# Java基础

1. **什么是Java?**

Java是一门面向对象的编程语言；

Java的三大特征：

1）封装（私有化属性、生成set get方法、生成有参午餐构造），

2）继承（子类继承父类拥有父类的所有属性和方法），

3）多态（重写：子类重写父类的方法，重载：同一类中方法名相同参数列表不同）

1. **Java的数据类型：**

基本数据类型：类型 位数 取值范围 包装类

整数型：byte 1个字节 -128——127（字节型） Byte

short 2个字节 -2的15次方——2的15次方-1（短整型） Short

int 4个字节 -2的31次方——2的31次方-1（整数型） Integer

long 8个字节 -2的63次方——2的63次方-1（长整型） Long

浮点型：float 4个字节 精度：小数点后7位（单精度） Float

Double 8个字节 精度：小数点后15位（双精度） Double

字符型：char 2个字节 只存储一个字符 Character

布尔型：boolean 1字节 只有 true 或false Boolean

引用数据类型：

类 Class、 接口 Interface、 数组Array

1. **包装类的常用方法**

Integer parseInt(String) String转Integer

valueOf(String或者int) String或者int 转 Integer

toString(Integer) Integer 转 String

Double parseDouble(String) String转 Double

valueOf(String) String 转 Double

toString(Double) double 转String

1. **基础语法**

if语句：

if（条件){方法体}else{}，if（条件）{}else if（条件）{}。。。。

For循环

For(初始值，循环条件，步长){}

While循环

while(条件){方法体；步长}//循环开始会先执行一次方法体

1. **String的常用方法：**

charAt（下标） 返回下标对应的字符；

indexOf（字符串) 返回字符串对应的下标；（不存在返回-1）

Length（） 返回字符串的长度；

Replace（字符串1，字符串2）将字符串1替换为字符串2；

Contains（）判断字符串是否包含参数中的字符串（返回true或false）；

toUpperCase()将字符串的字母转为大写；

toLowerCase()将字符串的字母转为小写；

Tirm()去掉字符串前后的空格；

Split(字符) 按字符将字符串进行分割得到字符串数组；

Substring(start,end) 从开始下标截取到结束下标的前一个字符（包前不包后）；

Concat(字符串) 将字符串进行拼接得到新的字符串；

1. **Date的相关方法**

getTime()获取毫秒值

转换类SimpleDateFormat，parse(String)将字符串转换成Date，format(Date)将date转换成String

1. **File的常用方法**

Exist() 判断文件夹是否存在 mkdir()创建文件夹 delete()删除文件

1. **Io流**

输入流 input （用于将本地文件上传到服务器中）

1. inputStream（字节输入流）
2. reader （字符输入流）

输出流 out（用于从服务器下载文件）

1）outputStream（字节输入流）

2）writer （字符输入流）

8、**继承的特点 extends**

①写在子类中

②格式：public class 子类类名 extends\* 父类类名{}

③子类继承父类，拥有父类所有的属性和方法；

但不能继承父类私有的特性和构造方法；

④创建子类对象时，会默认先调用父类的构造方法；

⑤一个子类只能有一个父类，但一个父类可以有多个子类（单继承）；

⑥将类中相同的属性和方法，向上提取并封装成父类，增加代码复用性，减少代码量

**9、重载的特点**

在同一个类中，方法名相同，参数的个数、类型、顺序至少有一个不同，就构成方法的重载，重载与修饰符和返回值类型无关。

**10、重写的特点**

①发生在继承关系中；

②方法名、返回值类型、参数列表必须与父类完全一致；

③子类不能重写父类私有方法和static方法；

④子类重写父类 访问修饰符权限不能小于父类，

不能报出比父类更大的异常。

**11、重写和重载的区别**

①重载要在同一个类中

重写要在存在继承关系的两个或多个类中，必须时非私有非静态的方法；

②重载要求参数的个数、类型、顺序至少有一个不同

重写要求方法名和返回值类型与父类完全一致；

③重载与访问修饰符无关,与异常无关

重写的访问修饰符权限不能小于父类；不能报出比父类更多的异常。

**12、抽象类 抽象方法的特点abstract 抽象的**

①abstract修饰的类叫做抽象类 修饰的方法叫做抽象方法

