目录

[一.core java 3](#_Toc21423464)

[1.1初识java 3](#_Toc21423465)

[1.1.1简介 3](#_Toc21423466)

[1.1.2概念 4](#_Toc21423467)

[1.1.3环境搭建 4](#_Toc21423468)

[1.1.4java文件 5](#_Toc21423469)

[1.1.5基本语法规范 5](#_Toc21423470)

[1.2基本语法 6](#_Toc21423471)

[1.2.1变量 6](#_Toc21423472)

[1.2.2分支结构 9](#_Toc21423473)

[1.2.3循环结构 9](#_Toc21423474)

[1.2.4函数 10](#_Toc21423475)

[1.2.5数组 10](#_Toc21423476)

[1.2.6排序 11](#_Toc21423477)

[1.3面向对象 12](#_Toc21423478)

[1.3.1对象 12](#_Toc21423479)

[1.3.2三大特性 14](#_Toc21423480)

[1.3.3一些特别的类 16](#_Toc21423481)

[1.4异常和线程 23](#_Toc21423482)

[1.4.1异常 23](#_Toc21423483)

[1.4.2线程 25](#_Toc21423484)

[1.5 IO及反射 28](#_Toc21423485)

[1.5.1流 28](#_Toc21423486)

[1.5.2反射 32](#_Toc21423487)

[1.5.3设计模式 33](#_Toc21423488)

[二.web开发 34](#_Toc21423489)

[2.1html 34](#_Toc21423490)

[2.2css 36](#_Toc21423491)

[2.3js 37](#_Toc21423492)

[2.4jdbc 42](#_Toc21423493)

[2.5jQuery 44](#_Toc21423494)

[2.6JSP 48](#_Toc21423495)

[2.7mybatis 52](#_Toc21423496)

[2.8oracle 55](#_Toc21423497)

[2.9servlet 61](#_Toc21423498)

[2.10Struts2 64](#_Toc21423499)

[三.最基础的企业开发 70](#_Toc21423500)

[3.1ajax 70](#_Toc21423501)

[3.2MySQL 71](#_Toc21423502)

[3.3bootstrap 72](#_Toc21423503)

[3.4ElasticSearch 72](#_Toc21423504)

[3.5layui 74](#_Toc21423505)

[3.6linux 75](#_Toc21423506)

[3.7redis 79](#_Toc21423507)

[3.8spring 82](#_Toc21423508)

[3.9springboot 88](#_Toc21423509)

[3.10SpringMVC 91](#_Toc21423510)

[四、扩展功能 94](#_Toc21423511)

[4.1项目开发流程 94](#_Toc21423512)

[4.2糊涂工具类 95](#_Toc21423513)

[4.3layui文件上传 95](#_Toc21423514)

[4.4POI 96](#_Toc21423515)

[4.5easypoi 96](#_Toc21423516)

[4.6echarts 97](#_Toc21423517)

[4.7日志 97](#_Toc21423518)

[4.8git 98](#_Toc21423519)

[4.9接口开发 100](#_Toc21423520)

[4.10shiro权限管理 101](#_Toc21423521)

[4.11springCloud 102](#_Toc21423522)

[五、知识补充（详情见资料） 105](#_Toc21423523)

[5.1mysql优化 105](#_Toc21423524)

# 一.core java

## 1.1初识java

### 1.1.1简介

Java是一门[面向对象](https://baike.baidu.com/item/%E9%9D%A2%E5%90%91%E5%AF%B9%E8%B1%A1" \t "_blank)编程语言，不仅吸收了[C++](https://baike.baidu.com/item/C%2B%2B" \t "_blank)语言的各种优点，还摒弃了C++里难以理解的[多继承](https://baike.baidu.com/item/%E5%A4%9A%E7%BB%A7%E6%89%BF" \t "_blank)、[指针](https://baike.baidu.com/item/%E6%8C%87%E9%92%88/2878304" \t "_blank)等概念，因此Java语言具有功能强大和简单易用两个特征。Java语言作为静态面向对象编程语言的代表，极好地实现了面向对象理论，允许程序员以优雅的思维方式进行复杂的编程[1]  。

Java具有简单性、面向对象、[分布式](https://baike.baidu.com/item/%E5%88%86%E5%B8%83%E5%BC%8F/19276232" \t "_blank)、[健壮性](https://baike.baidu.com/item/%E5%81%A5%E5%A3%AE%E6%80%A7/4430133" \t "_blank)、[安全性](https://baike.baidu.com/item/%E5%AE%89%E5%85%A8%E6%80%A7/7664678" \t "_blank)、平台独立与可移植性、[多线程](https://baike.baidu.com/item/%E5%A4%9A%E7%BA%BF%E7%A8%8B/1190404" \t "_blank)、动态性等特点[2]  。Java可以编写[桌面应用程序](https://baike.baidu.com/item/%E6%A1%8C%E9%9D%A2%E5%BA%94%E7%94%A8%E7%A8%8B%E5%BA%8F/2331979" \t "_blank)、[Web应用程序](https://baike.baidu.com/item/Web%E5%BA%94%E7%94%A8%E7%A8%8B%E5%BA%8F" \t "_blank)、[分布式系统](https://baike.baidu.com/item/%E5%88%86%E5%B8%83%E5%BC%8F%E7%B3%BB%E7%BB%9F/4905336" \t "_blank)和[嵌入式系统](https://baike.baidu.com/item/%E5%B5%8C%E5%85%A5%E5%BC%8F%E7%B3%BB%E7%BB%9F/186978" \t "_blank)应用程序等[3]  。《百度百科》

### 1.1.2概念

1.运行机制

先编译，后解释

2.特点

①跨平台（操作系统、服务器、数据库）

②简单（相对C++）

③面向对象

3.专有名词

JVM：虚拟机

JRE：虚拟机+解释器

JDK：JRE+编译器+工具+类库（运行环境）

### 1.1.3环境搭建

1.JDK的安装

2.配置环境变量

JAVA\_HOME:JDK安装路径

PATH:JDK安装路径\bin

CLASSPATH:.（表示当前路径）

3.DOS命令

windows+r :呼唤dos窗口

cmd ：打开命令窗口

java –version :查看JDK版本号

F：切换到F盘

cd 文件夹名字：进入某个文件夹

dir：查看当前目录下内容

del 文件名.后缀名：删除文件

rd 文件夹名字：删除文件夹

cd..:返回上级目录

cls：清屏

### 1.1.4java文件

1. 以.java结尾的文件（编译完成后会生成.class字节码文件）
2. 类：代码容器

①公开类 public class 类名（公开类名与生成的文件名一致）

②内部类 class 类名

③匿名内部类 作为方法的参数直接定义使用的类

1. 主函数：程序的入口（每个类只能有一个主函数）
2. 黑窗口中使用javac+文件名.java编译，java+类名运行
3. 包：package 包名。存在于代码第一行，相当于用来管理文件的文件夹

带包运行：javac –d . 文件名.java java 包名.类（全限定名）

### 1.1.5基本语法规范

1. 层次分明
2. 注释清晰
3. 命名要求

①标识符由数字、字母、\_、$组成，数字不能开头

②不能使用关键字和保留字（基本内容见资料）

③望文生义，见名知意

④类名首字母大写，方法名首字母小写，后面单词首字母大写；包名全小写；常量名全大写，多个单词间使用\_隔开

## 1.2基本语法

### 1.2.1变量

变量是内存中的一块存储空间，是保存数据的基本单位（java是强类型语言），作用范围从定义开始到所在的代码块结束。

成员变量：在类中定义的变量

局部变量：在方法内部或静态代码块中定义的变量

常量：被final修饰的变量

1. 声明赋值：数据类型 变量名 = 变量值
2. 特点：同范围内不允许出现相同的变量名，变量必须赋值后使用
3. 数据类型

基本类型：

①整数类型：

byte 1B -2^7 -- 2^7-1 -128 -- 127

short 2B -2^15 -- 2^15-1

int 4B -2^31 -- 2^31-1

long 8B -2^63 -- 2^63-1

注：long类型定义时需要在值后加L

②浮点类型：

float 4B 单精度浮点数

double 8B 双精度浮点数

注：float类型定义时需要在值后加F

③字符类型：

char 2B 无负数 0 – 65535 Unicode（万国码，支持ASCII）

（见资料ASCII码值）

④布尔类型：boolean true/false

引用数据类型：

⑤字符串类型：String

1. 基本类型转换

自动类型提升：目标类型大于原类型

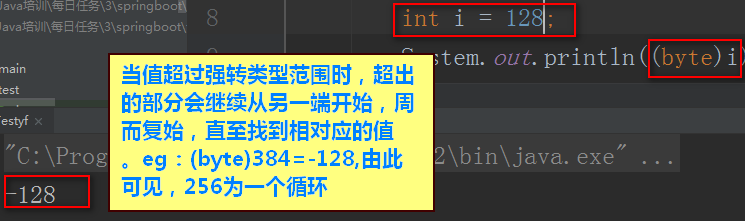
强制类型装换：目标类型小于原类型

强转语法：目标类型 变量名 = （目标类型） 强转前的值

特点：

①如果值符合目标类型长度，值不变

②如果不符合目标类型长度，值改变（超过部分从另一端开始继续累计）



③浮点数可以转整数，小数位舍弃，保留整数

④字符可以转int（详见ASCII码）

⑤boolean不允许强转

表达式：由运算符连接的变量或者字面值，并且得到一个结果的式子

运算符：

①算数运算符：+-\*/%

②赋值运算符：= += -= \*= /= %=

③比较运算符：> < >= <= == !=

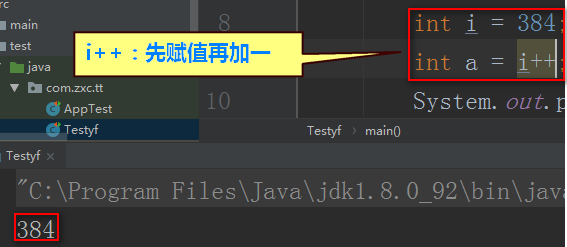
④逻辑运算符：&& || ！

⑤一元运算符：++ --

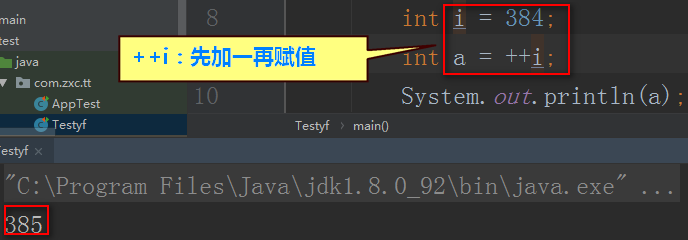
⑥三元运算符（三目运算符）：boolean？值1：值2

注：byte和short进行运算，结果为int；

a++：先运算再加一



++a：先加一在运算



1. 扫描器

Scanner sc = new Scanner(System.in);

int a = sc.nextInt();

char c = sc.next().charAt(0);

String s = sc.next();

1. 导包：import 类全限定名

静态导入：可以直接导入静态的方法全限定名，再类中直接使用

import static 方法全限定名

1. 转义字符：

\n:换行；\t:跳格；\’:单引号；\”:双引号；[\\:反斜杠](file:///\\\\:反斜杠)

### 1.2.2分支结构

1. if(){};if…else..;if…else if…..else…;
2. switch(值){

case 值1：代码；break；

case值2：代码；break；

default：代码；

}

break：结束并跳出结构

default：默认匹配，一般放在最后，放在前面需要加break（后面的就没用了）；

如果两个case执行的代码相同，则前面的可以不用写代码和break；

值可以是整数或者字符串

### 1.2.3循环结构

1. while（布尔表达式）{代码块}先判断再执行 不确定循环
2. do{代码块}while（布尔表达式）先执行再判断 不确定循环
3. for（开始条件；结束条件；迭代变量）{代码块}结束条件为真时执行 确定循环

注：break：强制结束并跳出循环

continue：跳过本次循环，执行下次循环

### 1.2.4函数

也叫方法，一段可以反复使用的完成特定功能的代码，存在类之内，其他函数之外

1. 语法：修饰符 返回值 函数名（参数列表）【异常】{代码块}
2. 形参：参数列表定义的参数
3. 实参：调用者传入的参数
4. 返回值：void不需要返回，其他需要return，分支需要多个返回
5. 调用：

①静态调用：静态方法可以通过类名.方法名直接调用

②非静态函数需要对象才能调用

③函数嵌套：函数内部调用另一个函数

④函数递归：自己调用自己，需要有出口

1. 作用：提高代码维护性，减少代码冗余，提高代码灵活性，方便分工合作

### 1.2.5数组

内存中一段连续的存储空间，可以存储多个相同数据类型的数据

声明：数组类型[] 变量名 = new 数组类型[数组长度]

赋值：数组类型[] 变量名 = {元素1，元素2，。。。}；

数组类型[] 变量名 = new 数组类型[]{元素1，元素2，。。。}；

变量名[下标] = 值；

取值：数据类型 变量名 = 数组名[下标]；

遍历：for（int i=0；i<数组名.length;i++）{数据类型 变量名 = 数组名[下标]；}

异常：java.lang.ArrayIndexOutOfBoundsException(下标越界异常，访问无效的下标时触发)

默认值：整数、浮点型：0/0.0；字符型：‘\u000’；布尔类型：false；对象类型：null；

内存存储原理：

①内存空间连续；

②数组类型的变量存储的是数组的首地址；

③数组元素的地址：首地址+下标\*数据类型字节数；

④下标从0开始，提高寻址效率；

⑤数组类型变量间传递的是数组的首地址

特点：寻址效率高，查询快，传递的是地址，可以作为参数和返回值，做最后一位参数时相当于可变长参数（参数类型。。。【参数个数不限，数组形式，只能放在形参列表的最后】）

数组扩容：

①新建数组，长度大于原数组，循环将原数组赋值给新数组

②java.util.Arrays.copyOf(原数组，新数组长度)通过此方法完成扩容，返回值是扩容后的新数组

③System.arraycopy(原数组，起始下标，新数组，存储元素的起始下标，长度)；此方法将元素中的从起始下标开始对应的元素一一赋值给新数组中存储的起始下标开始。

### 1.2.6排序

1. 冒泡排序（两两相比，大着后移）

for(int i=0;i<a.length-1;i++){

for(int j=0;j<a.length-1-i;j++){

if(a[j]>a[j+1]){

int temp = a[j];

a[j]=a[j+1];

a[j+1]=temp;

}

}

}

1. 选择排序（一挑七，弱者上位）

for(int i=0;i<a.length-1;i++){

for(int j=i+1;j<a.length;j++){

if(a[i]>a[j]){

int temp = a[i];

a[i]=a[j];

a[j]=temp;

}

}

}

1. 快速排序（通往成功的捷径，没错！就是Arrays.sort）

java.util.Arrays.sort(a);

print(a)

此方法可以直接对数组进行升序排序（从小到大）

## 1.3面向对象

### 1.3.1对象

客观事物在计算机语言中的定义，在java中一般和类相关联，类是对象的模板，对象是类的实例，有类才能创建对象。

1. 类：由属性和方法【代码块】组成
2. 属性：成员变量（静态部分），定义在类内部，方法外部，具有默认值。与局部变量相比，定义位置、作用范围、默认值和命名冲突规则都不同。当成员变量与局部变量命名冲突时，局部变量优先。
3. 方法：

①成员方法：在类之内，其他方法之外的方法

②构造方法：用于创建对象的一个特殊的方法，public 类名（）{}，此方法可以给属性赋值可以重载，不写的时候默认无参构造。

③方法重载（overload）：同一类中名字相同，参数列表不同的方法，跟返回值无关。作用：提高代码灵活性，屏蔽差异

④方法覆盖（override）:又叫方法重写，对同一方法进行重写，一般用于继承或者实现的类中。要求：①方法名、返回值、参数列表完全一致②访问修饰符相同或更宽③子类中方法不能抛出比父类更多、更宽的异常

