# Mybatis

## 1 一级缓存

* 1. Sqlsession在执行多次相同的查询的时候，会在第一次查询之后，将结果缓存起来。
  2. 若没有执行跟新的情况下，同一个Session执行相同的查询，会从缓存里查询。
  3. 若没有执行跟新的情况下，与第一次执行查询的Session不同的Session，去执行相同的查询也会从缓存里查询
  4. 在执行更新的情况下，执行跟新的Session会刷新缓存，再去执行与刚才相同的查询，则会从数据库查询。没有执行跟新Session不会刷新缓存，还是会从之前的缓存里查询。由此，会产生脏读的情况。

# Jdk8 新特性

## lambda表达式

* 1. lambda表达式格式：（）-> {}
  2. Lambda表达式作用之一：代替函数式接口的匿名内部类实现。
  3. 声明lambda表达式，必须是函数式接口。表达式是否返回值，应该与函数式接口里的方法一致。否则报错

## 函数式接口

a有且仅有一个抽象方法，可以有多个默认方法的接口

b函数式接口，在使用的时候，可以用lambda表达式来代替匿名内部类

## 接口的默认方法与静态方法

1.Jdk8以前，接口是不能拥有非抽象的方法

2.Jdk8以后，接口可以拥有默认方法，默认方法是费抽象的

3.Jdk8以后，接口可以有静态方法

4.Jdk8以后，有了default关键字，在接口的方法前面加上default，则变成了默认方法

5.Jdk8以后，抽象类和接口的比较

* + 1. 都有抽象方法和非抽象方法，不同的是，接口的非抽象方法需要用default来申明
    2. 单继承与多继承的区别
    3. 接口不能有属性，抽象类可以有
    4. 接口不能有构造器，抽象类可以有，但是不能实例化

## 方法的引用

a -----（调用静态方法） 类名：：方法名

b -----（调用构造方法） 类名：：方法名

c -----（调用实例方法） 对象：：方法名

# Oracle

## 零．杂想

修改密码：alter user 用户名 identified by 新密码；

设置密码永不过期：ALTER PROFILE DEFAULT LIMIT PASSWORD\_LIFE\_TIME UNLIMITED;

登录：键入sqlplus，根据提示以此键入用户和密码；若要以dba身份登录，输入密码的时候，要键入 “密码 as sysdba”

Natapp 外网穿

## 一．存储过程

1.1创建存储过程

Create procedure pro\_name([in/inout/out] var\_name var\_type)

Begin

\*\*\*\*\*\*

End ;

调用 call pro\_name();

2.1 申明变量

Declare x1,x2,x3 数据类型;

Declare x4 数据类型 Default ‘默认值’;

变量的作用于，只存在于申明所在的begin与end之间

2.2 给变量赋值

Set variable\_name=value [,variable\_name1=value1]

用户变量

用户变量一般以@开头

2.3 存储过程参数 in/out/inout

IN参数的值必须在调用存储过程时指定，在存储过程中修改该参数的值不能被返回，为默认值（个人理解：内部无法改变传入的值）

OUT:该值可在存储过程内部被改变，并可返回(个人理解：内部无法接收传入参数的值，仅仅是用于返回值的)

