JAVA 面试宝典 V4.0 版本

目录

基础

1.简述 JDK 跟 JRE 的区别

Jdk 是 java 开发人员在开发过程使用的软件开发包，他提供了 java 的开发环境和运行环境
JRE 是 Java Runtime Enviroment 是指 Java 的运行环境
如果你只想跑 java 程序，只要安装 jre 就够了，如果要从事开发就得安装 jdk

2.简述 path 跟 classpath 的区别

Path 是系统变量，跟 java 无关，里面存放的是各种可执行的应用程序的路径
Classpath 是给 java 使用的，从字面上来理解，就是类的路径，主要是模仿 path，将类文件的路径配置到 classpath 中实现在系统的任何位置可以对类文件进行编译和执行

3.Java 的关键字中有没有 goto

Goto 是 java 中的保留字不是关键字的一员

4."static"关键字是什么意思？Java 中是否可以覆盖(override)一个 private 或者是 static 的方法？

"static"关键字表明一个成员变量或者是成员方法可以在没有所属的类的实例变量的情况下被访问。
Java 中 static 方法不能被覆盖，因为方法覆盖是基于运行时动态绑定的，而 static 方法是编译时静态绑定的。static 方法跟类的任何实例都不相关，所以概念上不适用。

5.Java 中的方法覆盖(Overriding)和方法重载(Overloading)是什么意思？

Java 中的方法重载发生在同一个类里面两个或者是多个方法的方法名相同但是参数不同的情况。与此相对，方法覆盖是说子类重新定义了父类的方法。方法覆盖必须有相同的方法名，参数列表和返回类型。覆盖者可能不会限制它所覆盖的方法的访问。

6.Overload 和 Override 的区别?

方法的重写 Overriding 和重载 Overloading 是 Java 多态性的不同表现。重写 Overriding 是父类与子类之间多态性的一种表现，重载 Overloading 是一个类中多态性的一种表现。如果在子类中定义某方法与其父类有相同的名称和参数，我们说该方法被重写 (Overriding)。子类的对象使用这个方法时，将调用子类中的定义，对它而言，父类中的定义如同被“屏蔽”了。如果在一个类中定义了多个同名的方法，它们或有不同的参数个数或有不同的参数类型，则称为方法的重载(Overloading)。

7.接口和抽象类的区别是什么？

Java 提供和支持创建抽象类和接口。它们的实现有共同点，不同点在于：
接口中所有的方法隐含的都是抽象的。而抽象类则可以同时包含抽象和非抽象的方法。
类可以实现很多个接口，但是只能继承一个抽象类
类可以不实现抽象类和接口声明的所有方法，当然，在这种情况下，类也必须得声明成是抽象的。
抽象类可以在不提供接口方法实现的情况下实现接口。
Java 接口中声明的变量默认都是 final 的。抽象类可以包含非 final 的变量。
Java 接口中的成员函数默认是 public 的。抽象类的成员函数可以是 private，protected 或者是 public。
接口是绝对抽象的，不可以被实例化。抽象类也不可以被实例化，但是，如果它包含 main 方法的话是可以被调用的。
也可以参考 JDK8 中抽象类和接口的区别

8.接口是否可继承接口？抽象类是否可实现(implements)接口？抽象类是否可继承实体类(concrete class)?

接口可以继承接口。抽象类可以实现(implements)接口，抽象类是否可继承实体类，但前提是实体类必须有明确的构造函数。

9.Java 的基本数据类型跟引用数据类型分别有哪些？

Java 的基本数据类型有 8 个分别是 整数型 byte shrot int long char 浮点型 float double 字符型 char 布尔型 bollean

10.char 型变量中能不能存贮一个中文汉字？为什么？

char 型变量是用来存储 Unicode 编码的字符的，unicode 编码字符集中包含了汉字，所以，char 型变量中当然可以存储汉字啦。

11.简述&和&&的区别

&和&&都可以用作逻辑与的运算符，表示逻辑与（and），当运算符两边的表达式的结果都为 true 时，整个运算结果才为 true，否则，只要有一方为 false，则结果为 false。
&&还具有短路的功能，即如果第一个表达式为 false，则不再计算第二个表达式
&还可以用作位运算符

12.Java 中垃圾回收有什么目的？什么时候进行垃圾回收？

垃圾回收的目的是识别并且丢弃应用不再使用的对象来释放和重用资源。

13.如果对象的引用被置为 null，垃圾收集器是否会立即释放对象占用的内存？

不会，在下一个垃圾回收周期中，这个对象将是可被回收的。

多线程

1.进程和线程的区别是什么？

进程是执行着的应用程序，而线程是进程内部的一个执行序列。一个进程可以有多个线程。线程又叫做轻量级进程。

2.创建线程有几种不同的方式？

有三种方式可以用来创建线程：

继承 Thread 类
实现 Runnable 接口
应用程序可以使用 Executor 框架来创建线程池
实现 Runnable 接口这种方式更受欢迎，因为这不需要继承 Thread 类。在应用设计中已经继承了别的对象的情况下，这需要多继承（而 Java 不支持多继承），只能实现接口。同时，线程池也是非常高效的，很容易实现和使用。

3.概括的解释下线程的几种可用状态。

1. 新建(new)：新创建了一个线程对象。
2. 可运行(runnable) :线程对象创建后，其他线程(比如 main 线程) 调用了该对象 的 start ()方法。该状态的线程位于可运行线程池中，等待被线程调度选中，获 取 cpu 的使用权 。
3. 运行(running)：可运行状态(runnable)的线程获得了 cpu 时间片 （ timeslice ） ，执行程序代码。
4. 阻塞(block)：阻塞状态是指线程因为某种原因放弃了 cpu 使用权，也即让出了 cpu timeslice ， 暂时停止运行。直到线程进入可运行(runnable)状态，才有 机会再次获得 cpu timeslice 转到运行(running)状态。阻塞的情况分三种：
(一). 等待阻塞：运行(running)的线程执行 o . wait ()方法， JVM 会把该线程放 入等待队列(waitting queue)中。
(二). 同步阻塞：运行(running)的线程在获取对象的同步锁时，若该同步锁 被别的线程占用，则 JVM 会把该线程放入锁池(lock pool)中。
(三). 其他阻塞: 运行(running)的线程执行 Thread . sleep (long ms)或 t . join ()方法，或者发出了 I / O 请求时， JVM 会把该线程置为阻塞状态。 当 sleep ()状态超时、join ()等待线程终止或者超时、或者 I / O 处理完毕时，线程重新转入可运行(runnable)状态。
5. 死亡(dead)：线程 run ()、 main () 方法执行结束，或者因异常退出了 run ()方法，则该线程结束生命周期。死亡的线程不可再次复生。
举例：早上打车去上班
新建（准备叫一辆嘀嘀打车）
可运行（找到一辆可以带你去上班的车）
运行（司机接到你，带你去上班）
阻塞（路上堵车了）
死亡（到公司了，付钱下车）

4.同步方法和同步代码块的区别是什么？

同步方法默认用 this 或者当前类 class 对象作为锁；
同步代码块可以选择以什么来加锁，比同步方法要更细颗粒度，我们可以选择只同步会发生同步问题的部分代码而不是整个方法；

5.什么是死锁(deadlock)？

两个线程或两个以上线程都在等待对方执行完毕才能继续往下执行的时候就发生了死锁。结果就是这些线程都陷入了无限的等待中。

6.如何确保 N 个线程可以访问 N 个资源同时又不导致死锁？

使用多线程的时候，一种非常简单的避免死锁的方式就是：指定获取锁的顺序，并强制线程按照指定的顺序获取锁。因此，如果所有的线程都是以同样的顺序加锁和释放锁，就不会出现死锁了。

7.sleep() 和 wait() 的区别

sleep 指线程被调用时，占着 CPU 不工作，形象地说明为“占着 CPU 睡觉”，此时，系统的 CPU 部分资源被占用，其他线程无法进入，会增加时间限制。
wait 指线程处于进入等待状态，形象地说明为“等待使用 CPU”，此时线程不占用任何资源，不增加时间限制。

8.实现同步的方式

同步是多线程中的重要概念。同步的使用可以保证在多线程运行的环境中，程序不会产生设计之外的错误结果。同步的实现方式有两种，同步方法和同步块，这两种方式都要用到 synchronized 关键字。

1.同步方法

即有 synchronized 关键字修饰的方法。
由于 java 的每个对象都有一个内置锁，当用此关键字修饰方法时，
内置锁会保护整个方法。在调用该方法前，需要获得内置锁，否则就处于阻塞状态。
代码如：

```
public synchronized void save(){}
```

注： synchronized 关键字也可以修饰静态方法，此时如果调用该静态方法，将会锁住整个类

2.同步代码块

即有 synchronized 关键字修饰的语句块。
被该关键字修饰的语句块会自动被加上内置锁，从而实现同步
代码如：

```
synchronized(object){
}
```

注：同步是一种高开销的操作，因此应该尽量减少同步的内容。
通常没有必要同步整个方法，使用 synchronized 代码块同步关键代码即可。

集合

1.Java 集合类框架的基本接口有哪些？

Java 集合类提供了一套设计良好的支持对一组对象进行操作的接口和类。Java 集合类里面最基本的接口有：
Collection：代表一组对象，每一个对象都是它的子元素。
Set：不包含重复元素的 Collection。
List：有顺序的 collection，并且可以包含重复元素。
Map：可以把键(key)映射到值(value)的对象，键不能重复。

2.为什么集合类没有实现 Cloneable 和 Serializable 接口？

克隆(cloning)或者是序列化(serialization)的语义和含义是跟具体的实现相关的。因此，应该由集合类的具体实现来决定如何被克隆或者是序列化。

3.什么是迭代器(Iterator)？

Iterator 接口提供了很多对集合元素进行迭代的方法。每一个集合类都包含了可以返回迭代器实例的迭代方法。迭代器可以在迭代的过程中删除底层集合的元素,但是不可以直接调用集合的 remove(Object Obj)删除，可以通过迭代器的 remove()方法删除。

4.Iterator 和 ListIterator 的区别是什么？

Iterator 可用来遍历 Set 和 List 集合，但是 ListIterator 只能用来遍历 List。
Iterator 对集合只能是前向遍历，ListIterator 既可以前向也可以后向。
ListIterator 实现了 Iterator 接口，并包含其他的功能，比如：增加元素，替换元素，获取前一个和后一个元素的索引，等等。

5.hashCode()和 equals()方法的重要性体现在什么地方？

Java 中的 HashMap 使用 hashCode()和 equals()方法来确定键值对的索引，当根据键获取值的时候也会用到这两个方法。如果没有正确的实现这两个方法，两个不同的键可能会有相同的 hash 值，因此，可能会被集合认为是相等的。而且，这两个方法也用来发现重复元素。所以这两个方法的实现对 HashMap 的精确性和正确性是至关重要的。

6.HashMap 和 Hashtable 有什么区别？

HashMap 和 Hashtable 都实现了 Map 接口，因此很多特性非常相似。但是，他们有以下不

同点：
HashMap 允许键和值是 null，而 Hashtable 不允许键或者值是 null。
Hashtable 是同步的，而 HashMap 不是。因此，HashMap 更适合于单线程环境，而 Hashtable 适合于多线程环境。

7.ArrayList 和 LinkedList 有什么区别？

ArrayList 和 LinkedList 都实现了 List 接口，他们有以下的不同点：
ArrayList 是基于索引的数据接口，它的底层是数组。它可以以 O(1)时间复杂度对元素进行随机访问，因此查询某个元素的时间短。与此对应，LinkedList 是以元素列表的形式存储它的数据，每一个元素都和它的前一个和后一个元素链接在一起，在这种情况下，查找某个元素的时间长。
相对于 ArrayList，LinkedList 的插入，添加，删除操作速度更快，因为当元素被添加到集合任意位置的时候，不需要像数组那样重新计算大小或者是更新索引。

8.Comparable 和 Comparator 接口是干什么的？列出它们的区别。

java 提供了只包含一个 compareTo()方法的 Comparable 接口。这个方法可以个给两个对象排序。具体来说，它返回负数，0，正数来表明输入对象小于，等于，大于已经存在的对象。
Java 提供了包含 compare()和 equals()两个方法的 Comparator 接口。compare()方法用来给两个输入参数排序，返回负数，0，正数表明第一个参数是小于，等于，大于第二个参数。equals()方法需要一个对象作为参数，它用来决定输入参数是否和 comparator 相等。只有当输入参数也是一个 comparator 并且输入参数和当前 comparator 的排序结果是相同的时候，这个方法才返回 true。

9.Enumeration 接口和 Iterator 接口的区别有哪些？

Enumeration 速度是 Iterator 的 2 倍，同时占用更少的内存。但是，Iterator 远远比 Enumeration 安全，因为其他线程不能够修改正在被 iterator 遍历的集合里面的对象。同时，Iterator 允许调用者删除底层集合里面的元素，这对 Enumeration 来说是不可能的。

