面试题：

1. **实现多态的机制**

* 编译时多态（方法重载）
* 运行时多态（方法重写）

运行时多态一般有两种实现方法

* 子类继承父类（extends）
* 类实现接口（implements）

1. **Int, Integer区别**

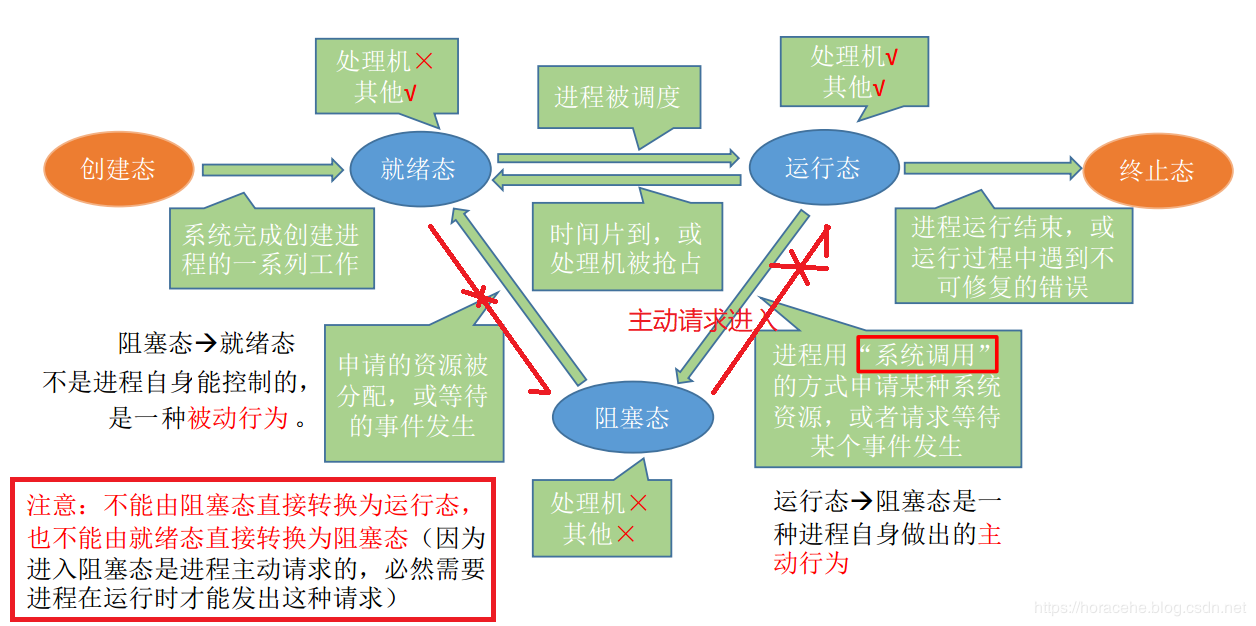
* Integer是int的包装类，int则是java的一种基本数据类型
* Integer变量必须实例化后才能使用（-128-127除外），而int变量不需要
* Integer实际是对象的引用，当new一个Integer时，实际上是生成一个指针指向此对象；而int则是直接存储数据值
* Integer的默认值是null，int的默认值是0

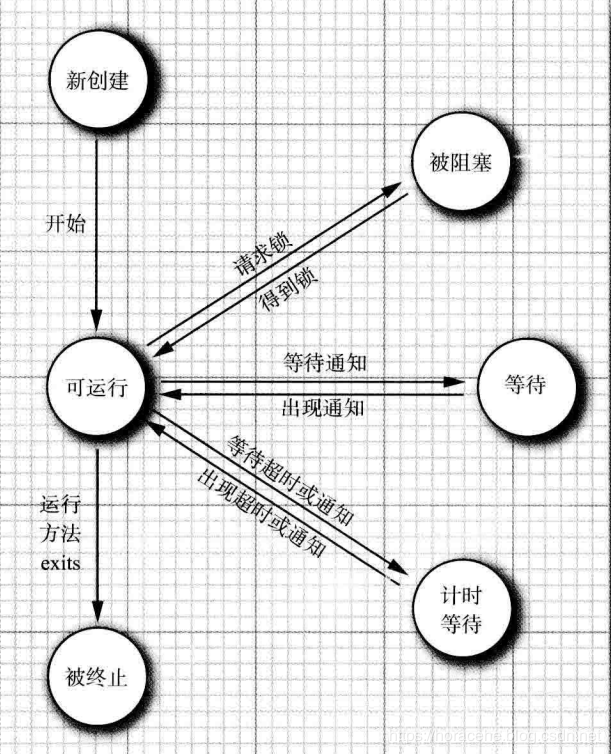
1. **New 一个子类继承父类过程**
2. 父类静态代码块
3. 子类静态代码块
4. 父类非静态(代码块)
5. 父类构造方法
6. 子类非静态(代码块)
7. 子类构造方法
8. **集合体系隐藏接口**



