Java笔记

1. jdk配置以及开发工具安装与使用
   1. eclipse包的使用
      1. 管理文件
      2. 可以重名
   2. jdk环境变量配置
      1. PATH
      2. JAVA\_HOME
      3. CLASSPATH
2. 变量、常量、字面量
3. 八大数据类型
   1. boolean 布尔型
   2. char 字符型
   3. short 短整型
   4. int 整型
   5. long 长整型
   6. float 浮点型
   7. double 双精度
   8. byte 字节型
4. 常用快捷键
   1. Alt+/
      1. 提示
   2. ctrl+shift+/
      1. 多行注释
   3. ctrl+/
      1. 单行注释
   4. ctrl+shift+Enter
      1. 在当前行的上一行插入一行
   5. ctrl+d
      1. 删除一行
   6. ctrl+alt+向下方向键
      1. 拷贝一行
   7. alt+上下键
      1. 上下移动代码
   8. ctrl+shift+t
      1. 搜索类
   9. shift+enter
      1. 在某行的光标切换到下一行
   10. ctrl+1
       1. 帮助
   11. 格式化:Ctrl+Shift+f
   12. 导包:Ctrl+Shift+o
   13. 查看源码:按住Ctrl+鼠标点击
   14. 搜索类：Ctrl+Shift+t
   15. 帮助键：Ctrl+1
   16. 把小写单词转为大写：Ctrl+Shift+s
   17. 把大写单词转为小写：Ctrl+Shift+y
   18. Alt+Shift+s+c添加无参构造
   19. Alt+Shift+s+o添加带参构造
   20. Alt+Shift+s+r添加getXxx与setXxx方法
   21. Alt+Shift+s+s添加toString方法
   22. Ctrl+o：查询当前当前类的方法
   23. Alt+Shift+a：多行一起写代码
5. 算术运算符
   1. 算数
      1. a++
      2. ++a
      3. b++
      4. ++b
   2. 赋值
      1. i=i+3 i+=3
      2. a+""+b ab
   3. 关系
   4. 逻辑
      1. &&短路
      2. &
      3. ||短路
      4. |
   5. 位运算符
      1. >>:缩小2(n)次方倍
      2. <<:扩大2(n)次方倍
      3. >>>:[无符号右移]只适用于正数,缩小2(n)次方倍
   6. 三目运算符
      1. 1?2:3 if(1){2}else{3}
6. 循环
   1. for
   2. while
   3. 分支
      1. 单分支

if(条件){代码块}

* + 1. 多分支

if...else...

if...else if ... else if... else

?:

* + 1. instanceof String

判断字符串类型

* + 1. switch...case
  1. do...while..

1. 数组
   1. 容器：指定数据类型的容器
      1. int[] a={1,2,3,4,5}
      2. int b[]=new int[5];b[0]=1
      3. int c[]=new int[]{1,2,3,4,5}
   2. 特征
      1. 地址单元连续的存储空间
   3. 概念
      1. 数组长度:元素的个数
      2. 数组的索引[下标]:从0开始=数组的长度-1
      3. 数组的赋值
      4. 数组的遍历
   4. 数组的弊端
      1. 定容

解决方法：定义一个新数组，长度为old。length+1，old指向到新数组，通过arraycopy方法

* + 1. 类型受限制,but Object[]

1. 方法
   1. 为了完成指定功能的代码块
   2. 要素
      1. 方法名,参数列表,返回值类型,访问修饰符,方法体
   3. 方法的重载
      1. 为了方法的多样性
   4. 特征
      1. 方法名相同，参数列表不同[参数个数、参数类型、参数位置]
      2. 与返回值类型没有关系
2. 面向对象
   1. 构造方法
      1. 方法名必须与类名相同
      2. 不要声明返回类型
      3. 带参构造方法

创建对象的同时给对象的属性赋值

* 1. 面向对象的内存分析
     1. 定义对象，当执行了new 对象()

JVM会为对象分配内存空间，但是对象的属性值为各自的默认初始值

* + 1. 栈区存放对象，堆区存放对象属性值
  1. 类和对象的理解
     1. 使用class定义类
     2. 类实际上是指一类具有共通属性和行为的对象的集合，起到一个模板作用
     3. 对象是类的一个具体实例
     4. 对象都具备类中的属性和功能
  2. 访问修饰符
  3. 面向过程以及面向对象的区别和联系
     1. 分析出解决问题所需要的步骤，然后用函数把这些步骤一步一步实现，使用的时候一个一个依次调用就可以了。
     2. 构成问题事务分解成各个对象，建立对象的目的不是为了完成一个步骤，而是为了描叙某个事物在整个解决问题的步骤中的行为
  4. this关键字
     1. 代表当前类中正在使用该方法或者属性的对象的一个指针引用,它是一个隐士参数
     2. this调用本类中的其他构造方法，调用时要放在构造方法的首行。
     3. this调用本类中的属性
     4. this调用本类中的其他方法
  5. static关键字
     1. 和类共存亡，属于类级别[类生成的时候就会初始化静态的属性或者方法]
     2. 静态代码块内不能使用非静态的属性和方法，反之则可以
     3. 静态代码块在类加载(会将类加载到类加载器[classLoader])完毕后就执行了，并且只被执行一次
  6. 垃圾回收机制
     1. java.lang.OutOfMemoryError

增加MaxPermSize值

增加最大堆内存到512M（-xmx参数）

* + 1. 垃圾回收器

分配内存

保证所有正在被引用的对象还存在于内存中

回收执行代码已经不再引用的对象所占的内存

* 1. setter和getter方法
     1. setter：设置成员属性的值
     2. getter：获取成员属性的值
  2. 特征
     1. 封装性