10.HashSet 和 TreeSet 有什么区别？

Treeset 中的数据是自动排好序的，不允许放入 null 值
HashSet 中的数据是无序的，可以放入 null，但只能放入一个 null，两者中的值都不能重复，就如数据库中唯一约束
HashSet 要求放入的对象必须实现 HashCode()方法，放入的对象，是以 hashcode 码作为标识的，而具有相同内容的 String 对象，hashcode 是一样，所以放入的内容不能重复。但是同一个类的对象可以放入不同的实例

JAVAWEB

1.Ajax

Ajax 的原理简单来说通过 XmlHttpRequest 对象来向服务器发异步请求，从服务器获得数据，然后用 javascript 来操作 DOM 而更新页面。这其中最关键的一步就是从服务器获得请求数据。
XMLHttpRequest 是 ajax 的核心机制，它是在 IE5 中首先引入的，是一种支持异步请求的技术。简单的说，也就是 javascript 可以及时向服务器提出请求和处理响应，而不阻塞用户。达到无刷新的效果。
优点：
Ajax 的给我们带来的好处大家基本上都深有体会：
1、最大的一点是页面无刷新，在页面内与服务器通信，给用户的体验非常好。
2、使用异步方式与服务器通信，不需要打断用户的操作，具有更加迅速的响应能力。
3、可以把以前一些服务器负担的工作转嫁到客户端，利用客户端闲置的能力来处理，减轻服务器和带宽的负担，节约空间和宽带租用成本。并且减轻服务器的负担，ajax 的原则是“按需取数据”，可以最大程度的减少冗余请求，和响应对服务器造成的负担。
缺点：
下面所阐述的 ajax 的缺陷都是它先天所产生的。
1、ajax 干掉了 back 按钮，即对浏览器后退机制的破坏。后退按钮是一个标准的 web 站点的重要功能，但是它没法和 js 进行很好的合作。这是 ajax 所带来的一个比较严重的问题.
2、安全问题
技术同时也对 IT 企业带来了新的安全威胁，ajax 技术就如同对企业数据建立了一个直接通道。这使得开发者在不经意间会暴露比以前更多的数据和服务器逻辑。ajax 的逻辑可以对客户端的安全扫描技术隐藏起来，允许黑客从远端服务器上建立新的攻击。还有 ajax 也难以避免一些已知的安全弱点，诸如跨站点脚步攻击、SQL 注入攻击和基于 credentials 的安全漏洞等。

2.Servlet

主要分四大选择器，分别是基本选择器、层次选择器、过滤选择器、属性过滤选择器。
常用的选择器分为以下几种。
1. 基本选择器：
Id 选择器、class 选择器、element 选择器、*选择器、并列选择器。
2. 层级选择器：
parent > child（直系子元素）、
prev + next（下一个兄弟元素，等同于 next()方法）、
prev ~ siblings（prev 元素的所有兄弟元素，等同于 nextAll()方法）、
3. 过滤选择器：

first 和:last（取第一个元素或最后一个元素）
even 和:odd（取偶数索引或奇数索引元素，索引从 0 开始，even 表示偶数，odd 表示奇数）
t（取非元素）
eq(x) （取指定索引的元素）
gt(x)和:lt(x)（取大于 x 索引或小于 x 索引的元素）
header（取 H1~H6 标题元素）

3.Cookie&Session

会话(Session)跟踪是 Web 程序中常用的技术 ,用来跟踪用户的整个会话。常用的会话跟踪技术是 Cookie 与 Session。Cookie 通过在客户端记录信息确定用户身份，Session 通过在服务器端记录信息确定用户身份。

4.什么是 Cookie

Cookie 实际上是一小段的文本信息。客户端请求服务器，如果服务器需要记录该用户状态，就使用 response 向客户端浏览器颁发一个 Cookie。客户端浏览器会把 Cookie 保存起来。当浏览器再请求该网站时，浏览器把请求的网址连同该 Cookie 一同提交给服务器。服务器检查该 Cookie，以此来辨认用户状态。服务器还可以根据需要修改 Cookie 的内容。

5.Cookie 机制

在程序中，会话跟踪是很重要的事情。理论上，一个用户的所有请求操作都应该属于同一个会话，而另一个用户的所有请求操作则应该属于另一个会话，二者不能混淆。例如，用户 A 在超市购买的任何商品都应该放在 A 的购物车内，不论是用户 A 什么时间购买的，这都是属于同一个会话的，不能放入用户 B 或用户 C 的购物车内，这不属于同一个会话。
而 Web 应用程序是使用 HTTP 协议传输数据的。HTTP 协议是无状态的协议。一旦数据交换完毕，客户端与服务器端的连接就会关闭，再次交换数据需要建立新的连接。这就意味着服务器无法从连接上跟踪会话。即用户 A 购买了一件商品放入购物车内，当再次购买商品时服务器已经无法判断该购买行为是属于用户 A 的会话还是用户 B 的会话了。要跟踪该会话，必须引入一种机制。
Cookie 就是这样的一种机制。它可以弥补 HTTP 协议无状态的不足。在 Session 出现之前，基本上所有的网站都采用 Cookie 来跟踪会话。

6.什么是 Session

Session 是另一种记录客户状态的机制，不同的是 Cookie 保存在客户端浏览器中，而 Session 保存在服务器上。客户端浏览器访问服务器的时候，服务器把客户端信息以某种形式记录在服务器上。这就是 Session。客户端浏览器再次访问时只需要从该 Session 中查找该客户的状态就可以了。
如果说 Cookie 机制是通过检查客户身上的“通行证”来确定客户身份的话，那么 Session 机制

就是通过检查服务器上的“客户明细表”来确认客户身份。Session 相当于程序在服务器上建立的一份客户档案，客户来访的时候只需要查询客户档案表就可以了。

7.Session 机制

除了使用 Cookie，Web 应用程序中还经常使用 Session 来记录客户端状态。Session 是服务器端使用的一种记录客户端状态的机制，使用上比 Cookie 简单一些，相应的也增加了服务器的存储压力。

8.如何利用 cookie 实现自动登录?

当用户在某个网站注册后，就会收到一个惟一用户 ID 的 cookie。客户后来重新连接时，这个用户 ID 会自动返回，服务器对它进行检查，确定它是否为注册用户且选择了自动登录，从而使用户无需给出明确的用户名和密码，就可以访问服务器上的资源.

9.保存 session id 有几种方法?

A 保存 session id 的方式可以采用 cookie，这样在交互过程中浏览器可以自动的按照规则把这个标识发送给服务器。
B．由于 cookie 可以被人为的禁止，必须有其它的机制以便在 cookie 被禁止时仍然能够把 session id 传递回服务器，经常采用的一种技术叫做 URL 重写，就是把 session id 附加在 URL 路径的后面，附加的方式也有两种，一种是作为 URL 路径的附加信息，另一种是作为查询字符串附加在 URL 后面。网络在整个交互过程中始终保持状态，就必须在每个客户端可能请求的路径后面都包含这个 session id。
C．另一种技术叫做表单隐藏字段。就是服务器会自动修改表单，添加一个隐藏字段，以便在表单提交时能够把 session id 传递回服务器

10.session 什么时候被创建?

一个常见的错误是以为 session 在有客户端访问时就被创建，然而事实是直到某 server 端程序(如 Servlet)调用 HttpServletRequest.getSession(true)这样的语句时才会被创建。

11.session 何时被删除?

A. 程序调用 HttpSession.invalidate()
B．距离上一次收到客户端发送的 session id 时间间隔超过了 session 的最大有效时间
C．服务器进程被停止
注意关闭浏览器只会使存储在客户端浏览器内存中的 session cookie 失效，不会使服务器端的 session 对象失效.

12.cookie 机制和 session 机制的区别？

具体来说 cookie 机制采用的是在客户端保持状态的方案，而 session 机制采用的是在服务器端保持状态的方案。同时我们也看到，由于在服务器端保持状态的方案在客户端也需要保存一个标识，所以 session 机制可能需要借助于 cookie 机制来达到保存标识的目的，但实际上还有其他选择。

MySQL 数据库

1.MySQL 事务

MySQL 事务主要用于处理操作量大，复杂度高的数据。比如说，在人员管理系统中，你删除一个人员，你即需要删除人员的基本资料，也要删除和该人员相关的信息，如信箱，文章等等，这样，这些数据库操作语句就构成一个事务！

1. 在 MySQL 中只有使用了 Innodb 数据库引擎的数据库或表才支持事务

2. 事务处理可以用来维护数据库的完整性，保证成批的 SQL 语句要么全部执行，要么全部不执行

3. 事务用来管理 insert,update,delete 语句

一般来说，事务是必须满足 4 个条件（ACID）： Atomicity（原子性）、Consistency（稳定性）、Isolation（隔离性）、Durability（可靠性）

1. 事务的原子性：一组事务，要么成功；要么撤回。

2、稳定性 ： 有非法数据（外键约束之类），事务撤回。

3、隔离性：事务独立运行。一个事务处理后的结果，影响了其他事务，那么其他事务会撤回。事务的100%隔离，需要牺牲速度。

4、可靠性：软、硬件崩溃后，InnoDB 数据表驱动会利用日志文件重构修改。可靠性和高速度不可兼得， innodb_flush_log_at_trx_commit 选项 决定什么时候吧事务保存到日志里。

在 MySQL 控制台使用事务来操作

1，开始一个事务

start transaction

2，做保存点

savepoint 保存点名称

3，操作

4，可以回滚，可以提交，没有问题，就提交，有问题就回滚。

2. MySQL 索引

MySQL 索引的建立对于 MySQL 的高效运行是很重要的，索引可以大大提高 MySQL 的检索速度。

打个比方，如果合理的设计且使用索引的 MySQL 是一辆兰博基尼的话，那么没有设计和使用索引的 MySQL 就是一个人力三轮车。

索引分单列索引和组合索引。单列索引，即一个索引只包含单个列，一个表可以有多个单列索引，但这不是组合索引。组合索引，即一个索包含多个列。

创建索引时，你需要确保该索引是应用在 SQL 查询语句的条件(一般作为 WHERE 子句的条件)。

实际上，索引也是一张表，该表保存了主键与索引字段，并指向实体表的记录。

上面都在说使用索引的好处，但过多的使用索引将会造成滥用。因此索引也会有它的缺点：虽然索引大大提高了查询速度，同时却会降低更新表的速度，如对表进行 INSERT、UPDATE 和 DELETE。因为更新表时，MySQL 不仅要保存数据，还要保存一下索引文件。

建立索引会占用磁盘空间的索引文件。

1. 普通索引

1.创建索引

这是最基本的索引，它没有任何限制。它有以下几种创建方式：

```
CREATE INDEX indexName ON mytable(username(length));
```

如果是 CHAR，VARCHAR 类型，length 可以小于字段实际长度；如果是 BLOB 和 TEXT 类型，必须指定 length。

2.修改表结构

```
ALTER mytable ADD INDEX [indexName] ON (username(length))
```

3.创建表的时候直接指定

```
CREATE TABLE mytable(
```

```
ID INT NOT NULL,

username VARCHAR(16) NOT NULL,

INDEX [indexName] (username(length))

);
```

4.删除索引的语法

```
DROP INDEX [indexName] ON mytable;
```

2. 唯一索引

它与前面的普通索引类似，不同的就是：索引列的值必须唯一，但允许有空值。如果是组合索引，则列值的组合必须唯一。它有以下几种创建方式：

1.创建索引

```
REATE UNIQUE INDEX indexName ON mytable(username(length))
```

2.修改表结构

```
ALTER mytable ADD UNIQUE [indexName] ON (username(length))
```

3.创建表的时候直接指定

```
CREATE TABLE mytable(

ID INT NOT NULL,

username VARCHAR(16) NOT NULL,

UNIQUE [indexName] (username(length))

);
```

3. 使用 ALTER 命令添加和删除索引

有四种方式来添加数据表的索引：

- **1. ALTER TABLE tbl_name ADD PRIMARY KEY (column_list):**
- 该语句添加一个主键，这意味着索引值必须是唯一的，且不能为 NULL。
- **2. ALTER TABLE tbl_name ADD UNIQUE index_name (column_list):**
- 这条语句创建索引的值必须是唯一的（除了 NULL 外，NULL 可能会出现多次）。
- **3. ALTER TABLE tbl_name ADD INDEX index_name (column_list):**
- 添加普通索引，索引值可出现多次。
- **4. ALTER TABLE tbl_name ADD FULLTEXT index_name (column_list):**
- 该语句指定了索引为 FULLTEXT ，用于全文索引。

