JDK：java开发工具包

jre:java运行环境

JVM:java虚拟机

JDK=JRE+bin

JRE=jVM+API

JVM=类加载器，字节码校验器，解释执行器

2. 局部变量和成员变量之间的区别

a) 定义的位置

成员变量：方法体外面

局部变量：方法体里面

b) 作用范围不同

成员变量：依赖于对象而存在 只要对象没被回收 那么都可以访问

局部变量：只能在定义的那一行开始到所在的方法体结束之前可以访问

c) 默认值的问题

成员变量：即使不赋值也有默认值

局部变量：在使用之前必须先赋值

switch case的参数类型可以传哪些？

答：jdk1.0 可以传char byte short int

jdk5.0 可以传枚举 enum

jdk7.0 可以传String类型

1. System.arraycopy(1,2,3,4,5)方法

1: 要复制的原数组

2：原数组的起始下标位置

3：要复制到的目标数组

4：目标数组的起始下标位置

5：要复制的长度

冒泡排序：

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

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

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

double tmp=data[j];

data[j]=data[j+1];

data[j+1]=tmp;

}

}

}

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

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

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

data[j]=data[j]^data[j+1];

data[j+1]=data[j+1]^data[j];

data[j]=data[j]^data[j+1];

}

}

}

重载重写的区别

首先第一个区别：一个是Override 一个是Overload

第二个区别：对于方法重载的时候 需要发生在同一个类体里面

对于方法覆盖的时候 需要发生在有继承关系的两个类中的子类当中

第三个区别：对于方法重载的时候 方法返回类型没有要求

对于方法覆盖的时候 返回类型5.0之前一模一样 5.0开始变成父类方法返回类型的子类类型

第四个区别：对于方法重载的时候 参数必须不同 包括类型 个数 顺序不同

对于方法覆盖的时候 参数必须和父类要覆盖的方法一模一样

a) 构造方法可以重载吗？

可以

b) 构造方法可以覆盖吗？

不可以方法覆盖

一． 参数传递

1. java里面只有值传递

2. java中的基本数据类型传值 引用数据类型传地址

二． String创建对象方式

1. String str = “Etoak”;

2. String str = new String(“Etoak”);（存在两个对象，str、etoak，str指向etoak）

3. 二者之间的区别是：不new涉及到常量池机制

String20方法：

int length() 得到字符串的字符个数

byte[] getBytes() 将一个字符串转换成字节数组

char[] toCharArray() 将一个字符串转换成字符数组

string[] split(String) 将一个字符串按指定内容劈开

boolean equals(S) 判断两个字符串内容是否相等

boolean equalsIgnoreCase(s) 同上（忽略大小写）

boolean contains(S) 判断字符串是否包含指定内容

boolean startsWith(S) 判断字符串是否以指定内容开始

boolean endsWith(S) 判断字符串是否以指定内容结尾

String toUpperCase() 将一个字符串全部转换成大写

String toLowerCase() 将一个字符串全部转换成小写

String replace(S,S)将字符串里面出现的某个内容全部替换成指定的内容

String replaceAll(S,S)将字符串里面出现的某个内容全部替换成指定的内容[支持正则表达式]

String replaceFirst(S,S)将字符串里面第一次出现的某个内容替换成指定的内容

String trim() 将字符串的前后空格全部去掉

String substring(int x,int y)从下标x对应的字符截取到下标y但是不包含y对应的字符

String substring(int)从指定下标开始截取到字符串最后

String StringBuffer StringBuilder的区别

keyword: 缓冲区 缓冲空间

String 没有预留任何缓冲空间

StringBuffer 底层大量使用了synchronized修饰符

同一时间只允许一个线程进行操作

效率相对较低 但是不会出现并发错误

StringBuilder 同一时间允许多个线程同时进行操作

效率较高 但是可能出现并发错误

单例模式

单例模式 之 醉汉式

class Moon{

//私有话构造方法 -》 防止类体之外随意的创建对象

private Moon(){}

//创建一个私有的 静态的 属于本类类型的对象

private static Moon only = new Moon();