1. **线程状态**

进程的5种状态：创建态、就绪态、运行态、阻塞态、终止态

****

线程可有6种状态：新创建、可运行、被阻塞、等待、计时等待、被终止。****

1. **网络模型有哪几层**

答：这7层是:[物理层](https://link.zhihu.com/?target=http://www.so.com/s?q=%E7%89%A9%E7%90%86%E5%B1%82&ie=utf-8&src=internal_wenda_recommend_textn" \t "https://zhuanlan.zhihu.com/p/_blank)、[数据链路层](https://link.zhihu.com/?target=http://www.so.com/s?q=%E6%95%B0%E6%8D%AE%E9%93%BE%E8%B7%AF%E5%B1%82&ie=utf-8&src=internal_wenda_recommend_textn" \t "https://zhuanlan.zhihu.com/p/_blank)、[网络层](https://link.zhihu.com/?target=http://www.so.com/s?q=%E7%BD%91%E7%BB%9C%E5%B1%82&ie=utf-8&src=internal_wenda_recommend_textn" \t "https://zhuanlan.zhihu.com/p/_blank)、[传输层](https://link.zhihu.com/?target=http://www.so.com/s?q=%E4%BC%A0%E8%BE%93%E5%B1%82&ie=utf-8&src=internal_wenda_recommend_textn" \t "https://zhuanlan.zhihu.com/p/_blank)、[会话层](https://link.zhihu.com/?target=http://www.so.com/s?q=%E4%BC%9A%E8%AF%9D%E5%B1%82&ie=utf-8&src=internal_wenda_recommend_textn" \t "https://zhuanlan.zhihu.com/p/_blank)、[表示层](https://link.zhihu.com/?target=http://www.so.com/s?q=%E8%A1%A8%E7%A4%BA%E5%B1%82&ie=utf-8&src=internal_wenda_recommend_textn" \t "https://zhuanlan.zhihu.com/p/_blank)和[应用层](https://link.zhihu.com/?target=http://www.so.com/s?q=%E5%BA%94%E7%94%A8%E5%B1%82&ie=utf-8&src=internal_wenda_recommend_textn" \t "https://zhuanlan.zhihu.com/p/_blank)。

1. **collection,collections区别**

Collection是集合体系的最顶层，包含了集合体系的共性

Collections是一个工具类，方法都是用于操作Collection

1. **Jsp九大内置对象**



1. **Cookies和session区别**

* Cookie和Session都是会话技术，Cookie是运行在客户端，Session是运行在服务器端。
* Cookie有大小限制以及浏览器在存cookie的个数也有限制，Session是没有大小限制和服务器的内存大小有关
* Cookie有安全隐患，通过拦截或本地文件找得到你的cookie后可以进行攻击。
* Session是保存在服务器端上会存在一段时间才会消失，如果session过多会增加服务器的压力。

1. **转发，重定向区别**

* 重定向请求两次，转发请求一次；
* 地址栏不同：重定向地址栏会发生变化，转发地址栏不会发生变化；
* 是否共享数据：重定向两次请求不共享数据，转发一次请求共享数据（在request级别使用信息共享，使用重定向必然出错）；
* 跳转限制：重定向可以跳转到任意URL，转发只能跳转本站点资源；
* 发生行为不同：重定向是客户端行为，转发是服务器端行为；

1. **Hashset和 treeset区别**

* 不同点1：底层使用的储存数据结构不同
* Hashset底层使用的是HashMap哈希表结构储存
* Treeset底层用的是TreeMap树(红黑树)结构储存
* 不同点2：储存的数据保存唯一方式不同
* Hashset是通过复写hashCode()方法和equals()方法来保证
* Treeset是通过Compareable接口的compareto方法来保证
* 不同点3： hashset无序   Treeset有序(通过指定比较器)
* 不同点4： HashSet可以存储null值，而TreeSet不可以

1. **什么是反射，反射应用于那些场景**

答：Java 的反射机制是指在运行状态中，对于任意一个类都能够知道这个类所有的属性和方法； 并且对于任意一个对象，都能够调用它的任意一个方法；这种动态获取信息以及动态调用对象方法的功能称为Java语言的反射机制

获取 Class 类对象三种方式：

* 使用 Class.forName 静态方法
* 使用类的.class 方法
* 使用实例对象的 getClass() 方法

1. **请你说一下Error和Exception的区别，Exception有哪些异常（分大类说），有哪些是运行时异常，IO异常是不是运行时异常，找不到类的异常是属于什么异常**

