1. &和&&的区别

&是非短路逻辑与运算符，运算符俩边为布尔表达式，都为true时，整个运算为true。

&也是位运算符

&&是短路逻辑与运算符，运算符俩边都为true时，整个运算为true。

当前边的表达式结果为false时，后面的表达式不再计算。

1. 数据类型

基本数据类型：

整型：byte（1个字节、-2的7次方到2的7次方-1）、short（2个字节、-2的15次方到2的15次方-1）、int（4个字节、-2的31次方到2的31次方-1） 、long（8个字节、-2的63次方到2的63次方-1）

浮点型：float（4个字节）、double（8个字节）

布尔型：boolean（2个字节、unicode编码值）

字符型：char

引用数据类型：

类、数组、接口

1. 包装类

包装类是引用数据类型，将基本数据类型封装成对象，默认值为null

char—character、int—Integer

byte—Byte、 short—Short、long—Long、boolean—Boolean、float—Float、double—Double

自动类型转换—从低到高、强制类型转换: 从高到低

级别从低到高:

byte,char,short(这三个平级) -->int-->float-->long-->double

包装类还提供了将字符串转换成基本数据类型的方法：

byteValue()、valueOf()

包装类还提供了将基本数据类型转换成字符串的方法：

toString()

自动装箱：Integer i= 4;

自动拆箱：int j =i;

1. 面向对象的特征

封装：隐藏对象的属性和实现细节，对外提供公有的访问方式。（安全性、重用性）

继承：子类继承父类所有非私有方法和属性，且可以新增方法或修改继承自父类的方法(override重写)。让类与类之间产生关系、是多态的前提。（重用性）

父类对象引用子类，可以调用子类方法，不能调用子类的私有属性。

多态：类的不同对象对同一消息所作出的响应不同，例如工厂类生成对象。（程序的扩展性）

1. 类型转换

小类转大类：向上转型、自动转型。例：父类引用指向子类对象 P p = new Son();

1. 方法的重写和重载

Override（重写）：体现父类与子类之间的多态性

《1》子类方法的权限必须大于等于父类方法权限

《2》同为静态或非静态

《3》不可以重写父类构造

《4》子类方法抛出的异常比父类方法抛出的异常要少或是父类异常的子异常

《5》方法名、参数列表相同

《6》修饰符为public

Overload（重载）：体现一个类中的多态性

1. 方法名相同、参数列表不同
2. 与异常无关。
3. 抽象类和接口

抽象类 (abstract class)、接口(interface)

同：都不能被实例化，接口通过实现类，抽象类通过子类。

异：①接口不能有抽象方法

②接口没有普通成员变量，只能有常量(public static final)

③接口的方法必须是抽象方法，且必须用public修饰范围

④一个类只能继承一个抽象类，可以实现多个接口

⑤类单继承、接口可以多继承。

总结：抽象类是一个可以有抽象方法的类，接口是抽象类的一个特例

1. 集合容器

Collection是集合类上级接口，子接口有Set、List

Set集合：①无序，不可重复（equals()判断）、保证元素唯一性

②取出方式只有iterator迭代器（可对集合元素操作）

③实现类：HashSet/TreeSet

H：底层数据结构是哈希表、线程不同步（不安全）、无序

通过元素的hashCode()和equals()保证元素唯一

T：底层数据结构是二叉树、线程不同步（不安全）、制定顺序排序(有序)

List集合：①有序（索引），可重复

②取出方式有迭代器（可对集合元素操作）、foreach循环（只能遍历不能操作）

③实现类：ArrayList/LinkedList/Vector

A：底层数据结构是数组、线程不同步（不安全）、查询快

V：底层数据结构是数组、线程同步（安全）、增删查都慢，被A替代

L：底层数据结构是链表、线程不同步（不安全）、增删快

Set、List方法：新增add/addAll、删除remove、修改set、查询get、判断contains、迭代器iterator

Map集合：①键值对、键唯一

②取出方式，通过keySet()/entrySet()将Map集合转成Set集合，再使用迭代器iterator遍历

③实现类：hashMap、hashtable

hashMap： 底层数据结构是哈希表、线程不同步（不安全）,可null键或null值。

hashtable：底层数据结构是哈希表、线程同步（安全）,不 可null键或null值。

应用：当存储的是一个元素时、使用Collection集合，是映射关系时、使用Map集合

保证唯一用Set、会重复用List

Collections：工具类，方法都是静态的，直接类名调用方法，搜索、排序、线程安全化

sort(list)：list集合自然顺序排序

List synchronizedList(list)：线程安全化

Map synchronizedList(map)：线程安全化

Map遍历：Map map = new Map();