1. 对象的创建：

类名 对象名 = new 类名（）；

使用属性：对象名.属性名（公开的属性才能用，一般调用其get/set方法）；

使用方法：对象名.方法名（参数）；

创建过程：①在内存中分配空间，为属性赋默认值；②为属性赋初始值；③构造方法赋值。（3次赋值机会）

5.this关键字：代表当前对象

this.属性名 表示当前对象的属性，用此可以区分局部变量与成员变量；

this.方法名（参数） 表示调用当前对象的方法 用在构造方法或成员方法中；

this（） 调用无参构造；

this（实参） 调用有参构造，只能定义在构造方法内有效代码第一行，避免构

造方法的递归调用

6.引用：对象类型的变量称为引用

①引用中存储的是对象在堆空间的首地址

②对象在对空间中相互独立

③引用之间相互赋值，赋的是首地址

④引用如果为空，会报空指针异常

⑤引用可以用于参数、返回值和数组

### 1.3.2三大特性

1. 封装 ：把数据和方法包装进类中，以及具体的实现的隐藏，常使用修饰符来完成

实现：所有属性私有化，提供get/set方法

①修饰符：可以修饰方法，构造方法和属性，只有default和public可以修饰类

public 公开的 全都能用

private 私有的 本类使用

protected 受保护的 子类可用

default 默认的 同包子类

②set/get方法

public void set属性名（首字母大写）（属性类型 变量）{this.属性名 = 变量名}

public 属性类型 get属性名（首字母大写）{return this.属性名}

作用：为属性赋值、获取属性值

1. 继承 ：是类与类之间一种机制，通过继承机制，子类可以使用父类中的属性和方法（构造方法和私有属性不可被继承），也可以对父类的方法进行覆盖，覆盖之后，子类方法优先使用，继承通过extends关键字实现。

作用：提高代码复用性和扩展性

特点：单继承，可以多级继承，属性和方法进行逐级叠加

对象创建过程：继承关系下，构建子类对象会先构建父类对象

①分配空间（父类+子类）②构建父类对象（初始化父类属性+调用父类构造）③初始化子类属性调用子类构造

super关键字：代表父类对象

super（）用于子类构造方法中，调用父类的无参构造；

super（参数）调用父类有参构造

super.父类属性名 调用父类属性

super.父类方法名 调用父类中方法

1. 多态：父类类型的引用指向子类类型的对象

语法：父类类名 引用名 = new 子类类名（）；

引用可以调用父类中的方法，若子类中有方法覆盖，则调用子类中的方法

类型装换：

小类型转大类型可以直接赋值：父类类型 引用名 = 子类类型引用

大类型转小类型需要强制类型装换：子类类型 引用名 = （子类类型）父类类型引用名，类型一致时，编译运行通过，类型不一致时，编译通过，运行报错java.lang.ClassCastException(强制类型装换异常)

instanceof：判断引用类型与对应的类型是否一致，一致为true，不一致为false；

语法 引用名 instanceof 类名;

应用：①数组，可以存放父类类型及所有子类类型

②参数，可以是父类及所有子类类型

③返回值，可以使父类及子类类型

作用：屏蔽子类差异，提高代码灵活性

1. 三大修饰符

①abstract(抽象修饰符)

修饰类和方法，抽象类可以声明引用，不能创建对象，可以定义成员变量和方法，有构造方法，创建子类对象构建父类对象时使用；抽象方法只有声明，没有实现，只能定义在抽象类中。继承抽象类，要么实现里面的方法，要么也是抽象类（强制使用多态）

②static(静态的)

可以修饰属性、方法和代码块。静态属性和静态方法可以直接通过类名.属性/方法使用；静态方法中不能直接访问非静态成员，不能使用this和super，可以被静态覆盖，但没有多态；静态属性和静态代码块在类加载的时候就会执行，而且只执行一次。

类加载：当JVM第一次使用一个类时，会根据ClassPath找到对应的.class文件，然后加载这个类的所有信息并保存在内存中。一般在创建对象时进行，声明引用时不会进行类加载。

③final（最终的）

修饰变量、方法和类。被修饰的变量叫常量，一旦被赋值不能被改变；被修饰的方法不允许被覆盖；修饰的类不能被继承（String、Math、System）。

④禁忌：abstract不能和其他的修饰符组合使用

1. 接口：一种规范/标准/约定，使用者和实现者的约定，接口会生成对应的.class文件，但不能创建对象，没有构造方法，可以定义属性，但必须是公开静态常量；方法只有声明，没有实现，默认公开抽象。
   1. 创建接口：public interface 接口名（不是类）
   2. 实现：class 类名 implements 接口名{}，此类必须实现接口中的所有方法，实现的方法必须公开，否则此类为抽象类
   3. 接口之间可以多继承，类可以实现多接口，中间用逗号隔开，实现类中可以继承父类，继承必须写在实现前面
   4. 作用：扩充子类功能，解耦合
   5. 接口回调：先有接口的使用者，后有接口的实现者

### 1.3.3一些特别的类

1. Object类
   1. java.lang.Object 所有类的父类，可以存储任意类型对象，方法可以被任意类使用。
   2. 常用方法：
      1. getClass():返回引用中存储的实际类型
      2. hashCode():返回对象地址的整数表现形式
      3. toString():返回该对象的String表现形式，通常在自定义类中重写
      4. equals（Object o）：比较两个对象的内容是否相同，通常需要重写，规则：①自反性②非空判断③类型判断④强转⑤属性对比
      5. finalize():垃圾回收器回收对象时调用的方法
   3. 垃圾回收相关：
      1. 垃圾对象：没有引用指向的对象
      2. 垃圾回收机制：定期清理垃圾对象，释放内存
      3. 垃圾回收器（GC）
      4. 回收时机：①自动垃圾回收，当JVM内存耗尽时，一次性回收所有垃圾对象②通过System.gc()手动通知JVM做垃圾回收，此时若GC空闲就会去回收，否则不予理会
2. 内部类：在一个类的内部定义的类，会生成对应的.class文件，文件名是外部类名$内部类名.class，可以不破坏封装的访问外部类的私有属性。
   1. 成员内部类：类似于成员变量，创建对象需要借助外部类的对象（外部类.内部类 引用名 = 外部类的对象.new 内部类()）,此类中不能定义静态成员。
   2. 静态内部类：类似静态变量，被static修饰，依赖外部类创建对象（外部类.内部类 引用 = new 外部类.内部类（）），此类中只能访问外部类的静态成员
   3. 局部内部类：类似局部变量，只能在方法内部创建对象，可以访问外部类的私有属性和方法内的局部常量
   4. 匿名内部类：特殊的局部内部类，一般在创建对象是使用，必须继承一个类或者实现一个接口，覆盖或者实现类中的方法。代码量少，可读性低。
3. 包装类：为8中基本数据类型提高的引用数据类型
   1. char –> Character; int 🡪Integer
   2. 类型装换：
      1. int—》Integer Integer in = Integer.valueOf(a)/new Integer(a)
      2. String 🡪Integer Integer in = Integer.valueOf(s)/new Integer(s)
      3. Integer🡪String String s = in.toString()/in+” ”;
      4. int 🡪String Stirng s = 10+” ”;
      5. String 🡪int int a = Integer.parseInt(s);字符串必须为数字
   3. JDK5.0之后提供自动拆箱装箱功能，基本类型与包装类可自动转换。自动装箱时，默认调用的是valueOf（），当基本数据类型的值在-128~127之间时，会去缓冲区找创建好的Integer对象，作用：节省内存
   4. 作用：区分0和null，区分有效数据和无效数据。
4. String（字符串）
   1. 常用方法：
      1. boolean contains(String s):判断s是否存在字符串中
      2. int length():返回字符串长度
      3. String trim():去除字符串的前后空格
      4. String[] split(String s):根据s做拆分，返回拆分后的数组
      5. String toUpperCase():将字符串装换成全大写
      6. String toLowerCase():将字符串转换成全小写
      7. boolean equals(Object o):比较字符串内容是否相同
      8. bookean endWirh(Stirng s):判断字符串是否以指定字符串s结尾
   2. 创建字符串
      1. String s = “hello”;创建一个字符串对象，保存在串池中
      2. String s = new String(“hello”);产生两个字符串对象，一个放在堆空间中，一个放在串池中
      3. 内容不可变
   3. 串池：字符串被创建之后都会放入串池，串池内字面值形式的字符串对象共享。
   4. 可变长字符串：
      1. StringBuffer:jdk1.0 线程安全，效率慢
      2. StringBulider:jdk5.0 线程不安全，效率快
      3. 通过构造方法创建可变长参数，通过append（）方法进行字符串拼接
   5. String与可变长字符串的区别

String对象不可变，因此在进行任何内容上的修改时都会创建新的字符串对象，一旦修改操作太多就会造成大量的资源浪费。

可变长参数StringBuffer和StringBuilder在进行字符串拼接的时候不会创建新的对象，而是在源对象上修改，不同之处是StringBuffer线程安全，StringBuilder线程不安全，所以在进行字符串拼接的时候推荐使用StringBuffer或者StringBuilder。

1. 集合：一类对象，可以存储多个对象
   1. Conllection:集合的根接口，常用方法如下
      1. boolean add(E e):向集合添加一个元素
      2. boolean contains(Object o):判断集合中是否包含某个元素
      3. boolean remove(Object o):移除某一元素
      4. int size():返回集合中的有效元素个数
   2. List接口：有序有下标，元素可重复
      1. 常用方法
         1. void add(int index,E element):向指定位置处添加元素
         2. Object get(int index):根据下标获取对应的元素
         3. int indexOf(Object o):返回指定元素对应的第一次出现的下标，没有则返回-1
         4. E remove(int index):移除指定下标对应的元素
         5. E set(int index,E element):向指定位置处添加元素并且覆盖原来的元素，更改
      2. 实现类
         1. ArrayList:

数组实现，查询快，增删慢，线程不安全，轻量级实现，运行效率高，jdk1.2

* + - 1. LinkedList:

链表实现，查询慢，增删快

* + - 1. Vector:

数组实现，查询快，增删慢，线程安全，重量级实现，运行效率低，jdk1.0

* 1. Set接口：无序无下标，元素不可重复
     1. 常用方法：从Collection中继承
        1. boolean add(E e):向集合中添加一个元素
        2. boolean contains(Object o):判断集合中是否包含某个元素
        3. boolean remove(Object o):移除某一元素
        4. int size():返回集合中的有效元素个数
     2. 实现类
        1. HashSet:

原理：基于hashCode方法进行，需要重写hashCode方法，将属性的内容转换成int类型进行返回，当hashCode值相同时，去调用equlas方法比较内容是否相同。数组实现，jdk1.2

* + - 1. LinkedHashSet：数组+链表实现，元素遍历的顺序与添加的顺序相同，jdk1.2
      2. TreeSet：红黑树，元素的遍历顺序按升序排列，jdk1.2
  1. Map集合：元素由键值对组成，key不可重复，value可以重复
     1. 常用方法
        1. V put(K key,V value):向集合中添加或覆盖一个元素，覆盖返回之前的value
        2. V remove(Object o):根据key删除集合中元素，并且返回删除的值
        3. size():获取集合中有效的键值对个数
        4. boolean containsKey(Object key):如果此集合包含指定键所对应的键值对，返回true
        5. boolean containsValue(Object value):如果集合中包含对应的值，返回true
     2. 实现类
        1. HashMap：线程不安全，轻量级实现，运行效率高，允许key和value为null值，自定义类型作为key时，必须重写hashCode和equals方法，jdk1.2

补充：底层数组+链表实现，初始size为16，扩容规则翻倍，即长度一定为2的n次幂，每次扩容元素依次重新计算存放位置，并重新插入，插入元素后才判断是否扩容，有可能无效扩容（插入后扩容，如果没有再次插入，就会产生无效扩容），当Map中元素总数超过Entry数组的75%时触发扩容操作，目的是减少链表长度，使元素分配更均匀，index = hash & (tab.length-1)

* + - 1. LinkedHashMap：自定义的类型作为key时，必须重写HashCode和equals方法，保证遍历的顺序跟添加的顺序一致,jdk1.2
      2. TreeMap：保证遍历的顺序为Key升序排列后的结果，自定义的类型作为key时，需要实现Comparable接口，以及实现compareTo方法，jdk1.2
      3. Properties：键和值都必须是String类型，通过用来读取配置文件，jdk1.0
      4. HashTable：线程安全，重量级实现，运行效率低，不允许key和value为null，jdk1.0

补充：底层数组+链表实现，修改数据时锁住整个HashTable，初始size是11，扩容规则：原容量翻倍+1；index=（hash & 0x7FFFFFFF）% tab.length，可以通过get（key）判断key是否存在

* + - 1. ConcurrentHashMap：底层采用分段的数组+链表实现，线程安全 ，通过把整个Map分为N个Segment,可以提供相同的线程安全，使效率提升N倍，默认提升16倍（读操作不加锁，由于HashEntry的value变量是volatile的，也能保证读取到最新的值），锁分离技术，允许多个修改操作并发进行，size()和containsValue()需要跨段，可能需要锁定整个表，即按顺序锁定所有的段，操作完毕后释放所有段的锁，扩容规则：段内扩容（段内元素超过该段对应Entry数组长度的75%触发扩容，不会对整个Map扩容）插入前会检测需不需要扩容，有效避免无效扩容；默认将hash表分为16个桶，get、put、remove等常用操作只锁住当前需要用到的桶。
      2. Hashtable和HashMap都实现了Map接口，但Hashtable的实现时基于Dictionary抽象类的，java5提供了ConcurrentHashMap,它是Hashtable的替代，比Hashtable的扩展性更好
      3. 锁分段技术：首先将数据分成一段一段的存储，然后给每一段数据陪一把锁，当一个线程占用锁访问其中一个段数据的时候，其他段的数据也能被其他线程访问
    1. 遍历方式
       1. 键遍历：通过键key获取值value

Set<Integer> set = map.keySet();

for(Integer key:set){map.get(key)}

* + - 1. 值遍历：只能获取值value

Collection<String> values = map.values();

for(String value:values){}

* + - 1. 键值对遍历：通过Map的内置对象获取键和值

Set<Map.Entry<Integer,String>> set = map.entrySet();

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

Integer key = entry.getKey();

String value = entry.getValue();

}

* 1. conllection集合遍历
     1. 下标遍历（list集合）

for(int i=0;i<list.size();i++){list.get(i)}

* + 1. for-each遍历（set集合）

for(元素类型 变量名：集合名/数组名){变量名即元素}

底层实现：iterator遍历，Iterator id = set.iterator();

while(it.hasNext()){it.next();}

* 1. 利用泛型解决集合不安全问题
     1. 泛型：参数化类型，通常使用单个大写字母来表示，可以用在接口、类和方法上面，指任意类型
     2. 泛型接口：interface IA<T>{void add(T t);T get();}
     3. 泛型类：class A<T>{}
     4. 泛型集合：List<String> list = new ArrayList<String>();jdk1.7后后面的泛型可以省略
        1. 定义泛型后，集合中的元素类型必须一致
        2. 引用中的泛型类型必须和实际对象的泛型类型保持一致
        3. 不同泛型的集合之间不能相互赋值
        4. 泛型只能是对象类型