访问修饰符

public（公共的）

无访问修饰符

private（私有的）

protected（受保护的）

属性私有化，对外隐藏细节

* + 1. 继承性

提高代码重复使用率

父类的私有属性可以被继承的

关键字extends

静态的属性和方法是不能被继承,但是可以被所有对象访问并且修改

super关键字（代表直接父类）

引用构造方法,super(),必须放在第一行代码处，一般情况下，在无参构造方法中不会调用有参的构造方法。

引用父类其它方法

引用成员属性

Java中只支持单继承

实现多个接口

间接继承

组合

从某种情况下是可以替换继承的 has a:一个类中包含类外类的引用

final关键字

final修饰的类表示该类不能被继承

final修饰的方法不能被重写

final修饰的变量时常量（常量的变量名用大写，多个单词组成，中间用\_分割，并且常量必须要赋初始值）

tostring

返回该对象的字符串表示形式

所有类都默认继承Object类

* + 1. 多态性

要有继承

要有重写

前提:必须在有继承的前提下

目的:体现子类的多样性、多态性

特征:方法名必须和分类方法名一致，参数列表也必须一致，返回值类型也要一样

访问修饰符不能小于父类的访问修饰符

父类引用指向子类对象

* 1. 抽象类
     1. 特征

抽象类不能创建对象

可以有普通方法，也可以有抽象方法

抽象方法:需要在返回值类型前加abstract,不能有方法体

普通类实现抽象类，必须实现所有地抽象方法

创建抽象类的关键字需要使用abstract关键字

* + 1. 抽象类、普通类、接口的关系

普通类--抽象类:继承 extends

普通类--接口:实现 implements 可以实现多个接口

抽象类--接口:实现 implements

抽象类--抽象类:继承 extends

接口--接口:继承 可以多重继承

* 1. 接口
     1. 接口使用关键字interface定义

接口必须是公开的

接口不能有构造方法

接口中的属性都是public static final类型的静态常量，并且只有一份，被所有子类所共享

* 1. 内部类
     1. 成员内部类

可以无条件地访问外部类的成员

外部类中如果要访问成员内部类的成员，必须先创建一个成员内部类的对象

* + 1. 局部内部类

访问仅限于方法内或者该作用域内

* + 1. 静态内部类

不需要依赖于外部类

不能使用外部类的非static成员变量或者方法

* + 1. 匿名内部类

1. 集合
   1. 本质：容器
      1. 规避了数组的弊端
      2. Collection

List

ArrayList

底层原理是Object[]

查询速度快

增删改较慢

ArrayList():构造一个初始容量为十10个大小的空列表

ArrayList(int initialCapacity):构造具有指定初始容量的空列表

add

remove

LinkList

底层原理:双向链表

查询速度慢

增删改较快

Vector

线程安全、低性能

使用与ArrayList基本相同

Set

TreeSet

HashSet

没有get() 因为无序性，不能通过index获取元素

addAll() 添加另一个集合的数据

SortedSet

* + 1. 键值对集合Map<K,V>

HashMap：底层数据结构为数组+链表

hash值:将任意长度的数据经过hash算法得到固定长度的那个值

Hash表：将hash值放在一起组成一张表叫做hash值

Hashcode：将对象的物理地址值经过Hash算法得到一个整数值

使用hashcode的目的：为了提高查询速度

hashcode碰撞：不同对象的Hashcode相同，将不同对象会放在同一个位置的现象

TreeMap：底层数据结构为红黑树

LinkedHashMap：放入元素的顺序和输出元素一致

HashTable：与HashMap基本相同（二者区别）

HashMap没有继承Dictionary，HashTable继承了

HashMap允许key和value允许放入null键和null值，HashTable不允许

HashMap是线程不安全，HashTable线程安全

ConcurrentHashMap：当前比较流行的线程安全并且效率较高的集合

* + 1. Collections类（专门用来操作集合的普通类）

reserve：反转集合内的元素

synchronizedList（map）：将HashMap转换为线程安全的

synchronizedList（list）：将HashList转换为线程安全的

sort：对集合元素进行排序（自然顺序），如果元素是对象时

实现Comparator接口，重写CompareTo方法

匿名内部类的方式，重写comparator接口

1. 基本类型--包装类型
   1. 基本类型-->包装类型
      1. 装箱

Byte Short Integer Long Float Double Boolean Character

int a=1;Integer b=Integer valueof(a)

* 1. 包装类型-->基本类型
     1. 拆箱

byte short int long float double boolean char

1. 常用类
   1. Math
      1. ceil():向上取整>=
      2. floor():向下取整<=
      3. round():四舍五入
   2. Random
      1. random():随机数[0.0,1.0)
   3. String：final类型
      1. charAt():指定索引处的char值
      2. str1.concat(str2):将str2连接到str1结尾
      3. contains():是否包含
      4. startWith()是否已str2开始
      5. endWith():是否已str2结束
      6. getBytes():转换为byte数组
      7. indexf():返回str2第一次出现的 索引
      8. subString():截取子串
      9. replace():将字符串中str1全部替换为str2
      10. split():分割字符串
      11. toCharArray():将str2转换为字符数组
      12. toUpperCase():将str2转换为大写
      13. trim():只能去除首位空格
      14. equals():比较内容是否相等
      15. ==：比较引用
      16. equalsIgnoreCase():乎略大小写比较相等
   4. StringBuffer：线程安全的
      1. append()追加字符串
      2. reverse():反转字符串
      3. insert():在指定位置插入字符串序列
   5. StringBuilder：线程不安全该类方法与StringBuffer基本相同
   6. Date
      1. getTime():返回现在距离1970.1.1日晚上12：00:00的毫秒值（long）
   7. Calendar：static
      1. get(fild):返回年月日分秒
      2. add():给年月日加定值
      3. format():日期转字符串
      4. parse():字符串转日期
      5. before(when):如果此Calendar的时间在when表示的时间之前，则返回true；
      6. after(when)
2. 异常
   1. 常见异常
      1. 检查性、运行时性、错误
   2. 异常体系结构
      1. 所有异常都继承自Throwable类
   3. 自定义异常
      1. 子类不能抛出比父类更多的异常
   4. 异常处理
      1. try{可能出现异常的代码}catch（异常情况）{要抛出的异常}finally{必须执行的代码块，finally可以不写}
      2. main方法后加上throws+异常情况
      3. throw+异常情况：发生在代码块里面
3. IO流
   1. java.io.File类(文件和目录路径名的抽象表示形式)
   2. 具体流
      1. FileInputStream，FileOutputStream：读取文件较快
      2. DataInputStream，DataOutputStream：读取基本数据类型较快
      3. BufferedReader，BufferedWriter：高级字符流
      4. InputStreamReader，OutputStreamWriter：桥接流，是字节流通向字符流的桥梁，并且可以指定编码
      5. FileReader，FileWriter：不需要低级流进行构建
      6. PrintWriter：打印流
      7. ObjectInputStream，ObjectOutPutStream：对象流（将对象写入磁盘，写是序列化，读是反序列化）对象要实现序列化接口
      8. RandomAccessFile：随机流
   3. File类常用方法
      1. File.separator：兼容不同的操作系统
      2. exists：判断文件是否存在
      3. isFile：判断文件是都是文件
      4. isDirectory：判断是否是文件夹
      5. length：测试文件有多少字节
      6. canExecute：是否是可执行文件
      7. createNewFile：创建一个新的文件
      8. delete：删除文件
      9. mkdirs：创建多层文件
      10. isAbsolute：判断当前路径是否为绝对路径
      11. list：返回一个字符串数组，这些字符串指定此抽象路径名表示的目录中的文件和目录。
   4. IO流通道
      1. 目的