Map.put(“a”,”aaa”);

Set entrys = map.entrySet();//获得键值对Entry对象的Set集合

iter = entrys.iterator();//使用迭代器

第一种：

While(iter.hasNext()){

Map .Entry entry=(Entry)iter.next();

String key = entry.getKey();//取得key/value

String value = entry.getValue();

System.out.println(“key=”+key+”value=”+value);

}

第二种：

for(Object object:entrys){

Map .Entry entry=(Entry) object;

System.out.println(“key=”+ object .getKey()+”value=”+ object .getValue());

}

1. 栈、堆

栈内存：存放局部变量（方法变量、局部变量）

堆内存：存放数组和对象，有垃圾回收机制，分为三个区

年轻代：存放new的对象

年老代：存放内存回收后还存活的对象

永久代：final常量、static变量、常量池

1. String、StringBuilder、StringBuffer

String：java.lang包下的final类，不可被继承，字符串一旦被初始化，就不可以被改变，存放在方法区中的常量池中。

String不可变性：是String 对象内容不可变，String对象存在于常量池内，而String对象的引用是可以改变的。

方法：

获取：length()字符串长度

charAt()指定位置的字符

indexOf()指定字符的位置

substring()获取子串

判断：contains()是否包含

equals()字符串内容是否相同

转换：valueOf()字符数组转成字符串

toLowerCase()/toUpperCase()大小写转换

split()分割字符串

替换：replace()替换字符串

concat()追加字符串

trim()清除两端空格

StringBuffer：java.lang包,字符缓冲区、可以对字符串内容修改、长度可变、线程不安全

StringBuilder：java.lang包,字符缓冲区、可以对字符串内容修改、长度可变、线程安全

方法：添加append()、insert()

删除delect()

修改replace()、setCharAt()、setLength()

查询indexOf()

获取substring()

反转reverse()

1. 静态

static关键字：修饰成员（成员属性和成员方法）

静态变量和方法直接类名.调用

静态成员随类的加载而产生

静态方法只能调用静态成员、不能使用this、super，因为它们需要依附在对象下

优先级：静态成员>静态代码块>静态方法。父类大于子类

1. final、finally、finalize

final关键字：关键字，被修饰的类不可被继承、被修饰的方法不能被覆盖（重写override）、被修饰的变量是常量，只赋值一次

finally语句：异常处理语句中的部分，final语句总执行

finalize方法：Object类的方法，在垃圾收集器（GC）删除对象之前调用，可覆盖该方法进行资源整理。

1. 异常

异常(Throwable)：程序运行时出现的不正常情况，分为Error(错误)/Exception(异常)

Error：程序无法恢复，一般与JVM相关，栈溢出(递归：方法调用方法本身，每次传入的参数有上次结果确定，次数过多会出现栈内存溢出)、内存溢出等错误。

Exception：程序可以处理的异常，子类有运行时异常(RuntimeException)和受检查异常(CheckedException)

R:运行过程中发生异常，编译器不检查。脚标越界异常（IndexOutOfBoundsException）空指针异常（NullPointerException）

C：编译器会检查到的异常，导致编译不通过，必须捕获（try…catch）或声明抛出异常(throw 、throws)。类存在异常(ClassNotFoundException)

类型转换异常(Classcastexception)

throw：抛出异常对象，用于方法中

throws：在方法上声明异常，后面可以跟多个异常。

try…catch…finally：异常处理机制，try：可能出现异常的语句。catch：捕获异常并处理。finally：总是执行的代码（一般为关闭资源）。可没有finally。

自定义异常：继承Exception或RuntimeException类。

1. 线程

进程：正在执行的程序。是一个程序运行时的内存分配空间，可能包含多个线程。

线程：CPU调度的基本单位，负责应用程序的执行顺序。同一进程下的线程之间互相通信。

创建线程：①继承Thread类

定义类继承Thread类，重写run()方法(线程运行的代码)，通过子类对象创建线程对象，

调用start()方法，开启线程，执行run()方法启动线程。

* + 1. 实现Runnable接口(常用，原因是java单继承，通过接口实现多继承)

定义类实现Runnable接口，重写run()方法，通过Thread类创建线程对象，将实现类的对象作为实参传递给Thread类的构造(要让线程对象明确要运行的run()方法所属的对象)。调用Thread类start()方法开启线程，运行实现类的run()方法。