## 1.4异常和线程

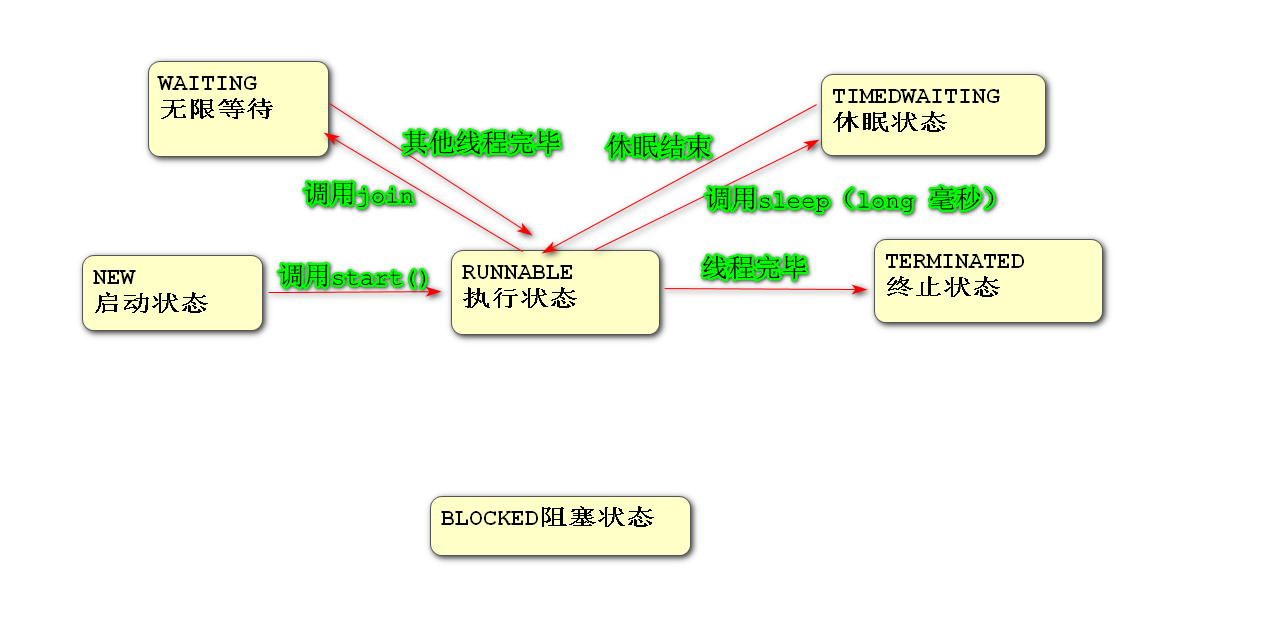
### 1.4.1异常

异常：程序运行过程中产生的不正常的情况

1. 分类
   1. java.lang.Throwable,异常的父类，常用方法：①getMessage()返回String类型的异常信息，②printStackTrace()打印跟踪方法调用栈而获得的详细异常信息
   2. Error：Throwable的子类，严重错误，程序本身无法解决的异常，例如栈溢出异常java.lang.StackOverflowError
   3. Exception：Throwable的子类，又分为以下两种
      1. RuntimeException：运行时异常，未检查异常。编译器不会对其进行检查，运行时会报错，此类异常可以处理也可以不处理，可以避免。常见运行时异常：①java.lang.ArrayIndexOutOfBoundsException下标越界异常②java.lang.ClassCastException强制类型装换异常③java.lang.NullPointerExcepeion空指针异常④java.lang.NumberFormatException数字格式转换异常
      2. 非RuntimeException：非运行时异常，已检查异常，编译器会对其进行检查，编译时会报错，此类异常必须处理。
2. 异常的产生
   1. 自动产生：当程序遇到错误代码，会产生异常，程序终止
   2. 手动产生：throw new 异常类名(),相当于return关键字，结束整个程序的执行，必须定义在方法内
3. 异常处理
   1. 作用：增强用户体验，减少不必要的损失
   2. 消极处理：通过throws对异常进行上抛，允许编译通过，当上抛的异常为非运行时异常时，方法的调用者必须处理，运行时异常可以处理也可以不处理
   3. 积极处理：使用try-catch对异常进行捕获处理。try后的catch代码块只能匹配成功一个；catch后声明的异常为父类时，可以捕获它本身及其所有子类异常（多态）,所以catch代码块捕获异常时，子类异常必须定义在父类异常前面；finally代码块中的代码是必须执行的代码，不论是否产生异常，其中不要定义return语句，一般用于关闭资源
4. 自定义异常
   1. 自定义异常类需要继承Exception或者其子类，运行时异常可以继承RuntimeException或者其子类
   2. 要求：需要有两个构造方法，一个无参，一个有参，参数为String message

### 1.4.2线程

1. 概念：①进程，操作系统中的任务，微观串行，宏观并行②线程，一个进程中并发的多个流程，JVM安装在操作系统上，java程序可以实现多线程；微观串行，宏观并行
2. 线程的组成：①CPU时间片 由os系统分配②内存空间 栈空间独立，堆空间共享（栈空间 存储局部变量，堆空间 存储实际对象）③代码
3. 线程的创建：在java中有一个Thread类代表线程，使用start()方法开启线程，JVM会调用该线程的run()方法。
   1. 方法一：继承Thread类，重写run方法（三无方法）
   2. 方法二：实现Runnable接口，重写run方法（三无方法）
   3. 方法三：实现Callable接口及其call()方法,该方法类似方法二，但是call方法提供有返回值和抛出异常,返回值为Future<V>类型，返回异步计算的结果
4. 线程的状态
   1. NEW：至今尚未启动的线程
   2. RUNNABLE：正在java虚拟机中执行的线程处于这种状态
   3. BLOCKED：阻塞状态
   4. WAITING：等待另一个线程执行完毕，再执行此线程。比如线程中的join()方法，允许其他线程加入当前线程，并且当其他线程执行完毕后才会继续执行当前线程
   5. TIMED\_WAITING：线程休眠状态。sleep(long 毫秒)，进入休眠状态
   6. TERMINATED：已退出的线程处于这种状态
   7. 状态关系如下图



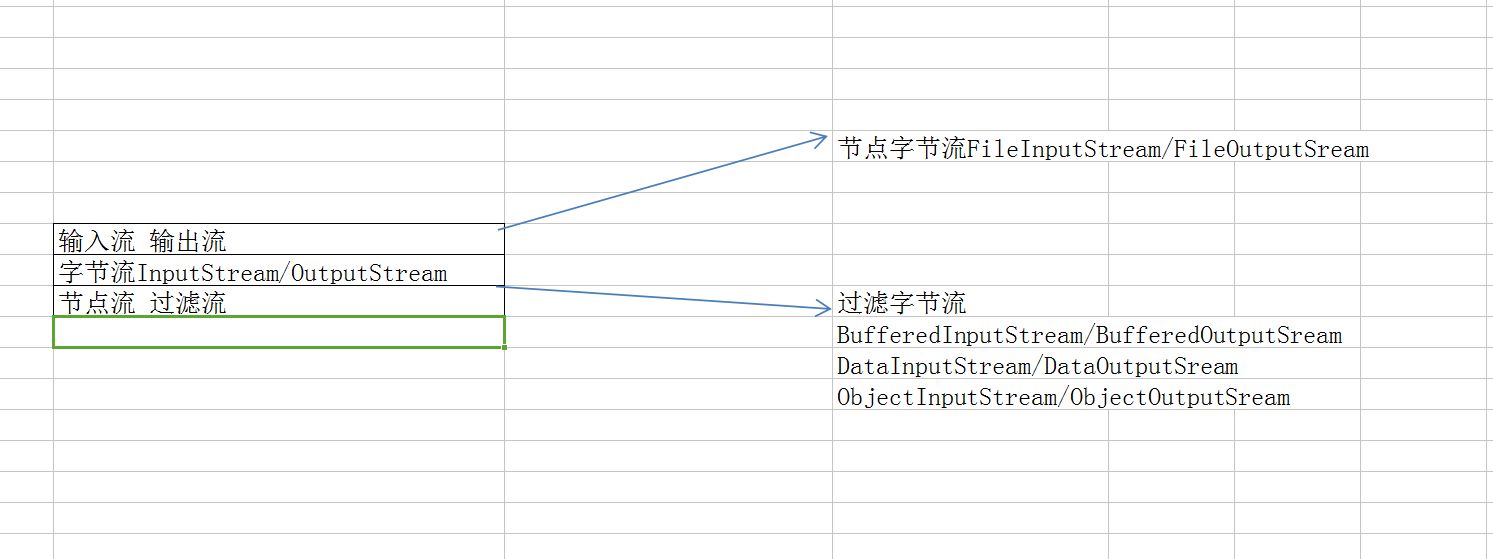
1. 多线程作用
   1. 提高硬件的利用率
   2. 提高用户的体验度
   3. 模拟现实世界
2. 线程同步
   1. 线程安全问题：当多线程并发访问临界资源时，如果破坏原子操作，可能会造成数据不一致
   2. 临界资源：共享资源（同一对象）。一次仅允许一个线程使用，才可保证其正确性
   3. 原子操作：不可分割的多步操作，被视作一个整体，其顺序和步骤不可打乱或缺省
   4. 实现线程同步方法：
      1. 对相关代码块加锁：synchronized(锁标记){需要加锁的代码，临界资源}。当t1线程拿到锁标记时，t1将加锁的代码执行完毕以后，释放了锁标记后，t2线程才能拿到对应的锁标记；反之亦然
      2. 对相关方法加锁：对方法添加修饰符synchronized，方法内部的代码，都是加锁的代码
      3. 线程是否同步是由是否是同一个锁标记来决定的
      4. 死锁：一个线程可以同时拥有多个对象的锁标记，当线程阻塞时，不会释放已经拥有的锁标记，由此可能造成死锁。即当第一个线程拥有A对象锁标记，并等待B对象锁标记，同时第二个线程拥有B对象锁标记，并等待A对象锁标记时，产生死锁。
3. 线程安全的并发集合（集合补充）
   1. Collection
      1. list：CopyOnWriterArrayList 写时复制，读写分离
      2. set：CopyOnWriterArraySet同上
      3. Queue：BlockingQueue 阻塞队列集合，先入先出（FIFO）生产者消费者。void put(E e)将指定元素插入此队列中，若没有可用的空间，将阻塞；E take()获取并移除此队列的头部元素，若其中没有可用的有效元素，将阻塞。实现类：①ArrayBlockingQueue 数组实现，有界队列②LinkedBlockingQueue 链表实现 无界队列
      4. ConcurrentHashMap jdk1.5 分段锁 默认16段，jdk1.7 CAS算法
      5. CAS算法：Compare and Swap，即比较再交换，是一种无锁算法，CAS有3个操作数，内存值V，旧的预期值A，要修改的新值B，当且仅当预期值A和内存值V相同时，将内存值V修改为B；实现了区别于synchronouse同步锁的一种乐观锁，JDK5之前java语言是靠synchronized关键字保证同步的，这是一种独占锁，也是悲观锁。
4. 线程池
   1. ExecutorService：线程池接口，常用方法①submit(Runnable task)提交任务给线程池，线程池会自动为你分配对应的线程执行任务。②void shutdown()关闭线程池
   2. Executors：工具类，①static ExecutorService newFixedThreadPool(int nThreads)返回一个创建固定线程的线程池对象②static ExecutorService newCachedThreadPool()返回一个线程池对象，新建的线程池没有线程，当提交任务时，新建一个线程
   3. 通过工具类中的方法获取线程池对象

## 1.5 IO及反射

### 1.5.1流

将数据从内存传输到外部存储设备的通道

1. 流的分类
   1. 按方向划分（以JVM为参照物）
      1. 输入流：将外部存储设备中的数据读入到内存中
      2. 输出流：将内存中的数据写入到外部存储设备中
   2. 按单位划分
      1. 字节流：以字节为单位，能够传输所有类型的文件
      2. 字符流：以字符为单位，只能传输文本文件（能够被记事本打开.java/.txt/.html,并且不改变内容的文件）
   3. 按功能划分
      1. 节点流：具有基本的读写功能
      2. 过滤流：在节点流的基础上，增加了新的功能
2. 字节流：①InputStream字节输入流🡪读操作。②OutputStream字节输出流🡪写操作
   1. 字节节点流
      1. FileOutputStream：文件字节输出流。
         1. 创建文件字节输出流：FileOutputStream fos = new FileOutputStream(文件路径)；若输入的文件不存在，则新建，若输入的文件夹不存在，则报java.io.FileNotFoundException文件找不到异常
         2. 绝对路径：盘符：\文件夹\文件
         3. 相对路径：从项目的根目录开始 文件夹\文件
         4. 路径使用 [\\ 或者 /](file:///\\\\或者/) 表示
         5. 多个参数的创建：FileOutputStream fos = new FileOutputStream(String name,boolean append);第一个表示文件名，第二个参数表示追加或者覆盖，传一个参数时，默认覆盖
         6. 常用方法：
            1. void write(int b)：向外部存储设备中写入数据
            2. void write(byte[] b)：向指定文件中写入 这些字节
            3. void write(byte[] b,int off,int len)：向指定文件中写入数组中指定的字节，b表示数组，off表示开始下标，len表示个数
            4. void close()：关闭资源
      2. FileInputStream：文件字节输入流
         1. 创建同上
         2. 常用方法：
            1. int read()：从输入流中读取一个字节，返回值表示字符的整数表现形式，若没有字符返回-1
            2. int read(byte[] b)：从输入流中读取多个字节，返回值表示读取的有效字节个数
            3. int read(byte[] b,int off,int len)：b数组，off存储到数组的指定下标，len存储个数
   2. 字节过滤流
      1. BufferedOutputStream/BufferedInputStream：缓冲流，提高IO的效率，减少磁盘的访问次数。数据存储在数据缓冲区中，需要使用flush方法清空缓冲区，也可以直接使用close方法关闭自动缓冲
      2. DateOutputStream/DataInputStream：
         1. 操作基本数据类型：WriteByte/readByte、writeint/readint。。。
         2. 操作字符串：writeUTF(String str)、String readUTF()
      3. ObjectOutputStream/ObjectInputStream：
         1. 增强缓冲区的功能
         2. 增强了操作基本数据类型和字符串类型的能力
         3. 可以操作对象：①writeObject(Object obj)向外部存储设备写入对象数据类型②Object readObject()从外部存储设备读入对象，返回值为读入的对象
         4. 对象序列化：对象放在流上进行传输的过程称为对象序列化，自定义的类必须实现Serializable接口；被transient修饰的属性，不参与序列化；java.io.EOFException 文件到达尾部的异常，文件中只有一个对象，连续读取两次就会报此异常；如果参与序列化对象有对象属性，这个属性也必须实现Serializable,当序列化对象为集合时，他的元素对象类型也必须实现Serializable接口



1. 编解码集：字符和整数—对应的关系需要进行保存，保存下来的就是编码集
   1. ASCII：英文字符
   2. ISO8859-1：西欧
   3. GBK：简体中文
   4. BIG5：繁体中文
   5. Unicode：万国码（UTF-8）
2. 字符流
   1. 字符流父类（抽象类）
      1. Reader：字符输入流---》读操作
         1. int read()：一次读取一个字符，返回值类型为int,表示读取的字符的整数表现形式，若没有字符，则返回-1
         2. int read(char[] c)：一次读取多个字符，存储到char数组中，返回值表示读取到的有效字符个数，若读取不到则返回-1
         3. int read(char[] c,int off,int len)：将读取的字符存储在char数组中存储的起始下标 off，读取的长度为len
      2. Writer：字符输出流---》写操作
         1. write(int a)：一次写入一个字符
         2. write(String s)：一次性写入多个字符（字符串）
         3. write(char[] c)：一次写入多个字符，为char数组的元素
   2. 节点流子类
      1. FileReader：文件字符输入流
      2. FileWriter：文件字符输出流
   3. 过滤流子类
      1. BufferedReader：常用方法String readLine()，读取一个文本行，若文件到达尾部，返回null
      2. PrintWriter：
         1. 可以操作基本数据类型
         2. 可以操作字符串：①print(String s)写入不换行②println(String s)写入换行
         3. 可以操作对象：println(Object o)写入对象，写入的数据是对象的toString的返回值
3. 桥转换流：字节流和字符流之间相互转换的桥梁
   1. InputStreamReader(InputStream in,CharsetDecoder dec)：参数1是字节流对象，参数2是解码方式
   2. OutputStreamWriter(OutputStream on,CharsetDecoder dec)：
   3. 步骤：①创建字节节点流②创建桥转换流，同时设置 编解码方式③包装过滤流，方便读写操作④读写操作⑤关闭流（只需要关闭最外层流）
4. File类：IO是对文件内容进行操作，File是对文件本身进行操作，如删除文件、创建文件、重命名等
   1. public String gerName()获取文件名（含扩展名）、文件夹名
   2. public String getAbsolutePath()获取绝对路径

### 1.5.2反射

1. 概念
   1. 类的对象：基于某个类new出来的对象，也称实例对象
   2. 类对象：类加载的产物，封装了类的所有信息（包括类名，包名，父类，接口，构造方法，属性等）
   3. 类加载：1.3.2.4.2
2. 获取类对象
   1. 引用名.getClass()
   2. 类名.Class
   3. Class.forName(类全限定名)
3. 常用方法
   1. newInstance()：获取对应的实例对象（默认调用无参构造）
   2. getDeclaredFields()：获取所有属性的数组
   3. getDeclaredMethods()：获取所有方法
   4. Method.invoke(Object o,Object… args)：调用方法，参数一是拥有此方法的对象，参数二是该方法需要的参数

### 1.5.3设计模式

1. 单例模式：在JVM中，一个对象只能存在一个实例对象，这个对象称为单例对象
   1. 饿汉式：

class ClassA{

private final static ClassA ca = new ClassA();

private ClassA(){}

private static ClassA getInstance(){return ca}

}

优先：并发效率高

缺点：空间利用率低

* 1. 懒汉式：

class ClassB{

private static ClassB cb;

private ClassB(){};

public synchronized static ClassB getInstance(){

if(cb==null)cb=new ClassB();

return cb;

}

}

优点：空间利用率高

缺点：并发效率低

* 1. 优化单例：

class ClassC{

private static class Inner{

static ClassC cc = new ClassC();

