放在 COOKIE 中

65.谈谈对 MVC 的理解?

1) 什么是 MVC?

MVC 是一种设计思想，根据职责不同将程序中的组件分成以下 3 个部分。

V（View 视图）：负责与用户交互。将数据展现，或者是接收数据

M（Model 模型）：负责业务处理。业务模型，数据模型

C（Controller 控制器）：负责协同模型和视图工作。视图有请求调用模型处理，模型处理完毕调用视图响应。

2)为什么使用 MVC?

MVC 是一个非常优秀的设计思想，基于该思想架构程序，可以提高程序的结构灵活性，便于日后维护、扩展和升级。

注意,下面内容助于理解：

1) 一个模型可以被多个视图共享模型只负责输出数据，不关心数据的表现形式，同一伴数据，可以使用多个不同的视图展现给用户。模型只负责处理数据，不关心是谁在调用，可以使用多种不同的界面来调用模型。

2) 方便测试

模型一般使用 java 类来开发，在开发完成之后，可以立即测试。如果业务逻辑直接写在 servlet

里面，则需要部署在服务器上面才能测试，比较麻烦。

3) 组件复用

控制器可以做成一个通用的模块。

4) 代码好维护，利于分工协作。

按照 mvc 的思想，可以对程序迸行分层，一般划分成表示层(包括 v,c)、业务层(m 中的业务逻辑部分)、持久层(m 中的数据访问逻辑部分)。下一层的代码发生改变，只要接口不变，不会影响到上一层的代码。

mvc 的缺点

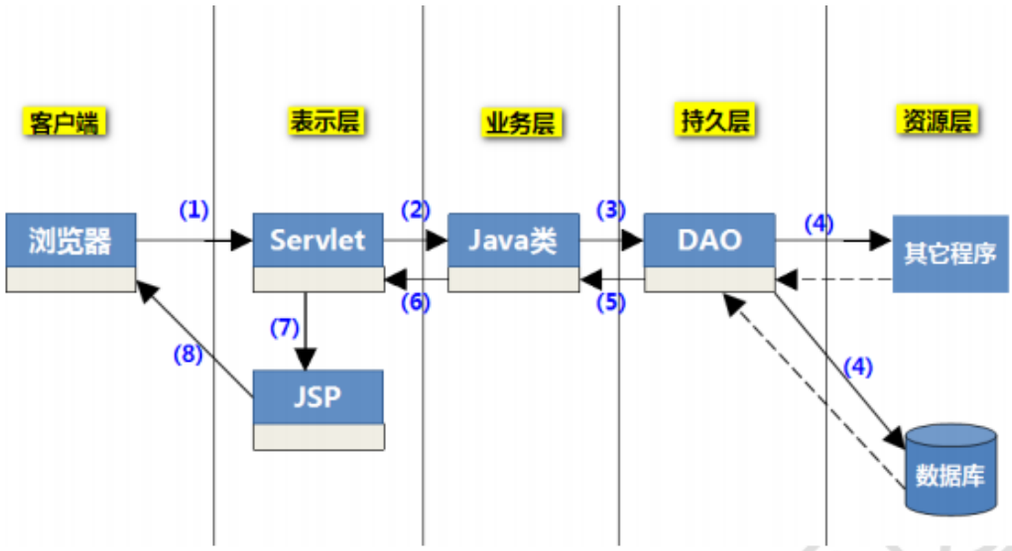
1) 采用 mvc 以后，会增加代码量，相应的开发周期以及开发的成本会相应增加。

2) 使用 mvc，需要良好的设计。如果设计不当，会增加开发的难度。

结论

一般来说，如果一个程序需要良好的架构，需要良好的代码的可维护性及可扩展性，需要使用 mvc

思想来架构。反之，则不必使用。



在表示层 Servlet 中调用业务层代码的接口，当业务层发生改变时不影响 Servelt；
在业务层 Service 中调用 DAO 的接口，DAO 发生改变不影响 Service 和其上层

66.什么是 Ajax?

asynchronous javascript and xml(异步的 javascript 和 xml)。

为了解决传统的 web 应用当中“等待-响应-等待"的弊端而创建的一种技术，其实质可以理解为：使用浏览器内置的一个对象(XmlHttpRequest)向服务器发送请求，服务器返回 xml 数据或者是文本数据给浏览器，然后在浏览器端，使用这些数据更新部分页面，整个过程，页面无任何的刷新。

67. ajax 编程步骤?