线程状态：创建：线程对象被创建

可运行(临时阻塞)：start()，在队列中还未执行(排队中)

运行：run()

阻塞（休眠）：sleep(time)使线程进入睡眠状态，释放执行权不释放锁，时间到进入可运行状态。

wait()使线程暂停，释放执行权并释放锁。notify()唤醒线程、notifyAll()唤醒所有线程

结束：stop()

多线程安全问题：当一个线程的多条语句对数据运算时，其他线程对数据进行了操作，导致错误数据产生。

解决方式：加上同步锁(同步代码块synchronized/同步方法)。

同步代码块：使用的锁是任意对象

同步方法：使用的锁是this

静态同步方法：使用的锁是该类的字节码文件对象(类.class)

类中有一个同步时，使用同步方法

类中有多个同步时，使用同步代码块

同步死锁：同步代码嵌套，同步方法中有同步代码块，同步代码块中有同步方法。

wait()、notify()、notifyAll()必须定义在同步内。

Lock接口：解决线程安全，使用同步的形式。能完成synchronized所实现的所有功能，准确、性能好。await()进入等待，unlock()释放锁，lock()请求锁，singnal()唤醒一个，singnalAll()唤醒所有。

Lock与synchronized的区别：

* 1. synchronized自动释放锁，lock手动释放(finally从句中释放)
  2. synchronized隐式的锁操作，lock显示锁操作
  3. synchronized锁是任意对象，lock锁自己就是对象

1. IO流

按操作分：输出流（写）、输入流（读）

按处理的数据类型分：字符流、字节流

字节流（处理二进制数据）：InputStream、OutputStream

字符流（处理文本）：Writer、Reader

W：子类必须实现的方法读read(cahr[],int.int)和关闭流close()

R ：子类必须实现的方法写write(cahr[],int.int)和关闭流close()、刷新flush()

四个基类都是抽象类)

对象序列化(Serializable)：将具体的对象持久化，可写入硬盘。实现Serializable接口,用FileoutputStream构造一个ObjectOutputStream对象流，调用writeObject(obj)将对象写出，反之用对象输入流读取。

1. JVM

内存回收时程序自动回收，俩次，第一次根据对象是否覆盖了finalized()，没有覆盖则回收，覆盖了就运行finalized()，可以救回。程序启动会启动俩个线程，一个是主线程mian，另一个就是垃圾回收线程(优先级低)，

垃圾回收（GC）主要针对的是堆区。

1. 设计模式

一套解决问题且被反复使用的代码设计。提高代码重用、保证代码可靠性。

Java有23种设计模式：分三类：

创建型模式(5种)：工厂方法、抽象工厂、单例、建造者、原型

结构型模式(7种)：适配器、装饰器、代理、外观、桥接、组合、享元

行为型模式(11种)：策略、模板方法、观察者、迭代子、责任链、命令、备忘录、状态、访问者、中介者、解释器。

六大设计原则：《1》开闭原则：对扩展开放、对修改封闭。(使用接口、抽象类)

1. 单一职责原则：每个类实现单一职责。(类拆分)
2. 里氏替换原则：子类对父类的方法尽量不要重写和重载。(父类更好)
3. 依赖倒转原则：面向接口编程，依赖抽象不依赖具体。

《5》迪米特法则(最少知道原则)：将复杂的逻辑封装到方法中，通过public方法提供给外部。

《6》合成复用原则：合成/聚合的方式优先于继承方式。

单例模式(single)：保证一个类在内存中的对象唯一。饿汉式、懒汉式

例：

多线程读取配置文件，为方便操作数据将配置文件封装成对象，要保证多个程序读到的是同一个配置文件

饿汉式：

class Single{

private Single(){} //私有化构造方法(创建对象需要调用构造，私有化后其他程序无法创建该类的对象)

private static Single s = new Single(); //创建私有的静态的类对象

public static Single getInstance(){ //定义公有的静态方法，将对象传递出去。

return s;

}

}

懒汉式：延迟加载

Class Single{

private Single(){}

private static single s= null;

public static Single getInstance(){

if(s==null)

s=new Single();

return s;

}

}

区别：饿汉式：一旦加载，单例初始化完成。线程安全

懒汉式：当调用getInstance()时，才初始化单例。线程不安全， getInstance()加上synchronized。

1. 反射技术

动态加载指定类，并获取该类的内容。将字节码文件、及文件的内容都封装成对象，便于操作，增强了程序的扩展性。

步骤：①获得Class对象(字节码文件对象)，getClass()/类.class/ Class.forName(Classpath)

