1. **String StringBuffer StringBuilder 的区别？**
2. String：字符串常量，字符串长度不可变。Java中String是immutable（不可变）的。
3. StringBuffer：字符串变量（Synchronized，即线程安全）。如果要频繁对字符串内容进行修改，出于效率考虑最好使用StringBuffer，如果想转成String类型，可以调用StringBuffer的toString()方法。出现在JDK1.0
4. StringBuilder：字符串变量（非线程安全）。在内部，StringBuilder对象被当作是一个包含字符序列的变长数组。出现在JDK1.5
5. **HashMap和HashTable的区别？**

1.同步特性不同

HashMap 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

Hashtable 同一时间只允许一个线程进行操作 效率较低 但是不会出现并发错误

\*:JDK5.0开始 Collections.synchronizedMap(hashMap);

2.对于null的处理不同

HashMap 无论主键还是值 都是可以存放null的 只不过主键要求唯一

所以只能存放一个null

Hashtable 无论主键还是值 都不允许存放null 会出现异常NullPointerException

3.底层实现的细节不同

HashMap 默认分为16个小组 分组组数可以指定 但是最终结果一定是2的n次方

正因为分组组数都是2的n次方 所以可以使用 &(分组组数-1)来计算散列

Hashtable 默认分为11个小组 分组组数完全可以随意指定

正因为如此 它不能使用&(分组组数-1) 而只能使用%分组组数

4.出现的版本不同

HashMap since JDK1.2

Hashtable since JDK1.0

1. **==和equals的区别？**

1.==首先是一个运算符。

2.equals则是一个String对象的方法，可以.出来

两种比较：

1.基本数据类型比较：==和equals都是比较两个值是否相等，true和false

2.引用对象比较：==和equals都比较内存中的地址是否相等，true和false

1.string是一个特殊的引用类型。对于两个字符串的比较，不管是 == 和 equals 这两者比较的都是字符串是否相同；

2.当你创建两个string对象时，内存中的地址是不相同的，你可以赋相同的值。

所以字符串的内容相同。引用地址不一定相同，（相同内容的对象地址不一定相同），但反过来却是肯定的；

3.基本数据类型比较(string 除外) == 和 Equals 两者都是比较值；

1. **HashMap如何调整性能和空间的取舍？**
2. **List set和Map存取元素的特点以及你所知道的实体类？**

Set中的对象不按特定方式排序，并且没有重复对象。但它的有些实现类能对集合中的对象按特定方式排序，例如TreeSet类，它可以按照默认排序，也可以通过实现java.util.Comparator<Type>接口来自定义排序方式。  
List中的对象按照索引位置排序，可以有重复对象，允许按照对象在集合中的索引位置检索对象，如通过list.get(i)方式来获得List集合中的元素。  
　Map中的每一个元素包含一个键对象和值对象，它们成对出现。键对象不能重复，值对象可以重复。

1. **创建线程有几种方式？分别是？**

1.承继自Thread类，代码逻辑写在子线程中，需要重写run()方法，主线程里start()就可以了

package com.myjava.thread;

public class ExtendsThread extends Thread{

private final static int THREAD\_NUM = 5;

public static void main(String[] args){

for (int i = 0; i <THREAD\_NUM; i++) {

new ExtendsThread("thread"+i).start();

}

}

public ExtendsThread(String name){

super(name);

}

@Override

public void run() {

// TODO Auto-generated method stub

for (int i = 0; i < this.THREAD\_NUM; i++) {

System.out.println(this.getName()+i);

}

}

}

2，实现Runnable接口

package com.myjava.thread;

public class ImplRunnable implements Runnable {

private static final int THREAD\_NUM = 5;

@Override

public void run() {

for (int i = 0; i < THREAD\_NUM; i++) {

System.out.println(Thread.currentThread().getName()+i);

}

}

public static void main(String[] args) {

// TODO Auto-generated method stub

for (int j = 0; j < THREAD\_NUM; j++) {

ImplRunnable implRunnable= new ImplRunnable();

new Thread(implRunnable,"thread"+j).start();

}

}

}

3，通过线程池

package com.myjava.thread;

import java.util.concurrent.ExecutorService;

import java.util.concurrent.Executors;

public class ThreadPool {

private static int POOL\_NUM = 10;

public static void main(String[] agrs){

ExecutorService executorService = Executors.newFixedThreadPool(5);

for (int i = 0; i < POOL\_NUM; i++) {

RunnableThread thread = new RunnableThread();

executorService.execute(thread);

}

}

}

class RunnableThread implements Runnable{

private int THREAD\_NUM = 10;

public void run() {

for (int i = 0; i <THREAD\_NUM; i++) {

System.out.println("线程"+Thread.currentThread()+i);