//提供一个公共的静态的 返回本类类型的对象

public static Moon getOnly(){//Moon.getOnly();

return only;

}

}

单例模式 之 懒汉式

class Sun{

private Sun(){}

private static Sun only;

public static synchronized Sun getOnly(){

if(only == null)

only = new Sun();

return only;

}

}

接口和抽象类的区别

接口当中定义的变量 默认添加三个修饰符public static final

接口当中定义的变量默认就是常量

抽象类当中定义的变量是普通的属性 每个这个类型的对象都有一份

接口当中定义的方法 自动添加两个修饰符 public abstract

抽象类当中定义的方法可以是非抽象方法 也可以是抽象方法

\*:从JDK8.0开始 接口当中的方法可以有具体的方法实现

a> static修饰的静态方法

b> default修饰的默认方法

集合：List: ArrayList LinkedList Vector Stack

Set: HashSet

SortedSet: TreeSet

Map: HashMap Hashtable

SortedMap: TreeMap

Collection : add(obj) remove(obj) contains(obj) clear() iterator() size()

List: get(int) remove(int)

Map: put(k,v) remove(k) get(k) containsKey(k) containsValue(v) clear() size()

keySet() values() entrySet()

LinkedList和ArrayList的区别

ArrayList底层使用数组实现的，优势是遍历查找和随机访问效率较高，劣势是添加或删除元素效率较低

LinkedList底层使用双向循环链表实现的，优势是添加或删除元素效率较高，劣势是遍历查找和随机访问效率较低

注意：尽量避开LinkedList的get(下标)方法

HashMap和Hashtable的区别

同步特性不同

首先HashMap同一时间允许多个线程同时进行访问，效率相对较高，但是可能出现并发错误

Hashtable底层大量的使用了synchronized修饰方法，同一时间只允许一个线程对其操作，效率相对较低，但是不会出现并发错误

\*：从jdk5.0开始集合的工具类Collections当中出现了一个静态方法synchronizedMap方法可以将一个不安全的map变成线程安全的

\*：在高并发的场景下推荐使用java.util.concurrent.ConcurrentHashMap 有更高的性能~

对于null的处理不同

然后，他们对null的处理方式不同，HashMap无论是主键还是值都能存放null，

但是由于主键要求唯一，所以主键只能存放一个null，但是值能存放多个空

Hashtable无论是主键还是值都不能添加null，会触发空指针异常

底层实现的区别

底层有点不同 HashMap分组组数可以指定，默认分为16个小组，但是最终的分组组数一定是2的n次方数，

在计算散列的时候，底层使用&(分组组数-1) [按位与运算]

Hashtable分组组数可以随意指定，默认分11组，可以随意指定分组模分组组数

出现的版本不同

HashMap jdk1.2 Hashtable jdk1.0

共享数据三种：

1.静态变量2.利用参数传递3.使用内部类

2. Error和Exception的区别

Error : 通常是指底层硬件或系统所导致的

程序员通过代码无法解决的问题

Exception : 程序运行过程当中出现的例外情况

3. 运行时异常和非运行时异常的区别

运行时异常 : 在编译的时候 不做任何提示直接通过编译

直到运行的时候 问题直接体现出来 - RuntimeException

非运行时异常 : 编译的时候必须明确该如何处理

否则根本无法通过编译 – Exception

4. 常见的运行时异常

ArithmeticException => 算术异常

NegativeArraySizeException => 负的数组大小异常

ArrayIndexOutOfBoundsException => 数组索引值超出边界异常

NullPointerException => 空指针异常

ClassCastException => 类造型异常

IllegalArgumentException => 非法参数异常

IndexOutOfBoundsException => 索引值超出边界异常

NumberFormatException => 数字格式异常

IllegalStateException => 非法状态异常

NoSuchElementException => 没有指定元素异常

CocurrentModificationException => 并发修改异常

处理异常：throws Exception try{}catch{}

创建异常：throw new Exception（XXX）

线程过程：新生，就绪，阻塞，运行，消亡

线程实现：

1 extends Thread{

@Override

public void run(){

System.out.println("first");