答：Exception和Error都是继承了Throwable类

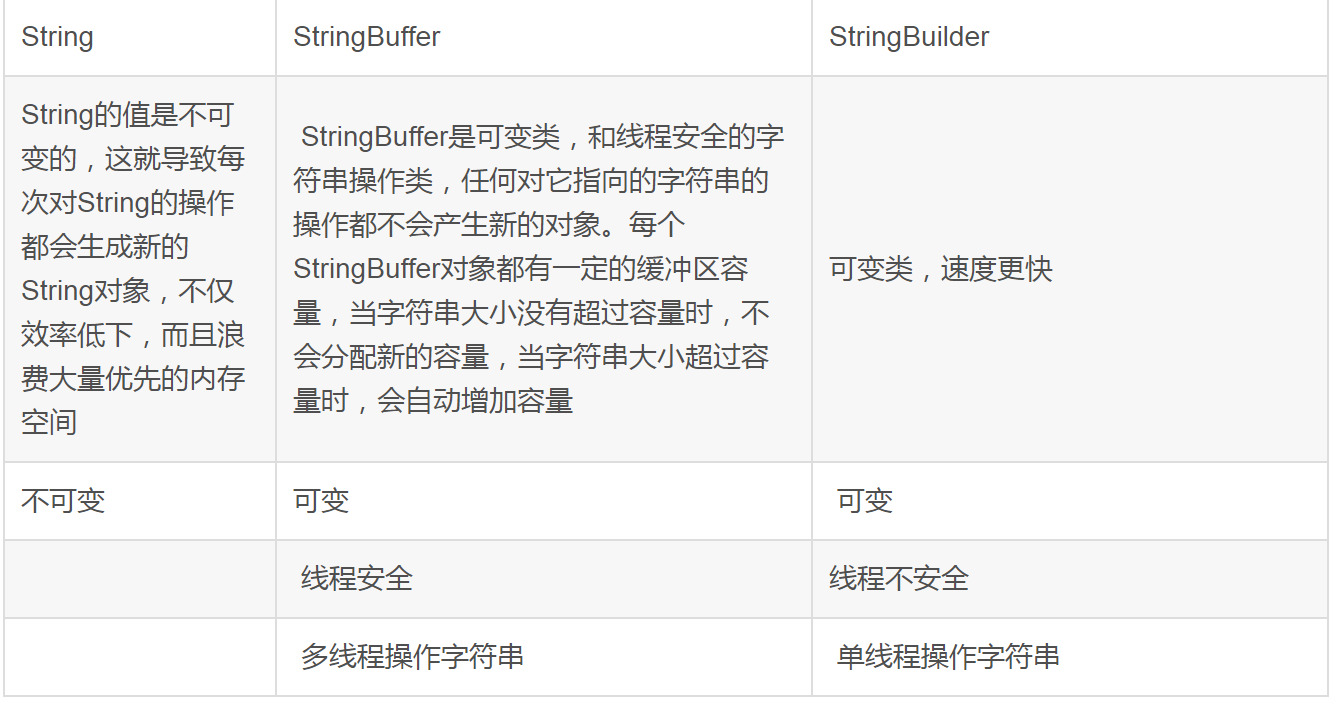
* Error是指系统级别的错误（一般是处理不了的）
* Exception是指应用程序级别的异常（是程序员能通过修改代码解决的）

Exception分为

* RuntimeException（运行时异常）
* CheckedException（受检异常）

IO异常是编译期异常(受检异常)，找不到类是运行时异常

1. **String,StringBuffer,StringBuilder的区别**

****

String s = "s";

String str = new String("str");

String strNew = str + s;

常量池 3 1 1 1

堆内存 2 1 1

1. **请你说一下List，Set，Map三个集合的区别**

* List和Set是存储单列数据的集合
* Map是存储键值对的双列数据的集合
* List集合 存储数据是有序的，允许重复
* Map集合存储数据是无序的，Map集合的Key是不能重复的，Value可以重复
* Set集合存储数据是无顺序的但（JDK1.8版本之后好像有序），不允许重复

1. **请你说一下ArrayList和Vector的区别**

* ArrayList和Vector底层都是数组，大部分功能一样
* ArrayList线程不安全
* Vector线程安全

1. **什么是Java的序列化，怎么去实现序列化，什么是持久化。**

在一定的介质内进行同化 implements Serializable 只是为了标注该对象是可被序列化的。

1. **请你列举抽象类和接口的区别（说它们的特征）**

* 抽象类代表的是“is-a”的关系，接口所代表的是"like-a"的关系(单继承多实现)
* 接口实际上是一个特殊的抽象类
* 抽象类可以有抽象方法，也可以有实例方法,接口中所有的方法都是公开抽象方法
* Jdk1.8之后，接口中默认方法和静态方法可以有方法体，1.9之后接口允许有私有方法
* 类使用抽象类，用extends（继承）；类使用接口，用implements(实现)
* 具体类实现接口，必须重写接口中的所有方法
* 抽象类可以有自己的构造方法，接口不能有构造方法