利用File类找到对应的文件，然后利用IO流操作文件内容

用于网络传输

* + 1. 按照流方向分类

输入流InputStream

输出流OutPutStream

* + 1. 按照操作的字节数分类

字节流：传输任何数据

字符流：只能传输文本

* 1. IO流体派
     1. 字节流： 可以传任何数据

读：InputStream

FileInputStream:读取文件快

FilterInputStream

BufferedInputStream

DataInputStream:读取基本数据类型快

ObjectInputStream

ByteArrayInputStream

写：OutputStream

FileOutputStream

FilterOutputStream

BufferedOutputStream

DataOutputStream

PrintStream

ObjectOutputStream

ByteArrayOutputStream

* + 1. 字符流：只能传文本

读：Reader

BufferedReader:高级字符缓冲流

InputStreamReader:桥节流，是字节流通向字符流的桥梁、可以指定编码

FileReader:不需要低级流进行构建

AtringReader

写：Writer

BufferedWriter

OutputStreamReader

FileWriter

PrintWriter

StringWriter

1. ORACLE数据库
   1. 数据库基础
      1. 什么是数据库？数据库是以一定方式储存在一起、能与多个用户共享、具有尽可能小的冗余度、与应用程序彼此 独立的数据集合，可视为电子化的文件柜——存储电子文件的处所
      2. 为什么使用数据库：存储数据、查询数据功能
      3. 怎样学好数据库

明白数据库基本概念，作用

表的知识

函数、存储过程、触发器

常见的简单查询、复杂查询[多表查询，子查询]

* + 1. Oracle数据库和一般数据库不一样

Oracle创建一个数据库等同于创建一个数据库实例

* 1. DML（数据库操作语言）
     1. select（查询语句）
     2. update（修改语句）
     3. insert（插入语句）

insert into 表名（列名）value（内容）

* + 1. delete（删除语句）
  1. DDL（数据库定义语言）
     1. create table（创建表）
     2. drop table（删除表）
  2. DCL（数据库控制语言）
     1. grant
     2. revoke
  3. Oracle的数据类型
     1. number:(4,1)
     2. char、nchar：定长查询
     3. varchar、varchar2(保留字)：变长查询
     4. date:年月日

副主题

* + 1. datetime:年月日时分秒
    2. timestamp:年月日时分秒+时区
  1. 主键:区分每条记录不同
     1. 不能重复
     2. 不能为null
     3. 推荐每个表有一个主键，并不是表必须要有主键
     4. 主键分类

自然主键:当前主键目的只是为了区分不同记录

代理主键:当前主键目的只是为了区分不同记录,而且具有一定业务意义

* 1. 常见函数处理
     1. -- 字符串转日期 select to\_date('2019-10-01','yyyy-MM-dd') from dual;
     2. -- 日期转字符串 select to\_char(sysdate,'yyyy-MM-dd') from dual;
     3. -- 按员工薪水高到低,默认是升序的(asc)，降序desc select ename,sal from emp order by sal asc;
     4. -- 选择员工姓名的第三个字母是a的员工信息 \_:1个字符 %:0-n个字符 %A% select ename,job from emp where ename like '\_\_A%';
     5. -- 对null值的处理 -- 算出所有员工的年薪 -- 数据库中任何数据和null运算，最后结果为null -- nvl(){nvl(comm,0)第一个参数为null返回第二个参数}、nvl2(){第一个参数为null返回第三个参数，不为null返回第二个参数} select ename,sal \* 12 + nvl2(comm,comm,0) from emp;
     6. select 字段名 from 表名 where 条件 group by havaing 分组条件 order by
     7. -- 去掉重复记录(值一模一样) select distinct job from emp;
     8. 判断一个值为空为is null而不是=null
     9. select \* from emp where sal not between 1000 and 2000;-- 包含边界
     10. select count（\*）计算数量
     11. -- 去掉两边空格，不会去掉中间空格 select trim(' ab cd ') from dual; select replace('jack and jue','j','bl') from dual;
     12. -- dual:虚表，为了做临时查询测试使用的 select lower('ABC') from dual; select upper('abc') from dual; select initcap('abc') from dual; select concat('a','b') from dual; select substr('abcdefg',2) from dual; select length('abc') from dual; select lpad('abc',15,'\*') from dual; select rpad('abc',15,'\*') from dual;
  2. 关系型数据库:表与表之间可以进行关联 主外键进行关联
  3. 笛卡尔积
     1. 笛卡尔集

-- 查询员工及其所在的部门信息 重复记录现象:笛卡儿积 select \* from emp,dept; -- 去掉笛卡儿积 select e.\*,d.dname,d.loc from emp e,dept d where e.deptno = d.deptno;