}

}

2. implements Runnable{

@Override

public vois run(){

System.out.println("seconde");

}

}

3. implements Callable<String> {

@Override

public String call()throws Exception{

System.out.println("third");

return "over";

}

}

newfixedThreadPool()可重用的线程

newCachedThreadpool()带缓存

newsingleThreadExecutor()单一实例

并发错误解决?

synchronized修饰符，修饰代码块，方法

java.util.concurrent.locks.ReentrantLock:方法lock（），unlock（），

5. 锁池和等待池的区别

进入的时候是否需要释放资源 锁池不释放 等待池释放

离开的时候是否需要调用代码

锁池不需要

等待池必须要唤醒

离开之后去到哪里

离开锁池后直接前往就绪

离开等待池后直接前往锁池

时间解码：SimpleDateFormat f=new SimpleDateFormat("yyyy-MM-dd HH:mm:ss");

Calendar cal = Calendar.getInstance();

cal.setTimeInMillis(t);年加1900月加1

Date now = new Date(t);年加1900月加1

端口号：Oracle 1521 MySQL 3306 Tomcat 8080

Sql sever 1433 ssh 22 ftp 21 telent 23

数据库数据类型

char(长度) 固定 2000B ANSI 一个汉字占两个字节

nchar(长度) 固定 2000B 本地国家字符集 一个汉字占一个字节

varchar(长度) 可变 4000B 2000个汉字

varchar2(长度) 可变 4000B 只有oracle识别

nvarchar(长度) 可变 4000B oracle不支持

nvarchar2(长度) 可变 4000B 只有oracle识别

3. 事务的特性

原子性：事务是一个整体 不可分割

一致性：事务中操所的数据及类型要与数据库保持一致

持久性：事务能够持久化数据

隔离性：事务与事务之间操作一个资源可能存在的问题

剧组函数 max sum avg count min

1. 左外连接

from 左表 left [outer] join 右表 on 关联关系

\* 左表关联上右表的数据(内连接)+左表关联不上右表的左表数据

1. 序列的作用

\* 一个单独的数据对象，是一个能够生成有序的整数列值的对象

\* oracle通过调用序列的形式来实现主键自增、

2. 序列的两个属性

currval: 序列生成到的当前值

nextval: 序列生成的下一个值

\*\*\* 在一个新会话(一次连接)中，必须先调用下一个值才能获得当前值

3. 创建序列的语句

create sequence 序列名;

minvalue 1

maxvalue 9999999999999999999999999999

start with 1 --start with是下一次缓冲位置

increment by 1 --每次增长多大

cache 20 --缓冲大小(不缓冲则用nocache)

nocycle -- 是否循环增(cycle)

索引: 类似于一个目录,但是不只是目录,能够帮助我们提高查询效率.

语句: create index 索引名 on 表名(列名,列名); 普通索引

unique index 唯一性索引

bitmap index 位图索引

create index 索引名 on 表名(函数名(列名)); 函数索引

1. 视图的有关特性

\* 是一张虚拟表 不是真实的表

\* 视图是将编译后查询语言保存到数据库中，下次调用视图

可以不需要编译，直接执行从原表中获得数据

2. 创建视图

create view 视图名 as select 列名 from 表名 [where 条件];

3. 调用视图

select \* from viewname;

4. 为什么使用视图？

\* 视图是编译后的查询语言，可以节省编译时间，提高查询效率。

\* 视图能够屏蔽原表中的字段：避免没有权限的用户查询其他字段。

\* 视图中的数据能够动态来源于原表

\* 简单视图能够更新原表中的数据;