1) 获得 XmlHttpRequest 对象
2) 使用 XmlHttpRequest 向服务器发请求。

a.*发送 get 请求:*

```
/* open(请求方式,请求地址,同步/异步)
   * 请求方式: get/post
   * 请求地址：如果是 get 请求，请求参数添加到地址之后。
  * 比如   check_user.do?username=zs
  * 同步/异步:true 表示异步。*/
xhr.open('get','check_user.do',true);
```

b. 发送 post 请求

```
xhr.open('post','check_username.do',true);
//必须添加一个消息头 content-type
xhr.setRequestHeader("Content-Type","application/x-www-form-urlencoded");
```

3). 在服务器端，处理请求。

4).在监听器当中，处理服务器返回的响应。

```
xhr.onreadystatechange=function(){
  //编写相应的处理代码
  if(xhr.readyState == 4){
    //只有 readyState 等于 4,xhr 才完整地接收到了服务器返回的数据。
    //获得文本数据
    var txt = xhr.responseText;
    //获得一个 xml dom 对象。
    var xml = xhr.responseXML;
    //dom 操作、更新页面
  }
   };
5.xhr.send(null)
```

Ajax 技术的优点?

1.页面无刷新

2.不打断用户的操作，用户的体验好

3.按需获取数据，浏览器和服务器之间数据的传输量减少

4.是一个标准技术，不需要下载任何的插件

5.可以利用客户端(浏览器)的计算能力

68.什么是 JQuery?

jQuery 是一个 JS 框架，设计思想是将原始的 dom 对象封装成一个 jQuery 对象，通过调用 jQuery 对象的方法来实现对原始的 dom 对象的操作。这样设计的目的是：是为了更好地兼容不同的浏览器,简化代码。

70.程序，线程，进程的区别?

一般操作系统都支持同时运行多个任务，一个任务通常就是一个程序，每个运行中的程序被称为一个进程，当一个程序运行时，内部可能包含多个顺序执行流，每个顺序执行流就是一个线程。

1) 程序　指令 + 数据的 byte 序列，如: qq.exe
2) 进程　正在运行的程序，是程序动态的执行过程（运行于内存中）
3) 线程　在进程内部，并发运程的过程（Java 中的方法可以看做线程）

71.在做项目的过程中，遇到了中文乱码问题，该如何处理?（比如网站主页查询页面显示乱码）

可以从页面，程序，以及数据库三个方面去查看问题的原因所在，主页面显示中文乱码，无外乎是因为从数据库中取出来的数据传输到主页面的过程中出现了编码问题，那么就应该从这三个方面去查找问题的所在。因为我们常用 UTF-8 的编码格式，所以我们就检查所有数据的编码是否都是 UTF-8 编码。

1. 查看一下页面 jsp 文件的编码格式(1.3.1/1.3.2)

1.1、项目文本文件默认编码：

【右击项目】->【Properties】->【Resource】->【Text file encoding】

1.2、文件默认编码：默认使用项目的默认编码

【右击文件】->【Properties】->【Resource】->【Text file encoding】

1.3、JSP 文件编码：由于 JSP 要翻译为 Servlet

1.3.1、JSP 文件编码：

【右击文件】->【Properties】->【Resource】->【Text file encoding】

1.3.2、JSP 翻译为 Servlet 时的编码：　(此项可以由 1.1,1.2,1.3.1 代替)

<%@ page language="java" pageEncoding="utf-8"%>

1.3.3、从服务器将 jsp 内容输出到浏览器

<%@page contentType="text/html;charset=utf-8"%>

在这次输出过程中，由 contentType 属性中的 charset 来指定，将 servlet 编译后的的二进制码以 charset 的编码形式来输出。

2. 服务器端编码设置

服务器端编码，将客户端传过来的数据进行解码：

浏览器默认使用 ISO-8859-1 进行编码数据，然后将数据传输到服务器，因此我们默认只需要将浏览器发送过来的数据转换为我们需要的编码即可。

GET 请求方式：

2.1 TOMCAT 默认 ISO-8859-1 因此可以设置默认编码为 UTF-8 解决，在 conf\server.xml 文件中设置如下

```
<Connector port="8080" protocol="HTTP/1.1"
connectionTimeout="20000"
redirectPort="8443" URIEncoding="UTF-8"/>
```