}

private ClassC(){};

private static ClassC getInstance(){return Inner.cc;}

}

补充：静态内部类只有在调用其内部的属性或方法时才会被加载

1. 工厂模式：解决对象创建的问题，工厂的职责在于创建一个对象，详情在spring中细述。

# 二.web开发

## 2.1html

1. 概念：HTML（HyperText Markup Language）超文本标记语言，用来生成一个静态网页。
2. 语法：html文件后缀名必须以.html或者.htm结尾
3. 常见标签
   1. &nbsp; 空格
   2. <br/> 换行标签
   3. <p></p> 段落标签
   4. <h1~6> 标题标签
   5. <hr/> 分割线标签
   6. <ul><li> 无序列表
   7. <ol><li> 有序列表
   8. <a></a> 超链接标签
   9. <img/> 图片标签
   10. <table> 表格标签
       1. <table>表示一个表格，<thead>表示表格头（标题栏），<tbody>表示表格体，<tr>表示行，<td>表示单元格
       2. 部分属性：
          1. align：位置布局left（左）center（中）right（右）
          2. bgcolor：背景颜色
          3. rowspan：跨行合并
          4. colspan：跨列合并
   11. <form> 表单标签
       1. action：数据提交的网址
       2. method：提交方式
          1. get：①数据不安全②数据数量有限③乱码解决麻烦
          2. post：①数据安全②数据数量相对比较大③乱码解决比较容易④上传文件必须使用post
   12. <input> 输入框
       1. name：提交数据的名字
       2. checked：默认选中，通常和单选、复选一起使用
       3. type：
          1. text：文本框
          2. radio：单选按钮，同一组按钮name属性值必须相同
          3. checkbox：复选框，同一组复选框name属性值必须相同
          4. file：用于上传文件
          5. hidden：隐藏输入框，页面上不显示，但仍然会作为表单的一部分提交数据，常用于提交不必要显示的数据
          6. button：普通按钮，常配合js使用
   13. <select> 下拉列表
   14. <option> 下拉列表的一个选项
       1. value：选项对应的值
       2. selected：默认选中
   15. <textarea> 文本域

## 2.2css

1. 概念：CSS(Cascading Style Sheets),层叠样式表
2. 语法
   1. 行内样式

标签内style属性设置

* 1. 内嵌样式

html中<style type=”text/css”>标签

* + 1. id选择器：根据标签id属性值选择，#id值{样式名：样式值；。。。}
    2. 类选择器：根据class属性值选择，.class值{样式名：样式值；。。。}
    3. 标签选择器：根据标签名选择，标签吗{样式名：样式值；。。。}
    4. 选择器优先级：id选择器>class选择器>标签选择器
    5. 伪类选择器：
       1. a:link{color：？}：超链接点击前颜色
       2. a:visited{}：点击后
       3. a:hover{}：鼠标悬停
       4. a:active{}：鼠标点击不松
  1. 外部引用样式

<link/>标签引入外部css文件

* 1. 常见样式
     1. 边框样式：border：10px(粗细) dotted（线条） blue（颜色）
     2. 外边距：两个标签之间的距离，margin：10px（像素）
     3. 内边距：标签边框和标签内容的距离，padding：10px（像素）
     4. 背景：
        1. background-color：颜色
        2. background-image：图片路径
        3. background-repeat：是否重复（no-repeat|repeat-x|repeat-y|repeat）
        4. background-position：图片位置（top|bottom|left|right|center）
        5. background：颜色 图片 是否重复 图片位置
     5. 字体颜色：color：red
     6. 文本水平位置：text-align（left|center|right）
     7. 行高：line-height（像素）
     8. 装饰线：text-decoration（overline|underline|line-through|blink）
     9. 宽度：width（像素）
     10. 高度：height（像素）

## 2.3js

1. 概念：JavaScript,网景公司的一个脚本语言，是前端技术中的编程语言（弱类型语言），可以减轻服务器端的压力，把要执行的一些代码分摊到浏览器上交给js处理，比如表单验证等。区别于①JScript（微软的脚本语言）②java（服务器端的编程语言）
2. 使用
   1. js代码要写在html的<script>标签里面
   2. 打印①alert弹出框②console.log()控制台打印③document.write()在页面打印
   3. 注释：和java一样，用//或者/\* \*/
   4. js代码逐行运行，错误不影响前面的代码
   5. 页面不会提示错误，要在控制台（F12）上查看
3. 变量
   1. 所有的变量都用var声明
   2. 一个变量可以被赋值为任意数据类型，并且可以在多个类型之间切换
   3. 变量可以不声明直接使用
   4. 每个指令后面的分号可以不用写
4. 变量类型
   1. string类型：js中不区分单引号和双引号，即字符就是字符串，比较使用==
   2. number类型：数字类型
   3. boolean类型：0，“”，null，undefined，false都是假，其余都是真
   4. object类型
   5. undefined：只声明未赋值
   6. null：声明并且赋值为null
   7. 可以通过typeof()查看变量类型
   8. 类型转换
      1. parseInt(“字符串”)：字符串转数字
      2. parseFloat（）：转换小数
      3. parseInt可以找到第一个非数字前面的数字，如果第一位就不是数字，则=Nan
      4. Nan：not a number 非数字
      5. isNaN（）：判断参数是不是非数字
5. 函数
   1. 定义：function 函数名（参数。。。）{函数体；return 返回值；}
   2. 调用：函数名（参数。。。）
      1. 函数调用时，形参的个数和实参的个数可以不一致，少的参数是undefined，多的参数会丢失
      2. 函数只会覆盖，没有重载
      3. 调用函数与定义函数的位置没有关系
      4. 变量可以直接等于一个函数，但是需要函数定义在赋值前
   3. 匿名函数：function（）{}，一般用在赋值后面或者方法里面
   4. 函数可以作为一个参数
6. dom对象：标签对象，html中的每一个标签都被js封装成了一个个对象，我们可以通过js代码获取到html标签对象。
   1. var obj = document.getElementById(id属性值)
   2. document.getElementByName(name属性值)返回一个数组
   3. 异常：set property value of null（获取不到标签对象）解决办法①把script标签写在body下面②用window.onload=function(){写要执行的代码}
7. 内置对象
   1. 数组：
      1. 创建数组：①new Array(),②var arr=[1,2,3];
      2. 放置元素：arr.push(元素)，arr[1]=元素；
      3. 遍历：for(var i in arr){alert(i)}
   2. 字符串：常见方法如下
      1. indexOf()：获取字符串的下标
      2. subString()：截断字符串
      3. Trim()：去除左右空格
      4. Split()：把字符串拆分成数组
      5. 。。。
   3. 日期
      1. 创建：var d = new Date();
      2. d.getFullYear()：年
      3. d.getMonth()：月
      4. d.getDate()：日
      5. d.getHours()：时
      6. d.getMinutes()：分
      7. d.getSeconds()：秒
   4. Json：js中的对象，一种特定格式的字符串{属性名：值；属性名：值；。。。}，ajax中会细述
8. 事件
   1. 事件名字
      1. onclick：点击事件
      2. onmouseover：鼠标移入事件
      3. onmouseout：鼠标移出事件
      4. onfocus：文本框获取焦点事件
      5. onblur：文本框失去焦点事件
      6. onsubmit：表单提交事件
      7. onchange：内容改变事件
      8. onload：页面加载完后事件
      9. 所有事件都是以on开头，值都是一个函数，有两种写法①dom元素.onclick=function(){}②function test(){};<input onclick=”test()”/>。window.onload也可以改为在body标签里面增加onload事件。
   2. 实例应用
      1. 点击按钮切换图片
      2. 修改超链接的地址
      3. 通过按钮实现复选框的选中或不选中
      4. 修改div的背景颜色
      5. 修改div的class属性（通过dom元素.className）
9. this的使用，在js中this表示window，在事件方法实参使用this表示当前dom对象
10. 表单验证
    1. 给文本框后面添加一个空的span标签，用于提示错误信息
    2. 给每个文本框添加onblur事件，当用户输入完光标离开时给出提示
    3. 每个事件需要返回true或false，用于最后的表单提交
    4. form中添加onsubmit事件，返回true提交，false不提交
    5. 当所有验证事件为true时提交表单
11. 级联下拉
12. bom操作：对整个浏览器对象的操作，包括刷新，后退，地址栏的操作，窗口弹出等
    1. bom：浏览器browser对象模型，根对象是当前窗口的对象，即window
    2. window.alert()：弹出框
    3. window.onload=function(){}：页面刷新
    4. window调用方法时window可省略
    5. window.location：表示当前窗口的地址栏，可以通过改变location的值做页面跳转
    6. window.confirm：确认框，当用户点击确定时，返回true，否则返回false
    7. 禁用超链接：①href=”javascript:void(0)”②href=”#”
    8. js可以使用el表达式，也可以使用s标签
    9. 定时器
       1. setTimeout(函数,时间（毫秒）)：等时间过完去执行函数里的代码
       2. setInterval(函数，时间)：函数会每过一段时间就执行一次
       3. clearInterval(dsq)：停止定时器

## 2.4jdbc

1. JDBC（Java DataBase Connectivity）java数据库连接技术
2. 相关api
   1. Driver：接口，该类型的对象用来和数据库打交道
   2. DriverManager：类，方便管理不同的驱动
   3. Connection：接口，该类型的对象，代表java和数据库之间的连接
   4. PreparedStatement：接口，该类型的对象，表示一个发送sql的工具
   5. ResultSet：接口，该类型对象，表示查询sql执行后的结果
3. 相关步骤
   1. 加载驱动（java代码） Driver实现类
   2. 输入用户名密码，建立和数据库之间的连接（DriverManager建立和数据库之间的连接）
   3. 新建一个发送sql的工具类（PreparedStatement对象）
   4. 发送并执行sql
   5. 处理结果集（ResultSet对象）
   6. 释放资源
4. ResultSet

ResultSet对象保存查询结果，内部有一个数据游标，游标初始指向第一行数据上方。next()游标向下移动一行，移动后当前行有数据返回true，否则返回false。getXxx（列的编号从1开始/列名）可以获取数据，Xxx值的是该列数据类型

1. 数据绑定
   1. 字符串拼接
   2. 占位符拼接
2. JDBCUtils
   1. 读取配置文件
   2. 创建connection对象
3. 常见异常
   1. java.lang.ClassNotFoundException:oracle.jdbc.OracleDriver：驱动类找不到，原因①类名写错②驱动jar包没有引入
   2. java.net.UnknownHostExcepeion:loclahost：无法建立连接，找不到主机loclahost,主机名写错应该是localhost
   3. invalid username/password：用户名或密码错误
   4. SQLSyntaxErrorException：sql语法错误，通常sql语句问题
   5. 索引中丢失In或OUT参数:1：未给第一个占位符赋值
   6. 列名无效：列名写错
   7. 空指针：工具类静态代码块中获取流失败，配置文件路径可能写错
4. ORM：object Relational Mapping（对象关系映射），即数据库中的一个表对应着java中的一个类，表中的一行数据对应该类的一个对象。
5. DAO：Data Access Object（数据访问对象），dao类，封装对一张表的增删改查操作，将java中对对象的操作转化为数据库中数据的操作，dao接口，抽取对表中数据的增删改查操作，可以使用不同的技术实现接口，面向接口编程，提高程序的可维护性，解耦合。
6. 三层结构
   1. dao层：对数据库进行操作
   2. 业务层：service层，业务逻辑判断
   3. 视图层：view层，功能入口用户交互的地方
7. 初级项目开发步骤
   1. 导入相关jar包
   2. 建表
   3. 配置文件工具类
   4. 实体类
   5. dao
   6. service
   7. test测试类
   8. view
8. 事务控制
   1. 事务：保障一个业务操作包含的多条sql要么同时成功，要么同时失败
   2. 业务层中每一个方法对应一个业务功能，为保证每一个业务功能的完整性，每一个业务方法中都必须添加事务控制的代码
   3. 提交事务：commit()
   4. 回滚事务：rollback()
   5. JDBC中一个链接代表一个client，要保证多条sql添加上事务，必须保证事务控制的连接和执行sql的连接是同一个。
      1. 连接注入：把获取的连接作为参数传入到dao层（使用麻烦，书写繁琐，还有API污染）
      2. ThreadLocal 线程绑定：线程对象内部有一块存储空间，可以保存数据，将连接保存在线程中，dao可以在同一个线程中获取连接。set(对象数据)将数据保存至当前线程，get()从当前线程获取数据，remove()从当前线程空间移除数据。
      3. 线程对象内容的存储空间是map类型的，键的类型时ThreadLocal,值的类型是ThreadLocal的泛型决定。

## 2.5jQuery

1. 概念：jQuery是一个JavaScript函数库，极大地简化了JavaScript编程，封装了JavaScript常用的功能代码，提供了简单的设计模式，优化了HTML文档操作、事件处理、动画设计和AJAX交互，兼容各种主流浏览器
2. 使用：
   1. 引入jQuery函数库
   2. 写jQuery代码：jQuery元素.方法名（参数）
3. jQuery元素：同dom元素一样，表示htnl标签对象，只不过方法不同
   1. 获取：$(“选择器”)
   2. 元素转换：
      1. jQuery转dom—》jQuery元素.get(0)
      2. dom转jQuery—》$(dom元素)
4. 选择器
   1. 基本选择器
      1. 标签选择器
      2. id选择器
      3. class选择器
      4. 交集选择器：两个选择器都满足的被选中
      5. 并集选择器：两个选择器用逗号分隔，只要符合一个就可以
   2. 表单选择器
      1. $(“:text”)：选择type= “text”的文本框
      2. $(“:button”)：botton按钮
      3. $(“:checked”)：选中状态的复选框
      4. $(“:checkbox”)：获取到复选框的表单元素
   3. 属性选择器：$(“[属性名=属性值]”)
   4. 过滤性选择器
      1. $(“input:first”)：第一个input标签
      2. ：eq（0）：索引为0的
      3. ：eq（5）：索引为5的
      4. ：last：最后一个
      5. ：gt（0）：索引大于0的
      6. ：not（：first）：选中不是第一个的
   5. 层次选择器
      1. 后代选择器：中间用空格隔开
      2. 子选择器：中间用>隔开
5. jQuery元素方法
   1. html（）：获取或者设置标签的内容，无参获取，有参设置
   2. prop（）：获取或设置元素的属性值，一个参数获取。两个参数赋值
   3. css（）：可以设置元素的样式，一个参数是获取样式，两个参数是设置样式
   4. size（）：可以获取到选择器选中的元素的个数
   5. is（）：参数是一个选择器。jQuery元素.is（选择器）：判断jQuery是否满足is参数里面的选择器
   6. prev（）：找元素紧邻的哥哥
   7. next（）：找元素紧邻的弟弟
   8. children（）：元素的亲儿子
   9. val（）：等价于prop（“value”），表示获取文本框的value属性
   10. parent（）：获取元素父标签
   11. remove（）：删除当前元素
6. 事件
   1. js中事件去掉on
   2. jQuery中事件是函数，js中是属性
   3. Hover(function(){},function(){})：第一个参数是mouseover，第二个是mouseout
7. 动画
   1. show（）：显示元素，改变元素的高、宽、亮度
   2. hide（）：隐藏元素
   3. slidedown（）：显示元素，改变元素的高度
   4. slideup（）：隐藏元素
   5. fadein（）：显示元素，改变透明度
   6. fadeout（）：隐藏元素
   7. 以上函数都可以添加时间类型参数
8. jQuery经典案例
   1. 点击按钮实现图片的切换（click）
   2. 修改超链接的连接地址（prop）
   3. 控制css样式（.css）
   4. 全选（checked，t）
   5. 鼠标悬浮（hover）
   6. 动画效果（show）
   7. 购物车删除（remove）
   8. 元素的遍历：each（function（i，obj））
9. this
   1. this表示的是调用这个函数的元素
   2. jQuery中this表示dom元素
10. 正则表达式：正则表达式是一个字符串格式的规则，能够准确的判断一个字符串是不是某种固定格式，比如邮箱验证等
    1. 定义：var reg = /正则表达式/
    2. 使用：var result = reg.test(字符串)验证字符串是否满足正则表达式
    3. 正则表达的符号
       1. ^：以什么开头
       2. $：以什么结尾
       3. \d：任意一位数字
       4. \w：数字，大小写英文，常见符号
       5. .：任意内容
       6. \：转义
       7. {n}：匹配前面的字符n次
       8. {n,m}：匹配前面的字符n到m次
       9. {n，}：至少匹配n次
       10. \*：匹配0次或多次
       11. +：至少一次
       12. ？：0次或者1次
11. 富文本编辑器。。。