存储过程：在服务器端，能够被一个或多个应用程序调用的一段SQL语句集

3. XML用途

保存数据

传输数据

\* 配置文件解析xml

解析xml

1.dom4j:dom sax

2.Document

3.xmlWriter

4.SaxREader

所有的数据库如果要支持JAVA语言必须实现此接口

oracle: oracle.jdbc.driver.OracleDriver

mysql : com.mysql.jdbc.Driver

链接：

oracle: jdbc:oracle:thin:@localhost:1521:orcl

mysql : jdbc:mysql://localhost:3306/etoak

Sql优化提高效率

1可以使用索引

2 SQL语句如果是多表关联可以将关联条件写在左侧,过滤条件尽量写在右侧

3 避免在子查询语句中使用in some any all等关键字

可以使用多表关联替代

4 避免使用like '%%'写法,这种写法或造成全盘扫描

Css在页面的几种

1、外链式

<link type = "text/css" href=" rel="stylesheet"/>

2、内嵌式

书写在head标签内

<style type="text/css"></style>

3、行内式

书写在标签内的

style=""

jQuery的选择器、怎么用

$("标签名")直接使用标签名作为选择元素的依据

$(".class名直接使用.class属性名作为选择元素的依据

$("#id名")直接使用#id属性名作为选择元素的依据

$("标签#id名") $("标签.class名")必须同时满足标签带有id或者相应class的元素

$("选择器,选择器,选择器")只要符合其中任意一个选择器就可以成功选取

$("选择器 选择器 选择器")根据左祖先右后代的原则精确选取子元素

$("\*")选择全部元素

$("选择器 > 选择器")仅仅选取字辈元素

$("选择器 + 选择器")从上向下选取必须紧邻必须互为兄弟

$("选择器 ~ 选择器")向下选取所有的兄弟元素，不需要紧邻

表单设置

enctype:表示使用流上传文件

multipart/form-data：表示使用字节流传递文件

application/x-www-form-urlencoded:表示使用字符流上传文件

<form action="servlet/Upload" method="post" enctype="multipart/form-data"></form>

封装json的三种方式

1.JsonObject(任意数据，复杂程度高) 2.JsonObject.fromObject(Map和自定义数据类型，复杂程度低) 3.JsonArray.fromObject(list,set,数组)

$.ajax

$.ajax({

//1.要提交到的目的地

url:"servlet/GetPage",

//2.提交的方式

type:"post",

//3.要传递的值

data:"currPage="+currPage+"&pageSize="+pageSize,

//4.返回的数据类型

dataType:"json",

success:function(data){

//此处返回的数据不再是一个String,而是已经被转换的object对象

JDBC：

1加载驱动：Class.forName是根据指定名字加载相应的类、Interface

2.获得链接：connection con=DriverManager.getConnection(url,user,pwd);

3.获取执行器PrepareStatement

4.执行 5.处理结果6.关闭资源

数据库连接池的工作原理？

当服务器启东时，创建一些连接放在连接池中等待客户端请求；如果客户端请求连接，则首先判断连接池中是否有可用连接，付过有则返回，如果没有则，判断当前连接数是否小于最大可用连接数，如果当前连接数小于最大可用连接数，则创建新的连接返回，如果当前连接数不小于最大可用连接数，则抛出无可用连接的异常，当客户端使用完连接之后，将连接放回到连接池中，从而实现连接的重用。

反射是JAVA语言提供的一套在运行期动态的获得类中信息的API。获得类中的属性和方法；执行类中的方法;创建类的对象；判断对象的类型

.getClass(), 得到Class cls = Emp.class;

得到发方法Method m1=cls.getDeclaredMethod();

Object value= m1.invoke（）

hibernate和mybatis的区别

hiberbate注重Orm思想，自动生成Sql语句，自动执行，自动返回结果

Mybatis注重sql语句灵活，自己书写sql语句，自动执行，自动返回结果。

Struts2流程？是不是单例的？Struts1用过吗？

Struts2处理请求的流程:容器启动时，首先启动Struts2的核心过滤器，来加载配置文件等待客户端请求，如果客户端发出请求，Struts2首先根据请求的名字来到struts.xml中寻找对应action【根据name属性来查找】，如果找到对应的，则创建class属性指定的类，并且根据method属性值调用对应的方法【如果没有method属性值，则调用execute方法】，然后再根据方法返回的字符串找到元素的子元素，根据中的name 和type属性跳转到其他的action或者页面。

拦截器必须实现Interceptor接口，就如同一个过滤器必须实现Filter接口是一样的

Spring：解决传统的javaEE开发的复杂性而提出的，管理javabean，整合其他框架，提高系统的维护和扩展性

核心IOC，依赖注入，控制反转。

核心AOP，动态代理，jdk和cglib

注解：Controller，Service，RequestMapping，Autowried

什么是事务?