2.2

```
String username = request.getParameter("username");
username = new String(username.getBytes("ISO-8859-1"), "UTF-8");
```

POST 请求方式：

2.3

```
// 必须在获取参数之前，调用如下方法先解码
request.setCharacterEncoding("UTF-8");
String username = request.getParameter("username");
response.setContentType("text/html;charset=utf-8");
```

3.数据库的编码设置

3.1 JDBC 连接数据库的编码设置

"jdbc:mysql://localhost:3306/jd1203db?**useUnicode=true&characterEncoding=utf8**","root","1234");

3.2 在建表或者建数据库时可以加上：

create database jsd1203db ENGINE=InnoDB DEFAULT CHARSET=utf8;

3.3 查看 MySql 数据中当前联接系统参数　show variables like 'char%'，显示各个编码类型。

可以 show create database dangdang;通过该语句查看建库的语句，看是否有设置编码。

如果各种编码都是对的，可是在数据库中查看数据时是乱码，可以用 set names 'gbk'，设置查看编码为 gbk，与系统一致(windows)。 注意，这个设置只对当前会话有效。

72.get 和 post 请求的中文乱码问题处理?

当表单提交时，浏览器会对表单中的中文参数值进行编码(会使用打开表单所在的页面时的编码格式来进行编码)。服务器在默认情况下，会使用 iso-8859-1 这种编码格式进行解码。所以会出现乱码。

1)get 请求方式的处理:

第一种方式:

要确保表单所在的页面，使用指定的编码格式打开。对于 html 文件，可以添加:<meta http-equiv="content-type" content="text/html;charset=utf-8">,确保表单中的编码为 utf8,能够存储中文。

在 Servlet 中使用 name = new String(name.getBytes("iso-8859-1"),"utf-8");来处理编码

第二种方式:

找到 tomcat---->conf----->server.xml----><Connector port="8080" protocol="HTTP/1.1" connectionTimeout="20000" redirectPort="8443" URIEncoding="UTF-8"/> 后面加上 URIEncoding="UTF-8"

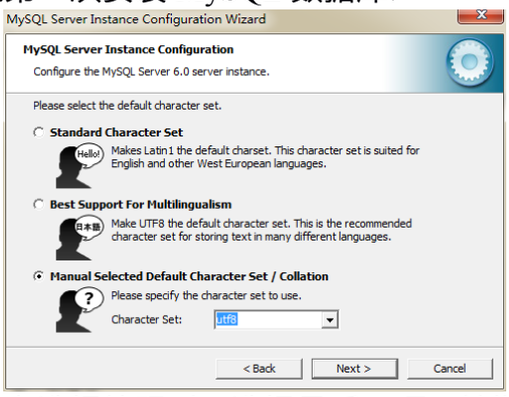
2)post 请求方式的处理:

2.1 首先确保 html 的编码为 utf-8

2.1 在服务器端，取数据之前,添加 request.setCharacterEncoding("utf-8");

MySQL 的乱码解决方案:

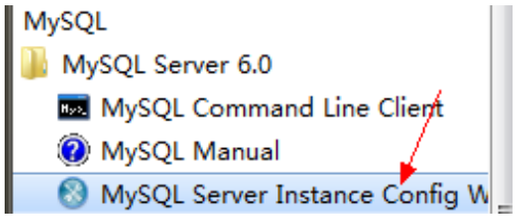
5. 如果第一次安装 MySQL 数据库:



在选择编码时，选择最后一项，并指定编码为 utf8 或者 gbk（这两种编码支持中文,是我们比较常用的）

6. 如果已经按照完成了 MySQL 数据库，那么可以进行重新配置修改:

在开始菜单中找到:



然后重复第一个步骤即可。(注意：这样修改完成后，那么数据库中原来保存的数据如果有中文，就会乱码，因为当前的编码已经变为了 utf8，而原来保存的可能是其它编码。)

7. 重启 MySql 的服务.右键管理员身份打开命令提示符窗口,

```
C:\Windows\system32>net stop MySQL
MySQL 服务正在停止..
MySQL 服务已成功停止。


C:\Windows\system32>net start MySQL
MySQL 服务正在启动 .
MySQL 服务已经启动成功。
```