INOUT:调用时指定，并且可被改变和返回过程体（个人理解：过程内部可以接收值，也可以返回值）

2.5 控制语句

IF-THEN-ELSE

If 表达式 then \*\*\*\* endif

If 表达式 then \*\*\*\* else \*\*\* endif

CASE-WHEN-THEN-ELSE

Case 变量

when value then \*\*\*\*

when value1 then \*\*\*\*\*

else \*\*\*\*

end case

WHILE-DO…END-WHILE

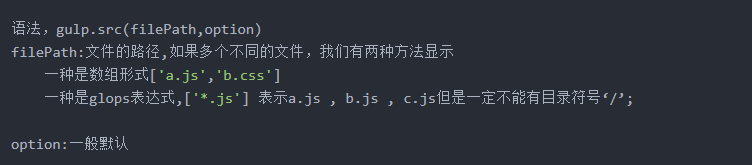
While 条件表达式 do \*\*\* endwhile

# Glup 流程化控制

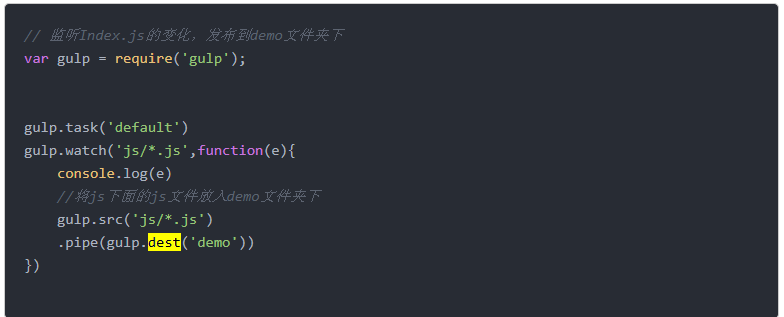
该插件是基于nodejs环境运行的

## 一.gulp API

1. .task一个个的任务
2. .src为我们提供这样的方法：将文件转换成stream流



1. .pipe，我们可以通过管道方法来处理每个阶段的流，这些管道依次排开，前一个管道的输出作为后一个管道的输入,对流执行，链式调用；
2. .dest方法表示将文件写入某个目录：
3. .wath提供了一个方法叫watch来监视文件变化，来实现实时发布；