## 2.6JSP

1. 概念：jsp（Java Server Page）动态页面，用来替换原有web应用中负责显示的Servlet，简化动态页面的开发
2. 格式：
   1. 以.jsp结尾的文件
   2. 由html标签和java代码构成
   3. 访问方式：<http://ip:port/项目名/jsp>文件路径
3. jsp本质上就是一个servlet，当第一次访问jsp时，tomcat会将jsp翻译成java并编译成class，为浏览器响应结果，后续访问过程中，只要不修改jsp文件，就使用第一次创建的class文件，如有修改会重新翻译和编译。先翻译，后编译，最后再给浏览器。
4. 语法：
   1. 脚本
      1. 输出脚本（表达式脚本）：<%=java表达式%>,将表达式的值输出到页面，类似打印语句，表达式后不能跟分号
      2. 普通脚本：<%java代码%>,可以写除定义函数以外的任意代码，脚本不能嵌套脚本和标签
      3. 声明脚本：<%!定义变量、函数%>用来定义变量和函数，供输出脚本和普通脚本使用
   2. 指令：<%@指令 属性=值 属性=值。。。%>用来设置jsp页面的属性
      1. page
         1. language：设置jsp支持的编程语言目前只有java
         2. session：设置jsp页面是否使用session，true使用（没有新建），false有就用（不新建）
         3. isELIgnored：是否禁用el表达式，true禁用，false不禁用
         4. import：导入java脚本中使用的类，类似import语句
         5. pageEncoding：设置jsp翻译成java代码时使用的编解码方式
         6. contentType：设置jsp响应给浏览器结果时，文件的响应类型和编码方式。（通常将pageEncoding和contentType以及html中charset设置相同的编码方式）
         7. errorPage：设置当前页面出现异常时跳转的页面
         8. isErrorPage：是否是错误页面，true|false
      2. taglib：<%@taglib uri=”” prefix=”” %>用来引入jsp所需要的标签
      3. include：<%include file=”引入的jsp路径” %>用来在当前页面引入包含其他页面
   3. 动作
      1. include：<jsp:include page=”jsp页面路径” %/>用来包含其他jsp页面，同上
      2. forward：<jsp:forward page=”jsp” %>用来向其他jsp页面跳转
   4. 内置对象：不需要创建可以直接使用的对象
      1. request：请求
      2. response：响应
      3. session：session对象
      4. application：ServletContext对象，代表一个应用
      5. pageContext：当前页面作用域，可操作其他作用域，也可以获得其他8个内置对象，还可以做页面包含和跳转
      6. page：当前页面，类似this
      7. config：ServletConfig对象，保存了jsp的一些启动配置信息
      8. out：流对象，可以向页面输出内容
      9. exception：异常对象，只能在错误页面使用
5. 批量删除：
   1. 使用复选框接受用户选择，并使用form提交
   2. 根据同一个key获取多个value
   3. 项目中可以使用request.getContextPath()指定项目路径
   4. 重定向可以在地址后面拼接数据来传递参数，中午需要使用URLEncoder编码后在传递
6. el表达式
   1. 作用：用来代表输出脚本在页面展示数据
   2. 语法：${值}、${作用域范围标识.对象.属性}，底层通过get方法实现
   3. 作用域范围标识：pageScope<requestScope<sessionScope<applicationScope（从小到大）
   4. 计算
      1. 数学运算：+-\*/%
      2. 关系运算：> >= < <= == !=
      3. 逻辑运算：&& || ！
      4. empty运算：${empty 作用域的命名属性}，true不包含该命名属性,false标识包含该命名属性
   5. 内置对象
      1. pageContext：类似jsp中的pageContext，可以获取其他8个jsp内置对象，例如获取当前项目的应用名${pageContext.request.contextPath}
      2. param：获取client传递来的数据
      3. cookie：获取浏览器发送来的cookie，session的底层实现就是cookie,session标记就是通过cookie.JSESSION获取。
7. jstl标签
   1. jstl：JavaServer Pages Standardtag library，配合el表达式显示结果
   2. 语法：<标签名 属性=值/>
   3. 使用
      1. 引入相关jar包
      2. 页面上引入标签
      3. 使用标签
   4. core标签库及格式化标签库常用标签
      1. <c:if test=”表达式”>：if判断
      2. <c:choose><c:when test=””>：条件判断
      3. <c:forEach>：数组或集合遍历。
      4. <c:url>：重用地址，方法拼接路径，将url的value以var属性值为命名属性名放置到pageContext作用域中。context和value属性同时出现，其属性值必须以/开头，context属性值长设为当前项目名，value值常设为要访问的url-pattern
      5. <c:param>：在上一个标签中做参数传递
      6. <c:set>：向指定作用域中设置数据
      7. <fmt:formatDate>：日期格式化
   5. c:foreach标签属性
      1. var：变量
      2. itens：集合或数组
      3. varStatus：状态属性
         1. first：判断当前变量是否为第一个元素
         2. last：判断当前变量是否为最后一个元素
         3. index：获取当前变量在集合或数组中的下标
         4. count：统计循环次数
8. 密码加密
   1. 对称加密
   2. 非对称加密
   3. 摘要加密：MD5、SHA（简单快速）
9. MVC模式
   1. model：模型，保存数据处理数据，代表业务数据和业务逻辑（service+dao+entity）
   2. view：视图，负责给用户提供交互的界面，收集用户数据，显示结果数据（jsp）
   3. controller：控制器，接受view收集的数据，调用model的业务逻辑，选择显示数据给视图（controller）
   4. 作用：提高代码重用性，解耦合，提高程序的灵活性和可维护性

## 2.7mybatis

1. 概念：一个持久层（dao）框架，提供了对数据库中数据的访问操作（CRUD）
2. 使用：
   1. 导入相关jar包
   2. 引入相关配置
   3. 在mapper文件中做映射
3. 集成struts2需要写配置文件，集成spring需要在spring配置文件中进行配置
   1. 数据源
   2. 起别名
   3. mapper映射
   4. 缓存
4. 常用api
   1. resources资源：常用来读取配置文件的数据，只用读取一次即可
   2. sqlsession：封装了connection，用来生成dao接口实现类，也可提供事务控制，和connection一样的使用条件，一个线程一个sqlsession
   3. sqlsessionFactory：用来创建sqlsession对象，重量级对象，内存占用多，功能多，线程安全
5. mapper文件数据绑定
   1. 单个参数时，除String需要使用@Param外，其余的可以直接用#{}，实体类对象用#{属性名}
   2. 多个参数时，可以使用@Param注解，也可以使用#{下标}获取，下标从0开始。实体类对象使用注解后，用#{注解名.属性名}
6. mybatis工具类：创建获取sqlsession，关闭sqlsession
7. Junit4测试框架
   1. 导入jar包
   2. 在测试类中方法上加@Test注解
8. config配置
   1. <typeAlias>：起别名
   2. <properties>：读取小配置文件
9. mapper配置文件
   1. resultMap：当数据库中查询结果和实体类属性不一致时，使用resultMap做映射
   2. sql和include标签：将共性的sql语句抽取出来，可以在需要的地方多次使用，sql抽取，inlude引用
   3. <association>：单个对象类型
   4. <collection>：集合类型
   5. xml特殊字符：
      1. <：&lt;
      2. >：&gt;
   6. ${}：字符串拼接，可以拼接表名，列名，关键字等，但存在sql注入风险
   7. #{}：？占位符，只能绑定数据，但可以避免sql注入攻击
   8. <if test=””>：类似if语句，满足条件就执行
   9. <where>：会自动删除sql多余的and或or前缀，会在动态sql前添加where关键字
   10. <set>：同上，自动删除多余的逗号后缀，动态添加set关键字
   11. <trim>：前两个的底层，清除指定内容，添加前缀
   12. <foreach>：用于遍历数组或集合，常用来做批量操作
10. 表间关系
    1. 1对1：可以定义对方类型的属性
    2. 1对多：1的一方定义多的集合，多的一方可以定义一的类型属性
    3. 多对多：定义对方的集合
    4. 外键存在于多的一方，相同时都可以
11. 缓存（cache）
    1. 一级缓存：默认开启，作用范围同一个sqlsession，脏数据处理一旦执行增删改立即清空缓存，作用范围小没用
    2. 二级缓存：需要手动开启，在配置文件中用<setting>开启，在mapper文件中<cache>说明，作用范围同一个namespace，脏数据处理执行同一个namespace下的增删改，自动清空，不支持持久化，没用
    3. Ehcache：独立缓存组件，纯java编写，config配置开启，mapper文件配置使用，src下添加ehcache配置文件，作用整个项目，执行增删改，立即清空缓存
    4. redis：也可作为缓存，并实现缓存持久化

## 2.8oracle

1. 概念
   1. 数据库：保存并组织管理数据的软件
   2. 内存：①速度快②瞬时性，一旦软件关闭或者电脑重启数据丢失
   3. 磁盘：可以长久保存数据
   4. 传统的关系型数据库
      1. 以关系模型组织数据
      2. 保存结构化数据
      3. 使用sql操作数据
      4. 性能一般
      5. 保障数据的完整性和正确性（强事务性）
      6. Oracle、DB2、SQLServer、Mysql、sqllife、javadb
      7. 保存对于数据准确性要求比较高的数据
   5. 新型nosql数据库
      1. 查询速度快
      2. 保存非结构化数据（文档、pdf、音频、图片、视频）
      3. 无法一定保证数据的完整性和正确性（弱事务性）
      4. HBase、MongoDB、Redis、Memcached
      5. 保存对于查询性能要求高，数据准确性不高的数据
2. 安装和使用：自行百度
3. 相关概念：
   1. 表：table，保存数据，组织管理数据的基本单位
   2. 行：row，代表表中的一条数据
   3. 列：column，用来描述数据的特征
   4. 主键：primary key，全表唯一，用唯一标识表中的一行数据
   5. 外键：foreign key，用来记录表和表之间的关系
   6. 用户：account，用于登陆认证连接数据库
4. sql：structure query language，结构化查询语言，用来对数据库中的数据进行操作（增删改查）
5. 普通查询
   1. 查询部分列：select 列名。。。from 表；
   2. 查询所有列：select \* from 表；\*代表所有列
   3. 对列进行数学运算：select 列(+-\*/)数字 from 表；不能使用%取余
   4. 给列取别名：select 列 as 别名 from 表；as可以省略
   5. 字符串拼接：select 列||字符串 from 表；||相当于+
   6. 数据去重：select distinct 列。。。from 表；distinct 去重
   7. case when：相当于if else，select 列，case when 表达式 then 结果 when。。。 else 结果 end from 表；对表中已经存在的列进行判断得出新的列
   8. 当前时间：sysdate
   9. dual：哑表，一行一列的表
6. 条件查询：select 列。。。 from 表 where 条件
   1. 比较查询：> >= < <= = !=
   2. 多条件查询：and or；与或，相当于&&和||
   3. 区间查询：between、not between
   4. 枚举查询：in
   5. null查询：is null、is not null
   6. 模糊查询：like ‘%\_。。。\_%’；\_表示一个字符，%表示多个任意字符
7. 补充
   1. 注释：--，相当于java中//
   2. 在Oracle中查询hr用户下的所有表：select table\_name from all\_tables where owner=’hr’；
   3. describe 表名：得到一个表的信息，只能在命令窗口使用
8. 排序：order by 列名 [asc（默认升序）/desc（降序）]，列名。。。
   1. order by 必须出现在sql语句的最后
   2. 多列排序，在第一列相同的基础下才会按第二列排序，依次类推
9. 函数
   1. 单行函数：作用于每一行数据，每一行数据得出一个结果
      1. mod（m,n）：取余函数，等价于m%n
      2. length（字符串）：获取字符串的长度
      3. to\_date（’字符串日期’,’日期格式’）：字符串转日期
      4. to\_char（日期,’日期格式’）：日期转字符串
   2. 多行函数（组函数）：作用于一组数据，得出一个结果，默认把表当做一组
      1. max（列）：求一列最大值
      2. min（列）：求一列最小值
      3. avg（列）：求一列平均值
      4. sum（列）：求一列值的和
      5. count（列）：求该列值的行数；count（\*）计算表中的行数
10. 分组：group by，对表中的数据使用特定的列进行分组，对分组后的各组数据分别使用组函数处理，分组后，select后只能跟分组列和组函数，一组数据只有一个结果
    1. having：类似where，进行条件判断，定义在group by之后，做分组后判断，必须和分组配合使用
    2. where：分组前判断，在group by 前使用
11. 伪列
    1. rowid：记录一行数据的物理地址
    2. rownum：尝试为每一行满足条件的数据进行编号，从1开始，在进行条件判断的过程中，为满足条件的行赋值，所以当rownum参与条件判断时，不能使用> >= != =,只能使用< <=
12. sql关键字顺序
    1. 定义顺序：select—from—where—group by—having—order by
    2. 执行顺序：from—where—group by—having--select—order by
13. 子查询：一个查询sql的条件依赖另一个查询sql的结果，即在一个sql中嵌套另一个sql
    1. where单值子查询：被嵌套的sql返回一行一列的结果，=
    2. where多值子查询：被嵌套的sql返回多行一列的结果，in
    3. from子查询：被嵌套的sql返回多行多列的结果，from
    4. 分页查询：
       1. 使用rownum对数据进行编号
       2. 对rownum起别名，把查询结果当做新表，对其进行区间查询
14. 表连接
    1. 内连接：select 列 from 表1 [inner] join 表2 on 连接条件
    2. 左外连接：select 列 from 表1 left [outer] join 表2 on 连接条件
    3. 右外连接：select 列 from 表1 right [outer] join 表2 on 连接条件
    4. 全外连接：select 列 from 表1 full [outer] join 表2 on 连接条件
    5. 多表连接：select 列 from 表1 left [outer] join 表2 on 连接条件
15. DDL：data definition language，数据定义语言，包含表、视图、索引、序列的创建和销毁
    1. 表
       1. 建表：create table 表名{列名 类型 【约束类型】}
       2. 数据类型：
          1. number（m，n）：小数，m表示整数位，n表示小数部分长度
          2. number（m）：整数
          3. char（n）：表示最多可以保存n个英文字符，定长字符串，无论保存多少个字符，一定占据n个字符空间
          4. varchar2(n)：可变长字符串，根据保存字符的实际个数，占据相应的空间，常用于保存内存不定的列值
          5. date：日期
          6. BLOB：大二进制文件类型
          7. CLOB：大文本类型
       3. 约束类型：
          1. primary key：主键约束，主键不能重复，并且不能为null
          2. not null：非空约束，限制一列的内容不重复
          3. unique：唯一约束，限制一列的内容不重复
          4. check：检查约束，对列中的内容范围进行限制，自定义约束
          5. references 表名 （列名）：外键约束，告知数据库该列的值参照哪张表的那一列
       4. 删除表：drop table 表名，如果两张表有外键引用关系，应先删子表后删父表
    2. 序列：sequence，Oracle提供的工具，用来生成自增的不重复的值
       1. 创建序列：create sequence 序列名 [start with 起始值 increment by 自增步长]
       2. 使用序列：序列名.nextval
       3. 删除序列：drop sequence 序列名
    3. 视图：view，一个复杂查询sql的别名
       1. 创建视图：create view 视图名 as 查询语句
       2. 使用视图：直接查询视图名代替复杂的sql
       3. 删除视图：drop view 视图名
       4. 特点：和表类似，但只是复杂sql的别名，不存储数据，不能提高查询效率，可提高开发效率
    4. index：索引
       1. 创建索引：create index 索引名 on 表名（列名）
       2. 使用索引：自动使用
       3. 删除索引：drop index 索引名
       4. 作用：提高查询效率，占用磁盘空间，降低增删改速度，主键自动创建索引，模糊查询及空值查询不走索引
16. DQL：data query language，数据查询语言，之前的查询语句都是
17. DML：data manipulation language，数据操语言，包含对表中数据的增删改
    1. 插入：insert into 表名 values（列1，列2。。。）
    2. 删除：delete from 表名 [where 条件]
    3. 表截断：truncate table 表名，先drop在生成，效率高
    4. 更新：update 表名 set 列名1=值，列名2=值。。。[where 条件]
18. DTL：data transaction language，数据事务语言
    1. 事务：数据库的一种机制，保证一个业务操作包含的多条sql，要么同时成功，要么同时失败。
    2. commit：提交事务
    3. rollback：回滚事务
    4. Oracle中DML自动开启事务
    5. 事务底层实现原理：Oracle数据库对应每个客户端都有一个回滚段，用来保存客户端执行sql的结果，提交事务后才同步到数据库，rollback后丢弃数据，不同步
    6. 事务特性：
       1. atomic：原子性，事务包含的多条sql要么同时成功，要么同时失败
       2. consistency：一致性，事务包含的sql执行前后，表中的数据必须保持一致
       3. isolation：隔离性，多用户并发访问数据库时，是否允许不同的客户端查看到对方未曾提交的数据
       4. durability：持久性，一个事务结束后，不能再更改事务的结果