②抽象类不能被实例化，使用抽象类中的属性与方法时，要去创建一个类

去继承抽象类；

③抽象方法没有方法体；

④抽象方法要在抽象类中；

⑤抽象类中可以有抽象方法，也可以有普通方法；

⑥一个类继承抽象类，要重写抽象类中所有的抽象方法；

**13、接口的特点interface 接口** implements 实现

①接口中所有的方法都是抽象方法；

②final修饰的属性成为常量，必须赋初始值，不能二次赋值；

③接口不能被实例化，使用接口中的属性和方法时，要创建一个类实现接口

④实现接口必须重写接口中所有的抽象方法；

⑤一个类可以实现多个接口（多实现）

14、**接口和抽象类的区别（异同点）**

相同点：

①都可以有抽象方法

②都不能被实例化

③一个类要继承抽象类 必须重写抽象类中所有的抽象方法

一个类实现接口 也必须重写接口中所有的抽象方法

不同点：

①接口不是类，用interface表示，

抽象类是一个用abstract修饰符修饰的类

②接口里面只能有抽象方法

抽象类中可以有抽象方法和普通方法

③一个类可以实现多个接口

但是只能继承一个类

④接口中的变量都是final修饰的常量，必须赋初始值不能二次赋值

而抽象类没有限制

1. **final的特点**

1）修饰的类不能被继承，修饰的方法不能被重写；

2）修饰的属性是常量，必须赋初始值，不能二次赋值；

**16、BigDecimal（数学类5）**

常用方法

1）add（） 加法

2）Subtract（） 减法

3）Multiply（） 乘积

4）Divide（） 除法

5）Remainder（） 求余数

6）Min（） 最小值

7）Max（） 最大值

8）Abs（） 绝对值

9）Negate（） 相反数

**17、Random类**

nextBoolean()在boolean取值范围内随机取值；

nextDouble()：0-1之间的小数 有效位15-16位；

nextFloat()：0-1之间的小数 有效位7-8位

nextInt()：int取值范围内的所有数的随机

nextInt(101)：[0-100]

nextInt(参数) 取值范围 是 0到参数-1

随机取出0--参数之前的书（不包含本身）

nextInt(参数)+数 取出指定范围内的随机一个数

括号内的数为终止数，括号外为从多少开始

**18、String和StringBuilder和StringBuffer的区别**

1）String是一个字符串常量，长度固定

2）StringBuffer是一个长度可变的量，线程安全，运行效率低高于String

3）StringBuilder是一个长度可变的量，线程不安全，运行效率高于StringBuffer高于String

**19、集合**

1）collection （顶层接口） 开来可深

作用：进行数据储存

collection是所有集合的所有集合接口

·创建对象

Collection集合名 = new 实现类();

Collection<泛型> 集合名 = new 实现类<泛型>();

·注意：泛型只能是类，如果要使用基本数据类型，请使用基本数据类型的包装类

·常用方法:

1.add(); 添加

2. size(); 集合的元素的个数

3. remove() 移除指定元素

4. iterator(); 迭代器

a. hasNext() 判断是否有下一个元素（true为有下一个元素。false为没有 下一个元素）

b. next() 取得迭代器中的下一个元素

2）List集合是Java.util包中的类

·创建对象的方法：

List 集合名 = new 实现类();

List<泛型> 集合名 = new 实现类<泛型>();

·List集合的特点：

List集合有下标，数据是有序的，可以存放重复数据

·List集合的常用方法：

add() 添加元素

remove() 移除元素 瑞姆

size() 获得集合的长度

get(index) 获得指定下标的元素

iterators() 获得迭代器 艾特瑞特

3）Set集合

①HashSet ②TreeSet

特点：无序 无下标 不能放重复数据

·ArrayList 和 LinkedList，Vector的区别：

·ArrayList基于数组的存储结构，查询效率快，删除和添加数据比较慢，线程不安全

· LinkedList基于链式存储结构，查询效率慢，删除和添加效率快， 线程不安全

·Vector线程安全的，已经不建议使用

**20、Map映射(不能被实例化，通过实现类创建Map对象）**

Map接口的实现类 HashMap，HashTable

1.对象的创建

Map<泛型，泛型> 变量名 = new 实现类<泛型，泛型>();

存值得方式：以键值对的形式存在，并且，键对象不允许重复，值对象 允许重复

map.put("键对象"，“值对象”);

常用方法

取值方式: get（） 通过键对象获取值对象 map.get(键对象);

得到所有的值对象：map.values();

得到所有的键对象：map.keySet();