持久层对一组sql进行统一的提交或回滚操作,以此保证数据执行的一致性!

事务的特性?

原子性 一致性 持久性 隔离性

注解事务

使用@Transactional注解，该注解既可以使用在类上，也可以使用在方法上。使用在类上表示该类的所有方法都设置事务，使用在方法上表示对特定的方法设置事务。

1.bootstrap:twitter 推特工程师 写的一套融合JS/HTML/CSS 的前端框架

Bootstrap 是最受欢迎的 HTML、CSS 和 JS 框架，用于开发响应式布局、移动设备优先的 WEB 项目

2.bootstrap:响应式布局 兼容移动设备

3.响应式布局？

相同的内容，根据显示设备的不同，显示不同的效果。

Springmvc和Struts有什么区别？

spring中用过的注解有哪些？分别有什么作用？@Autowired和@Resource有什么区别?

@Component @Service @Controller @Repository

类级别的注解;被标注的类会成为ioc容器实例化的对象;相当于在ioc容器中手动配置bean

@Controller也常用在spring-mvc中,标注请求处理器

@Autowired @Resource

标注属性或者setter方法

实现bean对象之间的被动注入

Autowired默认按类型装载;是spring内置的;如果想使用名字装载，可结合@Qualifier注解进行使用

@Autowired

@Qualifier("empDao")

private EmpDaoIf empDao;

Resource默认按名字装载;是javaEE提供的

@Resource(name="empDao")

private EmpDaoIf empDao;

@RequestMapping(path)

标注请求处理器中的方法，使用被标注的方法处理path指向的请求

当然也可以使用它标注Controller类，表示这个类中的方法处理url为前缀的请求

@ResponseBody

标注请求处理器中用于处理ajax请求的方法，将被标注方法的返回值封装成json返回给客户端的js

@RequestParam(paramName)

标注请求处理器方法中的参数，使用该参数封装'paramName'指向的参数

@SessionAttributes({key1,key2...})

标注请求处理器，将该处理器方法中添加到ModelMap中的数据key转存到session范围中

注意:数组中指向的key值不能和方法的参数名重名！

@Aspect

@Pointcut

@Before

@After

@AfterReturning

@AfterThrowing

@Around

@Transactional

redis怎样使用的？怎样存储数据的？主要存储的什么类型？（不是说以键值对的形式）

首先在服务器中启动redis服务 : redis-server.exe redis.conf

连接redis可以采用spring提供的spring-data-redis组件，该组件封装了jedis访问redis服务器的过程 ：

首先在ioc容器中配置redis连接池(JedisPoolConfig)、连接工厂(JedisConnectionFactory)和模板类对象(RedisTemplate)

然后将RedisTemplate注入给业务对象，通过使用模板类实现对redis服务器的连接和操作。

存储数据 ：

redisTemplate.set(key , value) get(key)

rt.opsForList() -> ListOperations -> rightPush() leftPush() range()

rt.opsForSet() -> SetOperations -> add() members()

rt.opsForHash() -> HashOperations -> put(key,value-key,value) entries(key) get(key,value-key)