4.显示索引信息

你可以使用 SHOW INDEX 命令来列出表中的相关的索引信息。可以通过添加 \G 来格式化输出信息。

尝试以下实例：

```
mysql> SHOW INDEX FROM table_name\G
.....
```

3. MySQL 临时表

MySQL 临时表在我们需要保存一些临时数据时是非常有用的。临时表只在当前连接可见，当关闭连接时，MySQL 会自动删除表并释放所有空间。

临时表在 MySQL 3.23 版本中添加，如果你的 MySQL 版本低于 3.23 版本就无法使用 MySQL 的临时表。不过现在一般很少有再使用这么低版本的 MySQL 数据库服务了。

如果你使用了其他 MySQL 客户端程序连接 MySQL 数据库服务器来创建临时表，那么只有在关闭客户端程序时才会销毁临时表，当然你也可以手动销毁。

4. MySQL 处理重复数据

1). MySQL 处理重复数据

有些 MySQL 数据表中可能存在重复的记录，有些情况我们允许重复数据的存在，但有时 候我们也需要删除这些重复的数据。

2).防止表中出现重复数据

你可以在 MySQL 数据表中设置指定的字段为 PRIMARY KEY（主键） 或者 UNIQUE（唯一） 索引来保证数据的唯一性。

让我们尝试一个实例：下表中无索引及主键，所以该表允许出现多条重复记录。

```
CREATE TABLE person_tbl
(
    first_name CHAR(20),
    last_name CHAR(20),
    sex CHAR(10)
);
```

如果你想设置表中字段 first_name，last_name 数据不能重复，你可以设置双主键模式来设置数据的唯一性， 如果你设置了双主键，那么那个键的默认值不能为 NULL，可设置为 NOT NULL。如下所示：

```
CREATE TABLE person_tbl
(
   first_name CHAR(20) NOT NULL,
   last_name CHAR(20) NOT NULL,
   sex CHAR(10),
   PRIMARY KEY (last_name, first_name)
);
```

如果我们设置了唯一索引，那么在插入重复数据时，SQL 语句将无法执行成功,并抛出错。

INSERT IGNORE INTO 与 INSERT INTO 的区别就是 INSERT IGNORE 会忽略数据库中已经存在的数据，如果数据库没有数据，就插入新的数据，如果有数据的话就跳过这条数据。这样就可以保留数据库中已经存在数据，达到在间隙中插入数据的目的。

以下实例使用了 INSERT IGNORE INTO，执行后不会出错，也不会向数据表中插入重复数据：

```
mysql> INSERT IGNORE INTO person_tbl (last_name, first_name)
    -> VALUES( 'Jay', 'Thomas');
Query OK, 1 row affected (0.00 sec)
mysql> INSERT IGNORE INTO person_tbl (last_name, first_name)
    -> VALUES( 'Jay', 'Thomas');
Query OK, 0 rows affected (0.00 sec)
```

INSERT IGNORE INTO 当插入数据时，在设置了记录的唯一性后，如果插入重复数据，将不返回错误，只以警告形式返回。 而 REPLACE INTO into 如果存在 primary 或 unique 相同的记录，则先删除掉。再插入新记录。

另一种设置数据的唯一性方法是添加一个 UNIQUE 索引，如下所示：

```
CREATE TABLE person_tbl
(
   first_name CHAR(20) NOT NULL,
   last_name CHAR(20) NOT NULL,
   sex CHAR(10)
   UNIQUE (last_name, first_name)
);
```

3). 查询重复记录

```
select user_name,count(*) as count
       from user_table
   group by user_name having count>1;
select *
       from people
where peopleId in (select peopleId from people group by peopleId having count(peopleId) > 1)
```

4) . 统计重复数据

以下我们将统计表中 first_name 和 last_name 的重复记录数：

```
mysql> SELECT COUNT(*) as repetitions, last_name, first_name
    -> FROM person_tbl
    -> GROUP BY last_name, first_name
    -> HAVING repetitions > 1;
```

以上查询语句将返回 person_tbl 表中重复的记录数。 一般情况下，查询重复的值，请执行以下操作：

- 确定哪一列包含的值可能会重复。
- 在列选择列表使用 COUNT(*)列出的那些列。
- 在 GROUP BY 子句中列出的列。
- HAVING 子句设置重复数大于 1。

5). **过滤重复数据**

如果你需要读取不重复的数据可以在 SELECT 语句中使用 DISTINCT 关键字来过滤重复数据。

```
mysql> SELECT DISTINCT last_name, first_name
    -> FROM person_tbl
    -> ORDER BY last_name;
```

你也可以使用 GROUP BY 来读取数据表中不重复的数据：

```
mysql> SELECT last_name, first_name
    -> FROM person_tbl
    -> GROUP BY (last_name, first_name);
```

6). **删除重复数据**

如果你想删除数据表中的重复数据，你可以使用以下的 SQL 语句：

```
mysql> CREATE TABLE tmp SELECT last_name, first_name, sex
    ->                  FROM person_tbl;
    ->                  GROUP BY (last_name, first_name);
mysql> DROP TABLE person_tbl;
mysql> ALTER TABLE tmp RENAME TO person_tbl;
```

当然你也可以在数据表中添加 INDEX（索引） 和 PRIMAY KEY（主键）这种简单的方法来删除表中的重复记录。方法如下：

```
mysql> ALTER IGNORE TABLE person_tbl
    -> ADD PRIMARY KEY (last_name, first_name);
```

5. MySQL 导出数据

MySQL 中你可以使用 SELECT...INTO OUTFILE 语句来简单的导出数据到文本文件上。

1).使用 SELECT ... INTO OUTFILE 语句导出数据

以下实例中我们将数据表 w3cschool_tbl 数据导出到 /tmp/tutorials.txt 文件中：

```
mysql> SELECT * FROM tutorials_tbl
    -> INTO OUTFILE '/tmp/tutorials.txt';
```

你可以通过命令选项来设置数据输出的指定格式，以下实例为导出 CSV 格式：

```
mysql> SELECT * FROM passwd INTO OUTFILE '/tmp/tutorials.txt'
    -> FIELDS TERMINATED BY ',' ENCLOSED BY '"'
    -> LINES TERMINATED BY '\r\n';
```

在下面的例子中，生成一个文件，各值用逗号隔开。这种格式可以被许多程序使用。

```
SELECT a,b,a+b INTO OUTFILE '/tmp/result.text'
FIELDS TERMINATED BY ',' OPTIONALLY ENCLOSED BY '"'
LINES TERMINATED BY '\n'
FROM test_table;
```

2).SELECT ... INTO OUTFILE 语句有以下属性:

- LOAD DATA INFILE 是 SELECT ... INTO OUTFILE 的逆操作，SELECT 句法。为了将一个数据库的数据写入一个文件，使用 SELECT ... INTO OUTFILE，为了将文件读回数据库，使用 LOAD DATA INFILE。
- SELECT...INTO OUTFILE 'file_name'形式的 SELECT 可以把被选择的行写入一个文件中。该文件被创建到服务器主机上，因此您必须拥有 FILE 权限，才能使用此语法。
- 输出不能是一个已存在的文件。防止文件数据被篡改。
- 你需要有一个登陆服务器的账号来检索文件。否则 SELECT ... INTO OUTFILE 不会起任何作用。
- 在 UNIX 中，该文件被创建后是可读的，权限由 MySQL 服务器所拥有。这意味着，虽然你就可以读取该文件，但可能无法将其删除。

---

3).导出表作为原始数据

mysqldump 是 MySQL 用于转存储数据库的实用程序。它主要产生一个 SQL 脚本，其中包含从头重新创建数据库所必需的命令 CREATE TABLE INSERT 等。

使用 mysqldump 导出数据需要使用 --tab 选项来指定导出文件指定的目录，该目标必须是可写的。

以下实例将数据表 tutorials_tbl 导出到 /tmp 目录中：

```
$ mysqldump -u root -p --no-create-info \

            --tab=/tmp W3CSCHOOL w3cschool_tbl

password ******
```

4). 导出 SQL 格式的数据

导出 SQL 格式的数据到指定文件，如下所示：

```
$ mysqldump -u root -p W3CSCHOOL w3cschool_tbl > dump.txt

password ******
```

以上命令创建的文件内容如下：

```
-- MySQL dump 8.23

--

-- Host: localhost      Database: W3CSCHOOL

---------------------------------------------------------

-- Server version         3.23.58


--

-- Table structure for table `w3cschool_tbl`

--


CREATE TABLE w3cschool_tbl (

  w3cschool_id int(11) NOT NULL auto_increment,

  w3cschool_title varchar(100) NOT NULL default '',

  w3cschool_author varchar(40) NOT NULL default '',

  submission_date date default NULL,

  PRIMARY KEY  (w3cschool_id),

  UNIQUE KEY AUTHOR_INDEX (w3cschool_author)

) TYPE=MyISAM;


--

-- Dumping data for table `w3cschool_tbl`
```

```
--

INSERT INTO w3cschool_tbl
    VALUES (1,'Learn PHP','John Poul','2007-05-24');
INSERT INTO w3cschool_tbl
    VALUES (2,'Learn MySQL','Abdul S','2007-05-24');
INSERT INTO w3cschool_tbl
    VALUES (3,'JAVA Tutorial','Sanjay','2007-05-06');
```

如果你需要导出整个数据库的数据，可以使用以下命令：

```
$ mysqldump -u root -p W3CSCHOOL > database_dump.txt
password ******
```

如果需要备份所有数据库，可以使用以下命令：

```
$ mysqldump -u root -p --all-databases > database_dump.txt
password ******
```

--all-databases 选项在 MySQL 3.23.12 及以后版本加入。

该方法可用于实现数据库的备份策略。

---

5). 将数据表及数据库拷贝至其他主机

如果你需要将数据拷贝至其他的 MySQL 服务器上，你可以在 mysqldump 命令中指定数据库名及数据表。

在源主机上执行以下命令，将数据备份至 dump.txt 文件中：

```
$ mysqldump -u root -p database_name table_name > dump.txt
password *****
```

如果完整备份数据库，则无需使用特定的表名称。

如果你需要将备份的数据库导入到 MySQL 服务器中，可以使用以下命令，使用以下命令你需要确认数据库已经创建：

```
$ mysql -u root -p database_name < dump.txt password *****
```

你也可以使用以下命令将导出的数据直接导入到远程的服务器上，但请确保两台服务器是相通的，是可以相互访问的：</p>

```
$ mysqldump -u root -p database_name \
```

```
| mysql -h other-host.com database_name
```

以上命令中使用了管道来将导出的数据导入到指定的远程主机上。

6. MySQL 导入数据

MySQL 中可以使用两种简单的方式来导入 MySQL 导出的数据。

使用 LOAD DATA 导入数据

MySQL 中提供了 LOAD DATA INFILE 语句来插入数据。 以下实例中将从当前目录中读取文件 dump.txt ，将该文件中的数据插入到当前数据库的 mytbl 表中。

```
mysql> LOAD DATA LOCAL INFILE 'dump.txt' INTO TABLE mytbl;
```

如果指定 LOCAL 关键词，则表明从客户主机上按路径读取文件。如果没有指定，则文件在服务器上按路径读取文件。

你能明确地在 LOAD DATA 语句中指出列值的分隔符和行尾标记，但是默认标记是定位符和换行符。

两个命令的 FIELDS 和 LINES 子句的语法是一样的。两个子句都是可选的，但是如果两个同时被指定，FIELDS 子句必须出现在 LINES 子句之前。

如果用户指定一个 FIELDS 子句，它的子句 （TERMINATED BY、[OPTIONALLY] ENCLOSED BY 和 ESCAPED BY) 也是可选的，不过，用户必须至少指定它们中的一个。

```
mysql> LOAD DATA LOCAL INFILE 'dump.txt' INTO TABLE mytbl
  -> FIELDS TERMINATED BY ':'
  -> LINES TERMINATED BY '\r\n';
```

LOAD DATA 默认情况下是按照数据文件中列的顺序插入数据的，如果数据文件中的列与插入表中的列不一致，则需要指定列的顺序。

如，在数据文件中的列顺序是 a,b,c，但在插入表的列顺序为 b,c,a，则数据导入语法如下：

```
mysql> LOAD DATA LOCAL INFILE 'dump.txt'
    -> INTO TABLE mytbl (b, c, a);
```

使用 mysqlimport 导入数据

mysqlimport 客户端提供了 LOAD DATA INFILEQL 语句的一个命令行接口。mysqlimport 的大多数选项直接对应 LOAD DATA INFILE 子句。