2.HashMap和HashTable的区别

HashMap 线程不安全的，效率高，允许键对象和值对象为null

HashTable线程安全的，效率低，不允许键对象和值对象为空，否则会报异常

Map是无序的，它的存储结构是哈希表<key,value>键值对，map中插入元素是根据key计算出的哈希值来存储元素的，因此他不是按照元素的添加顺序来存储对象的，所以Map是无序的

1. hashMap的底层原理

HashMap存储方式在jdk8之前用的是数组加链表，在之后用的是数组+链表+红黑树。当我们往hashMap里放值时，首先计算key的hashcode，再%（取模）数组的长度，从而获得该值在数组上的位置下标，若该位置上没有放置元素，则直接存放在该位置上；若已有元素，则插入该位置下的链表头部。Jdk1.8之后在此基础上增加了红黑树的存储方式，当链表上的key达到8个的时候将链表变为红黑树的存储方式，小于6个时候会再转换成链表的存储方式。（补充：我后来在网上查阅资料的时候看到个问题，就是说这个头插法有可能会在扩容时候出现问题，扩容的时候会重新放置数据，并且它会将链表中的数据倒序插入到一个新的链表，这时候当有一个新的元素存储过来的时候，会造成新元素和从链表尾部过来的值相互指向，造成死锁，jdk8之后采用了尾插法就避免了这个问题）

**21、get和post区别**

1）get是从服务器上获取数据，post是向服务器传送数据。

2）get传送的数据量较小，不能大于2KB。

post传送的数据量较大，一般被默认为不受限制。

3）get安全性非常低，post安全性较高。但是执行效率却比post方法好。

4）在进行文件上传时只能使用post而不是get。

22、**反射**

1. 通过类对象获取类的所有属性和方法；
2. 通过对象可以获取到属性的值和调用它的方法；

反射中常用的几个对象：

Class 类（可以获取所有的属性（getDeclaredFields()、getFields()、获取所有方法 getDeclaredMethods()））

Field 属性（获取属性名getName()、获取属性的数据类型getType() ））

Method 方法（get、获取返回值类型 getReturnType() 、执 行方法invoke() ）

Constructors 构造方法 （获取构造方法 getConstructors()）

Annotation 注解 （获取注解 getAnnotations()）

1. 自定义注解
   1. 自定义注解的关键字@Interface
   2. 自定义注解的四个元注解
      1. Target 声明注解的使用范围
      2. Retention 声明注解的生命周期
2. Lambda表达式

Lambda表达式是jdk1.8的新特性，是一种简单快捷实现接口的新写法，但是不是所有的接口都适合lambda表达式，需要以函数接口为前提（函数接口也就是有且只有一个抽象方法，都可以有多个普通方法），使用如下：

//接口

public interface EnumDemo {  
 public void query(String a);  
}

//实现方法及调用

String abb = "888";  
EnumDemo enumDemo = (a)->{  
 System.*out*.println(a);  
};  
enumDemo.query(abb);

1. Stream流
   1. Stream就是用来操作集合的；
   2. 创建Stream流的方法
      1. 集合名.stream();
   3. Stream的常用方法
      1. forEach()用来遍历集合
      2. filter()条件过滤
      3. Limit() 取定量数据（取几条）
      4. Skip()跳过几条
      5. Sorted()对集合进行排序
2. 多线程
   1. 什么是进程？什么是线程？

进程：进程指正在运行的程序。确切的来说，当一个程序进入内存运行，即变成一个进程，进程是处于运行过程中的程序，并且具有一定独立功能，进程主要用于获取CPU资源。

线程：线程是程序中最小的执行单元，一个进程至少有一个线程；

多线程：一个进程中有多个线程同时执行任务；

* 1. 线程的状态
     1. 创建（new之后）
     2. 就绪（执行start之后）
     3. 运行（执行run之后）
     4. 阻塞（执行sleep或wait之后，sleep不释放锁，wait释放锁）
     5. 死亡（run方法执行结束后）
  2. 创建线程的方法
     1. 实现Runable类
     2. 继承Thread类
     3. 实现callable类以及Future；(此线程可以有返回值)
  3. 线程安全问题
     1. 当多个线程同时操作同一个资源时会造成线程不安全问题；此时我们通过synchronized给代码块加锁
     2. 还有就是死锁，A线程拿着B线程需要资源不释放，B线程拿着A线程需要的资源也不释放，造成死锁，导致程序无法往下进行；尽量避免死锁的情况，或者线程礼让（线程礼让也不完全能解决死锁的问题）