* 1. 连接查询
     1. 内连接

-- 内链接 select e.\*,d.dname,d.loc from emp e inner join dept d on e.deptno=d.deptno;

* + 1. 外连接

-- 右外连接 以右表为主 select e.\*,d.dname,d.loc from emp e right join dept d on e.deptno=d.deptno;

-- 左外连接 以左表为主 select e.\*,d.dname,d.loc from emp e left join dept d on e.deptno=d.deptno;

-- 全外连接 select e.\*,d.dname,d.loc from emp e full outer join dept d on e.deptno=d.deptno;

* + 1. 子查询（跟在查询中的查询）

select后的子查询 from后的子查询 where后的子查询

子查询的结果集 单行单列 单行多列 多行单列 多行多列

-- 查询工资>scott的工资的员信息 select ename,sal from emp where sal > (select sal from emp where ename='SCOTT');

-- 查询工资最低的员工信息 select ename,sal from emp where sal = (select min(sal) from emp);

-- 单行子查询nul值问题 select ename,job from emp where job=(select job from emp where ename='MIKE');

-- 多行子查询 -- 查询大于本部门平均工资的员工的信息 -- 黄灯注意:where后不能使用聚合函数，但是havaing,order by select ename,sal,e1.deptno from emp e1, (select deptno,avg(sal) av from emp e group by deptno) e2 where e1.sal > e2.av and e1.deptno=e2.deptno;

* 1. 分组、聚合函数
     1. min
     2. max
     3. avg
     4. sum
     5. count:统计满足条件的记录数

count(\*) count(数字[不受限制]) count(字段名):如果列值为null值，不加入计数

select count(\*) from emp; select count(1) from emp; select count(comm) from emp;

-- 统计每年入职的员工的人数 ??? select sum(e.cnum),extract(year from e.hiredate) from (select count(\*) cnum,extract(year from hiredate),hiredate from emp group by extract(year from hiredate),hiredate) e ;

* 1. 分页要素
     1. 页码:page 当前页:pageNow 满足条件总记录数:total 100 每页记录数:pageSize 3 总页数:total%pageZie == 0 ? total / pageSize : total / pageSize + 1 每页的具体数据

Oracle使用rownum(伪列) 1、里面的2层子查询目的是为了让rownum重新排序(对结果集进行排序等操作rownum会编程无序的) 2、最终查询的字段以最内层的字段为准 page start end 1 1 3 2 4 6 3 7 9 4 10 12 start = (page - 1) \* pageSize + 1 end = page \* pageSize

* + 1. select \* from (select rm.\*,rownum r1 from (select ename,job,sal from emp where sal > 2000) rm) where r1 >=4 and r1 <= 6;
  1. 约束:保证数据的完整性
     1. 主键约束:唯一且不能null
     2. 外键约束:不给存在外键的主表删除
     3. 惟一约束:唯一但是可以为null
     4. 非空约束:不能为null
     5. 检查约束:check约束
  2. 删除
     1. delete

delete删除数据的时候会记录日志

* + 1. truncate

truncate不会记录日志

数据量大的时候采用truncate速度快

* + 1. 只删除数据，都留下表结构
  1. 事务
     1. 标准:事务要么同时成功，要么同时失败
     2. 事务:保证在一定的业务场景下的一个整体
     3. 特征

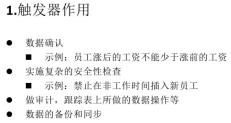
原子性

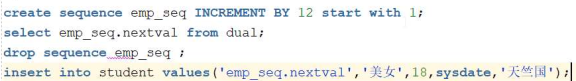
一致性

隔离性

脏读:脏读发生在一个事务A读取了被另一个事务B修改，但是还未提交的数据。 幻读:操作第一个事务的用户发现表中还有没有修改的数据行，就好象发生了幻觉一样。 不可重复读:这样在一个事务内两次读到的数据是不一样的，因此称为是不可重复读。 针对事务的隔离性出现的问题，采用事务的隔离级别来界别 读未提交:都解决不了 读已提交:解决脏读 oracle 可重复读:解决脏读和不可重复度 mysql 串行化:都能解决

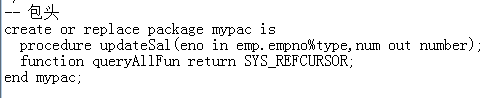
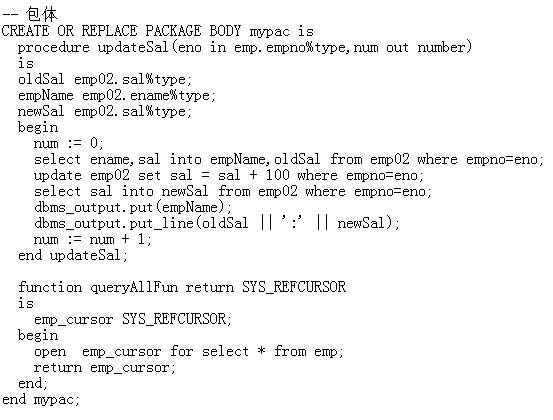
持久性

* 1. 触发器
     1. 
  2. 序列
     1. 是Oracle特有的功能，适用于表主键是整数且递增



* 1. 索引
     1. 提高查询效率，占用磁盘空间，主键默认有索引
     2. 使用场景：1、根据主键查询；2、外键查询；3、where 条件查询
     3. 注意：1、占用磁盘空间；2、数量大的时候，会降低查询效率
  2. 视图
     1. 提高查询效率；是一个表 的投影，占用磁盘空间的

create view emp\_view as select \* from emp; select \* from emp\_view

* 1. 程序包
     1. 
     2. 