8. 进入 MySQL 中，如果通过一下命令查看编码如下图所示，那么表示配置正确:

```
mysql> show variables like 'char%';
+--------------------------+----------------------------------------------------
-----------+
| Variable_name            | Value
          |
+--------------------------+----------------------------------------------------
-----------+
| character_set_client     | utf8
          |
| character_set_connection | utf8
          |
| character_set_database   | utf8
          |
| character_set_filesystem | binary
          |
| character_set_results    | utf8
          |
| character_set_server     | utf8
          |
| character_set_system     | utf8
          |
| character_sets_dir       | D:\Program Files (x86)\MySQL\MySQL Server 6.0\share
\charsets\ |
+--------------------------+----------------------------------------------------
-----------+
```

但是这个时候如果向表中插入数据，会报编码错误，如下图:

```
mysql> insert into users values(3,'张三','123123',1,'1998-12-11','2012-04-21',0,
'乱码啊');
ERROR 1366 (HY000): Incorrect string value: '\xD5\xC5\xC8\xFD' for column 'usern
ame' at row 1
```

这个时候，我们就可以使用 set names gbk;命令来设定字符集为 gbk。就可以正常插入数据了。(注意：这个只是让显示的时候正确，并不会改变表中的数据的编码。此命令只在当前会话中有效，也就是如果新开一个窗口打开 MySql，用 select 查询时会发现还是乱码，这个时候如果要正确显示，那么还得用此语句。)

73.Servlet 中的数据插入 MySql 乱码:（我们都用 UTF8 这样的编码来存储）

8. 首先在 Servlet 这个 java 文件得是 utf8 编码，在其中打印要插入 MySql 中的数据，查看是否能够正确显示，如果能够正确显示那么说明 Servlet 中的数据就是 utf8 编码。
9. 如果上一步不能通过检查，那么插入数据就会出现问题，直到检查第一步完成为止。
10. 建数据库之前首先保证 MySql 按照上面的安装正确:
11. 创建数据库: create database wuxi default character set 'utf8';
12. 可以使用 show create database wuxi;来查看这个数据库的 SQL 语句.
13. 建表时加上: create table t_pic(id bigint primary key auto_increment,picName varchar(100), userId bigint) ENGINE=InnoDB DEFAULT CHARSET=utf8;
14. JDBC 连接数据库时: conn = DriverManager.getConnection(

"jdbc:mysql://localhost:3306/jd1105db?useUnicode=true&characterEncoding=utf8","root","1234");

74.MySql 和 Oracle 数据库的区别?

查询当前所有的表	SQL> select * from tab;
	SQL> select * from cat where table_type='TABLE';//可能会有 view
	SQL>select * from user_tables;
		mysql> show tables;
		c:/mysql/bin>mysqlshow 库名

显示当前连接用户（库）	SQL> show user	mysql> connect
查看帮助	SQL> ?	mysql> help
显示表结构	SQL> desc 表名 SQL> describe 表名	mysql> desc 表名; mysql> describe 表名; mysql> show columns from 表名; c:/mysql/bin>mysqlshow 库名 表名

日期函数	SQL> select sysdate from dual;	mysql> select now();
		mysql> select sysdate();
		mysql> select curdate();
		mysql> select current_date;
		mysql> select curtime();
		mysql> select current_time;

日期格式化	SQL> select to_char(sysdate,'yyyy-mm-dd') from dual; SQL> select to_char(sysdate,'hh24-mi-ss') from dual;	mysql> select date_format(now(),'%Y-%m-%d'); mysql> select time_format(now(),'%H-%i-%S');

日期函数（增加一个月）	SQL> select to_char(add_months(to_date('20000101','yyyymmdd'),1),'yyyy-mm-dd') from dual;
	结果：2000-02-01
	SQL> select to_char(add_months(to_date('20000101','yyyymmdd'),5),'yyyy-mm-dd') from dual;
	结果：2000-06-01
		mysql> select date_add('2000-01-01',interval 1 month);
		结果：2000-02-01
		mysql> select date_add('2000-01-01',interval 5 month);
		结果：2000-06-01

别名	SQL> select 1 a from dual; SQL>select 1 as a from dual; //as 可以省略	mysql> select 1 as a;

字符串截取函数	SQL> select substr('abcdefg',1,5) from dual;
	SQL> select substr('abcdefg',1,5) from dual;
	结果：abcde
		mysql> select substring('abcdefg',2,3);
		结果：bcd
		mysql> select mid('abcdefg',2,3);
		结果：bcd
		mysql> select substring('abcdefg',2);
		结果：bcdefg
		mysql> select substring('abcdefg' from