从文件 dump.txt 中将数据导入到 mytbl 数据表中，可以使用以下命令：

```
$ mysqlimport -u root -p --local database_name dump.txt
password *****
```

mysqlimport 命令可以指定选项来设置指定格式,命令语句格式如下：

```
$ mysqlimport -u root -p --local --fields-terminated-by=":" \
   --lines-terminated-by="\r\n"  database_name dump.txt
password *****
```

mysqlimport 语句中使用 --columns 选项来设置列的顺序：

```
$ mysqlimport -u root -p --local --columns=b,c,a \
    database_name dump.txt
password *****
```

mysqlimport 的常用选项介绍

选项	功能
-d or --delete	新数据导入数据表中之前删除数据数据表中的所有信息
-f or --force	不管是否遇到错误，mysqlimport 将强制继续插入数据
-i or --ignore	mysqlimport 跳过或者忽略那些有相同唯一 关键字的行， 导入文件中的数据将被忽略。
-l or -lock-tables	数据被插入之前锁住表，这样就防止了， 你在更新数据库时，用户的查询和更新受到影响。
-r or -replace	这个选项与 –i 选项的作用相反；此选项将替代表中有相同唯一关键字的记录。
--fields-enclosed- by= char	指定文本文件中数据的记录时以什么括起的，很多情况下 数据以双引号括起。默认的情况下数据是没有被字符括起的。
--fields-terminated- by=char	指定各个数据的值之间的分隔符，在句号分隔的文件中，分隔符是句号。您可以用此选项指定数据之间的分隔符。默认的分隔符是跳格符（Tab）
--lines-terminated- by=str	此选项指定文本文件中行与行之间数据的分隔字符串或者字符。

默认的情况下 mysqlimport 以 newline 为行分隔符。
您可以选择用一个字符串来替代一个单个的字符：
一个新行或者一个回车。

mysqlimport 命令　　常用的选项还有-v 显示版本（version），-p 提示输入密码（password）等。

7. 存储过程

1).创建存储过程

1>.基本语法：

```
create procedure sp_name()
begin
.........
end
```

2>.参数传递

2)调用存储过程

基本语法：call sp_name()

注意：存储过程名称后面必须加括号，哪怕该存储过程没有参数传递

3).删除存储过程

1>.基本语法：

```
drop procedure sp_name//
```

2>.注意事项

(1)不能在一个存储过程中删除另一个存储过程，只能调用另一个存储过程

4.区块，条件，循环

1).区块定义，常用

```
begin
......
end;
```

也可以给区块起别名，如：

lable:begin

……..

end lable;

可以用 leave lable;跳出区块，执行区块以后的代码

2).条件语句

if 条件 then

statement

else

statement

end if;

3).循环语句

(1).while 循环

[label:] WHILE expression DO

statements

END WHILE [label] ;

(2).loop 循环

[label:] LOOP

statements

END LOOP [label];

(3).repeat until 循环

[label:] REPEAT

statements

UNTIL expression

END REPEAT [label] ;

5.其他常用命令

1).show procedure status

显示数据库中所有存储的存储过程基本信息，包括所属数据库，存储过程名称，创建时间等

2).show create procedure sp_name

显示某一个存储过程的详细信息

Oracle 数据库

1. 对索引的理解：

1、索引相当于字典的目录，作用在于提升查询效率，降低磁盘读写，提升数据库性能
2、索引是一种独立于表的数据库对象，可以存储在于表不同的磁盘中间或表空间
3、索引一旦创建，由 oracle 数据库自己来维护，并且由 oracle 管理系统来指定何时使用索引，我们不需要在查询语句中自己指定索引
4、索引的删除或损毁不会对数据库表带来影响，只会影响查询效率
5、创建索引的时候，如果没有指定表空间，会存储到默认的表空间
6、在删除表的时候，基于表的索引会被删除

2. 索引工作原理

创建索引的时候，oracle 会先在表空间中给索引开辟一个空间，并按索引字段对应的值进行分组，并把分好组的地址 rowid 存入索引空间中，当再次查询的时候，数据库会自动判断查询的条件中是否建有索引，如果有，则根据索引去查询符合条件的数据

3. 索引的创建

两种：1.自动创建--------即在创建主键 primarykey 或唯一性约束 unique 的时候，
数据库会自动在相应的列上　创建唯一性索引
一般业务过程中通过主键查询比较频繁，提升查询效率
2.手动创建--------即在不唯一的列上创建非唯一的索引，加速查询效率
create index index_xxx_xxx on 表（创建索引的列） tablespace 表空间（表空间可以不指定）工具创建

4. 索引的弊端：

一般在数据量比较大的表中并且经常查询的字段上才去建立索引，数据量小或不经常查询的字段建立索引的话，不仅会占用内存空间，而且会降低查询效率，索引不是越多越好，

下列情形不要创建索引

1、表很小
2、列不经常作为连接条件的
3、表经常更新的

系统日志历史表必须增加索引，效率超高

5. 执行计划 explain plan

作用：查询索引是否生效以及预估索引性能效果等
方式：1、explain plan window 窗口
　　2、对查询语句按 f5

6. 数据字典

主要存储的是数据库系统信息，由数据库系统自己去维护管理，一般我们只进行查询，不做修改

7. Pl/sql

是 oracle 对 sql 语言的过程化扩张，是存储过程的基础
组成：声明部分、可执行部分、异常处理部分
普通变量、引用变量(%type)、记录型变量（%rowtype）
1， 使用引用类型，当列中的数据类型发生改变，不需要修改变列的类型。而使用普通方式，当列的类型改变时，需要修改变列的类型
使用%TYPE 是非常好的编程风格，因为它使得 PL/SQL 更加灵活，更加适应于对数据库定义的更新。

8. 游标 cursor

可以理解为数据库表返回的结果集，它带有向前移动的指针，并且每次只向前移动一行数据
作用：可以临时存储返回的多行数据，通过变量游标，可以得到每一行的数据

9. 存储过程 program window--procedure

定义：是一组预编译的 sql 语句，也就是给 plsql 语句包装起来，完成一次创建任意调用的功能，相当于 java 中的方法
作用：在开发中，有时候为了某种特定的业务功能，需要对数据库进行多次的连接关闭，这种连接关闭是很耗费资源的，并且会对数据库进行多次的 io 读写，性能比较低，如果把业务功能放在 plsql 中，只需要

连接关闭一次数据库就可以实现我们的业务功能，大大提高了效率
2、更好的安全机制，对于没有权限执行存储过程的用户，也可以授权他们执行存储过程
3、对应大量的 sql 语句和重复执行的 sql 语句，存储过程执行要快
=
参数：
不带参数的、带输入参数的。带输入输出参数的

关于写存储的 3 个窗口的选择：
预发布的时候使用 command 测试使用 test

10. 存储过程和存储函数的区别：

1、存储过程可以有返回值也可以没有返回值，存储函数必须有返回值
2、存储过程和存储函数都可以通过输出参数 out 实现多个返回值
怎么选择：
原则上只有一个返回值的用存储函数，否则用存储过程
但是我们一般都使用存储过程，因为
1、存储过程可以有返回值也可以没有返回值，存储的灵活性
2、存储过程既然有返回值了，可以替代存储函数
3、Oracle 新版中已经不推荐适用存储函数了

Java 中通过 CallableStatement 调用存储过程

11. 触发器：

解释：
1、是一段 plsql 程序
2、它用来触发 dml(insert、update、delete)操作的
3、在进行 dml 操作时会自动触发执行的一段程序
换句话说：触发器就是在执行某个操作（增删改）的时候触发一个动作(一段程序)。
有点像 struts2 的拦截器，可以对 cud 增强

触发器类型
1、语句级触发器（表级触发器）
　在指定的操作语句执行之前或之后执行一次，不管它影响了多少行
2、行级触发器（for each row）
　触发语句作用都的每一条语句都会被触发

数据库三范式是什么?

第一范式（1NF）：

字段具有原子性,不可再分。所有关系型数据库系统都满足第一范式）
数据库表中的字段都是单一属性的，不可再分。例如，姓名字段，其中的姓和名必须作为一个整体，无法区分哪部分是姓，哪部分是名，如果要区分出姓和名，必须设计成两个独立的字段。

第二范式（2NF）：

第二范式（2NF）是在第一范式（1NF）的基础上建立起来的，即满足第二范式（2NF）必须先满足第一范式（1NF）。

要求数据库表中的每个实例或行必须可以被惟一地区分。通常需要为表加上一个列，以存储各个实例的惟一标识。这个惟一属性列被称为主关键字或主键。
第二范式（2NF）要求实体的属性完全依赖于主关键字。所谓完全依赖是指不能存在仅依赖主关键字一部分的属性，如果存在，那么这个属性和主关键字的这一部分应该分离出来形成一个新的实体，新实体与原实体之间是一对多的关系。为实现区分通常需要为表加上一个列，以存储各个实例的惟一标识。简而言之，第二范式就是非主属性非部分依赖于主关键字。

第三范式的要求如下：

满足第三范式（3NF）必须先满足第二范式（2NF）。简而言之，第三范式（3NF）要求一个数据库表中不包含已在其它表中已包含的非主关键字信息。
所以第三范式具有如下特征： 1，每一列只有一个值 2，每一行都能区分。 3，每一个表都不包含其他表已经包含的非主关键字信息。
例如，帖子表中只能出现发帖人的 id，而不能出现发帖人的 id，还同时出现发帖人姓名，否则，只要出现同一发帖人 id 的所有记录，它们中的姓名部分都必须严格保持一致，这就是数据冗余。

Struts2

1.1Struts2 的六个基本包

struts2-core-2.1.6.jar ： 开发的核心类库
freemarker-2.3.13.jar ： struts2 的 UI 标签的模板使用 freemarker 编写
commons-logging-1.0.4.jar ： 日志包
ognl-2.6.11.jar ： 对象图导航语言，通过它来读写对象属性
xwork-2.1.2.jar ：xwork 类库，struts2 在其上进行构建
commons-fileupload-1.2.1.jar：文件上传组件，2.1.6 版本后必须加入此 jar 包

1.2struts2 配置文件

（1）. web.xml 文件　主要完成对 StrutsPrepareAndExecuteFilter 的配置（在以前的版本中是对 FilterDispatcher 配置，新版本同样支持用 FilterDispatcher 配置），它的实质是一个过滤器，它负责初始化整个 Struts 框架并且处理所有的请求。这个过滤器可以包括一些初始化参数，有的参数指定了要加载哪些额外的 xml 配置文件，还有的会影响 struts 框架的行为。除了 StrutsPrepareAndExecuteFilter 外，Struts 还提供了一个 ActionContexCleanUp 类，它的主要任务是当有其它一些过滤器要访问一个初始化好了的 struts 框架的时候，负责处理一些特殊的清除任务。
（2）. struts.xml 文件　框架的核心配置文件就是这个默认的 struts.xml 文件，在这个默认的配置文件里面我们可以根据需要再包括其它一些配置文件。在通常的应用开发中，我们可

能想为每个不同的模块单独配置一个 struts.xml 文件，这样也利于管理和维护。这也是我们要配置的主要文件。

（3）. struts.properties（参 default.properties）　在 Struts 框架使用了很多属性，我们可以通过改变这些属性来满足我们的需求。要改变这些属性，只需在 struts.properties 文件中指定属性的 key 和 value 即可。属性文件可以放在任何一个包含在 classpath 中的路径上，但是通常我们都把它放在/WEB-INF/classes 目录下面。我们可以在 struts-default.properties 文件中找到一个属性的列表。

（4）struts-default.xml　此文件是 struts2 框架默认加载的配置文件，它定义了 struts2 一些核心 bean 和拦截器，它会自动包含(included)到 struts.xml 文件中(实质是通过<package extends="struts-default">)，并为我们提供了一些标准的配置。我们可以在 struts2-core.jar 中找到这个文件。

1.3Struts2 常用注解

@ParentPackage：定父包
@Namespace：指定命名空间
@Results：一组结果的数组
@Result(name="success",location="/msg.jsp")：　一个结果的映射（注意没有 s）
@Action(value="login")：指定某个请求处理方法的请求 URL。注意，它不能添加在 Action 类上，要添加到方法上。
@ExceptionMappings ：一级声明异常的数组
@ExceptionMapping：　映射一个声明异常自定义注解拦截器权限控制

1.4struts2 工作流程

Struts 2 框架本身大致可以分为 3 个部分：

核心控制器 FilterDispatcher、业务控制器 Action 和用户实现的企业业务逻辑组件。

核心控制器 FilterDispatcher 是 Struts 2 框架的基础，

包含了框架内部的控制流程和处理机制。

业务控制器 Action 和业务逻辑组件是需要用户来自己实现的。

用户在开发 Action 和业务逻辑组件的同时，还需要编写相关的配置文件，

供核心控制器 FilterDispatcher 来使用。

Struts 2 的工作流程相对于 Struts 1 要简单，与 WebWork 框架基本相同，

所以说 Struts 2 是 WebWork 的升级版本。基本简要流程如下：

1 、客户端初始化一个指向 Servlet 容器的请求；

2、 这个请求经过一系列的过滤器（Filter）

（这些过滤器中有一个叫做 ActionContextCleanUp 的可选过滤器，

这个过滤器对于 Struts2 和其他框架的集成很有帮助，例如：SiteMesh Plugin）

3 、接着 FilterDispatcher 被调用，

FilterDispatcher 询问 ActionMapper 来决定这个请是否需要调用某个 Action

4、如果 ActionMapper 决定需要调用某个 Action，

FilterDispatcher 把请求的处理交给 ActionProxy

5、ActionProxy 通过 Configuration Manager 询问框架的配置文件，

找到需要调用的 Action 类

6、ActionProxy 创建一个 ActionInvocation 的实例。

7、ActionInvocation 实例使用命名模式来调用，

在调用 Action 的过程前后，涉及到相关拦截器（Intercepter）的调用。

8、一旦 Action 执行完毕，ActionInvocation 负责根据 struts.xml 中的配置找到对应的返回结果。返回结果通常是（但不总是，也可 能是另外的一个 Action 链）一个需要被表示的 JSP 或者 FreeMarker 的模版。在表示的过程中可以使用 Struts2 框架中继承的标签。在这个过程中需要涉及到 ActionMapper