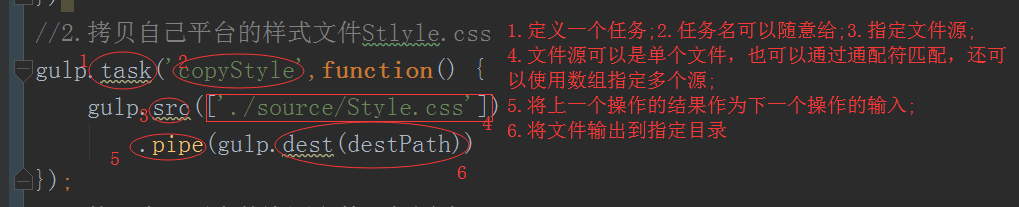


## 二．gulp示例

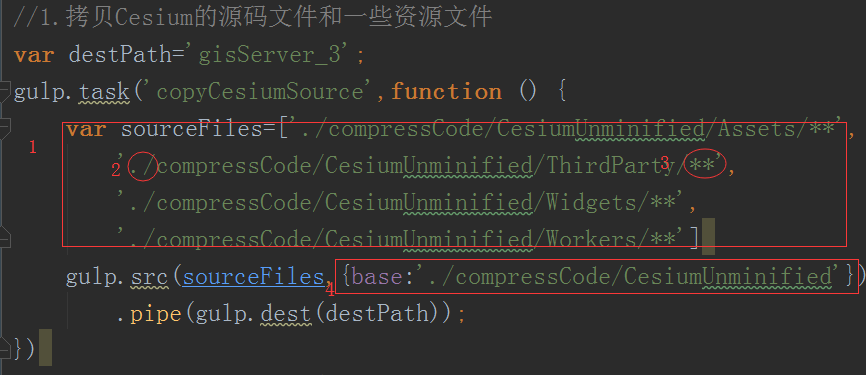
注意：使用这些工具之前，需要先引入相关插件



### 1．复制单个文件

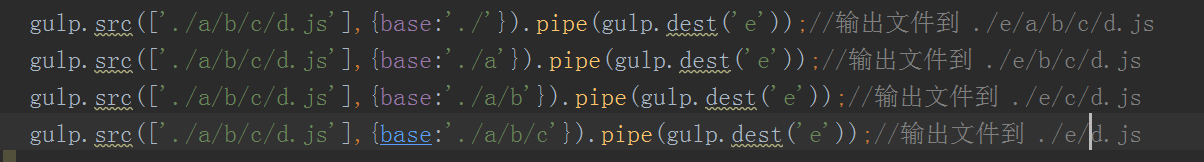


### 复制文件夹

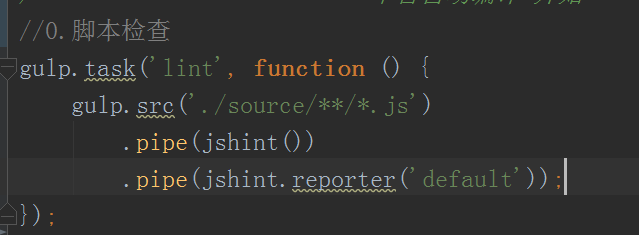


说明：

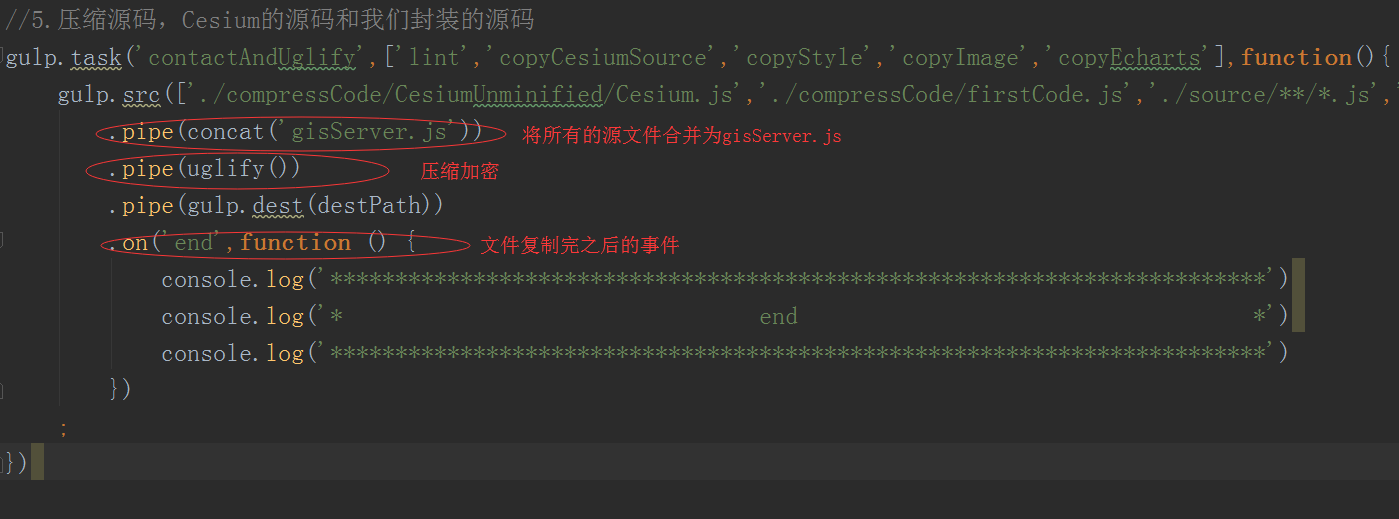
1. 通过数组指定了多个源文件
2. 以gulpfile.js也就是当前脚本文件所在目录为当前目录，使用相对路径指定源
3. 使用通配符指定文件（1个\*表示只匹配该级目录下的文件，若为文件夹，则复制空文件）
4. 指定文件源时，也可以同时指定base路径。指定了该值之后，源文件是保留了目录结构了的，但是保留的目录结构，是在base值之后的。例子如下