		2); 结果：bcdefg 另有 SUBSTRING_INDEX(str,delim,count) 函数 返回从字符串 str 的第 count 个出现的分隔符 delim 之后的子串。 如果 count 是正数，返回最后的分隔符到左边(从左边数) 的所有字符。 如果 count 是负数，返回最后的分隔符到右边的所有字符(从右边数)。
执行外部脚本命令	SQL >@a.sql	1：mysql> source a.sql 2：c:/mysql/bin>mysql <a.sql 3：c:/mysql/bin>mysql 库名 <a.sql
导入、导出工具	exp.exe exp73.exe imp.exe imp73.exe	mysqldump.exe mysqlimport.exe
改表名	SQL> rename a to b;	mysql> alter table a rename b;
执行命令	;<回车> / r run	;<回车> go ego
distinct 用法	SQL> select distinct 列 1 from 表 1; SQL> select distinct 列 1,列 2 from 表 1;	mysql> select distinct 列 1 from 表 1; mysql> select distinct 列 1,列 2 from 表 1;
注释	-- /*与*/	# -- /*与*/
当作计算器	SQL> select 1+1 from dual;	mysql> select 1+1;
限制返回记录条数	SQL> select * from 表名 where rownum<5;	mysql> select * from 表名 limit 5;
新建用户（库）	SQL> create user 用户名 identified by 密码;	mysql> create database 库名;
删用户（库）	SQL> drop user 用户名;	mysql> drop database 库名;
查询索引	SQL> select index_name,table_name from user_indexes;	mysql> show index from 表名 [FROM 库名];
通配符	“%”和“_”	“%”和“_”
SQL 语法	SELECT selection_list 选择哪些列 FROM table_list 从何处选择行	SELECT selection_list 选择哪些列 FROM table_list 从何处选择行

	WHERE primary_constraint 行必须满足什么条件 GROUP BY grouping_columns 怎样对结果分组 HAVING secondary_constraint 行必须满足的第二条件 ORDER BY sorting_columns 怎样对结果排序	WHERE primary_constraint 行必须满足什么条件 GROUP BY grouping_columns 怎样对结果分组 HAVING secondary_constraint 行必须满足的第二条件 ORDER BY sorting_columns 怎样对结果排序 LIMIT count 结果限定
自动增长的数据类型处理	create sequence myseq increment by 1 start with 1 maxvalue 99999;	主键列上加 auto_increment

Struts2 详细工作流程?

a.浏览器发出 welcome.action 请求

b.请求到达 Struts 的 Filter 控制器(由于 web.xml 配置)

c.Struts 控制器判断请求类型，如果是/welcome.action 或/welcome 格式请求，将调用 struts.xml 配置，寻找对应的 Action 组件

d.调用 Action 组件的 execute 方法处理请求，最后返回一个 String 视图标识

e.Struts 控制器根据视图标识寻找相应的 JSP（struts.xml 中 result 配置）

f.调用 JSP 视图生成响应界面给浏览器显示。

Struts2 控制流程?

1)请求到来

2)创建 ValueStack(Action 放栈顶)，进行初始化

3)调用拦截器 Interceptor，在拦截器中是可以访问 ValueStack 的

4)调用 Action,执行 execute()方法

5)调用 Result，Result 负责把数据显示给用户

6)最后到页面，通过标记库(Taglib)取出数据

Struts2 中常用的集中 Result 组件?

1）JSP 响应

dispatcher ： 采用请求转发方式调用 JSP 组件响应。

redirect：采用请求重定向方式调用 JSP 组件响应。(在重定向后,原 request 和 action 都被销毁掉，因此在 JSP 获取不到值)

2）Action 响应

redirectAction：采用重定向方式发出一个*.action 请求

chain：采用转发方式发出一个*.action 请求

3）流响应

典型功能：验证码和下载。

stream：可以将 Action 中的一个 InputStream 类型属性以流方式响应输出。

4）JSON 响应

负责对 Ajax 请求进行数据响应。

json：可以将 Action 中的一个属性或多个属性以 json 格式响应输出

(注意：使用前需要引入 struts-json-plugin.jar，然后将<package>元素的 extends 设置成"json-default")

filter（过滤器）和 interceptor（拦截器）的区别?