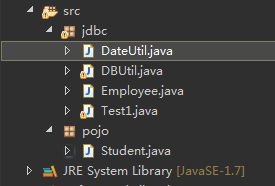
1. JDBC：使用不同语言操作不同的数据库（当前使用java操作Oracle数据库）
   1. 1、将ojdbc14的jar包复制到要创建连接的项目包
   2. 2、右键项目--Build Path --Add Library--JUnit--选择JUnit4--Finally
   3. 3、右键项目--Properties--Libraries--Add JARs--选中粘贴到项目里的ojdbc14的jar包--Apply and Close
   4. 4.创包，创类，编写代码
      1. @Test
      2. 加载驱动

Class.forName("oracle.jdbc.OracleDriver");

* + 1. 获取连接

Connection connection = DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:orcl","用户名","登陆口令");

* 1. 实体类包名
     1. pojo



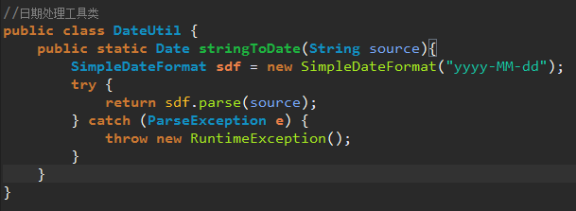
* + 1. domain
    2. entity
    3. bean
  1. 数据库连接池:管理数据库的连接
     1. 先和数据库创建10(自定义大小)连接[连接池],如果操作数据库需要使用连接从连接池中取连接 ,如果暂时不用，将该连接回收到连接池中

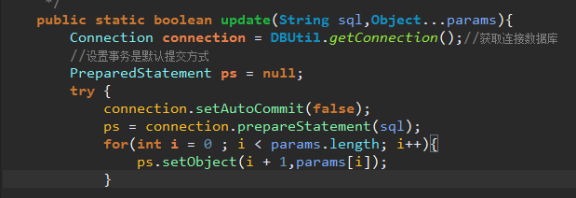
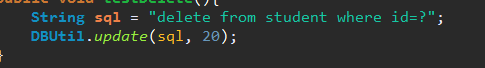
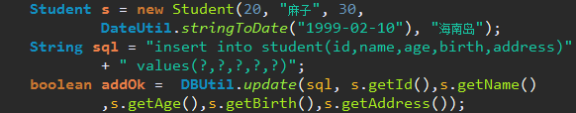
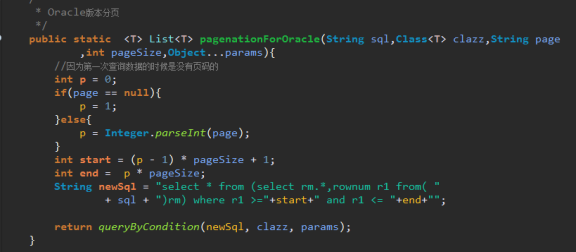
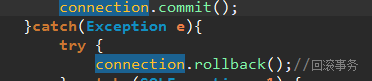


* + 1. 通过方法获取数据库连接方法



* 1. 优化代码过程
     1. 写一个获取连接的通用方法
     2. 使用PreparedStatement替换Statement
     3. 使用对象封装参数
     4. 编写通用的添加方法(注意日期类的处理)
  2. PreparedStatement（表示预编译的 SQL 语句的对象）
     1. 预编译过程
     2. 防止sql注入漏洞
  3. 日期类的工具类
     1. SimpleDateFormat可以选择任何用户定义的日期-时间格式的模式

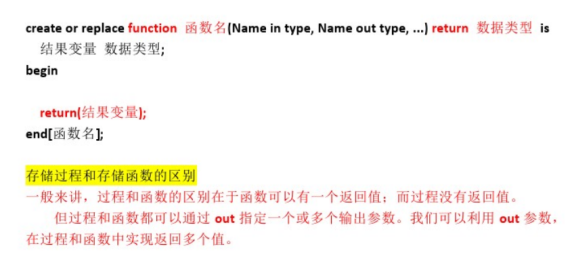


* 1. 通用的增删改方法
     1. 
     2. params[i]是从1开始的
     3. 
     4. 
     5. 
  2. 查询所有数据方法
     1. 
     2. 
  3. 分页工具类
     1. 
  4. 事务提交回滚
     1. 