9、响应的返回是通过我们在 web.xml 中配置的过滤器

10、如果 ActionContextCleanUp 是当前使用的，则 FilterDispatecher 将不会清理 sreadlocal ActionContext; 如果 ActionContextCleanUp 不使用，则将会去清理 sreadlocals。

1.5Struts2 问题合集

1.5.1、说下 Struts 的设计模式

MVC 模式: web 应用程序启动时就会加载并初始化 ActionServlet。用户提交表单时，一个配置好的 ActionForm 对象被创建，并被填入表单相应的数据，ActionServlet 根据 Struts-config.xml 文件配置好的设置决定是否需要表单验证，如果需要就调用 ActionForm 的 Validate（）验证后选择将请求发送到哪个 Action，如果 Action 不存在，ActionServlet 会先创建这个对象，然后调用 Action 的 execute（）方法。Execute（）从 ActionForm 对象中获取数据，完成业务逻辑，返回一个 ActionForward 对象，ActionServlet 再把客户请求转发给 ActionForward 对象指定的 jsp 组件，ActionForward 对象指定的 jsp 生成动态的网页，返回给客户。

1.5.2、拦截器和过滤器的区别

1、拦截器是基于 java 反射机制的，而过滤器是基于函数回调的。

2、过滤器依赖于 servlet 容器，而拦截器不依赖于 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，拦截器可以多次调用，而过滤器只能在容器初始化时被调用一次。

1.5.3struts2 有哪些优点？

1）在软件设计上 Struts2 的应用可以不依赖于 Servlet API 和 struts API。Struts2 的这种设计属于无侵入式设计；

2）拦截器，实现如参数拦截注入等功能；

3）类型转换器，可以把特殊的请求参数转换成需要的类型；

4）多种表现层技术，如：JSP、freeMarker、Velocity 等；

5）Struts2 的输入校验可以对指定某个方法进行校验；

6）提供了全局范围、包范围和 Action 范围的国际化资源文件管理实现

1.5.4struts2 框架的核心控制器是什么？它有什么作用？

1）Struts2 框架的核心控制器是 StrutsPrepareAndExecuteFilter。

2）作用：

负责拦截由<url-pattern>/*</url-pattern>指定的所有用户请求，当用户请求到达时，该 Filter 会过滤用户的请求。默认情况下，如果用户请求的路径不带后缀或者后缀以.action 结尾，这时请求将被转入 struts2 框架处理，否则 struts2 框架将略过该请求的处理。

可以通过常量"struts.action.extension"修改 action 的后缀，如：

```
<constant name="struts.action.extension" value="do"/>
```

如果用户需要指定多个请求后缀，则多个后缀之间以英文逗号（,）隔开。

```
<constant name="struts.action.extension" value="do,go"/>
```

1.5.5struts2 默认能解决 get 和 post 提交方式的乱码问题吗？

不能。struts.i18n.encoding=UTF-8 属性值只能解析 POST 提交下的乱码问题。

1.5.6 值栈 ValueStack 的原理与生命周期？

1）ValueStack 贯穿整个 Action 的生命周期，保存在 request 域中，所以 ValueStack 和 request 的生命周期一样。当 Struts2 接受一个请求时，会迅速创建 ActionContext，

ValueStack，action。然后把 action 存放进 ValueStack，所以 action 的实例变量可以被 OGNL 访问。 请求来的时候，action、ValueStack 的生命开始，请求结束，action、 ValueStack 的生命结束；

2）action 是多例的，和 Servlet 不一样，Servelt 是单例的；

3）每个 action 的都有一个对应的值栈，值栈存放的数据类型是该 action 的实例，以及该 action 中的实例变量，Action 对象默认保存在栈顶；

4）ValueStack 本质上就是一个 ArrayList；

5）关于 ContextMap，Struts 会把下面这些映射压入 ContextMap 中：

parameters : 该 Map 中包含当前请求的请求参数

request : 该 Map 中包含当前 request 对象中的所有属性 session :该 Map 中包含当前 session 对象中的所有属性

application :该 Map 中包含当前 application 对象中的所有属性

attr:该 Map 按如下顺序来检索某个属性: request, session, application

6）使用 OGNL 访问值栈的内容时，不需要#号，而访问 request、session、application、attr 时，需要加#号；

7）注意：　Struts2 中，OGNL 表达式需要配合 Struts 标签才可以使用。如：

<s:property value="name"/>

8）在 struts2 配置文件中引用 ognl 表达式 ,引用值栈的值 ，此时使用的"$"，而不是#或者%;

1.5.7ActionContext、ServletContext、pageContext 的区别？

1）ActionContext 是当前的 Action 的上下文环境，通过 ActionContext 可以获取到 request、session、ServletContext 等与 Action 有关的对象的引用；

2）ServletContext 是域对象，一个 web 应用中只有一个 ServletContext，生命周期伴随整个 web 应用；

3）pageContext 是 JSP 中的最重要的一个内置对象，可以通过 pageContext 获取其他域对象的应用，同时它是一个域对象，作用范围只针对当前页面，当前页面结束时，pageContext 销毁，

生命周期是 JSP 四个域对象中最小的。

1.5.8result 的 type 属性中有哪几种结果类型？

一共 10 种：
dispatcher
struts 默认的结果类型，把控制权转发给应用程序里的某个资源不能把控制权转发给一个外部资源，若需要把控制权重定向到一个外部资源，应该使用
redirect 结果类型
redirect 把响应重定向到另一个资源（包括一个外部资源）
redirectAction 把响应重定向到另一个 Action
freemarker、velocity、chain、httpheader、xslt、plainText、stream

1.5.9servlet 和 filter 的区别

Filter 可认为是 Servlet 的一种“变种”，它主要用于对用户请求进行预处理，

也可以对 HttpServletResponse 进行后处理，是个典型的处理链。它与 Servlet 的区别在于：它不能直接向用户生成响应。完整的流程是：Filter 对用户请求进行预处理，接着将请求交给 Servlet 进行处理并生成响应，最后 Filter 再对服务器响应进行后处理。

Hibernate

2.1Hibernate 的 jar 包

最基本的 Hibernate3.3.2 之 JAR 包（必要）：

hibernate3.jar　　核心 JAR 包

antlr.jar　　Another Tool for Language Recognition，可以构造语言识别器，解析 HQL 需要

commons-collections.jar　　包含了一些 Apache 开发的集合类，功能比 **Java**.util.*强大

dom4j.jar　越来越多的 Java 软件都在使用 dom4j 来操作 XML，Hibernate 也不例外

javassist.jar　操作字节码，跟 cglib 相关

jta.jar　　定义 JTA 规范的 JAR 包，当 Hibernate 使用 JTA 的时候需要

slf4j.jar　　整合各志框架种日的工具

slf4j-nop.jar　　包含了对 slf4j.jar 的实现类

2.2Hibernate 的核心配置文件

Hibernate 框架支持 properties 和 xml 两种方式的 Hibernate 属性的配置，对应的两种核心配置文件为：
hibernate.properties 配置常用的属性,必须手动加载 hbm 文件或持久化类。（了解）
hibernate.cfg.xml 配置常用的属性,配置加载 hbm 映射,配置缓存策略等。（推荐）

2.3Hibernate 的核心 API 接口

所有的 Hibernate 应用中都会访问 Hibernate 的 5 个核心接口。
Configuration 接口：配置 Hibernate,根启动 Hibernate，创建 SessionFactory 对象。
SessionFactory 接口：初始化 Hibernate，充当数据存储源的代理，创建 Session 对象。
Session 接口：负责保存、更新、删除、加载和查询对象。
Transaction 接口：管理事务。
Query 和 Criteria 接口：执行数据库查询。

2.3.1 Configuration 接口

Configuration 对象用于配置并且启动 Hibernate。Hibernate 应用通过 Configuration 实例来指定对象-关系映射文件的位置或者动态配置 Hibernate 的属性，然后创建 SessionFactory 实例。

2.3.2 SessionFactory 接口

一个 SessionFactory 实例对应一个数据存储源，应用从 SessionFactory 中获得 Session 实例。
SessionFactory 有以下特点：
它是线程安全的，这意味着它的同一个实例可以被应用的多个线程共享。
它是重量级的，这意味着不能随意创建或销毁它的实例。如果应用只访问一个数据库，只需要创建一个 SessionFactory 实例，在应用初始化的时候创建该实例。如果应用同时访问多个数据库，则需要为每个数据库创建一个单独的 SessionFactory 实例。
之所以称 SessionFactory 是重量级的，是因为它需要一个很大的缓存，用来存放预定义的 SQL 语句以能映射元数据等。用户还可以为 SesionFactory 配置一个缓存插件，这个缓存插件被称为 Hibernate 的第二级缓存。，该缓存用来存放被工作单元读过的数据，将来其他工作单元可能会重用这些数据，因此这个缓存中的数据能够被所有工作单元共享。一个工作单元通常对应一个数据库事务。

2.3.3 Session 接口

Session 接口是 Hibernate 应用使用最广泛的接口。Session 也被称为持久化管理器，它提供了和持久化相关的操作，如添加、更新、删除、加载和查询对象。
Session 有以下特点：
不是线程安全的，因此在设计软件架构时，应该避免多个线程共享同一个 Session 实例。
Session 实例是轻量级的，所谓轻量级，是指它的创建和销毁不需要消耗太多的资源。这意味着在程序中可以经常创建和销毁 Session 对象，例如为每个客户请示分配单独的 Session 实例，或者为每个工作单元分配单独的 Session 实例。
Session 有一个缓存，被称为 Hibernate 的第一级缓存，它存放被当前工作单元加载的对象。每个 Session 实例都有自己的缓存，这个 Sesion 实例的缓存只能被当前工作单元访问。

2.3.4Transaction 接口

Transaction 接口是 Hibernate 的数据库事务接口，它对底层的事务接口做了封装，底层事务接口包括：
JDBC API、JTA（Java Transaction API）、CORBA（Common Object Requet Broker Architecture）API
Hibernate 应用可通过一致的 Transaction 接口来声明事务边界，这有助于应用在不同的环境容器中移植。尽管应用也可以绕过 Transaction 接口，直接访问底层的事务接口，这种方法不值得推荐，因为它不利于应用在不同的环境移植。

2.3.5Query 和 Criteria 接口

Query 和 Criteria 接口是 Hibernate 的查询接口，用于向数据库查询对象，以及控制执行查询的过程。Query 实例包装了一个 HQL 查询语句，HQL 查询语句和 SQL 查询语句有些相似，但 HQL 查询语句是面向对象的，它引用类句及类的属性句，而不是表句及表的字段句。Criteria 接口完全封装了基于字符串的查询语句，比 Query 接口更加面向对象，Criteria 接口擅长执行动态查询。
Session 接口的 find（）方法也具有数据查询功能，但它只是执行一些简单的 HQL 查询语句的快捷方法，它的功能远没有 Query 接口强大。

2.4hibernate 常用注解

@Entity：声明实体 bean，每一个持久化 POJO 类都是一个实体 bean,这可以通过在类的定义中使用@Entity 注解来进行声明:
@Id：注解则声明了该实体 bean 的标识属性，对应相应表使用 id 列作为主键列
@Table：是类一级的注解，通过@Table 注解可以为实体 bean 映射指定表(table),目录(catalog)和 schema 的名字. 如果没有定义@Table,那么系统自动使用默认值：实体的短类名(不附带包名).
@Transient：实体 bean 中所有的非 static 非 transient 的属性都可以被持久化，除非你将其注解为@Transient，所有没有定义注解的属性等价于在其上面添加了@Basic 注解.
@GeneratedValue：定义该标识符的生成策略
AUTO - 可以是 identity column 类型,或者 sequence 类型或者 table 类型,取决于不同的底层数据库.
TABLE - 使用表保存 id 值
IDENTITY - identity column
SEQUENCE - sequence
@OneToOne 注解可以建立实体 bean 之间的一对一的关联.
@ManyToOne 注解来定义多对一关联

2.5 Hibernate 的执行流程

1.通过 Configuration.configure()读取并解析 hibernate.cfg.xml 配置文件；
2.由 hibernate.cfg.xml 中的<mapping resource=”com/xx/User.hbm.xml”/>读取并解析映射信息；
3.通过 config.buildSerssionFactory()创建 SessionFactory;
4.sessionFactory.openSession()打开 session；
5.session.beginTransaction()创建事务 transation；
6.persistent operate（持久化操作）；
7.session.getTransaction().commit()提交事务；
8.关闭 session;
9.关闭 SessionFactory。

2.6Hibernate 问题合集

2.6.1Hibernate 中 get 和 load 有什么不同之处?