* + 1. 实例化对象，获得类的属性和方法
    2. 访问属性、方法、构造、创建对象

1. 正则表达式

^开始 $结尾 \*重复>=0次 +重复>=1次 ? 重复0或1次 \转义 |或

\d数字 \w数字、字母、下划线、汉字 \s空白符

{n}重复n次 {n,}重复>=n次 {n，m}重复n到m次 [a-zA-Z0-9]

1. equals()与==

equals()是Object类的方法，很多类重写了该方法，比如String比较的是字符串的值相等。

没有重写的equals()与==作用相同，比较的是俩个对象是否相同。

“str”.equals(obj)建议这样使用

obj.equals(“str”)，当obj是null时，报空指针异常

1. 并发

Java并发库

同步器：是线程能够等待其他线程的对象，允许他们协调动作

Semaphore信号量，表示共享数量：

acquire()获取资源

release()释放资源

CyclicBarrier线程达到屏障后等待，当一组线程都达到屏障后才一起恢复执行：

await()每调用一次，计数减1，并阻塞当前线程。为0时，阻塞解除，所有在CyclicBarrier

上阻塞的线程全部运行

CountDownLatch倒计数门闩：

await()在倒计数为0之前阻塞当前线程

二、JavaWeb

1. JSP页面间的参数传递的方式

方式1：超链接：<a href=”P.jsp?username=zhang&pwd=123&age=25” >URL</a>

方式2： JS方式传值：

a. <input name=”btnModify” type=”button” class=”button1” onClick=”self.location=’client\_modify.jsp?id=<%=id%>’” value=”修改分销商”/>

b. <input name=”btnModify” type=”button” onClick=”modify()”/>

function modify(){

window. self.location=’client\_modify.jsp?id=<%=id%>;

}

方式3： form表单

a.显示控件 <input type=”text” name=”pwd” >

b.隐藏域 <input type=”hidden” name=”pwd” value=”<%=pwd%>”>

<form action=”XXX.jsp” method=”post”></form>

方式4：JSP的include和forward标签

<jsp:include flush=”true” page=”T.jsp?username=zhang&pwd=123”>

<jsp:param name=”age” value=”28”/>

</jsp:include>

forward传值

<jsp:forward page="login.jsp">

<jsp:param value="admin" name="userName"/>

<jsp:param value="password" name="password"/>

</jsp:forward>

统一接受值：request.getParameter("userName")

jsp:forward是动作标签，浏览器地址不变。

1. Jsp动作

<jsp:include/>包含

<jsp:forward/>转发到另一个页面，等价于request.getRequestDispatcher(1.jsp?name=XX).forward(request.response);

<jsp:usebean/>设置javaBean

<jsp:setProperty/>设置属性

<jsp: getProperty/>获取属性

<jsp:plugin/>设置插件

1. Jsp隐含对象(内置对象) 9个

就是在编译成servlet时，servlet里产生的可以直接用的内置对象

page: 表示当前jsp页面的一个实例，可以用this代替

out：jspWriter out =pageContext()用于标签对象中输出

request：HttpServletRequest接口的简单使用

response：HttpServletResponse接口的简单使用

exception：异常

application：servletContext的简单使用

session：HttpSeesion的简单使用

pageContext：当前页面对象

config：ServletConfig类，很少使用

1. forward()和redirect()
2. forward请求服务器一次，redirect请求服务器两次，所以forward可以获得request作用域的信息。
3. forward由request对象发出，redirect由response对象发起
4. redirect可以跨越工程
5. cookie禁止后如何使用session

session的sessionid是通过cookie保存的，浏览器禁用cookie后，将页面链接使用URL重写的方式(encodeURL/encodeRedirectURL)，将 sessionid当作参数的方式放在URL后面，使每次提交给服务器时都能获取到sessionid。

1. 网络协议

Socket：套接字，通信的俩端都有Socket，网络通信就是Socket之间的通过IO传输

UDP传输协议：数据封装到数据包中。用Socket服务(DatagramSocket )的send()发送，receive()接收。

TCP/IP传输协议：传输的是Socket流，Socket流即有读又有写。俩个端点，客户端(Socket对象)、服务端(ServerScokt对象)

客户端：创建Socket对象，初始化地址和端口，期间对指定ip和端口链接(三次握手：请求、定时、接收)，getInputStream()/ getOutputStream()获取流对象，关闭资源。

服务端：创建ServerScokt对象,初始化监听一个端口，accept()获取客户端Socket对象并通过该对象与客户端通讯，通讯结束，先关闭客户端资源、再关闭服务端资源。