}

}

}

1. **ArrayList 与LinkedList的区别？**

ArrayList采用的是数组形式来保存对象的，这种方式将对象放在连续的位置中，所以最大的缺点就是插入删除是非常麻烦。

LinkedList采用的将对此昂存放在独立的空间中，而且在每个空间中还保存下一个链接的索引 但是确定就是查找非常麻烦 要从第一个索引开始

1. **接口抽象类的区别？**

接口完全是抽象的，不存在方法的实现

抽象类有默认的方法实现

接口不能有构造器

抽象类可以有构造器

接口是完全不同的类型

抽象类除了不能实例化外，他和普通Java类没有任何区别

接口只可以继承一个或多个其他接口

抽象类可以继承一个类和实现多个接口

接口没有main方法，我们不能直接运行

抽象类有main方法，我们可以直接运行

1. **请介绍下你说所了解的设计模式?**

工厂方法模式 抽象工厂模式 单例模式 原型模式

适配器模式

1. **如何去遍历一个Map？**
2. 把值放在一个集合中，然后遍历集合

import java.util.Collection;

import java.util.Iterator;

import java.util.Map;

import java.util.HashMap;

/\*\*

\* 1.把值放到一个集合里，然后遍历集合

\*/

public class TraversalMap1 {

private static Map<Integer, String> map;

private static TraversalMap1 getMap(){

map = new HashMap<Integer, String>();

for(int i = 0 ; i < 10 ; i ++){

if(map.isEmpty()) map.put(i, "String " + i);

else if(map.get(i) == null) map.put(i,"String " + i) ;

}

return null;

}

public static void main(String[] args){

getMap();

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

Iterator i = c.iterator();

for(; i.hasNext() ;){

System.out.println(i.next());//遍历出map内的所有value

}

}

}

1. 把key放在一个集合中，遍历key值同时根据key得到值

import java.util.Set;

import java.util.Iterator;

import java.util.Map;

import java.util.HashMap;

/\*\*

\* 2.把key放到一个集合里，遍历key值同时根据key得到值

\*/

public class TraversalMap2 {

private static Map<Integer, String> map;

private static TraversalMap2 getMap(){

map = new HashMap<Integer, String>();

for(int i = 0 ; i < 10 ; i ++){

if(map.isEmpty()) map.put(i, "String " + i);

else if(map.get(i) == null) map.put(i,"String " + i) ;

}

return null;

}

public static void main(String[] args){

getMap();

Set set = map.keySet();

Iterator i = set.iterator();

for(; i.hasNext() ;){

System.out.println(i.next());//遍历出map内的所有key

}

}

}

1. 把一个map对象放到entry中，然后根据entry同时得到key和value

import java.util.Map;

import java.util.HashMap;

/\*\*

\* 3.把一个map对象放到放到entry里，然后根据entry同时得到key和value

\*/

public class TraversalMap3 {

private static Map<Integer, String> map;

private static int num = 0 ;

private static String str = "";

private static TraversalMap3 getMap(){

map = new HashMap<Integer, String>();

for(int i = 0 ; i < 10 ; i ++){

if(map.isEmpty()) map.put(i, "String " + i);

else if(map.get(i) == null) map.put(i,"String " + i) ;

}

return null;

}

public static void main(String[] args){

getMap();

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

num = entry.getKey();

str = entry.getValue();

System.out.println("the map key is : " + num + " || the value is : " + str);//显示出map的key和value

}

}

}

1. **Left join，right join 和 inner join 的区别？**

左外连接：from 旧表 left [outer] join 新表 on 关联关系

\* 旧表能够级联上新表的数据+两张表没有关系的旧表数据

select e.\* ,d.\*

from emp e

left join dept d on e.deptno = d.deptno;

右外连接：from 旧表 right [outer] join 新表 on 关联关系

\* 新表能够级联上旧表的数据+两张表没有关系的新表数据

select e.\* ,d.\*

from emp e

right outer join dept d on e.deptno = d.deptno ;

等值连接：只返回两个表中联结字段相等的行

select \* FROM table1 INNER JOIN table2 ON table1 . field1 compopr table2 . field2

1. **怎样去创建索引 什么样的字段适合创建索引？**

索引：index

\* 类似于一个目录结构，能够帮助我们快速检索数据。

\* 索引是被单独存放在索引块中

具有索引的列上的值会和rowid单独存放，通过rowid能够快速定位到某条数 据

\* 数据库会在具有唯一性的列上自动添加唯一性索引

创建索引：

create [unique/bitmap] index 索引名 on 表名(列名1,列名2..);