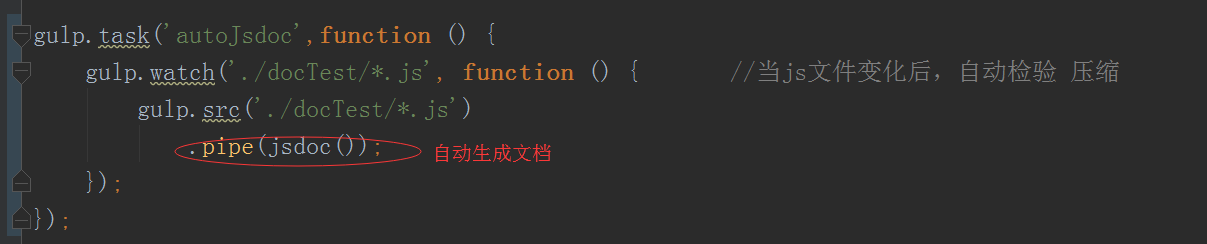
### js代码检查



### 合并和压缩解压



### 5.自动生成js文档

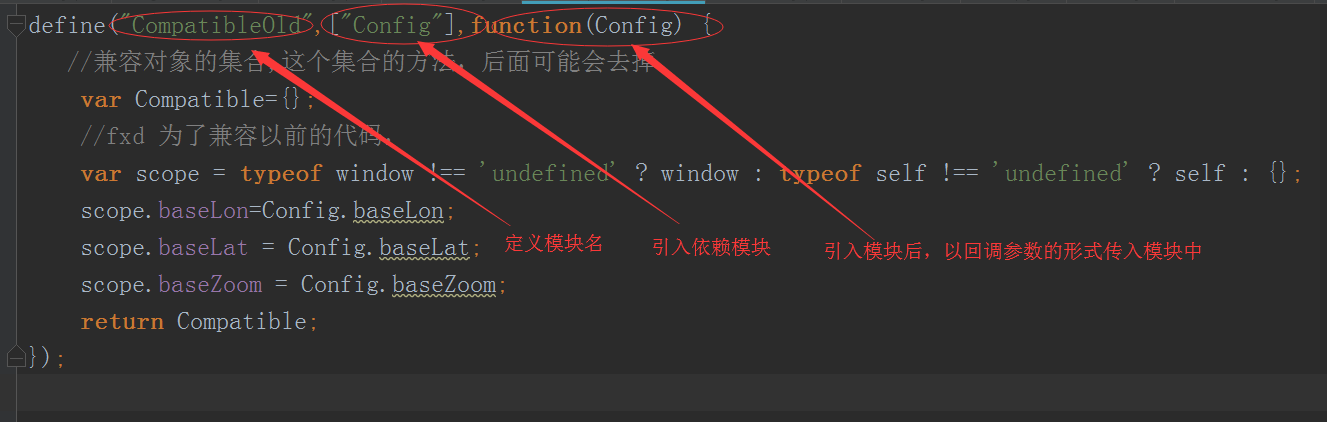


注意：使用该插件自动生成文档时，其源js文件必须严格按照jsdoc的规范写上注释，否则不能生成，或者生成什么也没有。并且生成文档是可以指定目录的，指定目录需要到jsdoc的插件处，修改配置文件。

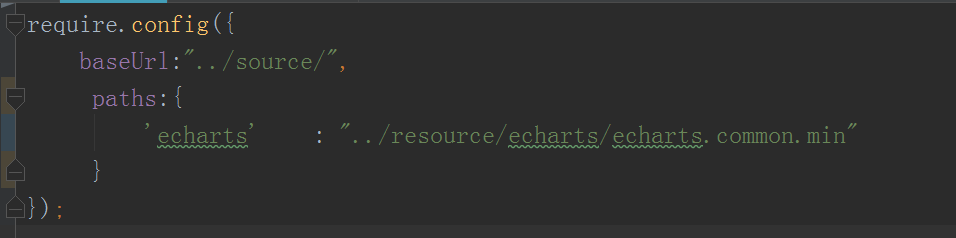
(配置文件路径为nodeJs的node\_modules下的gulp-jsdoc3\dist\jsdocConfig.json，指定destination值，其路径是以gulpfile.js为基础路径的相对路径)