TCP/IP通讯和UDP通讯的区别：

TCP/IP：面象连接(管发管到)、安全性高、点对点通信、数据大小不限制，通过Socket对象连接得到输入输出流，从流中读写通信。在网页(http)、邮件(SMTP)、文件(FTP)、远程都使用的TCP/IP。传输层使用

UDP：非面象连接(管发不管到) 、安全性低、一对多通信、每个数据包在64k以内，通过本机创建DatagramSocket 对象，将数据包装成DatagramPacket包来发送、自寻址(包里有发送的地址)。QQ、视频以及使用”ping”测试俩台主机的TCP/IP通讯是否正常就是象对方主机发送UDP数据包，对方主机确认接受，并返回消息，那么确定网络通畅。传输层使用

Http协议：应用层使用。

1. JDBC

操作数据库步骤：①Class.forName()：加载数据库驱动

* + 1. DriverManager.getConnection()：获取与数据库连接的对象
    2. conn.getPrepareStatement()/Conn.getStatement()：获取与数据库会话对象
    3. st.execute(sql)/executeQuery(sql)/executeUpdate(sql)：执行sql
    4. getResultSet()：获取结果集/getUpdateCount()：获取更新条数
    5. 关闭结果集、会话、连接

PrepareStatement：代表执行预编译的SQL语句。使用setter()传入查询变量，自动对特殊字符进行转义，避免SQL注入攻击，可以执行多次。

Statement：执行静态sql语句

execute()：结果为true(查询到结果)通过getResultSet()获取结果集，为flase(插入/修改)getUpdateCount()获取更新条数

executeQuery()：查询语句(DDL)，返回ResultSet。

executeUpdate()：插入/删除/修改(DML)，返回值int型。当放入DDL语句时，返回值0，放入DML时返回值是更新的条数

JDBC事务处理

事务：单个逻辑工作单元的一组操作。四大特性：

原子性A：这组操作要么全成功、要么全失败

一致性C：从一个一致状态编程另一个一直状态。A+B=200，A转给B五十。A+B=200不变。

隔离性I：多个并发事务相互隔离。多用户操作同一个表。

持久性D：一旦提交事务，数据的改变即永久。

事务处理步骤：开启(conn.setAutoCommit(false)关闭自动提交即开启)、提交(conn.commit())、异常--回滚(conn.rollback())。正常：开启—提交；出现异常：开启—回滚—提交。

隔离级别(4种)：

Serializable(串行化)：最高级别，全部避免。

Repeatable(可重复读)：可避免脏读、不可重复读。MySQL默认

Read committed(读已提交)：可避免脏读

Read uncommitted(读未提交)：最低级别，无法保证任何情况。

脏读：一个事务读取了另一个未提交的事务的脏数据。

不可重复读：一个事务多次读取表的同一数据，结果不同。读取了其他事务已提交的数据。

虚读(幻读)：一个事务读取到了其他事务插入或删除的数据，导致前后数据量不一致。

1. 数据库连接池

目的：避免应用程序直接访问数据库，造成数据库资源的浪费及造成数据库服务器内存溢出。

步骤：使应用程序从池中获取现有连接。

1. 编写DB连接池，先实现javax.sql.DateSource接口
2. 在DateSource 构造中批量创建DB连接，加入到LinkList中

mindle：最小空闲连接(初始化数量) maxdle：最大空闲连接(超过最大连接将进入等待队列)

maxwait：最大超时等待时间(单位ms)

1. Servlet

Servlet：在web容器中运行的服务器端java程序，主要用于响应HTTP请求。Servlet一般用于MVC中的控制器部分。

Servlet容器：管理servlet生命周期的应用程序(tomcat)

Servlet生命周期：①web容器加载servlet，生命周期开始

* + 1. 调用servlet的init()初始化Servlet
    2. 调用servlet的service()实现
    3. 根据请求方式调用doGet()/doPost()
    4. 结束服务，web容器调用servlet的destory()销毁