\* 如果一个索引作用在多个列上，我们成为复合索引/联合索引。

\* 普通索引 index

\* 唯一性索引 unique index

\* 位图索引 bitmap index

1. **如何创建序列，如何指定一个序列的步长(每次增长多少)**

创建序列：

\*\*简单创建语句：

create sequence 序列名;

【案例】

create sequence seq\_stu;

\*\*完整语句：

create sequence SEQ\_STU

minvalue 1 --最小值

maxvalue 999999999999999999999999999

start with 1 -- 下一次从哪个位置开始缓冲数据

increment by 1 -- 每次增长多大

cache 20 -- 每次缓冲多少个数

cycle -- 是否循环增长

1. **如何优化SQL语句？**

sql优化

1 尽量避免在 where 子句中对字段进行 null 值判断,否则将导致引擎放弃使用索引而进行全表扫描

2 尽量避免在 where 子句中使用!=或<>操作符，否则将引擎放弃使用索引而进行全表扫描。

3 尽量避免在 where 子句中使用 or 来连接条件，否则将导致引擎放弃使用索引而进行全表扫描

4 尽量避免使用in some any all 等子查询的用法

5 尽量避免在where之后的条件判断中符号左侧出现计算表达式的情况，负责条件中的列无法使用索引

6 尽量在where之后的条件中避免使用函数的形式

7 可以使用视图来加速查询

8避免使用select \*

1. **数据库当中in和exists的区别是什么？那个效率高？**

多行子查询：

\* 如果子查询语句能够查询出多个结果，则需要使用in some any all exists

\* in 表示外层的条件符合子查询结果中的某个值就成立

\* some 和any的用法与in相同，

in用在无符号的语句中

some或any用在有符号的语句中

\* all 表示比最大值大 比最小值小

\* exists 表示只要子查询结果有数据即成立

In是把外表和内表作hash链接，而exists是对外表作loop循环，每次loop循环再对内表进行查询。

一直以来认为exists比in效率高的说法是不准确的。如果查询的两个表大小相当，那么用in和exists差别不大。

如果两个表中一个较小，一个是大表，则子查询表最大的用exists,子查询表小的用in

1. **Mysql和Oracle如何实现分页?**

\* limit之后如果只有一个参数，表示查询前多少条数据

\* limit之后如果有两个参数c1,c2，表示从>c1的位置开始，数c2条数据

\*Select rownum,a.\* from(select e.\* from emp e order by sal desc)a

1. **在2亿条数据中取其中相同的两条数据，你要怎么取？**
2. **Union 和unionall的区别？**

union all: 重复数据查询多次

select \* from emp where ename like '%E%'

union all

select \* from emp where ename like '%S%';

Union :重复数据只计算一次

1. **数据库中如何将字符串转换成日期格式？**

to\_date(c1,c2) 按照c2指定的格式，将c1转变为日期类型

1. **简述悲观锁和乐观锁？**

悲观锁（pessimistic lock）,每次去拿数据的时候都认为别人会修改，所以每次在拿数据的时候都会上锁，这样别人想拿这个数据就会block直到它拿到锁。传统的关系型数据库里边就用到了很多这种锁机制，比如行锁，表锁等，读锁，写锁等，都是在做操作之前先上锁。

乐观锁（Optimistic Lock），每次去拿数据的时候都认为别人不会修改，所以不会上锁，但是在更新的时候会判断一下在此期间别人有没有去更新这个数据，可以使用版本号等 机制。乐观锁适用于多读的应用类型，这样可以提高吞吐量，像数据库如果提供类似于write\_condition机制的其实都是提供的乐观锁。

乐观锁适用于写比较少的情况下，就是冲突很少发生的时候，这样可以省去锁的开销，加大了系统的整个吞吐量。

悲观锁适用于经常产生冲突的情况

1. **在JS中怎么创建提个数组对象？**

Var arr=new Array(“sunday”,”monday”);

1. **什么情况下使用Ajax？**

请求提交是为了页面数据显示，用户不希望看到页面刷新

1. **JQuery循环数组？**
2. **下拉列表的级联如何实现？**

利用ajax，前台接收从后台返回的json数据，然后前台遍历显示，然后根据外键id拿取级联数据，对应显示

1. **JQuery的选择器(怎么给一个id为xxx的按钮加点击事件)**

Id选择器，类别选择器，标记选择器，全选选择器

$(“#xxx”).click()

1. **Document load 和 document ready区别？**

Document ready表示文档结构已经加载完成，但不包含图片等非文字媒体文件（机构加载完成）

Document load表示页面包含图片等文件在内的所有元素都加载完成(全文加载完成）

1. **JQuery的四个核心函数是？**

Jquery([sel,[context]]) jquery(html,[ownderDoc]) jquery(elements) jquery(callback)