# requireJs 模块化编程

## 一．定义模块在·



## 二．引入第三方模块

，这样配置后，买就可以在定义模块的时候，直接引入如echarts

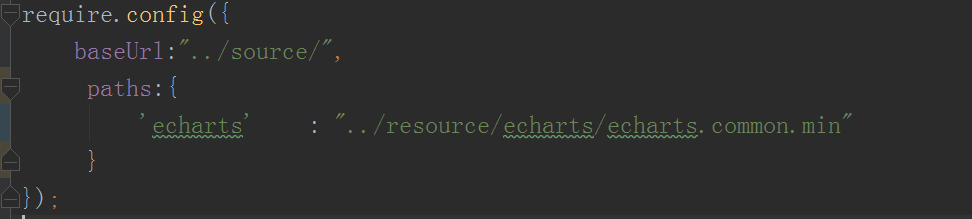
## 三．引入第三方非amd库

## 四．页面使用require

1.引入requirejs,并且指定js的主程序js



2．主程序的配置main.js



Baseurl指定该文件下下面的定义的js模块

Paths引入第三方的模块

# Cesium 开源地球

## 一．简单试例

### 1.1 初始化地球

### 1.2 定位

### 1.3 添加模型（gltf格式）

### 1.4 添加模型 （3dtiles）

### 1.5 添加label，billbord，point，等

### 1.6.1 添加影像（本地）

### 1.6.2 添加影像（在线）

### 1.7 添加地形

### 1.8切换2d,3d

viewer.scene.mode =Cesium.SceneMode.SCENE2D;

viewer.scene.mode =Cesium.SceneMode.SCENE3D;