1、拦截器是基于 java 的反射机制的，而过滤器是基于函数回调 。
2、过滤器依赖与 servlet 容器，而拦截器不依赖与 servlet 容器 。
3、拦截器只能对 action 请求起作用，而过滤器则可以对几乎所有的请求 起作用 。
4、拦截器可以访问 action 上下文、值栈里的对象，而过滤器不能 。
5、在 action 的生命周期中，拦截器可以多次被调用，而过滤器只能在容 器初始化时被调用一次 。

Hibernate 是什么，有什么作用和好处?

Hibernate 是一个数据库访问框架，用于实现对数据库的增删改查操作。使用 Hibernate 框架可以简化数据库访问操作，要程序员将更多的经历放在业务层编写上。

原有 JDBC 操作数据库存在一些问题，主要有以下几个方面:

a)需要编写大量复杂的 SQL 语句

b)需要设置大量的 SQL 参数或者将 ResultSet 取值封装成实体对象

c)当数据库移值时，需要修改部分 SQL 语句和操作

使用 Hibernate 框架可以解决以上问题。

什么是 ORM?

ORM(Object Relation Mapping)对象关系映射。意思是将程序中的实体对象和关系数据库表中的一行记录进行映射。这样在程序中使用该工具就可以将一个对象写入数据表，或者将数据表记录自动封装成一个对象返回。(这个也是 Hibernate 实现的原理)

Hibernate 的几个核心接口?

Configuration 类：Configuration 用于配置并启动 Hibernate。

SessionFactory 接口：一个 SessionFactory 对应一个数据源，它是个重量级对象，不可随意生成多个实例。它是线程安全的，同一个实例可以被应用中的多个线程共享。

Session 接口:Session 接口是 Hibernate 应用中使用最广泛的接口了,它是持久化管理器，提供添加、更新、删除、加载、查询对象。Session 不是线程安全的，所以应避免多个线程共享同一个 Session 实例。Session 是轻量级对象，它的创建和销毁不需要太多资源，这意味着在应用中可以经常创建和销毁 Session 对象。

Transaction 接口：Transaction 是 Hibernate 的数据库事务接口，它对底层的事务接口进行了封装。

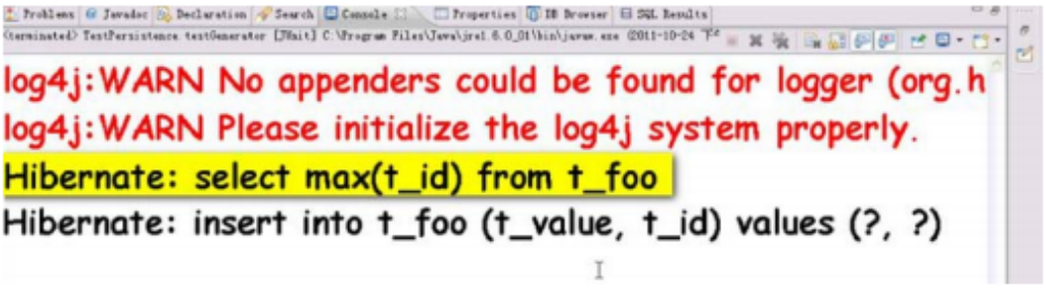
Query 和 Criteria 接口：这两个是 Hibernate 的查询接口，用于向数据库查询对象，以及控制执行查询的过程。

Hibernate 中的主键生成方式有哪些?

常用的主键生成方式有如下几种：

1)	identity	用于自动生成主键方式，除了 Oracle 不支持，其他数据库一般都支持（较常用）
2)	sequence	Oracle 中使用，用序列生成 ID 主键
3)	native	主键生成方式如果是 native，那就看配置文件 hibernate.cfg.xml 中方言<property name="dialect">是什么，如果方言是 Mysql，相当于 identity，如果方言是 Oracle，相当于 sequence
4)	increment	不常用
5)	assigned	不常用，手动生成 id

查看控制台，Hibernate 执行了 2 条 SQL



increment 生成主键方式是先“select max(t_id) from t_foo ”，从 t_foo 中找到最大的 id 之后将 max(t_id)加 1，这样就保证了主键唯一。

但是这样有风险，当并发访问时会有风险。

不建议使用。

Hibernate 一级缓存?