1. **Jsp常用指令有哪些？**

<jsp:usebean> <jsp:setproperty> <jsp:include> <jsp:getproperty> <jsp:forward> <jsp:param>

1. **批量上传数据是怎么操作的?**
2. **对MVC的理解？**

SpringMVC是一种设计模式,他有三个核心部件:模型,视图,控制器,他们各自处理各自的任务

1. **http和webservice请求的区别？**

Httpservice基于http协议用来调用本服务的内容，而webservice基于soap协议调用其他应用的服务，在处理数据上httpservice使用post和get效率较高，webservice能处理较复杂的数据类型。

1. **Spring当中如何处理事务？**

AOP应用 :

声明式事务

底层采用aop技术实现

将事务的管理过程(创建、使用(提交、回滚))封装在一个事务通知bean中;

通过为其提供切入点(xml中配置以及引用)的方式,描述需要将该事务通知bean提供的事务管理服务引用给哪些核心业务对象/方法.

事务管理服务从核心的业务逻辑中分离出来，封装在一个aop组件中

around+throws

1 核心方法执行之前开启事务

2 核心方法执行完毕之后

执行没有问题 提交

执行出现问题 回滚[throws]

1 事务

对一组sql进行统一的提交或回滚操作，保证操作的一致性

2 事务的特性

持久性

事务一旦提交成功，对数据库的操作是持久化的；后续的任何操作都不能回滚该结果

隔离性

事务操作的数据对其他事务是相互隔离的

原子性

不可分割的整体

一致性

数据的数据类型保持一致

3 事务的属性

传播途径

两个具有事务的方法相互调用时

被调用方法的事务以何种方式添加到主动方法的事务中

7

隔离级别

事务操作的数据对其他事务的隔离程度

未提交读

提交读

重复读

序列化读

脏读 幻读 不可重复读

oracle提交读 | mysql重复读

只读性 false

事务是否只读

对于只读事务，只能在事务中执行select操作，不允许执行insert delete update操作

超时时间 -1

事务在多长时间内有效

超过了指定时间，如果事务仍未提交，那么事务自动回滚

4 事务的分类

根据事务的编写位置

全局事务

在服务器中配置的事务(tomcat weblogic)

JTA java transaction

添加到该服务器上的工程都可以共享同一个全局事务

跨越多个connection连接

分布式事务管理方式

局部事务

在工程中配置的事务

JDBC事务 hibernate事务

局部事务的配置方式依赖于持久层采用的底层技术

技术不同，提供了不同的配置方式

只能局限在一个connection连接

1. **Spring中如何配置任务调度？**

使用spring内置的quartz组件实现任务调度

1：提供任务对象

2：ioc容器中配置任务对象

3：描述任务细节jobDetail

4：为任务设置触发器Trigger 设置触发时间（通过cron表达式）

5：配置总调度类 Scheduler

1. **用过的前端技术 EasyUI里面都有什么布局，有哪些组件？**

Layout布局：Panel面板、tabs选项卡、accordion分类、layout布局

组件：menu菜单、linkbutton按钮、form表单、window窗口、datagrid数据表格、tree树

1. **Hibernate的缓存机制和Hibernate对象之间的关系？**
2. **Maven是用来做什么的，如何使用？**

Maven是一个项目管理工具，管理项目报告，生成站点，管理jar文件等等。

Maven常用命令：

mvn archetype：create创建Maven项目

mvn compiled 编译源代码

mvn deploy 发布项目

mvn clean 清除项目目录中的生成结果

mvn package 根据项目生成.jar

1. **在之前的项目中都有用过什么一些封装的组件？**
2. **Hibernate和Mybatis的区别？**
   1. 联系:Hibernate和MyBATIS都是对JDBC进行了封装。
   2. 区别: Hibernate是一个”全自动”的ORM解决方案，提供了从实体类到数据库表的全套映射机制，程序猿不需要对SQL非常熟悉，只需要提供相应的映射关系文件，Hibernate自动生成SQL语句，并且自动执行SQL语句，并且自动返回对应的结果。不灵活，走自动生成，效率最低。
   3. 区别：JDBC所有的SQL语句都必须自己写，自动手动执行，自动手动返回结果，最快，冗余，重复代码多
   4. 区别:MyBATIS SQL语句需要自己写，SQL开发工作量大了。移植性差。但是灵活。
3. **Spring怎么整合hibernate?**

hibernate

HibernateTemplate

1 hibernate流程

【jar + javaBean + 映射文件】

a 添加hibernate配置文件

hibernate.cfg.xml

配置数据源 + 映射文件

配置SessionFactory对象

<session-factory>

<property>

<mapping>

b 解析 - 获取SessionFactory实例

c 获取Session会话对象

d 开启事务 session.beginTransaction()

insert delete update

e save() delete() update()

select

e 定义hql

f 编译hql session.createQuery(hql)

g 执行hql unqueObject() list()

h 提交、回滚 commit() rollback()

关闭会话 session

2 在DAO中获取HibernateTemplate实例

ioc被动注入

在ioc容器中配置HibernateTemplate?