### 1.9 获取位置（通过屏幕坐标获取世界坐标）

两种方法

通过camera.pickEllipsoid，



第一个参数是Cesium2（x,y）对象，表示窗口的屏幕的坐标

通过scene.pickPosition（para）（此法在2d下获取不了）

Para是Cesium2（x,y）对象，表示窗口的屏幕的坐标

### 1.10 坐标转换

### 1.11 画线，画圆，画多边形

### 1.12 鼠标拾取坐标位置

### 1.13 鼠标点击拾取模型

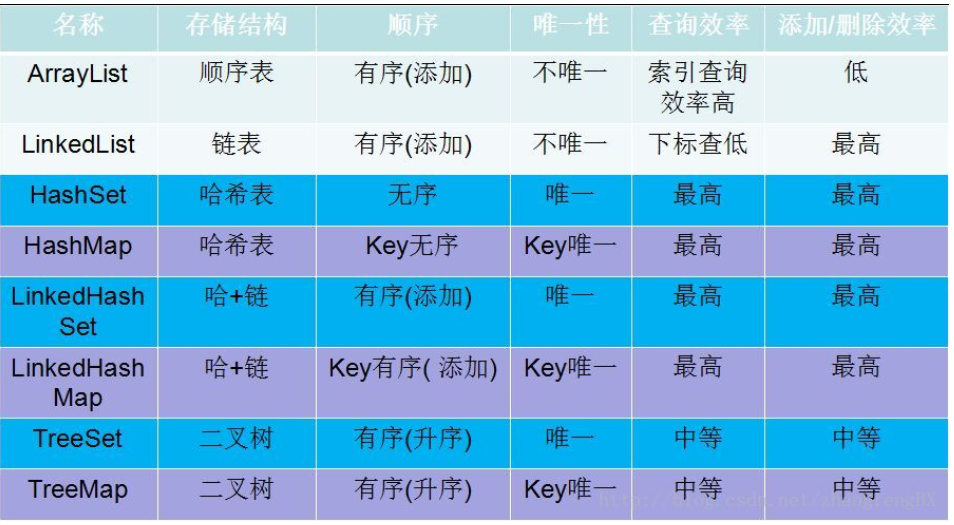
### 1.14 测距

### 1.15 粒子系统

## 二．地形发布

## 三．阿斯达

# 集合框架



## 常用集合（图）

### 1.1 Map



### 1.2 Collection



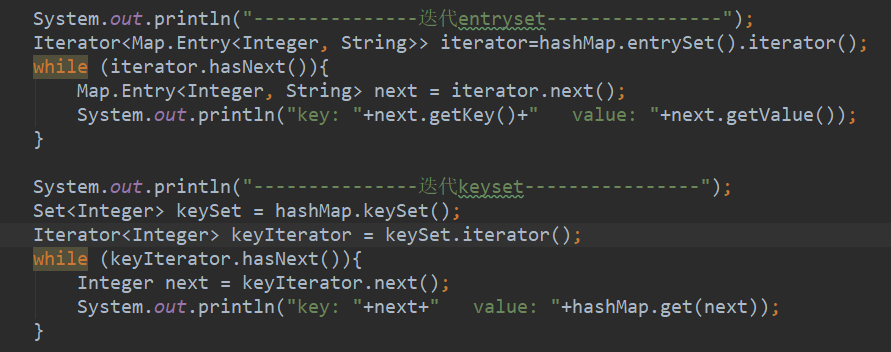
## HashMap

### 2.1简述

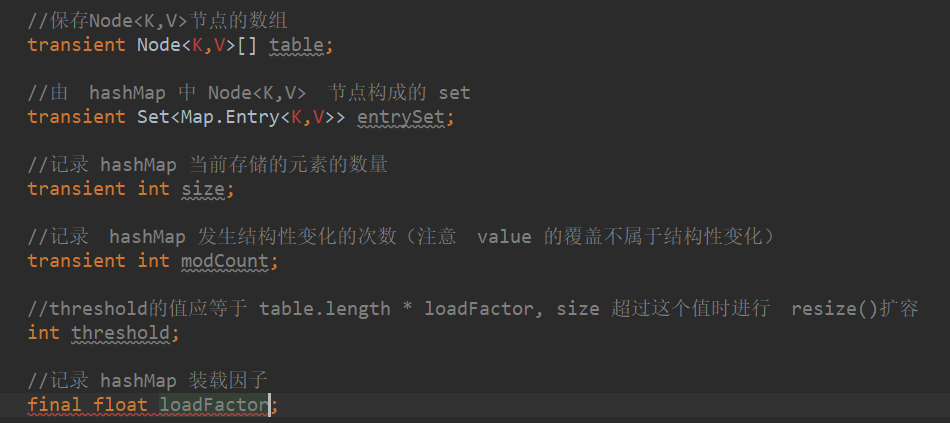
HashMap是以数组+链表来实现的。通过对key求hash值，找到数组对应的下标。数组实际存储的是对entry对象的引用。而链表的使用在于，因为数组的长度是有限的，而key值的个数是无限的，所以通过求hash值的方式来找下标，可能会出现相同的下标。这时候，下表相同的entiry就以链表的形式存储。几个相同下标的entry中，第一个entry中，就会存储第二个entry的引用，第二个存第三个的引用以此类推。

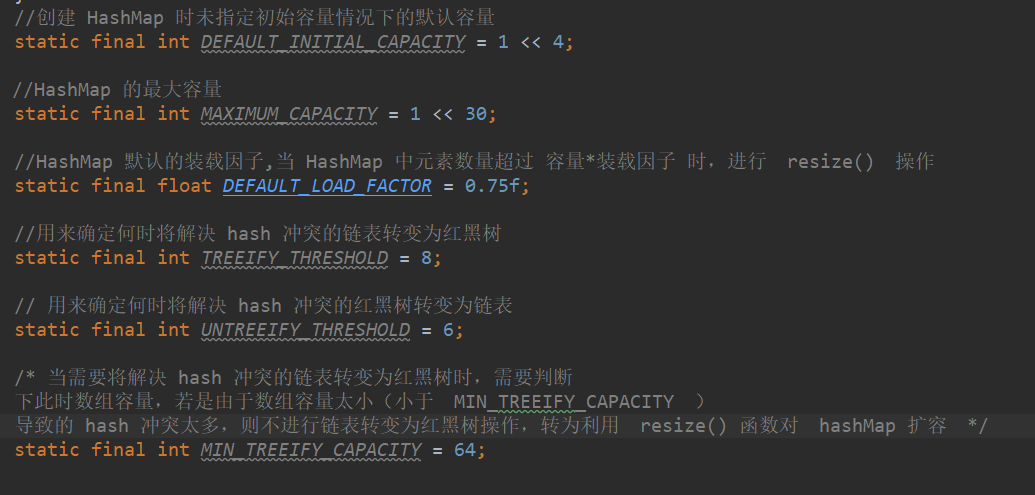
### 2.2 遍历操作

一共两种途径：1.对map的entriySet操作，2.对Map的KeySet操作。在获取到这个两个对象后，又可以通过迭代和foreach来进行遍历。所以说细一点，是四种遍历方式



### 2.3 HashMap的几个变量和常量





### 2.4 相关方法

putMapEntries