1. 线程池
   1. 什么是线程池？
      1. 线程池就是事先创建若干个可执行的线程放入一个容器中，当有任务需要执行时，直接从线程池拿空闲线程使用，执行完任务后将线程归还给线程池，这样就实现了线程的复用，从而减少了创建和销毁线程对象的开销。
   2. 线程池的优点？
      1. 降低系统消耗；
      2. 提升响应速度；
      3. 提高线程的管理性；
   3. 线程池运行原理
      1. 提交多线程任务，判断线程池核心线程数是否已经满，若没有满足则创建线程执任务，
      2. 若核心线程都在占用，判断等待队列是否已满，未满则进入队列；
      3. 队列已满则判断线程数是否达到最大线程，若为达到则直接创建线程执行任务；
      4. 未达到最大线程数则进入拒绝策略；
   4. 线程池的核心类ThreadPoolExecutor的主要参数
      1. c**orePoolSize：线程池中核心线程数**
      2. **maximumPoolSize：线程池最大线程数**
      3. **keepAliveTime：非核心线程闲置超时时间**
      4. Unit**：keepAliveTime的时间单位**
      5. **workQueue：线程池中的任务队列**
      6. **threadFactory：线程池创建线程时使用的线程工厂**
      7. **handler：线程池的饱和策略**
   5. 常用的线程池：

FixedThreadPool

创建一个固定工作线程数量的线程池。核心线程数和最大线程数是一样的，任务如果过多会在队列中阻塞。如果某个线程因为执行异常而结束，那么线程池会补充一个新线程。

|  |
| --- |
| 线程池中的线程处于一定的量，可以很好的控制线程的并发量  线程可以重复被使用，在显式关闭之前，都将一直存在  超出一定量的任务被提交时候需在队列中等待 |

CachedThreadPool

创建一个可缓存的线程池。核心线程数是0，最大线程数是 Integer.MAX\_VALUE。当线程池中的线程都处于活动状态的时候，线程池会创建新的线程来处理新任务，否则会用空闲的线程来处理新任务。这类线程池的空闲线程都是有超时机制的，keepAliveTime在这里是有效的，时长为60秒，超过60秒的空闲线程就会被回收，当线程池都处于闲置状态时，线程池中的线程都会因为超时而被回收，所以几乎不会占用什么系统资源。

|  |
| --- |
| 线程池中线程的数量没有固定大小，可达到最大值（Interger. MAX\_VALUE）  线程池中的线程可进行缓存重复利用和回收（回收默认时间为1分钟）  当线程池中没有可用线程，会重新创建一个线程 |

SingleThreadExecutor

创建一个单线程的线程池。这个线程池的核心线程数和最大线程数都是1，也就是相当于单线程串行执行所有任务。如果这个唯一的线程因为异常结束，那么会有一个新的线程来替代它。此线程池保证所有任务的执行顺序按照任务的提交顺序执行。

|  |
| --- |
| 线程池中最多有1个线程，之后提交的任务将会排在队列中依次执行 |

ScheduledThreadPool

创建一个支持定时及周期性的任务执行的线程池。

|  |
| --- |
| 可定时或者延迟执行线程活动 |

SingleThreadScheduledExecutor

创建一个单线程的支持定时及周期性的任务执行的线程池。

|  |
| --- |
| 可定时或者延迟执行线程活动  线程池中最多有1个线程 |

1. js基础

**1、js概念**

Js全称javascript，是一种基于对象和事件驱动的具有安全性能的脚本语言

**2、js数据类型**

基本数据类型（3种）

字符串(string) "hello"、"学习" 'javascript'