Get：1.请求参数在地址栏中显示，不安全

2.请求参数限制在1K以内

3.没有请求体，无法通过request.setCharacterEncoding()来设置参数的编码。

Post：1.请求参数不显示在地址栏，相对安全

2.请求参数长度没有限制

Servlet/jsp：JSP编译后是类Servlet。JSP侧重视图，Servlet侧重控制逻辑。

1. 拦截器、过滤器

①拦截器是基于Java反射机制的，而过滤器是基于函数回调；

②过滤器依赖与servlet容器，而拦截器不依赖与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。

⑤在action的生命周期中，拦截器可背调用多次，过滤器只在容器初始化时调用

过滤器：1.判断用户是否登陆2.聊天非法文字3.统一解决编码。

创建过滤器：1.实现javax.servlet.Filter接口2.重写init()、doFilter()、destroy()3.在web.xml配置

1. Ajax和JSON

Ajax(异步JS和XML)：实现客户端与服务器异步通信，页面局部刷新 ，并不是所有的用户请求都提交给服务器,像—些数据验证和数据处理等都交给Ajax引擎自己来做

使用：在JavaScript标签中写

* 1. 先创建XmlHttpRequest对象

②规定请求的类型、URL 以及是否异步处理请求：XmlHttpRequest对象的open(GET/POST、url、true(异步)/false(同步))

1. 设置响应服务器请求的回调函数：使用XmlHttpRequest对象的onreadystatechange属性
2. 向服务器发送请求：XmlHttpRequest对象的send(String),String只限于POST请求
3. 获取服务器响应：使用XmlHttpRequest对象的responseText/responseXML属性获取
4. 使用JS的DOM实现局部刷新：document.getElementById("div").innerHTML

JSON：一种数据交换格式，支持复合数据类型(数组、数字、对象、字符串)

数据是键值对，用逗号分开，大括号保存对象，方括号保存数组

JSON数据：”name”:”abc”

JSON对象：{”name”:”abc”,”url”:”www.abc.com”}

JSON数组：”s”:[

{”name”:”a”,”url”:”www.abc.com”},

{”name”:”b”,”url”:”www.abc.com”},

{”name”:”c”,”url”:”www.abc.com”},

]

obj = JSON.parse(jsonString)：将JSON字符串转换成JS对象

JS函数使用ajax异步请求

function(){

$.ajax({

url:”/user/login”,

dataType:”jsonp”,

type:”GET”,

success:function(data){

if(data.data){

doSubmit();

}else{

alert(“message”);

}

},

});

}

三、数据库

1.SQL语句

2.优化

## 1. SQL语句优化

避免where子句中使用!=或<>，会全表扫描

避免where子句对字段进行null值判断，会全表扫描

用Where子句替换HAVING 子句 因为HAVING

用 exists 代替 in

## 2索引优化

数据库索引：是数据库管理系统中一个排序的数据结构，以协助快速查询、更新数据库表中数据。索引的实现通常使用B树及其变种B+树。

优点：创建唯一索引，保证数据库的数据唯一性，查询快

缺点：创建索引费时，占用数据库的物理空间，增删改时索引也要对应维护，降低了速度

## 3.数据库结构优化

范式优化： 比如消除冗余

反范式优化：适当加冗余

拆分表：不同分区的数据可以制定保存在不同磁盘上的数据文件里

3.存储过程与触发器

相同：都是SQL语句集

区别：触发器不能用EXECUTE语句调用，而是在用户执行Transact-SQL语句时自动触发（激活）执行。触发器是在一个修改了指定表中的数据时执行的存储过程。通常通过创建触发器来强制实现不同表中的逻辑相关数据的引用完整性和一致性。由于用户不能绕过触发器，所以可以用它来强制实施复杂的业务规则，以确保数据的完整性。触发器不同于存储过程，触发器主要是通过事件执行触发而被执行的，而存储过程可以通过存储过程名称名字而直接调用。当对某一表进行诸如UPDATE、INSERT、DELETE这些操作时，SQLSERVER就会自动执行触发器所定义的SQL语句，从而确保对数据的处理必须符合这些SQL语句所定义的规则。

总结：触发器通过事件执行触发而被执行的，存储过程通过名字直接调用。

4.数据库范式

1.一范式(1NF)：字段不可再分

2.二范式(2NF)：每行为一个实例的所有属性。

3.三范式(3NF)：每个属性与主键存在直接关系。例如：学号-->学生、年龄、学院；

学院-->学院地址、学院电话

5.数据库优化面试

(1) 根据服务层面：配置mysql性能优化参数；

(2) 从系统层面增强mysql的性能：优化数据表结构、字段类型、字段索引、分表，分库、读写分离等等。

(3) 从数据库层面增强性能：优化SQL语句，合理使用字段索引。

(4)从代码层面增强性能：使用缓存和NoSQL数据库方式存储，如MongoDB/Memcached/Redis来缓解高并发下数据库查询的压力。

(5) 减少数据库操作次数，尽量使用数据库访问驱动的批处理方法。

(6) 不常使用的数据迁移备份，避免每次都在海量数据中去检索。

(7) 提升数据库服务器硬件配置，或者搭建数据库集群。

(8) 编程手段防止SQL注入：使用JDBC PreparedStatement按位插入或查询；正则表达式过滤（非法字符串过滤）；

四、框架

1.MVC(Model-View-Controler)