把 get 和 load 放到一起进行对比是 Hibernate 面试时最常问到的问题，这是因为只有正确理解 get()和 load()这二者后才有可能高效地使用 Hibernate。get 和 load 的最大区别是，如果在缓存中没有找到相应的对象，get 将会直接访问数据库并返回一个完全初始化好的对象，而这个过程有可能会涉及到多个数据库调用；而 load 方法在缓存中没有发现对象的情况下，只会返回一个代理对象，只有在对象 getId()之外的其它方法被调用时才会真正去访问数据库，这样就能在某些情况下大幅度提高性能。你也可以参考 Hibernate 中 get 和 load 的不同之处， 此链接给出了更多的不同之处并对该问题进行了更细致的讨论。

2.6.2Hibernate 中 save、persist 和 saveOrUpdate 这三个方法的不同之处?

除了 get 和 load，这又是另外一个经常出现的 Hibernate 面试问题。 所有这三个方法，也就是 save()、saveOrUpdate()和 persist()都是用于将对象保存到数据库中的方法，但其中有些细微的差别。例如，save()只能 INSERT 记录，但是 saveOrUpdate()可以进行 记录的 INSERT 和 UPDATE。还有，save()的返回值是一个 Serializable 对象，而 persist()方法返回值为 void。你还可以访问 save、persist 以及 saveOrUpdate，找到它们所有的不同之处。

2.6.3Hibernate 中的命名 SQL 查询指的是什么?

Hibernate 的这个面试问题同 Hibernate 提供的查询功能相关。命名查询指的是用<sql-query>标签在影射文档中定义的 SQL 查询，可以通过使用 Session.getNamedQuery()方法对它进行调用。命名查询使你可以使用你所指定的一个名字拿到某个特定的查询。 Hibernate

中的命名查询可以使用注解来定义，也可以使用我前面提到的 xml 影射问句来定义。在 Hibernate 中，@NameQuery 用来定义单个的命名查询，@NameQueries 用来定义多个命名查询。

2.6.4Hibernate 中的 SessionFactory 有什么作用? SessionFactory 是线程安全的吗？

这也是 Hibernate 框架的常见面试问题。顾名思义，SessionFactory 就是一个用于创建 Hibernate 的 Session 对象的工厂。SessionFactory 通常是在应用启动时创建好的，应用程序中的代码用它来获得 Session 对象。作为一个单个的数据存储，它也是 线程安全的，所以多个线程可同时使用同一个 SessionFactory。Java JEE 应用一般只有一个 SessionFactory，服务于客户请求的各线程都通过这个工厂来获得 Hibernate 的 Session 实例，这也是为什么 SessionFactory 接口的实现必须是线程安全的原因。还有，SessionFactory 的内部状态包含着同对象关系影射有关的所有元数据，它是 不可变的，一旦创建好后就不能对其进行修改了。

2.6.5Hibernate 中的 Session 指的是什么？可否将单个的 Session 在多个线程间进行共享？

前面的问题问完之后，通常就会接着再问这两个问题。问完 SessionFactory 的问题后就该轮到 Session 了。Session 代表着 Hibernate 所做的一小部分工作，它负责维护者同数据库的链接而且 不是线程安全的，也就是说，Hibernage 中的 Session 不能在多个线程间进行共享。虽然 Session 会以主动滞后的方式获得数据库连接，但是 Session 最好还是在用完之后立即将其关闭。

2.6.6Hibernate 中二级缓存指的是什么？

这是同 Hibernate 的缓存机制相关的第一个面试问题，不出意外后面还会有更多这方面的问题。二级缓存是在 SessionFactory 这个级别维护的缓存，它能够通过节省几番数据库调用往返来提高性能。还有一点值得注意，二级缓存是针对整个应用而不是某个特定的 session 的。

2.6.7Hibernate 中的查询缓存指的是什么？

这个问题有时是作为上个 Hibernate 面试问题的后继问题提出的。查询缓存实际上保存的是 sql 查询的结果，这样再进行相同的 sql 查询就可以之间从缓存中拿到结果了。为了改善性能，查询缓存可以同二级缓存一起来使用。Hibernate 支持用多种不同的开源缓存方案，比如 EhCache，来实现查询缓存。

Spring

3.1Spring 常用 jar 包

1. spring.jar 是包含有完整发布模块的单个 jar 包。
2. org.springframework.aop 包含在应用中使用 Spring 的 AOP 特性时所需的类。
3. org.springframework.aspects 提供对 AspectJ 的支持，以便可以方便的将面向方面的功能集成进 IDE 中，
   比如 Eclipse AJDT。
4. org.springframework.beans 所有应用都要用到的，它包含访问配置文件、创建和管理 bean 以及进行 Inversion of Control / Dependency Injection（IoC/DI）操作相关的所有类。
5. org.springframework.context 为 Spring 核心提供了大量扩展。可以找到使用 Spring ApplicationContext
   特性时所需的全部类，JDNI 所需的全部类，UI 方面的用来与模板（Templating）引擎如 Velocity、FreeMarker、
   JasperReports 集成的类，以及校验 Validation 方面的相关类。
6. org.springframework.core 包含 Spring 框架基本的核心工具类，Spring 其它组件要都要使用到这个包里的类，
   是其它组件的基本核心。
7. org.springframework.expression Spring 表达式语言。
8. org.springframework.oxm Spring 对 Object/XMl 的映射支持,可以让 Java 与 XML 之间来回切换。
9. org.springframework.test 对 Junit 等测试框架的简单封装。
10. org.springframework.transaction 为 JDBC、Hibernate、JDO、JPA 等提供的一致的声明式和编程式事务管理。

3.2Spring 核心 IOC AOP

3.2.1 IOC，DI

IoC Inverse of Control 反转控制的概念，就是将原本在程序中手动创建 UserService 对象的控制权，交由 Spring 框架管理，简单说，就是创建 UserService 对象控制权被反转到了 Spring 框架
DI：Dependency Injection 依赖注入，在 Spring 框架负责创建 Bean 对象时，动态的将依赖对象注入到 Bean 组件

3.2.2AOP

面向切面编程（AOP）提供另外一种角度来思考程序结构，通过这种方式弥补了面向对象编程（OOP）的不足，除了类（classes）以外，AOP 提供了切面。切面对关注点进行模块化，

例如横切多个类型和对象的事务管理
Spring 的一个关键的组件就是 AOP 框架，可以自由选择是否使用 AOP 提供声明式企业服务，特别是为了替代 EJB 声明式服务。最重要的服务是声明性事务管理，这个服务建立在 Spring 的抽象事物管理之上。允许用户实现自定义切面，用 AOP 来完善 OOP 的使用,可以把 Spring AOP 看作是对 Spring 的一种增强

3.2.2.1AOP 中的名词

切面（Aspect）：一个关注点的模块化，这个关注点可能会横切多个对象。事务管理是 J2EE 应用中一个关于横切关注点的很好的例子。 在 Spring AOP 中，切面可以使用通用类（基于模式的风格） 或者在普通类中以 @Aspect 注解（@AspectJ 风格）来实现。
连接点（Joinpoint）：在程序执行过程中某个特定的点，比如某方法调用的时候或者处理异常的时候。 在 Spring AOP 中，一个连接点 总是 代表一个方法的执行。 通过声明一个 org.aspectj.lang.JoinPoint 类型的参数可以使通知（Advice）的主体部分获得连接点信息。
通知（Advice）：在切面的某个特定的连接点（Joinpoint）上执行的动作。通知有各种类型，其中包括“around”、“before”和“after”等通知。 通知的类型将在后面部分进行讨论。许多 AOP 框架，包括 Spring，都是以拦截器做通知模型， 并维护一个以连接点为中心的拦截器链。
切入点（Pointcut）：匹配连接点（Joinpoint）的断言。通知和一个切入点表达式关联，并在满足这个切入点的连接点上运行（例如，当执行某个特定名称的方法时）。 切入点表达式如何和连接点匹配是 AOP 的核心：Spring 缺省使用 AspectJ 切入点语法。
引入（Introduction）：（也被称为内部类型声明（inter-type declaration））。声明额外的方法或者某个类型的字段。 Spring 允许引入新的接口（以及一个对应的实现）到任何被代理的对象。例如，你可以使用一个引入来使 bean 实现 IsModified 接口，以便简化缓存机制。
目标对象（Target Object）： 被一个或者多个切面（aspect）所通知（advise）的对象。也有人把它叫做 被通知（advised） 对象。 既然 Spring AOP 是通过运行时代理实现的，这个对象永远是一个 被代理（proxied） 对象。
AOP 代理（AOP Proxy）：AOP 框架创建的对象，用来实现切面契约（aspect contract）（包括通知方法执行等功能）。在 Spring 中，AOP 代理可以是 JDK 动态代理或者 CGLIB 代理。注意：Spring 2.0 最新引入的基于模式（schema-based）风格和@AspectJ 注解风格的切面声明，对于使用这些风格的用户来说，代理的创建是透明的。
织入（Weaving）：把切面（aspect）连接到其它的应用程序类型或者对象上，并创建一个被通知（advised）的对象。 这些可以在编译时（例如使用 AspectJ 编译器），类加载时和运行时完成。 Spring 和其他纯 Java AOP 框架一样，在运行时完成织入。

3.3spring 常用注解

@Component：所有受 Spring 管理组件的通用形式，不推荐使用
@Controller：对应表现层的 Bean，也就是 Action
使用@Controller 注解标识 UserAction 之后，就表示要把 UserAction 交给 Spring 容器管理，在 Spring 容器中会存在一个名字为"userAction"的 action，这个名字是根据 UserAction 类名来取的。注意：如果@Controller 不指定其 value【@Controller】，则默认的 bean 名字为这

个类的类名首字母小写，如果指定 value【@Controller(value="UserAction")】或者【@Controller("UserAction")】，则使用 value 作为 bean 的名字。
@Scope("prototype")：表示将 Action 的范围声明为原型。可以利用容器的 scope="prototype"来保证每一个请求有一个单独的 Action 来处理，避免 struts 中 Action 的线程安全问题。spring 默认 scope 是单例模式(scope="singleton")，这样只会创建一个 Action 对象，每次访问都是同一 Action 对象，数据不安全，struts2 是要求每次次访问都对应不同的 Action，scope="prototype" 可以保证当有请求的时候都创建一个 Action 对象
@Service：对应的是业务层 Bean
@Repository：对应数据访问层 Bean
补充：理解 Spring 中的"控制反转"
如，在 UserAction 中要用到 UserServiceImpl 中的方法，最初的方法是在 UserAction 中通过 UserService userService = new UserServiceImpl;需要时就 new 出来，控制权在自己手里，但是使用 spring 后，UserAction 就不用主动去创建 UserServiceImpl 的实例了，创建 UserServiceImpl 实例已经交给 Spring 来做了，Spring 把创建好的 UserServiceImpl 实例给 UserAction，UserAction 拿到就可以直接用了。UserAction 由原来的主动创建 UserServiceImpl 实例后就可以马上使用，变成了被动等待由 Spring 创建好 UserServiceImpl 实例之后再注入给 UserAction，UserAction 才能够使用。这说明 UserAction 对 UserServiceImpl 类的"控制权"已经被"反转"了

3.4Spring 常见问题

3.4.1 spring 配置 bean 实例化有哪些方式？

1）使用类构造器实例化(默认无参数)

```
<bean id="bean1" class="cn.itcast.spring.b_instance.Bean1"></bean>
```

2）使用静态工厂方法实例化(简单工厂模式)

```
//下面这段配置的含义：调用 Bean2Factory 的 getBean2 方法得到 bean2
<bean id="bean2" class="cn.itcast.spring.b_instance.Bean2Factory" factory-
method="getBean2"></bean>
```

3）使用实例工厂方法实例化(工厂方法模式)

```
//先创建工厂实例 bean3Facory，再通过工厂实例创建目标 bean 实例
<bean id="bean3Factory" class="cn.itcast.spring.b_instance.Bean3Factory"></bean>
<bean id="bean3" factory-bean="bean3Factory" factory-method="getBean3"></bean>
```

3.4.2 Spring Bean 的生命周期？

①instantiate bean 对象实例化
②populate properties 封装属性
③如果 Bean 实现 BeanNameAware 执行 setBeanName
④如果 Bean 实现 BeanFactoryAware 或者 ApplicationContextAware 设置工厂 setBeanFactory 或者上下文对象 setApplicationContext