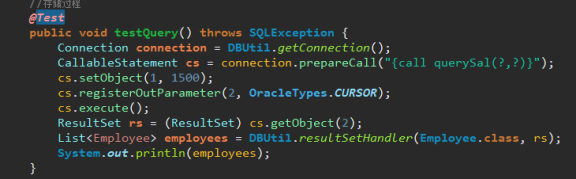
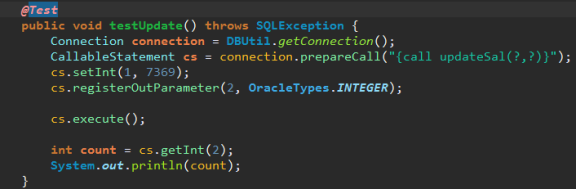
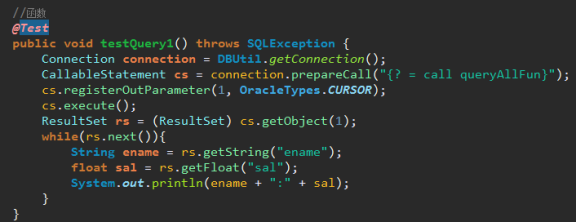
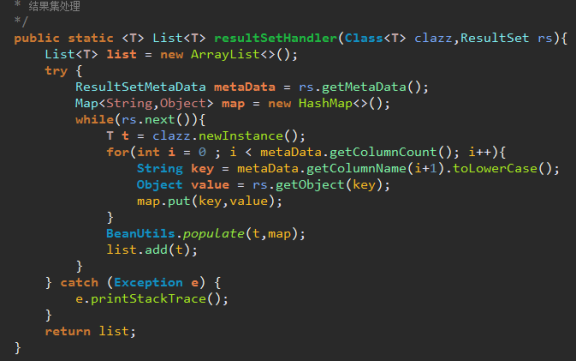
1. PL/SQL编程语句
   1. -- 打开控制台输出流 set serveroutput on; -- 匿名块 declare 说明部分（变量说明，光标申明，例外申明） begin 语句序列（DML语句） end;
   2. 使用匿名块将查询的单行单列的结果放入到变量中
      1. set serveroutput on; declare -- empname varchar2(20); empname emp.ename%type;-- 指定定义的变量和表中对应字段的数据类型一样 begin select ename into empname from emp where empno=7369; dbms\_output.put\_line(empname); end;
   3. 使用rowtype定义一行记录变量
      1. set serveroutput on; declare emp\_row emp%rowtype;-- 记录变量 begin select \* into emp\_row from emp where empno=7369; dbms\_output.put\_line(emp\_row.ename || emp\_row.sal || emp\_row.hiredate); end;
   4. 使用匿名块给定义变量赋值
      1. set serveroutput on; declare empname varchar2(20); esal float; sex varchar2(4):='男'; begin dbms\_output.put\_line(sex); end;
   5. loop循环
      1. set serveroutput on; declare i number:=1; j number:=1; begin loop exit when i = 10; loop exit when j = i + 1; dbms\_output.put(i || '\*' || j || '=' || i \* j || ' '); j := j + 1; end loop; dbms\_output.put\_line(''); --换行 i := i + 1; j := 1; end loop; end;
   6. for循环
      1. set serveroutput on; declare num number:=9; begin for i in 1..num loop exit when i = 10; for j in 1..i loop exit when j = (i + 1); dbms\_output.put(i || '\*' || j || '=' || i \* j || ' '); end loop; dbms\_output.put\_line(''); end loop; end;
   7. while循环
      1. set serveroutput on; declare i number:=1; j number:=1; begin while i <= 9 loop while j <= i loop dbms\_output.put(i || '\*' || j || '=' || i \* j || ' '); j := j + 1; end loop; j := 1; dbms\_output.put\_line(''); --换行 i := i + 1; end loop; end;
   8. 游标遍历结果集
      1. set serveroutput on; declare cursor emp\_cursor is select \* from emp; -- 查询出emp表的结果集，让游标指向该结果集 emp\_row emp%rowtype; begin open emp\_cursor;--打开游标 loop fetch emp\_cursor into emp\_row;--取一行到变量中 exit when emp\_cursor % notfound;--游标关闭条件 dbms\_output.put(emp\_row.ename || ':' || emp\_row.sal); dbms\_output.put\_line(''); end loop; close emp\_cursor;--关闭游标 end;
   9. 存储过程 注意:参数(in/out:输入输出参数)
      1. 

create or replace procedure updateSal(eno in emp.empno%type,num out number) is oldSal emp02.sal%type; empName emp02.ename%type; newSal emp02.sal%type; begin num := 0;--out不可以再is后面定义，只有in可以 select ename,sal into empName,oldSal from emp02 where empno=eno; update emp02 set sal = sal + 100 where empno=eno; select sal into newSal from emp02 where empno=eno; dbms\_output.put(empName); dbms\_output.put\_line(oldSal || ':' || newSal); num := num + 1; end updateSal; set serveroutput on; call updateSal(7369);--传入参数运行语句

-- 利用存储过程查询工资>1500的员工信息 create or replace procedure querySal(eSal in emp.sal%type,outResult out SYS\_REFCURSOR) is begin open outResult for select \* from emp where sal > eSal; end;

* 1. 存储参数 如果无参函数不用加括号
     1. 

create or replace function queryAllFun return SYS\_REFCURSOR is emp\_cursor SYS\_REFCURSOR; begin open emp\_cursor for select \* from emp; return emp\_cursor; end;

* 1. java调用存储过程
     1. 
     2. 
  2. java调用存储函数 区别在于传入参数
     1. 
  3. java封装结果集
     1. 

1. JavaWeb
   1. B/S C/S
      1. B/S:相对不安全，不需更新 （浏览器、服务器）
      2. C/S：相对安全，需要定期更新 （客户机、服务器）
   2. HTML+CSS+JS
      1. HTML

超文本标记语言(标签)

作用：静态页面的展示

标签必须要由浏览器解释并执行

标签

单标签

双标签

h1~h6

table

form:表单

select：下拉列表

span：独占一行

tr定义表格的行

td定义表格单元

option定义下拉列表中的选项

rowspan单元格垂直合并

colspan单元格行合并

input定义输入域

* + 1. CSS

层叠样式表

作用：美化渲染页面

使用场景：内联样式>内部>外部

内联：标签属性

内部：HTML任意位置

外部：通过link标签引入

元素选择器

\*{}通配符

.a{}:class类选择器

id选择器

div{}布局块

层次选择器

E F:查找所有子元素

E>F:只查找第一层级子元素

E+F{ sRules }:选择紧贴在E元素之后F元素。

E~F{ sRules }:选择E元素后面的所有兄弟元素F。

E[att]{ sRules }:选择具有att属性的E元素

E[att="val"]{ sRules }:选择具有att属性且属性值等于val的E元素。

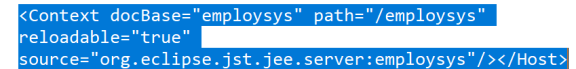
E:active{ sRules }:设置元素在被用户激活（在鼠标点击与释放之间发生的事件）时的样式。

E:checked{ sRules }:匹配用户界面上处于选中状态的元素E。(用于input type为radio与checkbox时)

* + 1. JS（Javascript）

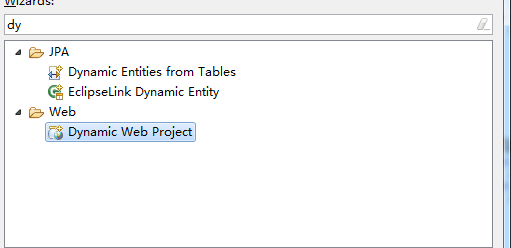
作用：提高和用户的交互效果，页面上的动态效果

get和post区别 get:参数会在地址栏显示[保密性不好],post不会[相对安全]