是 session 级别的缓存。

a.当查询某个对象时，Hibernate 首先去一级缓存中查找，如果存在将缓存中的对象取出返回。如果不存在才去数据库查询。

b.当查询某个对象时，Hibernate 会自动将数据库查询出的对象放入一级缓存。(例如 session 的 load 和 get 方法查出的对象)

c.一级缓存默认启用，一般被称为 Session 级别的缓存。因此一级缓存随着 Session 对象创建，随着 Session 对象释放而销毁。

d.每个 Session 都有自己一个独立的一级缓存空间，不能访问其他 Session 的缓存空间。

----------一级缓存的好处----------

在使用同一个 Session 查询同一个对象若干次的时候，只在第一次时查询数据库，后续几次都会从缓存取出。从而减少对数据库的查询次数。

----------**使用建议（重点）**----------

为了将一级缓存的优势更好的发挥出来，建议在处理一个请求时，使用一个 Session 对象处理。可以使用 ThreadLocal 技术封装 Session.

在Hibernate中，可以 <!-- 指定将session与处理线程绑定，实现线程单例 -->

```
<property name="current_session_context_class">
    thread
```

</property>

Hibernate 的二级缓存?

二级缓存也称为进程级的缓存或 SessionFactory 级的缓存，二级缓存可以被所有的 session 共享 ，可以被多个不 Session 对象访问,二级缓存的生命周期和 SessionFactory 的生命周期一致，SessionFactory 可以管理二级缓存。

线程单例及其使用?

线程单例

回到服务器中，只要是服务器，每一个浏览器访问服务器时，服务器会为每个浏览器创建一个线程。
假设 Some 就是 Session，如果使用这种机制获取 Session，当同一个用户浏览器不论怎么调用 session 都是同一个（只要在相同的线程中）。
这种机制就叫做线程单例。
线程单例的实现原理就是如上 SomeFactory 做的。

```
private static ThreadLocal<Session> tl = new ThreadLocal<Session>();
```

```
Session session = tl.get();
if (session == null) {
        session = factory.openSession();
        tl.set(session);
}
```

Hibernate 中 session 的 get 和 load 方法区别?

相同点：作用相同，都是按照主键条件查询某个对象。

不同点：

1).get 方法不使用延迟加载机制，load 采用延迟加载机制

2).如果没有满足条件的记录，get 方法返回 null，load 则抛出异常

3）.load 方法返回的对象是一个动态代理类（Hibernate 框架动态生成的，是实体类的子类型）。

4).get 方法返回的对象类型就是原实体类型

注意:a.在程序中，如果使用上述延迟加载操作，需要避免 Session 对象过早关闭问题。(could not initialize proxy - no Session)

b.为避免上述异常，一般采用 OpenSessionInView 模式。

c.可以将 Session 关闭采用 Filter 或 Interceptor(Struts2 推荐)封装。

HQL 和 SQL 的区别?

Hibernate Query Language（Hibernate 查询语言）

结构化查询语言(Structured Query Language)简称 SQL

HQL 语句结构与 SQL 语句相似， SQL 语句是面向数据表和字段进行查询，而 HQL 是面向映射后的对象和属性进行查询。因此 HQL 被称为面向对象查询语句。

HQL 语句与 SQL 相似点如下:

a.HQL 语句支持 select,from,where,order by ,group by,having 子句

b.HQL 语句支持分组函数 max,min,avg,sum,count

c.HQL 语句支持运算符和表达式

d.HQL 语句支持>,>=,<,<=,in,not in,like,between... and...,

<>,!=等查询条件,也支持 and,or 关键字

e.HQL 语句支持 inner join,left outer join,full join 等连接

HQL 语句与 SQL 不同点如下：(下面内容为重点)

a.HQL 区分大小写（大小写敏感）,除关键字之外

b.HQL 语句使用的类名和属性名（将 SQL 表名替换成类名，字段名替换成属性名）

c.HQL 不支持 select *写法，但支持 select count(*)

d.HQL 语句不支持表连接的 on 子句，on 关联条件可以通过关联映射自动追加

Hibernate 中操作并发处理?（乐观锁和悲观锁）

Hibernate 框架可以使用锁的机制来解决操作并发。

a.悲观锁

在数据查询出来时，就给数据加一个锁，锁定。这样其他用户再执行删、改操作时不允许。当占用着事务结束，锁会自动解除。

Hibernate 采用的是数据库锁机制实现悲观锁控制。

缺点：将并发用户操作同步开，一个一个处理。当一个用户处理时间比较长时，效率会比较低。

b.乐观锁

允许同时更新提交，但是最快的会成功，慢的失败。

在记录中追加一个字段值，用该字段值当做版本。当最先提交者提交后，会自动将版本字段值提升，这样其他用户提交，会发现版本低于数据库记录目前版本，因此抛出异常提示失败。

特点：允许用户同时处理，但只能有一个成功，其他失败，以异常方式提示。

Spring 框架的作用和好处?