将应用程序的输入、处理、输出分开。

视图：与用户交互的界面，向用户显示数据并接收用户输入，无任何业务逻辑处理。

模型：业务的数据和处理。一个模型为多个视图提供数据。

控制器：当用户点击web页面的提交按钮时，控制器接收到请求并调用相应的模型去处理请求，根据处理结果调用相应的视图显示数据。

2.Spring

Spring：一个轻量级的IOC和AOP的容器框架。

特点：方便解耦、简化开发、面向切面编程，通过配置完成对事务的管理，方便集成各种优秀的框架，对一些常用但不好做的API进行了封装、降低了开发难度。

1.IOC(Inverse of Control)：控制反转

概念：将创建对象的权力反转给Spring

应用：①通过application.xml配置文件，通过<bean>标签配置

②在application.xml配置文件中开启组件扫描 <context:component-scan base-package = “Path” > 扫描path下的所有内容。在类或方法上加注解。

@Repository(value="userDao") 持久层注解

@Service(value="userService") 业务层注解

@Controler 控制层注解

@RunWith(SpringJUnit4ClassRunner.class) JUnit4测试类注解

@ContextConfiguration("classpath:applicationContext2.xml") 读取配置文件

DI(Dependency Injection)：依赖注入，对象与对象之间有相互依赖的关系，Spring可以动态的将依赖对象或属性注入到Bean组件中。

优点：降低各个对象之间的耦合度

注入的方式：①Setter属性注入，例将Dao作为Service的属性注入

②构造注入

③P名称空间注入

④SPEL Spring表达式注入

⑤集合注入

2.AOP(Aspect Oriented programming)：面向切面编程

概念：通过预编译的方式和运行期间的动态代理先实现程序功能的统一维护。采取横向抽取机制代替传统继承体系重复性代码(性能监测、事务管理、安全检查、缓存)。

优点：在不修改源码的基础上对程序进行增强

底层实现：

①基于JDK的动态代理：必须面向接口，只有实现了具体接口的类才能生成代理对象

②基于CGLIB动态代理：对于没有实现接口的类，以产生这个类的子类的方式产生代理

AOP术语：

1. Joinpoint(连接点)：拦截到的方法

2. Pointcut(切入点)：切入点是指我们要对哪些Joinpoint进行拦截的定义

3. Advice(通知/增强)：拦截到Joinpoint之后所要做的事情就是通知.通知分为前置通知,后置通知,异常通知,最终通知,环绕通知(切面要完成的功能)

4. Introduction(引介)：一种特殊的通知在不修改类代码的前提下, Introduction可以在运行期为类动态地添加一些方法或Field

5. Target(目标对象) ：代理的目标对象

6. Weaving(织入) ：把增强应用到目标对象来创建新的代理对象的过程

7. Proxy（代理）：一个类被AOP织入增强后，就产生一个结果代理类

8. Aspect(切面) ：切入点和通知的结合，自己编写和配置的

实现步骤：①在application.xml配置文件中配置好切面类的bean，完成AOP的配置

<aop:config>

<aop:aspect ref=”切面类的beanId”>--引入切面类

<aop:beforemethod=”log”pointcut=”execution(public\*com.fth.demo.UserDaoImpl.save(..))”/ >--定义通知类型：切面的方法和切入点的表达式