⑤如果存在类实现 BeanPostProcessor（后处理 Bean），执行 postProcessBeforeInitialization，BeanPostProcessor 接口提供钩子函数，用来动态扩展修改 Bean。(程序自动调用后处理 Bean)
⑥如果 Bean 实现 InitializingBean 执行 afterPropertiesSet
⑦调用<bean init-method="init"> 指定初始化方法 init
⑧如果存在类实现 BeanPostProcessor（处理 Bean），执行 postProcessAfterInitialization
⑨执行业务处理
⑩如果 Bean 实现 DisposableBean 执行 destroy
⑪调用<bean destroy-method="customerDestroy"> 指定销毁方法 customerDestroy

3.4.3 Bean 注入属性有哪几种方式？

```
1.接口注入
public interface Injection{
      public void injectionName(String name);
}
为对象注入name属性
public class User implements Injection{
      private String name;
      public void injectName(String name){
            this.name = name;
      }
}
```

```
2.构造器注入
public class User{
      private String name;
      public User(String name){
          this.name = name;
      }
}
```

```
3.set注入
public class User{
      private String name;
      public void setName(String name){
          this.name = name;
      }
}
```

spring 支持构造器注入和 setter 方法注入
构造器注入，通过 <constructor-arg> 元素完成注入
setter 方法注入， 通过<property> 元素完成注入【开发中常用方式】

3.4.4Spring 如何处理线程并发问题？

Spring 使用 ThreadLocal 解决线程安全问题

我们知道在一般情况下，只有无状态的 Bean 才可以在多线程环境下共享，在 Spring 中，绝大部分 Bean 都可以声明为 singleton 作用域。就是因为 Spring 对一些 Bean(如

RequestContextHolder、TransactionSynchronizationManager、LocaleContextHolder 等)中非线程安全状态采用 ThreadLocal 进行处理，让它们也成为线程安全的状态，因为有状态的 Bean 就可以在多线程中共享了。

ThreadLocal 和线程同步机制都是为了解决多线程中相同变量的访问冲突问题。
在同步机制中，通过对象的锁机制保证同一时间只有一个线程访问变量。这时该变量是多个线程共享的，使用同步机制要求程序慎密地分析什么时候对变量进行读写，什么时候需要锁定某个对象，什么时候释放对象锁等繁杂的问题，程序设计和编写难度相对较大。

而 ThreadLocal 则从另一个角度来解决多线程的并发访问。ThreadLocal 会为每一个线程提供一个独立的变量副本，从而隔离了多个线程对数据的访问冲突。因为每一个线程都拥有自己的变量副本，从而也就没有必要对该变量进行同步了。ThreadLocal 提供了线程安全的共享对象，在编写多线程代码时，可以把不安全的变量封装进 ThreadLocal。
由于 ThreadLocal 中可以持有任何类型的对象，低版本 JDK 所提供的 get()返回的是 Object 对象，需要强制类型转换。但 JDK5.0 通过泛型很好的解决了这个问题，在一定程度地简化 ThreadLocal 的使用。

概括起来说，对于多线程资源共享的问题，同步机制采用了“以时间换空间”的方式，而 ThreadLocal 采用了“以空间换时间”的方式。前者仅提供一份变量，让不同的线程排队访问，而后者为每一个线程都提供了一份变量，因此可以同时访问而互不影响。

3.4.5 介绍一下 Spring 的事物管理

Spring 支持两种类型的事务管理：
编程式事务管理：这意味你通过编程的方式管理事务，给你带来极大的灵活性，但是难维护。
声明式事务管理：这意味着你可以将业务代码和事务管理分离，你只需用注解和 XML 配置来管理事务。

3.4.6 通知有哪些类型？

前置通知（Before advice）：在某连接点（join point）之前执行的通知，但这个通知不能阻止连接点前的执行（除非它抛出一个异常）。
返回后通知（After returning advice）：在某连接点（join point）正常完成后执行的通知：例如，一个方法没有抛出任何异常，正常返回。
抛出异常后通知（After throwing advice）：在方法抛出异常退出时执行的通知。
后通知（After (finally) advice）：当某连接点退出的时候执行的通知（不论是正常返回还是异常退出）。
环绕通知（Around Advice）：包围一个连接点（join point）的通知，如方法调用。这是最强大的一种通知类型。 环绕通知可以在方法调用前后完成自定义的行为。它也会选择是否继续执行连接点或直接返回它们自己的返回值或抛出异常来结束执行。

环绕通知是最常用的一种通知类型。大部分基于拦截的 AOP 框架，例如 Nanning 和 JBoss4，都只提供环绕通知。
切入点（pointcut）和连接点（join point）匹配的概念是 AOP 的关键，这使得 AOP 不同于其它仅仅提供拦截功能的旧技术。 切入点使得定位通知（advice）可独立于 OO 层次。 例如，一个提供声明式事务管理的 around 通知可以被应用到一组横跨多个对象中的方法上(例

如服务层的所有业务操作)。

3.4.7 BeanFactory 接口和 ApplicationContext 接口有什么区别？

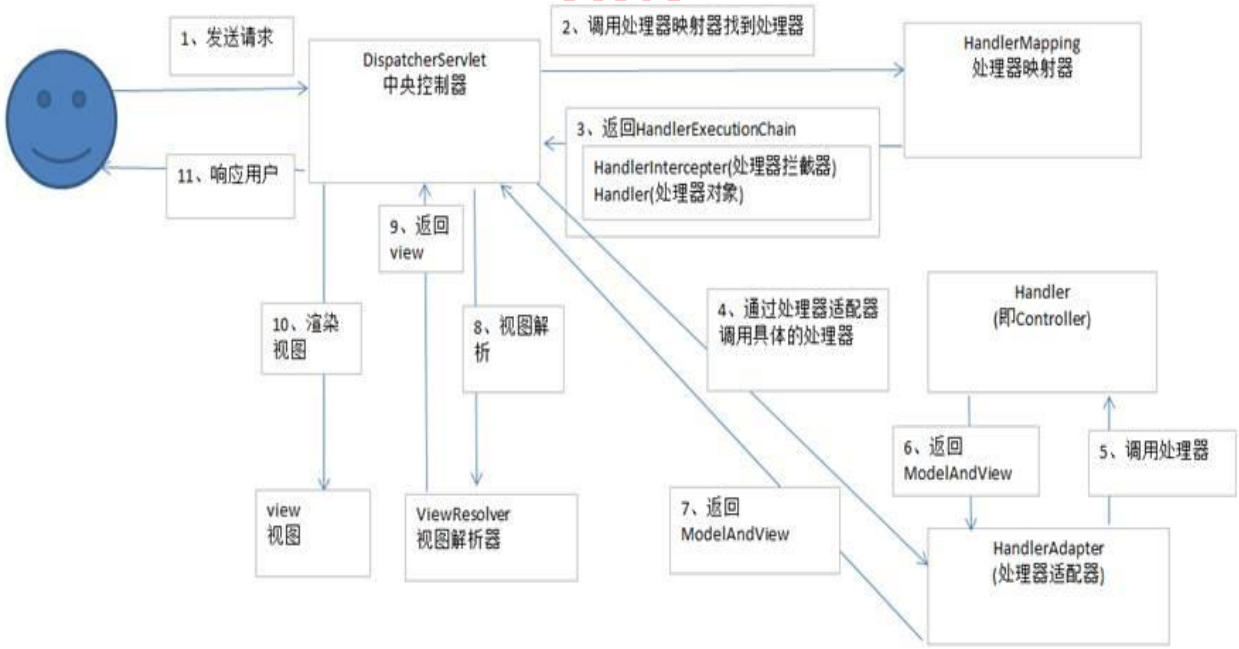
① ApplicationContext 接口继承 BeanFactory 接口，Spring 核心工厂是 BeanFactory ,BeanFactory 采取延迟加载，第一次 getBean 时才会初始化 Bean, ApplicationContext 是会在加载配置文件时初始化 Bean。

②ApplicationContext 是对 BeanFactory 扩展，它可以进行国际化处理、事件传递和 bean 自动装配以及各种不同应用层的 Context 实现

开发中基本都在使用 ApplicationContext, web 项目使用 WebApplicationContext，很少用到 BeanFactory

SpringMVC

1.简单的谈一下 SpringMVC 的工作流程？



流程

1、用户发送请求至前端控制器 DispatcherServlet

2、DispatcherServlet 收到请求调用 HandlerMapping 处理器映射器。

3、处理器映射器找到具体的处理器，生成处理器对象及处理器拦截器(如果有则生成)一并返回给 DispatcherServlet。

4、DispatcherServlet 调用 HandlerAdapter 处理器适配器

5、HandlerAdapter 经过适配调用具体的处理器(Controller，也叫后端控制器)。

6、Controller 执行完成返回 ModelAndView

7、HandlerAdapter 将 controller 执行结果 ModelAndView 返回给 DispatcherServlet
8、DispatcherServlet 将 ModelAndView 传给 ViewReslover 视图解析器
9、ViewReslover 解析后返回具体 View
10、DispatcherServlet 根据 View 进行渲染视图（即将模型数据填充至视图中）。
11、DispatcherServlet 响应用户

2.如何解决 POST 请求中文乱码问题，GET 的又如何处理呢？

在 web.xml 中加入：

```
<filter>
    <filter-name>CharacterEncodingFilter</filter-name>
    <filter-
class>org.springframework.web.filter.CharacterEncodingFilter</
filter-class>
    <init-param>
        <param-name>encoding</param-name>
        <param-value>utf-8</param-value>
    </init-param>
</filter>
<filter-mapping>
    <filter-name>CharacterEncodingFilter</filter-name>
    <url-pattern>/*</url-pattern>
</filter-mapping>
```

以上可以解决 post 请求乱码问题。对于 get 请求中文参数出现乱码解决方法有两个：

修改 tomcat 配置文件添加编码与工程编码一致，如下：

```
<Connector    URIEncoding="utf-8"    connectionTimeout="20000"
port="8080" protocol="HTTP/1.1" redirectPort="8443"/>
```

另外一种方法对参数进行重新编码：

```
String                userName                =                new
String(request.getParamter("userName").getBytes("ISO8859-
1"),"utf-8")
```

ISO8859-1 是 tomcat 默认编码，需要将 tomcat 编码后的内容按 utf-8 编码

3.问题：springmvc 常用注解有哪些？

@RequestMapping：是一个用来处理请求地址映射的注解，可用于类或方法上。用于类上，表示类中的所有响应请求的方法都是以该地址作为父路径。
@PathVariable：用于将请求 URL 中的模板变量映射到功能处理方法的参数上，即取出 uri 模板中的变量作为参数。
@requestParam：主要用于在 SpringMVC 后台控制层获取参数，类似一种是 request.getParameter("name"), 它有三个常用参数：defaultValue = "0", required = false, value = "isApp"；defaultValue 表示设置默认值，required 铜过 boolean 设置是否是必须要传入的参数，value 值表示接受的传入的参数类型。
@ResponseBody：该注解用于将 Controller 的方法返回的对象，通过适当的 HttpMessageConverter 转换为指定格式后，写入到 Response 对象的 body 数据区。使用时机：返回的数据不是 html 标签的页面，而是其他某种格式的数据时（如 json、xml 等）使用
@RequestBody：该注解常用来处理 Content-Type: 不是 application/x-www-form-urlencoded 编码的内容，例如 application/json, application/xml 等；
@RequestHeader ：可以把 Request 请求 header 部分的值绑定到方法的参数上。
@CookieValue ：可以把 Request header 中关于 cookie 的值绑定到方法的参数上。

4.问题：一个 bean 配置在 springmvc 的配置文件如 springmvc-servlet.xml 跟配置在 spring 全局配置文件 applicationContext 中有什么区别。

答：配置在 springmvc 配置文件中的 bean 属于子容器中内容。配置在全局配置文件中的 bean 属于父容器。子容器可以获取父容器中的内容，而父容器不可以获得子容器中的内容。

Mybatis

1.mybatis 比 IBatis 比较大的几个改进是什么？

a.有接口绑定,包括注解绑定 sql 和 xml 绑定 Sql，
b.动态 sql 由原来的节点配置变成 OGNL 表达式,
c. 在一对一,一对多的时候引进了 association,在一对多的时候引入了 collection 节点,不过都是在 resultMap 里面配置

2.接口绑定有几种实现方式,分别是怎么实现的?

a.一种是通过注解绑定,就是在接口的方法上面加上@Select@Update 等注解里面包含 Sql 语

句来绑定,
b,另外一种就是通过 xml 里面写 SQL 来绑定,在这种情况下,要指定 xml 映射文件里面的 namespace 必须为接口的全路径名.