数值(number) 18, 365.25, -12.7

布尔(boolean) true, false

复合数据类型（2种）

数组Array

对象 object

特殊数据类型（2种）

空 null

未定义 undefined

**3、js检测数据类型**

typeof() 检测类型。

**4、请写出js的5个内置函数并说明其作用**

isNaN() 判断一个值是否为数值,如果是数值返回false,否则返回true

Number() 把指定值转换为数字,如果无法转换返回 NaN

parseInt() 将指定值转为整数,如果无法转换返回 NaN

parseFloat() 将指定值转为小数,如果无法转换返回 NaN

eval() 将传入的字符串解析为JavaScript 代码并执行

**5、js中的内置对象**

--math对象(掌握5个)：

ceil（）向上取整 floor（）向下取整 abs（）绝对值

round（）四舍五入 random（）随机数 min() 最小值 max() 最大值

--string对象(掌握5个)：

indexOf（）查找子字符串在字符串中首次出现的位置

lastIndexOf() 查找子字符串在字符串中最后一次出现的位置

split（）将字符串分割为数组

replace（）替换字符串

substr（）按下标和个数截取字符串

substring（）按下标截取字符串

toLowerCase() 将字符串转为小写

toUpperCase() 将字符串转为大写

Replace()替换

--date对象(掌握5个)：

getFullyear（）返回指定日期的年份

getMonth（）返回指定日期的月份

getDay（）返回指定日期星期中的第几天（0-6）

getDate（）返回指定日期月份中的第几天（1-31）

getHours() 返回指定日期对象的小时数

getMinutes() 返回指定日期对象的分钟数

getSeconds() 返回指定日期对象的秒数

getMilliseconds() 返回指定日期对象的毫秒数

toLocalestring（）根据当前区域设置并转化为字符串返回

toString（）将日期转化为字符串返回

--array对象(掌握5个)：

concat（）将两个数组或多个数组拼接成一个新的数组

shift() 移除数组中第一个元素并返回该元素

unshifit() 向数组头部添加一个或多个元素，并返回数组的新长度

pop（）移除数组中最后一个元素并返回该元素

push（）向数组中添加一个元素并返回数组的长度

splice() 在任意位置添加或删除任意个元素

toString（）将数组以字符串的形式返回

join（）返回一个字符串，字符串是以指定的符号分隔

**6、DOM和BOM 对象**

--document:

getElementById() 返回对拥有指定 id 的第一个对象的引用。

getElementsByName() 返回带有指定名称的对象集合。

getElementsByTagName() 返回带有指定标签名的对象集合。

--window:

alert() 弹出警告框

confirm() 弹出确认框

prompt()弹出输入框

open() 打开新窗口

close() 关闭窗口

--history

forward() 加载历史列表中的下一个url，相当于浏览器前进按钮功能

back() 加载历史列表中的前一个url，相当于浏览器后退按钮的功能

go() 加载历史列表中某个具体页面

--location

href 设置或返回浏览器地址栏

reload() 重新加载当前页面，相当于浏览器刷新按钮的功能

**7、dom常用操作(掌握5个)**

getElementById() 通过id获取html元素

getElementsByName() 通过name获取html元素数组

getElementsByTagName() 通过标签名获取html元素数组

body 获取页面中body元素

innerHTML 设置或获取html元素html内容

childNodes 返回html元素的子节点数组

 firstChild 返回html元素的首个子节点

 lastChild 返回html元素的最后一个子节点。

 nextSibling 返回节点的下一个兄弟节点

 parentNode 返回节点的父节点

 previousSibling 返回节点的上一个兄弟节点

 appendChild() 向元素添加新的子节点，并作为最后一个子节点。

 cloneNode(是否深度克隆) 克隆元素

 insertBefore(新节点,已有子节点) 在指定的已有的子节点之前插入新节点。

 removeChild(已有子节点) 从元素中移除子节点

 replaceChild(新节点,旧节点) 替换元素中的子节点

**8、js全局变量和局部变量**

全局变量：声明在函数外(var可用可不用)一般为了规范加上var

局部变量：声明在函数内部(加上var是局部变量，不加var是全局变量)

**9、js事件**

**事件就是用户对网页或浏览器做出的动作.通常来说,鼠标和键盘的动作称之为事件。**

页面事件：

载入页面 onload

离开页面 onunload

鼠标事件:

单击事件 onclick

双击事件 ondblclick

鼠标按下事件 onmousedown

鼠标抬起事件 onmouseup

鼠标移上事件 onmouseover【实战】

鼠标移开事件 onmouseout 【实战】

键盘事件：

键盘按下事件 onkeydown

键盘抬起事件 onkeyup

键盘按下并抬起事件 onkeypress

表单事件：

表单提交事件 onsubmit

表单重置事件 onreset

获得焦点事件 onfocus

失去焦点事件 onblur

内容改变事件 onchange

**10、JS正则表达式**

元字符:元字符是具有特殊含义的字符

\d 匹配一个数字字符，等价于[0-9]

\w 匹配一个大小写字母、数字、下划线，等价于[0-9a-zA-Z\_]。

\s 匹配一个空白字符(空格、制表符(tab)、换行符(\n))