## 2.9servlet

1. 概念：特殊的java类，用于生成动态页面，属于javaEE规范的一部分
2. 使用
   1. 实现servlet接口
   2. 重写service方法
   3. 配置web.xml
3. servlet收参：request.getParameter(“参数名”)
4. 常见问题：
   1. 404：资源找不到，路径写错或没有响应
   2. 405：service方法重写错误
   3. 访问servlet，浏览器提醒下载：response.setContentType（“类型”）类型写错
   4. 500：服务器内部错误，代码错误
5. servlet对象生命周期：在第一次访问时创建，调用init方法，每次请求调用service方法，当应用关闭时，对象销毁，销毁前调用destroy方法。在整个应用中servlet对象是单例的。
6. servlet工作原理：
   1. 浏览器通过url发送对servlet的请求
   2. tomcat通过web.xml找到相应的类，如果是第一次访问就创建servlet对象，否则使用已经创建好的servlet对象，通过service方法为浏览器做出响应
7. servlet注解开发
   1. 注释：源代码中描述代码的文字，人看的
   2. 注解：annotation，源代码中描述代码的代码，机器看的，本质上是特殊的接口
      1. 标记注解：@注解名
      2. 单值注解：@注解名（属性名=值）
      3. 多值注解：@注解名（属性名1=值，属性名2=值。。。）
   3. @WebServlet（“url-pattern”）：相当于web.xml中的配置
8. 乱码问题
   1. 原因：编码方式和解码方式不一致
   2. 解决：使服务器、页面。工具等编解码方式保持一致
      1. 页面设置：charset=utf-8
      2. 服务器配置：URIEncoding=utf-8
      3. 后台：setCharacterEncoding（“utf-8”）
9. servlet跳转：
   1. forward：请求转发，地址栏不变，跳转中多个servlet属于同一个请求，可以通过request作用域传递参数【request.getRequestDispatcher(url-pattern).forward(request,response)】
   2. redirect：重定向，地址栏会改变，二次请求，不能使用request作用域【response.sendRedirect(url)】
10. cookie：当浏览器发送请求时，服务器会创建cookie响应回浏览器，下次浏览器可携带cookie访问做身份识别，cookie由浏览器负责保管，服务器只识别自己发放的cookie，存在缺点如下
    1. 不安全
    2. 可以被禁用
    3. 数据量有限制 4k
    4. 不能直接存中文，除非使用URLEncoder和URLDecoder进行编解码
11. session：会话技术，session对象由服务器为每一个浏览器提供，保存在服务器中，有有效期（距离上次访问30分钟）
    1. session创建：如果之前已经存在对应的session，直接返回，否则新建
    2. 可作为作用域存储数据
       1. setAttribute：存数据
       2. getAttribute：取数据
       3. remove：移除数据
    3. session的销毁：session.invalidate（）立即销毁或者存活时间到时销毁
    4. 底层实现：浏览器第一次访问服务器获取session时，服务器会将session的id以cookie的形式写会浏览器JSESSIONID；后续访问时，会自动携带JSESSIONID到服务器，根据sessionid的值找到浏览器对应的session。
    5. 使用案例
       1. 强制登陆
       2. 验证码
12. filter：过滤器，可以过滤一个或者多个servlet的请求，请求会先经过过滤器，再经过servlet，响应时也一样，先经过过滤器，再响应结果给浏览器
    1. 作用：定义多个servlet的共性代码，避免代码冗余
    2. 使用：
       1. 实现filter中的doFiter方法
       2. 配置web.xml
    3. 案例
       1. 设置字符编码集
       2. 强制登陆
13. 监听器：listener，负责监听特殊事件发生的对象，在监听到特殊事件发生之后，执行一段java代码
    1. 实现ServletContextListener：监听servletContext对象的创建和销毁，监听网站的启动和销毁
    2. 实现HttpSessionListener：监听session对象的创建和销毁
    3. 配置listener-class
14. filter生命周期：在web应用启动时创建，过滤请求时调用，web关闭时销毁

## 2.10Struts2

1. 概念
   1. 框架：程序开发过程的半成品，解决项目开发过程中通用的问题，从而提高项目开发效率
   2. Struts2框架：用来完成MVC模式中C的部分，替换原项目中servlet的部分代码
2. 使用：
   1. 添加相关jar包
   2. 配置web.xml核心过滤器
   3. 写action类[实现Action中的execute方法]
      1. ServletActionContext：通过该工具类获取request和response
      2. request获取session
      3. session获取servletcontex
   4. 配置Struts.xml文件
      1. package：包管理
      2. name：包的唯一标识
      3. extends：继承父包，默认Struts-default
   5. dtd：Document type define 文档类型定义，用于约束一个xml文件的内容
3. 跳转：在Struts.xml中result标签中配置，type设置跳转类型
   1. action跳jsp
      1. dispatcher：默认请求转发（可以不写）
      2. redirect：重定向
   2. action跳action
      1. chain：请求转发
      2. redirectAction：重定向
   3. 跨包跳转：
      1. 在result标签中使用param标签声明包名
      2. 同样声明跳转的地址
   4. 全局跳转
      1. <global-result>全局跳转，解决action中跳转路径冗余问题
      2. 一般在后期重构时使用
      3. 全局与局部配置冲突时，局部优先
      4. 只在同一个包下生效
4. 配置文件标签
   1. package：对action标签进行分类管理，将同一个模块的放在一起，提高项目后期的可维护性，重用功能
      1. namespace：命名空间，在访问action时，路径必须加此值，必须以/开头，相当于包名
      2. extends：继承父包，默认继承父包的配置
5. DMI：dynamic method invoke，动态方法调用，可以在一个action中定义多个方法完成多个功能，将同一个模块的功能定义在一个类中，提高开发效率，同时提高后期维护,struts.xml配置文件中可以使用通配符。
6. 工作流程：
   1. 根据项目名在webapps目录下匹配项目
   2. 核心过滤器拦截所有请求进行判断
   3. 动态资源有Struts2处理，根据地址在配置文件中的映射，执行方法并响应结果
   4. 静态资源由服务器其他servlet处理
   5. 响应结果给浏览器
7. 收参
   1. 零散数据：直接定义相关属性，名字和参数的name对应
   2. 对象数据：参数name应写成对象.属性名
   3. 数组、集合：多个参数name一致
8. action开发形式
   1. 实现action接口
   2. 继承actionsupport类：无需一定实现execute方法，可以继承到更多方法
   3. POJO方式：Plain Old Java Object,不需要继承任何父类，不需要实现任何接口，简单，开发效率高
9. action生命周期：
   1. 每次访问创建
   2. 每次访问工作
   3. 执行完毕方法销毁
   4. 一次请求创建一个action，action对象是多例的
   5. 线程安全，可以使用属性接收数据
10. 作用域的使用
    1. 使用ServletActionContext获取到3个作用域对象
    2. 实现XXXAware接口（RequestAware、SessionAware、ApplicationAware）定义map<String,Object>集合
11. 数据处理机制：值栈存储，ognl表达式获取，s标签显示
    1. OGNL表达式（对象图导航语言）：用于获取数据，是一个独立的技术，不是Struts的组件
       1. 从root区获取数据（简单对象）直接获取
       2. 从contextMap区获得数据（Map集合）需要加#
    2. 值栈：ValueStack，用来保存一次请求过程中数据，包括客户端发送的数据及作用域中的数据
       1. 好处：统一管理，和视图层解耦合
       2. 获取：ActionContext context = ActionContext.getContext();ValutStack vs = context.getValueStack()
       3. 生命周期：一次请求，每一次请求创建一个新的值栈，请求发生改变，值栈对象随之改变，但其中的session和application对象不会重复创建
    3. s标签
       1. 显示单个数据：<s:property value=”ognl表达式”/>
       2. 有条件的显示数据：<s:if test=”ognl表达式”><s:elseif test><s:else>
       3. 遍历集合：<s:iterator value>获取集合，<s:property value>获取元素属性
       4. 显示值栈内容：<s:debug/>
    4. BaseAction：将获取值栈和操作值栈共性的代码抽取到父类中，子类继承到操作作用域的方法，提高开发效率和可读性
    5. 值栈root区：root区保存对象。栈数据结构，保存的数据后进先出，action对象会以栈的特点存放，请求会从栈顶开始向栈底搜索，匹配到就返回，所以应避免多个action的属性名重复
12. 拦截器：Interceptor，类似过滤器
    1. 编码：实现interceptor接口
    2. 配置：在配置文件中的包下面配置
    3. 特点：
       1. 类似过滤器，用于抽取action中共性的代码，也可以改变请求执行流程
       2. 执行顺序跟在标签中使用的顺序一致
       3. 无法拦截静态资源
       4. 拦截器只在一个包下有效
    4. 拦截器栈：interceptor-stack，组合多个已经存在的拦截器，统一使用
    5. 默认拦截器：default-interceptor-ref，在没有显式设置拦截器时使用，当显式的设置浏览器后，默认浏览器失效
    6. 继承Struts-default其实就是继承内置的拦截器
    7. token拦截器：
       1. form标签添加<s:token/>
       2. action标签中配置<interceptor-ref name=token>
       3. 配置错误页面跳转：<result name=”invalid.token”>xx.jsp</>
       4. action需要继承actionsupport
13. 文件上传
    1. 提供上传文件的页面
       1. form标签
       2. enctype=“multipart/form-data”
       3. file
    2. action
       1. File对象接受页面传递的文件，名字必须和页面name保持一致
       2. File对象属性名+FileName：文件名
       3. File对象属性名+ContentType：文件类型
       4. servletContext.getRealPath(“/相对路径”)：根据相对路径获取绝对路径
14. 文件下载
    1. 强制浏览器以附件形式下载
       1. response.setContentType(“text/plain”)
       2. response.setHeader(“content-Disposition”,”attachement;filename=”+filename)
       3. 如果文件名为中文，需要对其进行编码，URLEncoder.encode(文件名,”utf-8”)
       4. struts.xml中的constant标签可以设置上传文件大小，jar包中的default配置文件中也可以设置
15. 类型转换器
    1. 继承StrutsTypeConverter实现方法
       1. 参数一：map表示值栈中contextMap，读取三大作用域的数据
       2. value参数：相关字符串数据
       3. 参数三：要转换的对象类型
    2. 配置
       1. 在action同包下配置properties文件，文件名为actionname-conversion.properties,属性名=转换器全类名
       2. src下，xwork-conversion.properties,属性全类名=转换器全类名

# 三.最基础的企业开发

## 3.1ajax

1. 概念：Asynchronous Javascript And Xml(eXetension Markup Language)异步js和XML，核心语言是js，数据传输工具是xml，异步请求，即可以在不刷新页面的前提下，发送请求并且接受响应，无刷新请求。
   1. 同步请求：请求发送出去之后，需要等待响应结束才可以进行一些事情，页面会刷新，地址栏会改变
   2. 异步请求：不会刷新页面，不需要等待，地址栏不变，局部刷新
   3. 可以通过dom、css在js中和html做交互
   4. 通过字符串、xml和json在js中和java后台做交互
2. 执行原理：
   1. 创建XNLHttpRequest对象
   2. 建立连接:open方法
   3. 发送请求：send方法
   4. 等后台响应结束并且正常响应，就可以获取后台响应回来的数据
3. ajax发送post请求需要做模拟表单：通过xhr设置请求头部信息
4. 手动封装ajax需要4个参数：method请求方式，url请求地址，params请求需要携带的参数，success响应结束后要调用的函数。
5. JSON：JavaScriptObjectNotation(js对象标记)，特定格式的字符串，可以跨计算机语言做数据传输
   1. 数组/list/set：[元素，元素]
   2. 对象类型/Map：{属性名/key：属性值/value,属性名/key：属性值/value}
   3. 解析工具：
      1. gson
      2. fastjson
      3. jkson
6. jqurey的ajax
   1. 引入jQuery脚本库
   2. $.ajax({url:’’,type:’’。。。})
7. Struts2集成ajax
   1. package标签继承json-default
   2. result标签中type=”json”
   3. param标签name=“root”，转换指定属性

## 3.2MySQL

1. 概念：同Oracle一样，关系型数据库，开源免费，区别
   1. Oracle默认端口号1521，MySQL默认端口号3306
   2. Oracle不需要用户处理字符编码，MySQL需要设置
   3. Oracle有多级用户sys/system/hr/scott，MySQL只有root
   4. O主键序列完成，M自动递增
2. 列类型：int、varchar、date、bigint、smallint、float、double等
3. 批量添加：insert into 表名 values(列属性),(列属性)。。。
4. 分页：limit 起始数，每页量
5. 拼接：CONCAT（‘%’，‘a’，‘%’）
6. jdbc操作mysql
   1. 添加MySQL驱动jar包
   2. 修改配置文件
   3. mapper中写sql

## 3.3bootstrap

1. 概念：前端框架，在jQuery基础上封装了大量的样式，提供了很多组件，支持自定义组件，使用前需要引入jQuery和bootstrap样式、脚本库
2. 帮助文档
   1. 模态框
   2. 表单验证
   3. 展示数据
      1. 请求参数：①offset起始位置②limit每页条数
      2. 响应数据：①json②总条数total，当前页数据rows

## 3.4ElasticSearch

1. 概念：
   1. restful风格：representational state transfer，表述性状态传递，参数作为地址的一部分
   2. http请求方式：服务器操作CRUD（增删改查）
      1. get：查
      2. post：添加、修改
      3. delete：删除
      4. put：修改、添加
   3. 幂等性：一次操作和多次操作，对数据影响一样，如get/delete提交，restful风格可以充分表达请求方式，如get/a/b/101。
   4. 关系型数据库概念
      1. DBMS：DataBaseManageSystem数据管理系统，
      2. DB：DataBase数据库
      3. table：表
      4. user：用户
      5. record：记录，表中的一行数据
      6. column：列，存储同类型的数据
      7. schema：模式，一系列内容的集合
   5. es，全文检索（full-textsearch）工具
      1. 基于Lucence的全文检索工具
      2. 可以提供分布式，多用户访问
      3. Apache的开源java项目
      4. 分布式、高扩展、高实时的搜索与数据分析引擎
2. 相关概念
   1. 近实时：near real time（NRT），检索一个数据，1s以内从输入到响应，接近于实时处理
   2. 索引：index，索引小写，同数据库索引
   3. 类型：type，类似数据库table
   4. 映射：mapping：schema，管理类型及文档结构
      1. text文本
      2. ip地址
      3. date日期
      4. long、double、integer
   5. 文档：document，类似record一行记录，以json方式存储管理数据
3. kibana基本操作
   1. 索引操作
      1. put/索引名：创建索引
      2. delete/索引名：删除索引
      3. delete/\*：删除所有索引
      4. get/\_cat/indices?v：查看索引信息
   2. 类型的操作：put/索引名并创建类型（product）
   3. 文档操作
      1. 添加：put/索引/类型/id
      2. 查询：get/索引/类型/id
      3. 删除：delete/索引/类型/id
      4. 更新：post /索引/类型/id/\_update①更新原有数据②添加新数据③在原数据上更新
   4. 批量操作：put|post/索引/类型/\_bulk
4. 检索方式
   1. url检索：get/index\_name/type\_name/\_search?参数
   2. DSL检索：详情见资料
5. IK分词器
6. java操作es
   1. 添加依赖
   2. 创建索引并制定映射
   3. 添加、更新、删除文档
7. springboot操作es：继承接口直接使用里面的方法+自定义方法
8. 集群

## 3.5layui

1. 分页：
   1. 请求参数：
      1. page：当前页码
      2. limit：每页条数
   2. 响应
      1. code：0
      2. msg：“”
      3. count：总条数
      4. data：当前页数据
2. 节点概念
   1. 节点：菜单中每一个分支
   2. 树形菜单
   3. 子节点：一个节点下的若干个子节点
   4. 叶子节点：没有子节点的节点
3. tree属性
   1. title：String
   2. id：String
   3. childre：list/set/数组