Spring 框架提供了一个容器，该容器可以管理应用程序的组件，还提供了 IoC 和 AoP 机制，实现组件之间解耦，提高程序结构的灵活性，增强系统的可维护和可扩展性。

在 SSH 整合开发中，利用 Spring 管理 Service、DAO 等组件，利用 IoC 机制实现 Action 和 Service,Service 和 DAO 之间低耦合调用。利用 AoP 机制实现事务管理、以及共通功能的切入等。

功能是整合，好处是解耦。

IoC(Inverse of Controller,控制反转)?

控制权：A 调用 B 组件，可以说成 A 组件拥有控制权。控制权可以代指对象的创建、初始化、销毁等操作。

控制反转：是将控制权转移，转移给第三方（Spring 容器），当需要改变对象关系时，只需要修改 Spring 容器注入配置即可。

IoC 是一种思想，Spring 框架通过 DI（依赖注入:setter 注入和构造方法注入）技术实现了控制反转。

什么是 AOP?

Aspect Oriented Programming 面向方面编程。

面向方面编程侧重点是关注方面组件（共通处理部分），可以将方面组件作用到某一批目标对象的方法上。

面向对象编程侧重点是关注对象，如何构建一个对象类型。

面向方面编程是基于面向对象编程的，主要用于改善程序结构，降低组件耦合度。

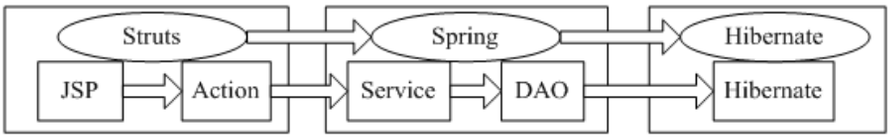
在 SSH 的组合框架模式中，三者各自的作用?

Struts 是一个很好的 MVC 框架，主要技术是 Servlet 和 Jsp。Struts 的 MVC 设计模式可以使我们的逻辑变得很清晰, 让我们写的程序层次分明。基于 Struts 开发可以简化开发难度，提高开发效率。

Spring 提供了管理业务对象的一致方法, 并鼓励注入对接口编程而不是对类编程的良好习惯，使我们的产品在最大程度上解耦。

Hibernate 是用来持久化数据的, 提供了完全面向对象的数据库操作。Hibernate 对 JDBC 进行了非常轻量级的封装，它使得与关系型数据库打交道变得非常轻松。

在 Struts+Spring+Hibernate 系统中，对象之间的调用流程如下：



Struts——>Spring——>Hibernate

JSP——>Action——>Service——>DAO——>Hibernate

SSH 工作流程?

a.启动服务器，加载工程以及 web.xml.

(实例化 Lisener,Filter 等组件，将 Spring 容器和 Struts2 控制创建)

b.客户端发送请求，所有请求进入 Struts2 控制器。控制器根据请求类型不同，分别处理。

(action 请求，*.action 会进入 struts.xml 寻找<action>配置.

其他请求，*.jsp 会直接调用请求资源，生成响应信息)

c.Struts2 控制器根据<action>配置调用一个 Action 对象处理。

整合方法一：将 Action 交给 Spring 容器

(Action 对象由 struts2-spring-plugin.jar 插件提供的 StrutsSpringObjectFactory 负责去 Spring 容器获取)

整合方法二：将 Action 置于 Spring 容器之外

(Action 对象由 struts2-spring-plugin.jar 插件提供的 StrutsSpringObjectFactory 负责创建，然后到 Spring 容器中寻找与 Action 属性匹配的 Bean 对象，给 Action 对象注入。(默认采用名称匹配规则)

d.Struts2 控制器执行 defaultStack 拦截器、Action 对象、Result 等组件处理.

e.执行 Action 的 execute 业务方法时，如果使用 Service 或 DAO 采用 Spring 的 IoC 机制调用。

f.执行 Result 生成响应信息，执行后续拦截器处理

g.将响应信息输出。