3.什么情况下用注解绑定,什么情况下用 xml 绑定

当 Sql 语句比较简单时候,用注解绑定,
当 SQL 语句比较复杂时候,用 xml 绑定,一般用 xml 绑定的比较多

4.myBatis 实现一对一有几种方式?具体怎么操作的

有联合查询和嵌套查询
联合查询是几个表联合查询,只查询一次,通过在 resultMap 里面配置 association 节点配置一对一的类就可以完成;
嵌套查询是先查一个表,根据这个表里面的结果的外键 id,去再另外一个表里面查询数据,也是通过 association 配置,但另外一个表的查询通过 select 属性配置

5.myBatis 实现一对多有几种方式,怎么操作的

有联合查询和嵌套查询
联合查询是几个表联合查询,只查询一次,通过在 resultMap 里面配置 collection 节点配置一对多的类就可以完成;
嵌套查询是先查一个表,根据这个表里面的结果的外键 id,去再另外一个表里面查询数据,也是通过配置 collection,但另外一个表的查询通过 select 节点配置

6.myBatis 里面的动态 Sql 是怎么设定的?用什么语法?

MyBatis 里面的动态 Sql 一般是通过 if 节点来实现,通过 OGNL 语法来实现,但是如果要写的完整,必须配合 where,trim 节点,where 节点是判断包含节点有内容就插入 where,否则不插入,trim 节点是用来判断如果动态语句是以 and 或 or 开始,那么会自动把这个 and 或者 or 取掉

7.讲下 myBatis 的缓存

MyBatis 的缓存分为一级缓存和二级缓存,
一级缓存放在 session 里面,默认就有,二级缓存放在它的命名空间里,默认是打开的,
二级缓存属性类需要实现 Serializable 序列化接口(可用来保存对象的状态),可在它的映射文件中配置<cache/>

8.myBatis(IBatis)的好处是什么

ibatis 把 sql 语句从 Java 源程序中独立出来，放在单独的 XML 文件中编写，给程序的维护带来了很大便利。
ibatis 封装了底层 JDBC API 的调用细节，并能自动将结果集转换成 Java Bean 对象，大大简化了 Java 数据库编程的重复工作。
因为 Ibatis 需要程序员自己去编写 sql 语句，程序员可以结合数据库自身的特点灵活控制 sql 语句，因此能够实现比 hibernate 等全自动 orm 框架更高的查询效率，能够完成复杂查询。.

Struts2 和 SpringMVC 对比

1. **机制**：spring mvc 的入口是 servlet，而 struts2 是 filter（这里要指出，filter 和 servlet 是不同的。以前认为 filter 是 servlet 的一种特殊），这样就导致了二者的机制不同，这里就牵涉到 servlet 和 filter 的区别了。
2. **性能**：spring 会稍微比 struts 快。spring mvc 是基于方法的设计，而 sturts 是基于类，每次发一次请求都会实例一个 action，每个 action 都会被注入属性，而 spring 基于方法，粒度更细，但要小心把握像在 servlet 控制数据一样。spring3 mvc 是方法级别的拦截，拦截到方法后根据参数上的注解，把 request 数据注入进去，在 spring3 mvc 中，一个方法对应一个 request 上下文。而 struts2 框架是类级别的拦截，每次来了请求就创建一个 Action，然后调用 setter getter 方法把 request 中的数据注入；struts2 实际上是通过 setter getter 方法与 request 打交道的；struts2 中，一个 Action 对象对应一个 request 上下文。
3. **参数传递**：struts 是在接受参数的时候，可以用属性来接受参数，这就说明参数是让多个方法共享的。
4. **设计思想上**：struts 更加符合 oop 的编程思想， spring 就比较谨慎，在 servlet 上扩展。

Hibernate 和 Mybatis 对比

1.开发对比开发速度

Hibernate 的真正掌握要比 Mybatis 来得难些。Mybatis 框架相对简单很容易上手，但也相对简陋些。个人觉得要用好 Mybatis 还是首先要先理解好 Hibernate。
开发社区
Hibernate 与 Mybatis 都是流行的持久层开发框架，但 Hibernate 开发社区相对多热闹些，支持的工具也多，更新也快，当前最高版本 4.1.8。而 Mybatis 相对平静，工具较少，当前最高版本 3.2。
开发工作量

Hibernate 和 MyBatis 都有相应的代码生成工具。可以生成简单基本的 DAO 层方法。
针对高级查询，Mybatis 需要手动编写 SQL 语句，以及 ResultMap。而 Hibernate 有良好的映射机制，开发者无需关心 SQL 的生成与结果映射，可以更专注于业务流程。

2 对象管理与抓取策略对象管理

Hibernate 是完整的对象/关系映射解决方案，它提供了对象状态管理（state management）的功能，使开发者不再需要理会底层数据库系统的细节。也就是说，相对于常见的 JDBC/SQL 持久层方案中需要管理 SQL 语句，Hibernate 采用了更自然的面向对象的视角来持久化 Java 应用中的数据。
换句话说，使用 Hibernate 的开发者应该总是关注对象的状态（state），不必考虑 SQL 语句的执行。这部分细节已经由 Hibernate 掌管妥当，只有开发者在进行系统性能调优的时候才需要进行了解。
而 MyBatis 在这一块没有文档说明，用户需要对对象自己进行详细的管理。
抓取策略
Hibernate 对实体关联对象的抓取有着良好的机制。对于每一个关联关系都可以详细地设置是否延迟加载，并且提供关联抓取、查询抓取、子查询抓取、批量抓取四种模式。 它是详细配置和处理的。
而 Mybatis 的延迟加载是全局配置的。

3 缓存机制对比 Hibernate 缓存

Hibernate 一级缓存是 Session 缓存，利用好一级缓存就需要对 Session 的生命周期进行管理好。建议在一个 Action 操作中使用一个 Session。一级缓存需要对 Session 进行严格管理。
Hibernate 二级缓存是 SessionFactory 级的缓存。 SessionFactory 的缓存分为内置缓存和外置缓存。内置缓存中存放的是 SessionFactory 对象的一些集合属性包含的数据(映射元素据及预定 SQL 语句等),对于应用程序来说,它是只读的。外置缓存中存放的是数据库数据的副本,其作用和一级缓存类似.二级缓存除了以内存作为存储介质外,还可以选用硬盘等外部存储设备。二级缓存称为进程级缓存或 SessionFactory 级缓存，它可以被所有 session 共享，它的生命周期伴随着 SessionFactory 的生命周期存在和消亡。

4 优势对比

Mybatis 优势
MyBatis 可以进行更为细致的 SQL 优化，可以减少查询字段。
MyBatis 容易掌握，而 Hibernate 门槛较高。
Hibernate 优势
Hibernate 的 DAO 层开发比 MyBatis 简单，Mybatis 需要维护 SQL 和结果映射。
Hibernate 对对象的维护和缓存要比 MyBatis 好，对增删改查的对象的维护要方便。
Hibernate 数据库移植性很好，MyBatis 的数据库移植性不好，不同的数据库需要写不同 SQL。
Hibernate 有更好的二级缓存机制，可以使用第三方缓存。MyBatis 本身提供的缓存机制不

佳。

面试题数据库(附件)

1.用一条 SQL 语句查询出每门课都大于 80 分的学生姓名

name	kecheng	fenshu
张三	语文	81
张三	数学	75
李四	语文	76
李四	数学	90
王五	语文	81
王五	数学	100
王五	英语	90

准备数据的 sql 代码：

```
create table score(id int primary key auto_increment,namevarchar(20),subject
varchar(20),score int);
insert into score values
(null,'张三','语文',81),
(null,'张三','数学',75),
(null,'李四','语文',76),
(null,'李四','数学',90),
(null,'王五','语文',81),
```

(null,'王五','数学',100),
(null,'王五 ','英语',90);
提示：当百思不得其解时，请理想思维，把小变成大做，把大变成小做，

答案： A: select distinct name from score where name not in (selectdistinct name from score where score<=80)
B:select distince name t1 from score where 80< all (selectscore from score where name=t1);

2.所有部门之间的比赛组合

一个叫 department 的表，里面只有一个字段 name,一共有 4 条纪录，分别是 a,b,c,d,对应四个球对，现在四个球对进行比赛，用一条 sql 语句显示所有可能的比赛组合.

答：select a.name,b.name from team a, team b where a.name < b.name

3.显示文章标题，发帖人、最后回复时间

表：id,title,postuser,postdate,parentid

准备 sql 语句：
drop table if exists articles;

create table articles(id int auto_increment primary key,titlevarchar(50), postuser varchar(10), postdate datetime,parentid int referencesarticles(id));

insert into articles values
(null,'第一条','张三','1998-10-10 12:32:32',null),
(null,'第二条','张三','1998-10-10 12:34:32',null),
(null,'第一条回复 1','李四','1998-10-10 12:35:32',1),
(null,'第二条回复 1','李四','1998-10-10 12:36:32',2),
(null,'第一条回复 2','王五','1998-10-10 12:37:32',1),

(null,'第一条回复 3','李四','1998-10-10 12:38:32',1),
(null,'第二条回复 2','李四','1998-10-10 12:39:32',2),
(null,'第一条回复 4','王五','1998-10-10 12:39:40',1);

答案：
select a.title,a.postuser,
(selectmax(postdate) from articles where parentid=a.id) reply
from articles a where a.parentid is null;
注释：子查询可以用在选择列中，也可用于 where 的比较条件中，还可以用于 from 从句中。

4.删除除了 id 号不同,其他都相同的学生冗余信息

学生表如下:

2.学生表如下:

id 号	学号	姓名	课程编号	课程名称	分数
1	2005001	张三	0001	数学	69
2	2005002	李四	0001	数学	89
3	2005001	张三	0001	数学	69

A: delete from tablename where id 号 not in(select min(id 号) from tablename group by 学号,姓名,课程编号,课程名称,分数)

实验：

```
create table student2(id int auto_increment primary key,codevarchar(20),name
varchar(20));
insert into student2 values(null,'2005001','张三'),(null,'2005002','李四'),(null,'2005001','张
三');
//如下语句，mysql 报告错误，可能删除依赖后面统计语句，而删除又导致统计语句结果不
一致。
delete from student2 where id not in(select min(id) fromstudent2 group by name);
//但是，如下语句没有问题：
select * from student2where id not in(select min(id) from student2 group by name);
//于是，我想先把分组的结果做成虚表，然后从虚表中选出结果，最后再将结果作为删除
的条件数据。
delete from student2 where id not in(select mid from (selectmin(id) mid
from student2 group by name) as t);
```

或者：

```
delete from student2 where id not in(select min(id) from (select* from s
tudent2) as t group by t.name);
```

5.航空网的几个航班查询题：

表结构如下：

```
flight{flightID,StartCityID ,endCityID,StartTime}
city{cityID, CityName)
```

实验环境：

```
create table city(cityID int auto_increment primary key,cityNamevarchar(20));
create table flight (flightID int auto_increment primary key,
        StartCityID intreferences city(cityID),
        endCityID   int references city(cityID),
        StartTimetimestamp);
//航班本来应该没有日期部分才好，但是下面的题目当中涉及到了日期
insert into city values(null,'北京'),(null,'上海'),(null,'广州');
```

```
insert into flight values
        (null,1,2,'9:37:23'),(null,1,3,'9:37:23'),(null,1,2,'10:37:23'),(null,2,3,'10:37:23');
```

1、查询起飞城市是北京的所有航班，按到达城市的名字排序

参与运算的列是我起码能够显示出来的那些列，但最终我不一定把它们显示出来。各个表组合出来的中间结果字段中必须包含所有运算的字段。

```
select * from flight f,city c
where f.endcityid =c.cityid and startcityid =
      (select c1.cityidfrom city c1 where c1.cityname = "北京")
order by c.citynameasc;

select flight.flightid,'北京' startcity, e.cityname
from
 flight,city e
where flight.endcityid=e.cityid and flight.startcityid=(
  selectcityid from city where cityname='北京');

select flight.flightid,s.cityname,e.cityname fromflight,city s,city e wh
ere flight.startcityid=s.cityid and s.cityname='北京' andflight.endCityId=e.cit
yID order by e.cityName desc;
```

2、查询北京到上海的所有航班纪录（起飞城市，到达城市，起飞时间，航班号）

```
select c1.CityName,c2.CityName,f.StartTime,f.flightID
from city c1,city c2,flight f
where f.StartCityID=c1.cityID
and f.endCityID=c2.cityID
and c1.cityName='北京'
and c2.cityName='上海'
```

3、查询具体某一天（2005-5-8）的北京到上海的的航班次数

```
select count(*) from
(select c1.CityName,c2.CityName,f.StartTime,f.flightID
from city c1,city c2,flight f
where f.StartCityID=c1.cityID
and f.endCityID=c2.cityID
and c1.cityName='北京'
and c2.cityName='上海'
and  查帮助获得的某个日期处理函数(startTime) like '2005-5-8%'
mysql 中提取日期部分进行比较的示例代码如下：
select * from flight wheredate_format(starttime,'%Y-%m-%d')='1998-01-02'
```