.匹配除\n之外的任意一个字符

\D 匹配一个非数字字符,等价于[^0-9]

\S 匹配一个非空白字符(非空格、非制表符(tab)、非换行符(\n))

\W 匹配一个非大小写字母、非数字、非下划线，等价于[^0-9a-zA-Z\_]

边界符：

^ 匹配字符串的起始位置

$ 匹配字符串的结束位置

量词

? 匹配前面的子表达式0次或1次

+ 匹配前面的子表达式1次或多次

\* 匹配前面的子表达式0次或多次

{n} 匹配确定的n(非负整数)次

{n,} 至少匹配n(非负整数)次，最多匹配无数次

{n,m} 至少匹配n次，最多匹配m次(n必须小于等于m)

修饰符

g 全局匹配

m 多行匹配

i 忽略大小写匹配

**11、oracle数据类型**

字符类型：

varchar2:最多4000字节,可变长度,最为常用

nvarchar2:最多2000个字符，可变长度，内部实现存在问题不常用

char:最多2000字节，固定长度，长度不够以空格填充 大文本数据类型

数值类型：

number(总长度,小数点后的位数) 最大精度38位

(小数点后的位数:小数位置四舍五入)

日期类型：

date (年月日 时分秒)

通常情况下如果是GBK编码则一个汉字对应2个字节，如果是UTF-8编码则一个汉字对应3个字节。

**12、Oracle的5个聚合函数 ，并说明用途**

-oracle的聚合函数

--count（）统计数量

--max（）最大值

--min（）最小值

--avg（）平均值

--sum（）总和

**13、主键外键**

主键--唯一标识一条记录，不能有重复的，不允许为空

primary key

constraint pk\_id primary key(字段名)

外键--表的外键是另一表的主键, 外键可以有重复的, 可以是空值

constraint 外键名 foreign key(字段名) references 主表(主键字段名)

**14、oracle常用函数函数**

 --数学函数

abs(字段名|值) 绝对值

ceil(字段名|值) 向上取整

floor(字段名|值 ) 向下取整

trunc(字段名|值) 截断

round(字段名|值) 四舍五入

 --字符串函数

concat(字段名|值, 字段名|值) 字符串拼接

initcap(字段名|值) 将首字母转为大写，剩余的全转为小写

instr(字段名|值,子字符串[,起始位置[,第几次出现]]) 按照起始位置和指定次数查找子字符串在字符串中出现的位置

length(字段名|值) 字符串长度

lower(字段名|值) 转为小写

upper(字段名|值) 转为大写

rpad|lpad(字段名|值,字符串总长度,填充的字符串) 左填充和右填充

rtrim|ltrim (字段名|值,要删除的字符串) 左去除|右去除

substr(字段名|值,起始位置,个数) 按照起始位置和个数截取字符串

replace(字段名|值,s1,s2) 替换字符串

reverse(字段名|值) 字符串反转，这里反转的是字节不是字符

--转换函数

to\_char(值|字段名[,格式] ) 将日期或数字转为字符串

to\_date(值|字段[,格式] ) 将字符串转为日期

 --日期和时间函数:

add\_months 增加或减去月份

last\_day 返回日期所在月的最后一天

next\_day(date,day) 给出日期date和星期day之后计算下一个星期的日期

 --通用函数:

nvl(字段名|值,默认值) 如果指定值为空则显示默认值，否则显示指定值

nvl2(字段名|值,默认值1,默认值2) 如果指定值为空则显示默认值2，否则显示默认值1

 decode(字段名|值,条件1,值1,条件2,值2,…,默认值)

如果指定值等于条件1则显示值1，

如果指定值等于条件2则显示值2，

…

如果指定值不等于任何条件则显示默认值

**15、Oracle分页：**

--假设现在数据库里有839条记录，页面上每页显示37条，现在查出第13页要显示的数据

select \* from (

select rownum rn,t.\* from (

select \* from t\_student

) t where rownum <= (13\*37)--页数\*每页显示条数

) where rn > (13-1)\*37--(页数-1)\*每页显示条数;

**16、DML数据操作语言**

--增加

insert into 表名 (字段1,字段2,字段3,...) values (值1,值2,值3,...);

insert into 表名 values (值1,值2,值3....);

--删除

delete from 表名 [where子句];

--修改

update 表名 set 字段1=值1,字段2=值2,… [where子句];