3 整合过程

a 创建java工程，添加spring框架支持(核心层core)

b 添加hibernate支持(jar+hibernate.cfg.xml)

c 添加整合jar(spring-orm层的核心jar)

4 整合hibernate注解版

1. **Spring怎么整合Struts？**

struts2整合

a 整合原理

struts2的作用 ? 处理HTTP请求

struts2处理请求使用 - Action [核心]

action本身 不具备数据库的连接能力

持久层的DAO

struts2的action 持久层的DAO

业务逻辑层对象 持久层对象

服务层技术的管理 - spring

ioc容器管理 Action-DAO

DAO -> IOC

JdbcTemplate / HibernateTemplate

Action -> IOC

spring对struts的整合原理 :

使用spring的ioc容器管理struts2中用于处理请求的Action

将struts2的Action配置成ioc容器中的bean

b 整合方式

遵循JavaEE的分层体系架构

前段层 整合 后端层

spring jdbc/hibernate

struts2框架提供用于整合spring方式 :

json插件 对ajax整合

struts2-json-plugin.jar

struts-plugin.xml

插件 实现对spring的整合

struts2-spring-plugin.jar

struts-plugin.xml

struts2内置了三种配置文件

struts-default.xml

struts-plugin.xml

struts.xml

c 整合过程

1) 创建web工程，添加struts2框架支持

StrutsPrepareAndExecuteFilter

2) 添加spring框架支持

ContextLoaderListener

3) 实现两个框架的整合

struts2提供的插件实现对spring的整合

'spring' StrutsSpringObjectFactory extends ObjectFactory

ObjectFactory

struts2内置的[Action]对象创建工厂

单独使用struts2时，action是由谁创建的

请求 - StrutsPrepareAndExecuteFilter - ActionProxy -

ActionInvocation - Action对象 <action name="" class=""/> -

ObjectFactory - 根据该对象中的配置，决定谁具备实例化Action的权利

ObjectFactory 指定 ActionInvocation具备初始化Action对象的权利

1. **Mybatis动态查询标签有哪些？**

<!-- 动态SQL查询 -->

<sql id="commonsql">select \* from student where 1=1 </sql>

<!-- 查询Student,如果 名字不为null，根据名字查询，如果年龄不为null 按年龄查询 -->

<select id="getByNameAndAge" resultType="com.etoak.pojo.Student"

parameterType="map">

<include refid="commonsql"></include>

<if test="name!=null">

and name=#{name}

</if>

<if test="age!=null">

and age = #{age}

</if>

</select>

<!-- 查询年龄是 12 23 34 45 56的学生信息 -->

<select id="getByForeach" resultType="com.etoak.pojo.Student"

parameterType="map">

<include refid="commonsql"></include>

and age in //data是dao里边的key值

<foreach collection="data" item="age" open="("

close=")" separator=",">

#{age}

</foreach>

</select>

1. **SpringMVC的核心是什么？**

1. **使用SpringMVC是不是每一次请求都创建一个Controller对象？**
2. **SpringMVC常用的注解有哪些？**

@Controller

@RequestMapping

@PathVariable

ModelAttribute

1. **JSONObject是哪个包底下的？**

Json-lib.jar

import net.sf.json.JSONObject

1. **POI是做什么用的，请具体写其中涉及到的一个类？**
2. **请写一个校验数据格式为10-500的正则表达式？**
3. **使用数据库连接池的优点？**

1）资源重用

2）更快的系统响应速度

3）新的资源分配手段

4）统一的连接管理，避免数据库连接泄露

5）使用连接池的最主要的优点是性能。创建一个新的数据库连接所耗费的时间主要取决于网络的速度以及应用程序和数据库服务器的(网络)距离，而且这个过程通常是一个很耗时的过程。而采用数据库连接池后，数据库连接请求可以直接通过连接池满足而不需要为该请求重新连接、认证到数据库服务器，这样就节省了时间。

1. **Mybatis当中如何处理分页？**
2. **Spring的AOP是怎么影响代码的？**