</aop:aspect>

< /aop:config >

②在切面类上添加注解@Aspect定义切面类

在切面类的通知方法上加入通知类型的注解

\* @Before -- 前置通知

\* @AfterReturing -- 后置通知

\* @Around -- 环绕通知

\* @After -- 最终通知

\* @AfterThrowing -- 异常抛出通知

在application.xml配置文件中配置好切面类的bean，开启自动代理

<aop:aspectj-autoproxy/>

通知类型：

1. 前置通知

\* 在目标类的方法执行之前执行。

\* 配置文件信息：<aop:after method="before" pointcut-ref="myPointcut3"/>

\* 应用：可以对方法的参数来做校验

2. 最终通知

\* 在目标类的方法执行之后执行，如果程序出现了异常，最终通知也会执行。

\* 在配置文件中编写具体的配置

<aop:after method="after" pointcut-ref="myPointcut3"/>

\* 应用：例如像释放资源

3. 后置通知

\* 方法正常执行后的通知。

\* 在配置文件中编写具体的配置

<aop:after-returning method="afterReturning" pointcut-ref="myPointcut2"/>

\* 应用：可以修改方法的返回值

4. 异常抛出通知

\* 在抛出异常后通知

\* 在配置文件中编写具体的配置

<aop:after-throwing method="afterThorwing" pointcut-ref="myPointcut3"/>

\* 应用：包装异常的信息

5. 环绕通知

\* 方法的执行前后执行。

\* 在配置文件中编写具体的配置

<aop:around method="around" pointcut-ref="myPointcut2"/>

\* 要注意：目标的方法默认不执行，需要使用ProceedingJoinPoint对来让目标对象的方法执行。

Spring事务配置：

3.SpringMVC

SpringMVC是表现层框架，是Spring框架的一个组件。

核心：DispatcherServlet (前端控制器)

作用：将请求分发给不同的后端处理器，将业务层处理的数据和对应的视图拼接成一个对象(ModelAndView对象)

工作原理：

1.客户端发送请求到DispacherServlet(前端控制器)

2.根据请求路径，分发给处理请求的Cotorller(后端控制器)

3.后端控制器调用相应的逻辑层代码，完成处理并返回视图对象给前端控制器

4.前端控制器查询视图解析器，找到制定视图

5.视图负责将结果显示到客户端

4.Mybatis

数据对象的持久化引擎，是对JDBC的封装。

Mybatis接口绑定：将XXMapper接口的方法与映射文件中的sql语句绑定，直接调用方法即可(使用mybatis的逆向工程自动生成对应的pojo、Dao接口和Mapper映射文件)。使用OGNL从sql参数对象中计算表达式的值，动态拼接sql。

Dao接口的工作原理是JDK动态代理，Mybatis运行时会使用JDK动态代理为Dao接口生成代理proxy对象，代理对象proxy会拦截接口方法，转而执行MappedStatement所代表的sql，然后将sql执行结果返回。

一级缓存：Session缓存。默认开启，作用域是一个sqlSession。在同一个sqlSession中，执行相同的查询SQL，第一次会去数据库中查询，并写入缓存，第二次直接去缓存中取；当执行SQL查询中间发生增删改操作，mybatis会将sqlSession的缓存清空。不想使用一级缓存，通过mybatis配置文件，将localCacheScope设置为statement。

二级缓存：同一命名空间下的缓存。需要手动设置，作用域是同一个命名空间(namespace)下的mapper映射文件内容，多个sqlSession共享。增删改操作会清空二级缓存

单表上使用，查询多于修改时使用

$、#

#（常用）：数据加上双引号，将数据当成字符串，当sql动态参数传递，防止sql注入

$：数据直接显示，将数据当成不变的字符串，传表名或在order by后使用

五、项目经验

1.单点登陆(SSO)：一次登录，多次使用。实现原理：登陆成功，用户生成token(相当于jsessionid)，将token存入cookie，key为token，value为对象信息(密码在存入cookie前删除或者设为null)，当跨域时请求用户信息，从cookie中取得token，根据token查询出用户信息。

jsonp解决js跨域请求，页面上将ajax中的dataType属性设置为jsonp，服务器端将json数据设置成js语句，通过js跨域加载绕过js跨域请求

2.柱状图、饼图、折线图：Echarts(纯JS的图形化库)、jfreechart(Java类库)

3.支付

4.项目部署

Weblogic：

1.控制台部署：将项目打成war包，登陆Weblogic控制台直接安装、激活后即可。

2.自动部署：将项目打成war包，放在autodeploy目录下，启动startWeblogic.cmd即可。

3.配置文件部署：config.xml配置

定时器：quartz

5.项目遇到的难点：根须需求做出相应的改变；业务需求不明白。项目组长说代码冗余，比如if.else语句，不需要else就得删掉。遇到强制类型转换报错，编译器没报错，运行时报错，原来是转换成long型时，没有判断数据是否为空。我就用了if语句判空，但老同事建议我使用三目运算符，并用long的包装类方法将数据类型转换。

六、自我介绍

我叫冯天辉，于2016年7月毕业于郑州轻工业学院软件工程专业，本科学历，拥有扎实的JAVA语言基础，良好的编程习惯，熟悉Spring、SpringMVC、Mybatis、hibernate等开源框架、熟悉Tomcat服务器。

七、算法

1.冒泡

2.堆排序

3.快速排序