1. 拦截器是基于java的反射机制的，而过滤器是基于函数回调。

2. 拦截器不依赖与servlet容器，过滤器依赖与servlet容器。

3. 拦截器只能对action请求起作用，而过滤器则可以对几乎所有的请求起作用。

4. 拦截器可以访问action上下文、值栈里的对象，而过滤器不能访问。

5. 在action的生命周期中，拦截器可以多次被调用，而过滤器只能在容器初始化时被调用一次

拦截器的代码实现(以struts2为例)：

jsp流程

Web容器将JSP转译成Servlet源代码。

Web容器将产生的源代码进行编译。

Web容器加载编译后的代码并执行。

把执行结果响应至客户端。

Springmvc工作原理

1. 客户端发送请求（login.do）,找到

DispatcherServlet并解析xml信息。

2. 找到处理映射器（<mvc:annotation-driven>），然后找到对应的handler（@RequestMapping("/login")）

3. 找到handler之后返回给DispatcherServlet，DispatcherServlet带着handler找对应的适配器（<mvc:annotation-driven>），也就是找后台方法（login），然后接收参数，并对参数进行转换处理，处理对应的业务逻辑。

4. 业务逻辑处理完之后，返回视图（ModelAndView），

将ModelAndView交给DispatcherServlet，DispatcherServlet拿到视图之后，

进行视图解析器解析（InternalResourceViewResolver），解析视图之后，转交给DispatcherServlet，DispatcherServlet拿到解析好的视图，交给view类，然后跳转真正视图，并返回给客户端。

springmvc和struts2区别

1.web.xml配置不一样，springmvc使用servlet元素

，struts2使filter元素Filter是过滤器，初始化加载，

Servlet是服务器请求，初始化不加载，发送第一次请求时，初始化。

2.接收请求参数，springmvc是局部变量，struts是成员变量

3.Springmvc不依赖拦截器，struts2依赖拦截器，

4.springmvc自己封装json，ObjectMapper，使用的注解

接收json参数注解@requestbody

响应json参数注解@responsebody，

Struts2集成了第三方json插件。

5. springmvc默认单例，也可以设置非单例@scope

struts2每次发送请求都会new一个新对象，struts2和spring整合也可以设置单例@scope

Session中get方法和load方法的区别?

1.get:立即加载模式，load是延迟加载模式，当执行到get方法时，会立即发出SQL语句，

当执行到load方法时，不会立刻发出SQL语句，只有当真正使用到时才会发出SQL语句。

2.当数据库中没有对应记录时，get返回null，load返回ObjectNotFoundException

3.Session中的close\clear\evict 方法的区别？

close:关闭session

clear:清空缓存

evict:从缓存中将指定的对象清除

说一下hibernate的缓存机制

A：hibernate一级缓存

（1）hibernate支持两个级别的缓存，默认只支持一级缓存；

（2）每个Session内部自带一个一级缓存；

（3）某个Session被关闭时，其对应的一级缓存自动清除；

B：hibernate二级缓存

(1) 二级缓存独立于session，默认不开启；

hibernate的配置文件：hibernate.cfg.xml;hibernate.properties;

hiberbate中的sql和Hql

Sql:面向数据库表，查询字段不区分大小写，关联条件根据类中的属性来

Hql:面向对象查询，查询字段区分大小写，关联条件是根据事先写好的主外键关系。

得到复选框的值

Jquery:

$(document).ready(function(){

var checked = [];

$("#submitButton").click(function(){

$('input:checkbox:checked').each(function() {

checked.push($(this).val());

});

alert(checked);

});

});

js:function fun(){

obj = document.getElementsByName("test");

check\_val = [];

for(k in obj){

if(obj[k].checked)

check\_val.push(obj[k].value);

}

alert(check\_val);

}

Js:

document.getElementById("id").value;//根据 Id获得对象中的值

document.getElementByName("name")[0].value;//根据name获得对象中的值

document.getElementsByTagName("input")取得所有input对象后过滤，也可以通过其它一些浏览器特定的方法去取（获得input的标签的所有对象）

document.getElementById("address").innerHTML;

document.getElementsByName("formName")[0].submit();

document.getElementById("id").getAttribute("calss");//获取id的对象的class的值

jquery:

获得input的值.val()

获得div的值.text()，.html(),.atttr(atrr);