**17、序列**

--创建序列语法

create sequence 序列名

minvalue 最小值

maxvalue 最大值

start with 起始值

increment by 增量(每次递增的值)

cache 缓存的序列值的个数

order;

--序列名.nextval 获取下一个值

--序列名.currval 获取当前值

**18、两表联查**

内连接

内连接 inner join 只显示两张表有关联的数据

select 字段名,字段名 from 表1 inner join 表2 on 表1和表2的关联条件

select \* from t\_student s inner join t\_class c on s.classid=c.id;

左连接

左连接 left join 会将左边表的全部数据进行展示，另外一张表有和它关联的就展示，没有就展示为null

select 字段名,字段名 from 表1 left join 表2 on 表1和表2的关联条件

select \* from t\_student left join t\_class on t\_class.id=t\_student.classid;

右连接

右连接 right join 会将右边表的全部数据进行展示，另外一张表有和它关联的就展示，没有就展示为null

select 字段名,字段名 from 表1 right join 表2 on 表1和表2的关联条件

**19、JDBC操作数据库的实现步骤及其主要代码**

--导jar包

--加载驱动为数据库连接做准备 关键字：Class.forName

--连接数据库 关键字：DriverManager.getConnection()

--写sql语句

--创建数据库操作对象 关键字：preparestatement

--执行sql语句 关键字：executeupdate()/executequery()

--关闭连接释放资源 关键字：.close()

**20、端口**

tomcat默认端口8080

oracle默认端口 1521

**21、Servlet的生命周期**

1.实例化：由web服务器进行servlet的实例化

2.初始化：使用init()方法进行初始化

3.提供服务：使用service()方法提供服务。（具体使用doGet()和doPost()方法提供服务）

4.销毁：使用destroy()方法进行销毁

5.不可用：标记为垃圾回收

**22、jsp运行原理**

--jsp的运行原理:

1、用户发送一个请求---->服务器(tomcat)

2、服务器根据请求的url(地址)----->找到相应的jsp文件

3、将找到的jsp文件转译成java源文件(servlet)

4、将转译出来的java源文件进行编译，得到一个可执行文件(.class)

5、执行.class文件并且告诉服务器(tomcat)

6、服务器返回一个响应信息------>用户

**23、Jsp 9大隐式对象。**

**输入输出对象**：

request 对应的实体类是(HttpServletRequest)

获取请求的信息(这是用户提交上来的信息)

同时还是作用域通讯对象

在当前请求中，一直有效

response 对应的实体类是(HttpServletResponse)

响应用户的信息(这是要返回给用户的信息)

out 对应的实体类是(JspWriter)

输出对象，一般输出一些调试信息

out.println(); 向页面输出一些信息

System.out.println(); 向控制台输出一些信息

**作用域通讯对象**：

application 对应的实体类是(ServletContext)

只要服务器不关闭，一直有效

session 对应的实体类是(HttpSession)

只要当前会话不关，一直有效

pageContext 对应的实体类是(PageContext)

只在本页面上有用(离开里本页面就没用了)

request

在当前请求中，一直有效

**servlet对象：**

page 对应的实体类是(Page)

相当于new了一个实例，类似于java中的this关键字

config 对应的实体类是(ServletConfig)

可以获取servlet初始化的一些信息，不太常用

**错误对象：**

exception 对应的实体类是(Throwable)

专门处理错误

**24、Jsp的四个作用域、作用域范围、以及对应的作用域对象**

page 当前页面 pageContext

request 当前请求 可以跨越多个页面 request

session 当前会话 多次请求 session

application 整个项目 整个服务应用 application

**25、Redirect和forward的区别**

转发和重定向的区别；

redirect重定向 特点：

地址栏会发生变化

数据不共享(用户请求信息不会带到新页面)

执行效率低，好几次请求

写法：

response.sendRedirect(目标文件地址);

forward转发 特点：

地址栏不会发生变化

数据共享(用户请求信息会被带到新页面)

执行效率高，只有一次请求

写法：

request.getRequestDispatcher(目标文件地址).forward(request, response);

26、**get和post的区别：**

1.get是一般用于从服务器上获取数据， post一般用于向服务器传送数据。

2.get提交的数据会存放在URL之后,以？分割url和传输数据，参数之间以&相连。get对数据不安全，post相对安全

3.get提交的数据大小有限制，最多只能有1024字节(因为浏览器对url长度有限制)而post提交没有限制。

4.通常用post做文件上传而get是不支持的。