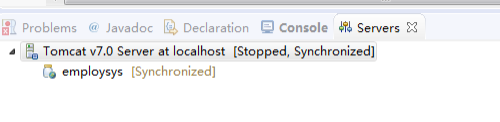


创建一个jsp工程，使用tomcat

创建环境：1、新建web工程



2、创建路径，启动服务器



3、架包放置lib文件夹下

4、新建jsp工程

5、新建servelet

session:域对象 存在服务器端，占用服务器内存的，服务器端技术 1、容器:存放数据 2、作用范围:在一个会话内生效或者在指定生命周期内生效

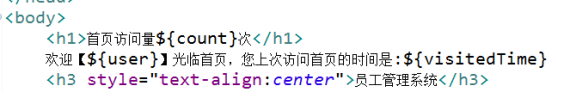
cookie:客户端技术，存在于客户PC机端

ServletContext:域对象 1、容器:存放数据 2、作用范围:与服务器共存亡

跳转页面方式 重定向response:地址栏发生变化，并且请求2次，重定向不可以 转发request:地址栏没有发生变化，请求1次，可以携带值到页面上

Servlet中的域对象 域对象: 1、容器:存放数据 2、作用范围:本次请求未中断生效

jsp界面输出值为${输出参数}



jsp输出日期型



dao层

DAO层叫数据访问层，全称为data access object，属于一种比较底层，比较基础的操作，具体到对于某个表的增删改查，也就是说某个DAO一定是和数据库的某一张表一一对应的，其中封装了增删改查基本操作，建议DAO只做原子操作，增删改查。

service层

Service层叫服务层，被称为服务，粗略的理解就是对一个或多个DAO进行的再次封装，封装成一个服务，所以这里也就不会是一个原子操作了，需要事物控制。

controller层

Controler负责请求转发，接受页面过来的参数，传给Service处理，接到返回值，再传给页面。

bean层

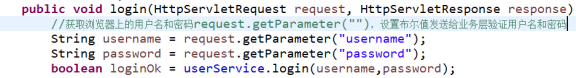
可复用的类

util

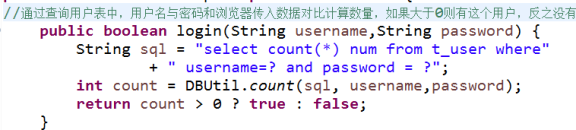
工具类（通用方法层）

* 1. javaweb项目总结
     1. 登录









* + 1. 访问量以及上次登陆时间



* + 1. 分页查询显示

1、设计jsp表格形式

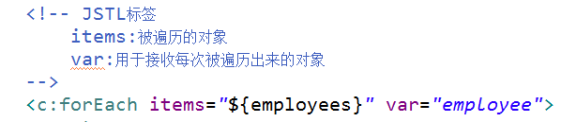
<a href="/employsys/UserServlet?page=1&method=pagenation">首页</a> <a href="/employsys/UserServlet?page=${page - 1 <= 0 ? 1 : page - 1}&method=pagenation">上一页</a> <c:forEach begin="1" end="${pageCount}" var="page"> <a href="/employsys/UserServlet?page=${page}&method=pagenation">${page}</a> </c:forEach> <a href="/employsys/UserServlet?page=${page + 1 >= pageCount ? pageCount : page+1}&method=pagenation">下一页</a> <a href="/employsys/UserServlet?page=${pageCount}&method=pagenation">尾页</a>

2、servlet请求方法



3、业务层，数据层

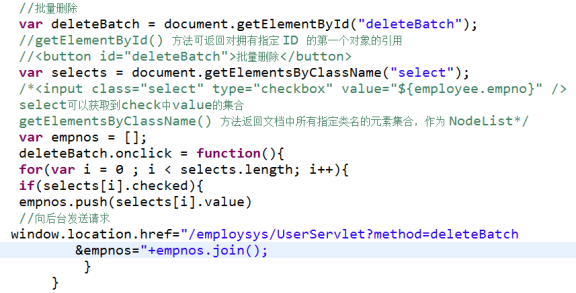
* + 1. 修改

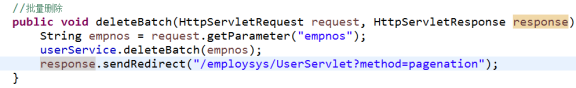


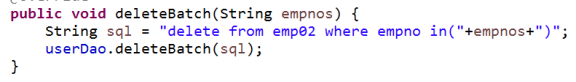




* + 1. 添加
    2. 批量删除







* + 1. 查看
    2. 模糊查询

1、设计jsp



2.servlet



* + 1. ajax局部加载动态



1. 框架基础
   1. Struts2框架
      1. 框架？

框架是产品的半成品

* + 1. 主流框架

Struts2、Spring、Hibernate、Mybatis

* + 1. 分类

web层

封装和接收请求数据，给用户响应

Struts2、xwork、struts1、Springmvc

持久层（Dao）

操作数据库

Hibernate(ORM)、Mybatis、templete

前端UI

easyui、bootstrap

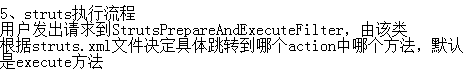
业务层：

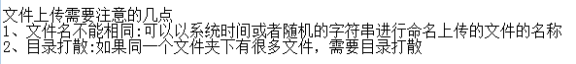
整合框架的作用

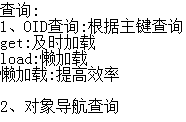
* + 1. Struts2

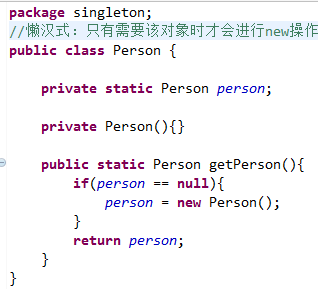
src下建Struts.xml配置文件

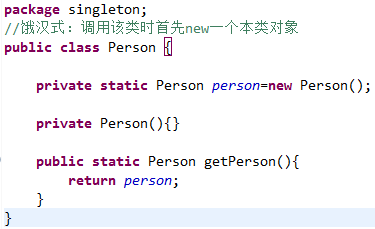
copy需要的配置文件





* 1. Hilbernate框架（持久层）
     1. 
     2. 
     3. Hibernate是单例模式，所谓单例模式就是对象类私有构造方法，禁止 外界创建该类对象，但是提供一个公共方法，返回一个该类对象，有两 种饿汉式、懒汉式两种方法