## 3.6linux

1. 概念：就是一个操作系统，同Windows不同的是，Linux主要应用在服务器上，是开源免费的生产环境，性能、安全和效率上都优于Windows，并且支持多用户。
   1. 操作系统：对计算机软件和硬件的统一管理，可以和软件和硬件做交互
2. 安装虚拟机VMware及CentOS7的Linux系统（详细步骤见资料）
   1. 快照：相当于Windows的备份与恢复
      1. 拍摄快照：备份
      2. 还原快照：恢复
   2. 克隆：复制一台机器，关机状态下才能克隆
   3. 分配ip并查看
      1. centos7：ipaddr
      2. centos6：ifconfig
3. 安装客户端xmanager
4. linux常用命令
   1. 语法：命令名 –参数，区分大小写，可以通过命令名 –help查看该命令的帮助信息
   2. 基本操作命令
      1. dhclient：分配动态ip
      2. ipaddr：查看ip
      3. clear：清屏
      4. reboot：重启
      5. shutdown：关机
      6. ping ip：检测网络，Ctrl+c结束
   3. 文件命令
      1. 原则：一切皆文件
      2. 绝对路径：从根目录/开始
      3. 相对路径：当前目录下
   4. 操作文件
      1. pwd：显示当前所在的目录
      2. cd 路径：切换目录
      3. ls：显示当前目录下的文件，①-l以列表方式显示②-a所有包含隐藏文件
      4. mkdir 目录名：创建一个目录/文件夹，-p创建父级目录
      5. touch filename：创建一个文件
      6. cp、mv、rm：复制、剪切、删除
      7. more、less、head、tail：查看文件内容，-f显示滚动信息
      8. grep：到文件中模糊搜索，grep –n 关键字 文件名
   5. 连接
      1. 软连接：类似Windows中的快捷方式，ln –s创建
      2. 硬链接：一个文件，和源文件相互独立又相互影响，ln 创建
   6. 管道
      1. >：把执行命令之后的结果写入文件，覆盖写
      2. >>：追加写
      3. 命令1|命令2：命令1执行之后的结果作为命令2执行的目标，命令2一般使用grep
   7. 编辑
      1. vi：进入编辑模式
      2. ：wq保存并退出
      3. ：q！不保存退出
      4. nyy复制n行
      5. p/P粘贴
      6. ndd删除n行
   8. 软件安装
      1. tar –zxvf 压缩包：解压
      2. rpm –ivh rpm结尾的文件：安装rpm软件
5. 安装jdk+tomcat
6. yum安装+nginx
   1. yum：yum install –y gcc，
   2. gcc：C语言环境，nginx需要
   3. wget下载nginx的yum原
   4. yum安装
7. Nginx
   1. 启动：systemctl start nginx
   2. 关闭防火墙：systemctl stop firewalld
   3. 默认端口号80
   4. 配置管理tomcat
8. 负载均衡：保证多台代理的服务器接收均衡的请求
   1. 代理
      1. 正向代理：代理客户端，客户端能够知道访问的真实服务器
      2. 反向代理：代理服务器，不知道真实服务器
      3. nginx是反向代理服务器
   2. 负载均衡策略
      1. 轮询策略：默认的策略，一个挨着一个访问
      2. 权重策略：在配置文件中添加weight属性
      3. ip黏着：用户ip和服务器绑定，原理对用户的ip地址去hash值，然后对存活的服务器数量取模，绑定对应服务器，在配置文件中添加ip\_hash;
9. mysql安装
   1. 通过wget+yum安装
   2. 启动：sysytemctl start mysql
   3. 登陆：mysql –uroot –p
10. 主从复制
    1. 读写分离：主机负责写操作，多台机器（从机）负责读操作，可以提升数据库服务器响应的效率，依赖主从复制
    2. 主从复制：主机负责写操作以外，还需要负责把数据同步到从机中
    3. 主从架构：主机负责管理，从机负责执行任务
    4. 原理：以日志的方式做增量备份，即把主机接受到的写操作的sql语句形成日志文件，然后传输到从机中
11. 读写分离
    1. mycat：开源java项目，用于数据库集群，也叫逻辑库中间件，实现主从架构的读写分离
12. 权限管理
    1. 创建组：groupadd groupname
    2. 创建用户：useradd –g groupname,每个用户属于一个组
    3. 设置密码：password 用户名
    4. 切换用户：su 用户名
    5. 分配权限：
       1. =号授权：chmod u=rwx，g=r--,o=r-- filename
       2. +/-添加/撤销某一权限：chmod u+x,g-x,o+w filename
       3. 数字分配：chmod 754 filename
    6. 权限：user group other；r4w2x1

## 3.7redis

1. 概念：非关系型数据库，在内存中以key-value的方式存储数据，并且可以自动持久化
2. 非关系型数据库特点：
   1. 结构简单：key-value方式存储数据、文档方式存储数据
   2. 内存存储：弱化了事务的处理
   3. 自动持久化：可以自动把内存数据写入磁盘
   4. 支持脚本语言
   5. 支持集群环境
3. 安装
4. 基本操作
   1. 启动redis：./redis-server ../etc/redis.conf
   2. 关闭redis：./redis-cli shutdown
   3. 打开客户端：./redis-cli
   4. 退出客户端：exit/quit
5. java操作redis：通过jedis完成
   1. 添加jedis依赖
   2. 创建jedis对象，调用方法完成操作
6. 序列化和反序列化
   1. 序列化：把java对象转换成一个json串或者字节数组写入磁盘文件
   2. 反序列化：从磁盘文件获取数据转换成java对象
   3. jedis中的get和set方法如果要操作java对象，需要对象做转换处理
      1. fastjson：都转成json串
      2. SerializationUtils：都转成字节数组（常用）
7. redis数据结构：根据value的数据类型进行分组
   1. String：字符串
   2. List：可重复集合
   3. Set：不可重复集合
   4. Hash：value是一个Map集合，方便对数据做统一管理，如购物车
   5. Sorted\_set(ZSet)：可排序的set集合，集合中的每一个元素都有一个分数，根据分数进行排序
   6. help @组名：获取该组下的所有命令帮助信息
8. 常用操作
   1. set str str：往redis中添加一个字符串结构的key，value
   2. get str：根据key获取到对应的value
   3. keys \*：获取所有的key
   4. select dbindex：选择使用的数据库，dbindex表示数据库编号，默认16个数据库，编号0~15，默认使用0
   5. jedis中的方法名和redis中的命令名一样，可以通过help @组名/help 命令名获取帮助信息
9. redis作为缓存服务器（可以持久化）：自定义缓存实现Cache接口，对比着PerpetualCache完成缓存类。
10. 解决集群环境下session数据丢失问题
    1. session复制：把新创建的session复制到集群环境下的其他服务器中，浪费内存，占用网络资源
    2. IP黏着：有可能负载不均衡，session数据还会有可能丢失
    3. cookie管理：把需要存储的session数据存储在cookie中，再次发生请求时从客户端携带的cookie中获取，cookie存储数量有限，存储中文比较麻烦，客户端行为，不能存储重要数据，有可能被禁用
    4. 借助非关系型数据库实现：把session写入redis中，使用时从redis中获取
11. redis持久化机制：把数据从内存写入磁盘，永久保存
    1. RDB机制：全本快照，默认磁盘文件dump.rdb，全部备份，默认
    2. AOF：以日志的方式做增量备份，把对redis写操作的命令形成日志的方式，追加到磁盘文件中，在配置文件中开启，开启之后RDB和AOF同时运行，恢复数据时通过AOF恢复，默认磁盘文件appendonly.aof
    3. 日志重做：确定appendonly.aof文件中的命令都是对于恢复数据来说有意义的，即根据内存中的现有数据，生成对应的命令写入到磁盘文件中，把文件中原本的内容覆盖掉
    4. 热切换：RDB切换AOF时会有数据丢失，可以在redis正常运行时，通过config set appendonly yes开启AOF机制，然后在停止服务修改配置

## 3.8spring

1. 工厂设计模式：通过工厂创建对象
   1. 好处：解耦合，有利于项目的后期维护
   2. 配置文件+反射+面向接口编程
2. spring：企业级框架，轻量级框架，解耦合，创建对象，管理对象，公共内容的统一处理，如事务处理和日志记录
   1. 对jdbc有封装，可以操作数据库
   2. 对servlet有封装，可以完成和页面的交互，springmvc
   3. 对junit有封装，在spring环境下测试更简单
3. spring应用：通过spring完成对象的创建
   1. spring通过工厂设计模式创建对象，利用反射加载无参构造方法创建对象，默认单例，配置scope=“prototype”可以创建非单例对象
   2. 通过set方法或构造方法给对象属性赋值
      1. 构造注入
      2. 自动装配：可以根据名字和类型进行自动装配
      3. 设置注入
4. SpringIOC：inversionOfController，控制反转，把创建对象并且给属性赋值的控制权交给spring框架
   1. DI：依赖注入，给对象中依赖的属性赋值，三种赋值方式如上
   2. 复杂对象的创建：写一个类实现spring提供的接口FactoryBean里面的getObject方法
5. spring对象生命周期
   1. 单例对象：加载spring配置文件时创建，只创建一次
   2. 非单例对象：每次使用时都会创建
   3. 初始化：通过bean标签的init-method属性进行配置，值为类中的方法名
   4. 销毁：工厂销毁时，对象销毁，销毁前操作destroy-method
6. 加载小配置文件
   1. 通过context：property-placeholder加载小配置文件
   2. import标签负责导入其他配置文件
7. log4j：log for java，用来做日志记录
   1. 添加jar包
   2. 添加配置文件
   3. 在执行到使用log4j的代码时，就会在指定位置显示日志信息
   4. 创建：Log log = LogFactory.getLog(该类的类对象)
   5. 使用：log.debug(日志信息)
   6. 日志级别：debug<info<warn<error
8. spring动态代理
   1. 概念
      1. 公共功能/额外功能：指的是在很多方法中都需要的功能，比如日志记录、事务控制
      2. 核心功能：最根本的内容，比如业务逻辑的处理
      3. 被代理类/对象：只有核心功能的类/对象
      4. 代理类/对象：有核心功能以及公共功能的类/对象
      5. 静态代理：编译时代理类已经存在，对于开发没有意义
      6. 动态代理：编译时代理类不存在，运行时动态生成的代理对象
   2. 使用
      1. 写被代理类，只有核心功能
      2. 交给spring管理
      3. 写公共功能类，需要实现spring提供的接口，比如MethodBeforeAdvice
      4. 交给spring管理
      5. 在配置文件中通过aop标签进行配置，config标签下，pointcit配置切入点，advice配置编织
   3. 公共处理：增强处理
      1. 前置增强：实现MethodBeforeAdvice及before方法，在目标方法执行之前的增强处理（参数:method目标方法，params调用方法传递的参数，target目标方法所在的对象）
      2. 后置增强：AfterReturningAdvice,在方法执行之后做增强（参数result目标方法返回值）
      3. 异常增强：ThrowsAdvice，发生异常时执行的处理，标记性接口，方法需要在注释中复制（参数ex目标方法发送的异常信息）
      4. 环绕增强：MethodInterceptor，类似拦截器，目标方法执行前后以及异常都会做的增强处理（参数methodInvocation.proceed()请求目标放行）
   4. 切入点：需要做增强处理的地方，一般都在service层，通过pointcut标签完成定义
      1. execution表达式：execution（方法返回值 方法全限定名 参数列表），可以使用通配符
      2. args表达式：args（目标方法参数类型），表示所有带这个参数类型的方法都需要增强
      3. within表达式：within（包全限定名..\*），包及其子包下的方法
      4. @Annotation表达式：@annotation（注解全限定名），所有被注解标记的方法，都需要增强处理，这里的注解可以自定义
      5. 表达式运算：①表达式1and表达式2，方法必须符合1和2才做增强。②表达式1or表达式2，方法符合表达式1或者2都会做增强
9. AOP：面向切面编程（OOP面向对象编程）
   1. 相关概念
      1. 切入点：需要增强处理的地方，做一个标记
      2. 编织：增强处理添加到切入点的这个过程
      3. 切面：编织之后形成的一个几何概念
   2. AOP：以横切的思想，在运行的过程中，把某一功能添加在指定的位置，AOP是在OOP的基础上完成的，是对OOP的补充
      1. 应用：公共处理，功能扩展
      2. 原理：默认使用jdk动态代理创建对象，面向接口编程，也可以使用cglib动态代理，面向继承编程
10. springJDBCTemplate：spring对JDBC的封装处理，类似于connection和sqlsession
    1. 添加相关jar包
    2. 加载配置文件
    3. 创建数据源对象
    4. 创建jdbcTemplate对象，需要数据源做参数
    5. 配置dao实现类
    6. 查询：jdbcTemplate.query(sql,rowMapper,args)
    7. 增删修：jdbcTemplate.update(sql,args)
11. 事务处理
    1. 增删改操作，springJDBCTemplate会自动在dao层添加事务处理
    2. 编程式事务处理：在代码中写事务处理的代码，很少使用
    3. 声明式事务处理：以AOP思想，把事务处理定义为增强处理，在配置中定义切入点，运行时把事务编织到切入点位置
    4. 过程
       1. 定义事务管理器
       2. 配置增强处理，通过tx：advice标签完成配置
       3. 定义切入点
       4. 编织，一旦service层使用声明式事务处理，原本dao层的事务就不会起作用了
12. 集成Struts2
    1. 添加相关jar包，在web.xml中进行相关配置
    2. 由tomcat加载spring配置文件（context-param标签），创建spring工厂（listener标签）
    3. Struts2使用的action对象，应该到spring工厂中获取
13. struts2配置文件的加载顺序
    1. 核心jar包中的Struts-default.xml（1个）
    2. 对其他技术的支持的jar包配置struts-plugin.xml（多个）
    3. 项目中自定义的配置struts.xml（1个）
14. web.xml中标签的加载顺序：context-param>listener>filter>servlet
    1. context-param:上下文参数，容器参数，所有的listener、filter、servlet都可以通过参数名获取到参数值，类似于类中的属性
    2. filter/servlet标签中可以定义init-param：这个标签只能在对应的类中使用，类似于局部变量
    3. filter/servlet都有对应的mapping标签，filter是根据mapping标签的上下顺序执行，servlet根据匹配条件执行，精确匹配>后缀匹配>模糊匹配>缺省匹配
15. 整合MyBatis
    1. 添加相关jar包
    2. 添加配置文件
    3. 配置公共功能（加载配置文件，创建数据源，创建SqlsessionFactory，dao）
16. 注解开发
    1. 注解完成IOC
       1. @Component（bean标签的id）：相当于配置文件中的bean标签
       2. @Autowired：按照类型自动装配,可以标记在属性或属性的set方法上，标记在属性上时可以不要set方法
       3. 配置文件中使用context:component-sacn标签开启注解，注解创建的对象，默认名字类名首字母小写
       4. @Repositury：用来标记数据访问层（dao）
       5. @Service：标记业务逻辑处理层（service）
       6. @Controller：标记控制器层（controller）
       7. @Scope（“prototype”）：配置创建的对象是否单例，prototype表示非单例，一般用于controller
    2. 注解完成事务控制
       1. @Transactional：可以标记在方法或者类上面，表示该方法或类中的所有方法开启事务，注解可以加属性可以指定事务属性
       2. 在spring配置文件中开启事务注解，通过tx:annotation-driven标签开启
17. spring集成Junit
    1. 添加jar包
    2. 写测试类
18. 事务属性详解
    1. read-only：true只读，主要用于查询上，提高查询效率
    2. timeout：超时时间，默认-1，由数据库的超时机制决定
    3. rollback-for/no-rollback-for：遇到异常做回滚或者不回滚，默认RuntimeException及其子类做回滚，非run及其子类做不回滚，自定义异常时可以考虑此项内容
    4. propagation：事务传播机制
       1. REQUIRED：需要一个事务，只要有一个事务就ok，默认值，可满足企业开发的绝大多数需求
       2. REQUIRES\_NEW：需要一个新的事务，每次都开启一个新事务
       3. SUPPORTS：支持事务
    5. isolation：隔离级别
       1. READ\_UNCOMMITTED：读未提交，一个事务可以读取到另一个事务没有提交的数据，脏读
       2. READ\_COMMITTED：读提交，可以解决脏读问题，但是一个事务多次读取，读取的结果不一致，可重复读
       3. REPEATABLE\_READ：可重复读，可以解决不可重复读问题，行锁，可能会产生幻读
       4. SERIALIZABLE：串行，解决幻读，表锁
       5. Oracle支持READ\_COMMITTED以及SERIALIZABLE
       6. MySQL以上4种都支持
    6. mysql数据库引擎
       1. InnoDB：支持事务（默认）
       2. MyISAM
       3. MEMORY
    7. sql语句
       1. show variables like ‘default\_storage\_engine’：确认mysql数据库使用的引擎
       2. select @@tx\_isolation：查看mysql数据库隔离级别
       3. set @@session.tx\_isolation=0/1/2/3：修改隔离级别，从低到高，read\_uncommitted<read-committed<repeatable-read<serializable

## 3.9springboot

1. 概念：springboot是由pivotal团队提供的全新框架，spring+springmvc
   1. 不需要大量的配置文件：简化spring环境搭建
   2. 内置tomcat：可以直接main运行
   3. 约定大于配置：在开发过程中按照规范完成
2. 环境搭建：springboot3.+，maven环境3.+，jdk1.8+
   1. 引入依赖
   2. 引入配置文件：在resources/main下引入application.yml/properties
   3. 创建入口类：
      1. 创建在根基包下面，和controller、service等同级
      2. 默认情况下类名：artifactld+Application.java
      3. 在类上添加springboot提供的注解@SpringBootApplication
      4. 在main方法中通过SpringApplication.run(类对象，args)方法运行
3. 启动
   1. 默认访问项目名是/
   2. 可以在配置文件中对端口号和访问项目名进行配置
      1. server.port：配置端口号
      2. server.servlet.context-path：配置访问时项目名
4. springboot相关注解
   1. @SpringBootApplication：组合注解，有了这个注解，就可以指定这个类是springboot的入口类，同时会加载和这个类同级的包下面的子包下面的spring相关的注解
   2. @RestController组合注解：@Controller+@ResponseBody
   3. 之前的注解都可以正常使用
5. 集成jsp：springboot中没有内置对jsp的支持
   1. 引入jsp支持的jar包
   2. 引入jstl支持的jar包
   3. 引入jsp运行插件
6. 页面跳转
   1. 视图解析器：配置spring.mvc.view.prefix:/（前缀） suffix:.jsp（后缀）
   2. redirect：/index.jsp,不经过视图解析器，直接跳转
7. springboot集成mybatis
   1. 引入依赖
   2. 数据源、数据库驱动
   3. 配置数据源：spring.datasource.username/password/url/driver-class-name
   4. 加载映射文件：mybatis.mapper-locations（加载指定位置映射）,type-aliases-package（重命名）
   5. 入口类加注解：@MapperScan（dao包全名），用来创建dao对象
8. 集成测试：@RunWith（SpringRunner.class） @SptingBootTest（classes=入口类.class）
9. 集成logback：引入依赖，添加配置文件即可
10. 面向切面编程：注解完成开发
    1. 引入依赖
    2. 相关注解
       1. @Aspect：切面，添加在类上面，表示这个类是一个增强处理
       2. @Before：前置增强，添加在方法上面，参数指定切入点
       3. @After：后置增强
       4. @Around：环绕增强
11. 事务处理：
    1. @Transactional，一般加在service实现类的方法上面，可以通过参数获取设置事务属性
    2. 在入口类上添加@EnableTransactionManagement开启注解
12. 文件上传
    1. jsp页面
       1. form表单必须post方式提交
       2. form表单必须以复合表单数据的方式提交（enctyoe=“multipart/form-data”）
       3. type=”file”的input
    2. controller
       1. 方法中声明MultipartFile类型参数，参数名和文件名一致
       2. MultipartFile对象.transeferTo（file）完成文件处理
       3. 重命名解决同名问题
    3. 文件大小限制可以在配置文件中改
13. 文件下载：同springmvc一样
14. 拦截器：springmvc中自定义拦截器
    1. 实现springboot提供的拦截器接口HanlerInterceptor
    2. 把自定义类交给spring管理
    3. 配置
       1. springboot提供了配置类，可以写代码完成配置
       2. 实现WebMvcConfigurer
       3. 实现里面的方法，在方法中配置拦截那些请求，不拦截那些请求
15. war包部署，见笔记

## 3.10SpringMVC

1. 简述：在spring的基础上，对servlet进行封装处理，形成一个mvc框架，运行效率比Struts2快
2. 快速使用
   1. 引入相关jar包
   2. web.xml中配置分发器（核心控制器）
   3. springmvc配置文件中开启controller注解，做视图解析器
   4. 控制类加注解@Controller，方法上加@RequestMapping
3. 执行流程
   1. 页面发请求给tomcat，经过web.xml
   2. 判断是否符合分发器规则，符合找springmvc，否则直接请求
   3. 到springmvc找同名的映射地址，找对应的方法
   4. 执行完后返回字符串，通过视图解析器跳转
4. 参数接收：通过方法形参接收参数，要求参数名与页面传递参数名一致
   1. springmvc默认解析yyyy/MM/dd HH:mm:ss格式字符串为日期格式
   2. 对象类型页面直接写对象的属性名
5. 流程跳转
   1. 视图解析器，为返回的字符串加上指定的前缀和后缀形参地址
   2. forward：/index.jsp 请求转发
   3. redirect：/index.jsp 重定向
   4. 带前缀的跳转不经过视图解析器
   5. 方法返回ModelAndView,视图和模型组成一个对象，视图是跳转的地址，模型默认放request作用域
6. 数据传输
   1. 方法中定义Map形参，springmvc会把map中的数据放到request作用域
   2. servletAPI，在方法中定义request接收，session也可以在方法形参中直接定义
   3. ModelAndView
   4. Model/ModelMap：同Map类似只是方法不同
   5. @ModelAttribute：加载参数前面，默认直接放入reques作用域
   6. @SessionAttribute：标记在类上面，把request作用域的数据复制到session中
7. 自定义类型转换器
   1. 写一个转换器类实现Conver
   2. 把这个类交个springmvc管理
   3. 让springmvc使用自定义的类
8. 中文乱码解决
   1. get提交乱码
      1. 服务器配置
      2. 重构字符串，先编码，在解码
   2. post提交乱码
      1. request.setCharacterEncoding(“utf-8”)
      2. 通过字符编码过滤器解决
9. 文件上传
   1. 原理：通过输入流读取客户端磁盘文件，通过输出流把文件写入到服务器磁盘
   2. 前端：post提交，form表单，enctype=“multipart/form-data”，file框
   3. 代码：定义MultipartFile类型接收，参数名同上传参数名一致
   4. 调用transferTo方法上传至指定位置
   5. 注意重命名问题和路径问题
10. 文件下载
    1. 直接写地址
    2. 后台控制下载，需要传入文件名
11. 拦截器
    1. 写一个类实现spring拦截器接口HandlerInterceptor
    2. 实现里面的方法
    3. 配置文件配置拦截信息
12. 异常处理
    1. 错误页面跳转
    2. 全局异常处理机制
       1. 写类实现springmvc提供的全局异常处理接口HandlerExceptionResolver
       2. 实现方法
       3. 把类交给spring管理
13. rest风格：表述性传递语言
    1. 参数作为地址的一部分
    2. 映射中以{xxx}形式表示
    3. 参数中注解接收@PathVariable（“xxx”）
14. 静态资源处理：<mvc:resource>标签处理
15. 大小工厂
    1. spring大工厂，父工厂，处理service和dao，listener配置
    2. springmvc小工厂，子工厂，处理Controller和视图，servlet配置

# 四、扩展功能

## 4.1项目开发流程

1. 需求分析：需求文档，当前项目的项目背景和简介，以及项目中拥有哪些模块和功能
2. 库表设计：根据需求文档设计库表结构
   1. 显性字段
   2. 隐性字段
   3. 表关系
3. 概要设计
   1. 当前模块有哪些功能
   2. 功能实现的简单思路
4. 【详细设计】：可以认为已经有了详细的代码的注释和思路
5. 写代码：项目开发
6. 测试
7. 上线

## 4.2糊涂工具类

1. 导入依赖
2. 查阅文档
3. 根据实际业务需求修改代码

## 4.3layui文件上传

1. layui文件上传组件默认自动提交，修改成按钮提交
2. 没有选择图片不能提交，加点击事件进行判断，有图片进行layui提交，没图片进行ajax提交
3. jquery图片上传
4. layui图片上传需要清空队列

## 4.4POI

1. 简介：excel文件的导入和导出
2. 导入：
   1. 导入依赖
   2. 创建一个HSSFWorkbook Excel文件对象
   3. 创建一个工作表对象HSSFSheet
   4. 创建一个行对象HSSFRow
   5. 创建一个单元格对象HSSFCell
   6. 在单元格对象中写入数据
   7. 将对象写出到本地 生成Excel文件
3. 导出
   1. 读取文件 得到流
   2. 通过POI处理流
   3. 获取对象中的值
4. 结合笔记做文件上传和下载

## 4.5easypoi

1. 简介：通过注解完成POI的简单开发
2. 使用：
   1. 添加新的依赖（之前的POI要去掉）
   2. 给要导出的数据实体类属性加注解@Excel（name=“xx”），确定映射关系
   3. 通过工具类直接完成导出或导出
   4. 还可以更改注解属性完成值的替换以及图片的导出（需要配置静态资源的路径）
   5. 配置静态资源路径：之后打包部署时不需要外置的tomcat，可以直接通过java命令启动
   6. 图片导入需要注意路径问题

## 4.6echarts

1. 简介：图表展示，柱状图、折线图、饼状图等图片的数据展示
   1. 表格展示：展示原始数据
   2. 图表展示：展示统计结果 数据变化的趋势
2. 基本使用
   1. 导入js
   2. 根据官方案例写代码
   3. 根据需求修改代码
   4. 使用dto和map做映射和 接收参数
3. 优化
   1. 索引：优化的基本手段，能不能用索引，需要通过执行计划
   2. 缓存：数据变动频率比较高，并且数据一致性要求比较高，不适合用缓存
   3. 多线程：计算量特别大的任务不适合使用多线程，一个方法内多个数据库任务也不适合

## 4.7日志

1. 简介：记录运行信息，有利于错误回顾和代码流程的记录
2. springboot中logback的使用
   1. 添加相关配置文件开启
   2. 在配置中开启（不需要配置文件）
      1. root：根日志
      2. com.baizhi：具体包日志
   3. 创建日志对象
   4. 在需要打印的地方使用，开发一般使用debug，上线用info及以上
3. 记录业务运行日志：以aop思想做日志记录
   1. 定义一个注解
   2. 定义增强类，切注解
   3. 在增强类中获取注解的内容

## 4.8git

1. 简介：分布式版本控制工具
2. 使用
   1. 安装
   2. 配置，添加当前电脑身份
      1. git config –global user.name
      2. git config –global user.email
      3. 支持linux命令
   3. 创建文件夹并初始化：git init
   4. 创建文件提交版本库
      1. git add .
      2. git commit –m “版本信息”
   5. 查看日志：git log,git reflog查看所有的操作记录和版本信息
   6. 版本切换
      1. git reset –hard HEAD^ ：回退^的个数的版本
      2. git reset –hard HEAD~n：回退n个版本
      3. git reset –hard 版本序列号：回到指定版本
      4. git status：查看文件的状态，红色表示有文件没有被添加到暂存区，绿色表示已经添加暂存区，但没有提交版本库
      5. git checkout：撤销和上个版本之间的新增内容
3. 集成码云
   1. 准备一个码云账号
   2. 在电脑上生成SHH密钥和公钥，执行ssh-keygen -t rsa -C [xxxxx@xxxxx.com](mailto:xxxxx@xxxxx.com)命令，一只回车就可以
   3. 在码云配置本机的公钥
   4. 新建 仓库，赋值连接
   5. 在电脑上执行克隆操作：git clone 地址
   6. 在本地新建文件，暂存（git add .），提交（git commit -m “v1.0”）并推送（git push）
   7. 远程仓库修改代码本地做更新：git pull
4. 集成idea
   1. 配置码云插件
   2. 配置git命令行
   3. 克隆项目
   4. 版本切换通过命令做！！
5. 分支：相当于版本库
   1. git branch：查看拥有分支
   2. git branch：创建分支
   3. git checkout 分支名：切换分支
   4. git merge/rebase 被合并分支名：分支合并
6. 合并冲突，需要手动解决，解决后的版本添加提交

## 4.9接口开发

1. restful
   1. 参数拼接的方式
   2. 请求类型做了细分处理
      1. get：获取数据
      2. delete：删除数据
      3. post：添加数据
      4. patch：修改数据
      5. put：添加数据（添加一个属性值）
   3. spring中的使用
      1. @GetMapping(“test/{name}”)：定义请求类型必须是get方式
      2. @PostMapping
      3. @DeleteMapping
      4. @PatchMapping
      5. @PutMapping
   4. 测试工具postman可以模拟发送各种请求
2. 接口开发流程
   1. APP的需求分析
   2. 库表设计
   3. 接口文档的确定
   4. 根据接口文档同时开发
   5. 接口联调
   6. 上线
3. 接口文档
   1. 请求地址
   2. 请求方式
   3. 请求参数
   4. 响应参数的格式和具体字段含义

## 4.10shiro权限管理

1. 概念
   1. 权限管理分为认证和授权
   2. 认证（登陆）：身份认证，系统校验一个用户是不是合法用户的过程
   3. Subject主体（user）：指用户实体在项目中对应的对象，封装了和认证相关的属性和方法
   4. 凭证信息（密码）：Credential
   5. 身份信息（用户名）：principal
   6. 授权：也称访问控制，控制不同的主体能够拥有不同的权限
   7. 资源：一切皆资源，一般指库表数据
   8. 权限（primission）：对资源可以进行的操作，相当于操作数据的方法
   9. 角色：一组权限的集合
2. 使用
   1. 导入依赖
   2. 自定义Realm继承AuthorizingRealm抽象类
   3. 实现对应的方法
   4. 写配置类，配置过滤器工厂，创建安全管理器，创建realm
   5. session：SecurityUtils.getSubject().getSession()
   6. 授权方式
      1. 代码式
      2. 标签式：在jsp中通过shiro标签完成授权
      3. 注解式：在配置类中配置后，对对应方法添加注解
   7. 注解式异常处理
      1. 前端处理
      2. 跳转404页面
      3. 全局异常捕获

## 4.11springCloud

1. 微服务基本概念
   1. 单体应用架构：一个归档包包含所有功能的应用程序，通常称为单体应用
   2. 微服务
      1. 每个服务可独立运行在自己的进程里
      2. 一系列独立运行的微服务共同构建起整个系统
      3. 每个服务为独立的业务开发，一个微服务只关注某个特定的功能，例如订单管理，用户管理等
      4. 微服务之间通过一些轻量的通信机制进行通信，例如Restful API进行调用
      5. 可以使用不同的语言与数据存储技术开发
2. springcloud基本概念
   1. SpringCloud是一个含盖多个子项目的开发工具集，集合了众多的开源框架，利用SpringBoot开发的便利性实现了很多功能，如服务注册，负载均衡等，简化了分布式架构的开发
   2. 特点
      1. 服务注册和发现
      2. 路由
      3. 服务之间的调用
      4. 负载均衡
      5. 断路器
3. Eureka组件：服务发现组件，基于rest的服务，包含客户端和服务端
   1. 服务提供者
      1. 服务注册：当微服务client启动时，向eureka服务发起注册并上报该节点的相关信息
      2. 服务续约（心跳机制）：在完成注册后，服务提供者会对其进行维护，防止该服务被剔除，默认30s进行一次心跳，90s未响应认为服务失效，进行剔除
   2. 服务消费者
      1. 获取服务
      2. 服务调用
      3. 服务下线
   3. 服务注册中心
      1. 失效服务剔除
      2. 自我保护
      3. 高可用
   4. RestTemplate服务调用
      1. restTemplate.getForEnty()
      2. restTemplate.getObject()
   5. Ribbon负载均衡
4. Feign声明式服务调用
   1. 创建feign接口
   2. 声明请求的服务
   3. 配置日志信息
   4. 自带断路器，负载均衡
5. Hystrix
   1. 雪崩效应：基础故障导致级联服务故障的现象
   2. @EnableHystrix：开启断路器
   3. @HystrixCommand（fallbackMethod=“方法名”）
6. 配置中心
   1. 配置中心服务器
      1. 添加依赖
      2. 添加注解支持
      3. 远程仓库创建配置文件
      4. 配置相关配置文件
      5. 启动配置中心服务
   2. 配置中心客户端
      1. 导入依赖
      2. 添加配置
      3. 配置文件名改为bootstrap.yml
7. 网关
   1. zuul：主要功能是路由转发和过滤器
   2. 路由转发
   3. 服务过滤

# 五、知识补充（详情见资料）

## 5.1mysql优化

1. 索引优化：对常用字段添加索引
2. 缓存：对查询操作远多于增删改操作的情况使用缓存
3. sql优化：sql语句细节的处理