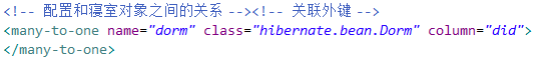
* + 1. 关联查询

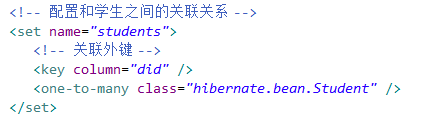
1-1:人和身份证



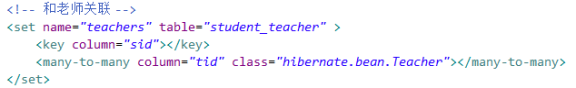


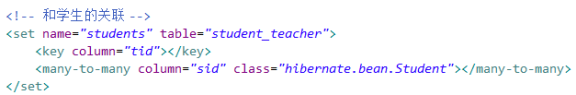
1-n:学生和寝室



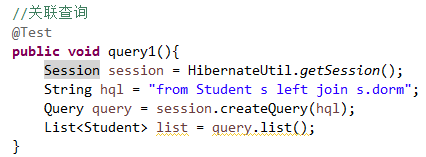


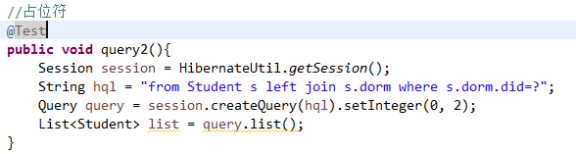
n-n:学生和老师

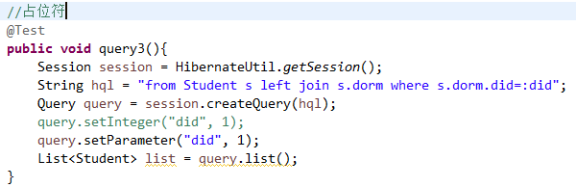


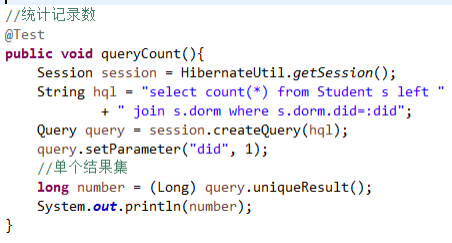


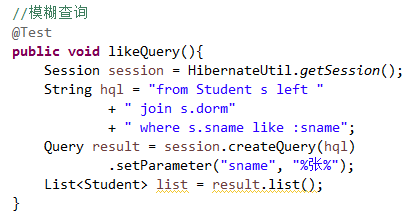
* + 1. 多种查询方法



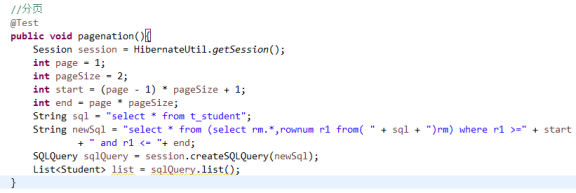








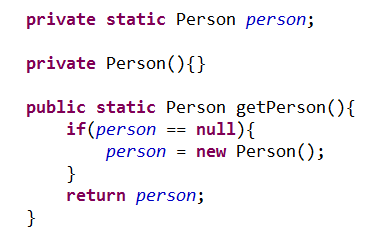
* + 1. 分页



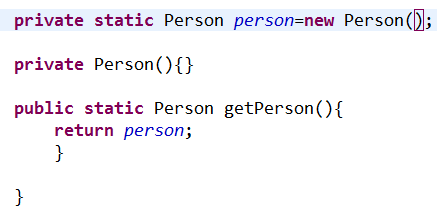
* + 1. 单利模式

懒汉式：线程安全

属性私有；构造器私有；提供方法



饿汉式：线程不安全



* + 1. 缓存







* 1. Spring框架
     1. Spring是一个开源的控制反转和面向切面的容器框架，主要目的是简化企业开发
     2. IOC: Inversion Of Control

由Spring的applicationContext.xml文件创建一个Bean工厂, 根据工厂中的bean标识取出对象

原先程序员自己管理的对象，现在由Spring(Spring容器)管理

@Repositroy,@Controller,@Service,@Component, @Autowired,@Resource,@Qualifier;@Value;

<!-- 开启扫描 --> <context:component-scan base-package="com.bwie"></context:component-scan> </beans>

* + 1. AOP

代理模式

静态代理

动态代理

jdk代理

目标对象一定要实现接口

以实现同个接口方式

cglib代理

目标类不能为final，目标方法不能final/tatic

继承目标类为父类方式

注解方式实现AOP

引入相关jar包

配置文件

注解

@Aspect 执行一个类为切面类

@Before 前置通知，目标方法执行前执行

@After 后置通知，目标方法指定后执行

@AfterReturning 返回后通知，目标方法执行成功后执行

@AfterThrowing 异常通知，目标方法出现异常

@Aroud 环绕通知，环绕目标方法执行

@Pointcut 定义一个切入点表达式用于其他注解引用

配置方式实现AOP

* + 1. DI依赖注入

构造方法赋值

setter方法注入

普通字段

list

map

property

对象

内部bean写法

property内部添加bean，不需要id 外部bean是单例的话内部bean也是单例，外部多例，内部也多例

p名称空间

自动装配

配置在指定的bean节点

配置到整个xml文件

注解方式

@Component

表示一个逐渐，把当前组件加入IOC容器

子类

@Repository 标识当前加入IOC容器的对象时数据访问层对象

@Service 标识当前加入IOC容器的对象时业务逻辑层对象

@Controller 标识当前加入IOC容器的对象时控制对象

@Scope()

指定当前对象在IOC容器中是单例还是多例

@Resource/@Autowrite

做依赖注入，写在属性上面或者setter()方法上

配置