1. **throw和throws的区别。**

* Throw：作用在方法内，表示抛出具体异常，由方法体内的语句处理。具体向外抛出的动作，所以它抛出的是一个异常实体类。若执行了Throw一定是抛出了某种异常。
* Throws：作用在方法的声明上，表示如果抛出异常，则由该方法的调用者来进行异常处理。主要的声明这个方法会抛出某种类型的异常，让它的使用者知道捕获异常的类型。

1. **请你说一下线程跟进程的区别**

**进程：**资源分配的最小单位

**线程：**程序执行的最小单位

**区别：**

* 地址空间： 同一进程的所有线程共享本进程的地址空间，而不同的进程之间的地址空间是独立的。
* 资源拥有： 同一进程的所有线程共享本进程的资源，如内存，CPU，IO等。进程之间的资源是独立的，无法共享。
* 执行过程：每一个进程可以说就是一个可执行的应用程序，每一个独立的进程都有一个程序执行的入口，顺序执行序列。但是线程不能够独立执行，必须依存在应用程序中，由程序的多线程控制机制进行控制。
* 健壮性： 因为同一进程的所以线程共享此线程的资源，因此当一个线程发生崩溃时，此进程也会发生崩溃。 但是各个进程之间的资源是独立的，因此当一个进程崩溃时，不会影响其他进程。因此进程比线程健壮。
* 系统开销：线程执行开销小，但不利于资源的管理与保护；进程的执行开销大，但可以进行资源的管理与保护。进程可以跨机器前移。

1. **JavaIO体系顶层有哪些接口或类,字节流和字符流之间怎么转换，有哪些转换流**

InputStream、OutputStream、Reader、Writer

转换流:InputStreamReader、OutputStreamWriter

1. **如何避免SQL的非法注入**

PreparedStatement(占位符)

1. **请你说一下SpringMVC有哪些组件**

* DispatcherServlet 前端控制器 请求入口
* HandlerMapping　处理器映射器
* HandlerAdapter 处理器适配器
* Controller　　　　 处理器
* ModelAndView　　封装模型信息和视图信息
* ViewResolver　　　视图解析器

1. **SpringMVC是什么以及它的运作流程**

* Spring MVC是一个基于Java的实现了MVC设计模式的请求驱动类型的轻量级Web框架，通过把Model，View，Controller分离，将web层进行职责解耦，把复杂的web应用分成逻辑清晰的几部分，简化开发，减少出错，方便组内开发人员之间的配合
* 运作流程如下：
* 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(可以根据xml配置、注解进行查找)，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ViewResolver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

1. **你对spring框架的理解，什么叫控制反转，什么叫依赖注入**

* Spring框架是一个轻量级一站式的开源性框架(非侵入式)，它能降低应用程序开发的复杂度。
* Spring核心是IOC和AOP，IOC是控制反转，AOP是面向切面编程（AOP就是通过[预编译](https://baike.baidu.com/item/%E9%A2%84%E7%BC%96%E8%AF%91/3191547" \t "https://baike.baidu.com/item/AOP/_blank)方式和运行期间动态代理实现程序功能的统一维护的一种技术——拦截器，底层实现原理是动态代理）
* 控制反转是思想(**容器托管)**
* 依赖注入DI是在编译阶段尚未知所需的功能是来自哪个的类的情况下，将其他对象所依赖的功能对象实例化的模式，它也是控制反转的基础
* 依赖注入有三种方式：setter注入、构造器注入和接口注入(Spring不支持)
* Spring框架能降低程序的耦合性，同时它能融合许多流行框架，如持久层框架Hibernate和Mybatis，还有MVC框架 Struts2等(万能胶水)

1. **HashMap和HashTable的区别，哪个的性能更好**

* HashMap可以存储null值，但是线程不安全
* HashTable不可以存储null值，线程安全

1. **静态变量和实例变量的区别**

**区别：**

* 生命周期不同
* 实例变量随着对象的创建而存在，随着对象的回收而释放
* 静态变量随着类的加载而存在，随着类的消失而消失
* 存储位置不同
* 实例变量存储在堆内存的对象中，所以也叫对象的特有数据
* 静态变量数据存储在方法区（共享数据区）的静态区，所以也叫对象的共享数据
* 调用的方式也不同(静态变量属于类，实例变量属于对象)
* 静态变量：类名.xx
* 实例变量: 对象.xx
* 修饰(命名)方式不同
* 静态变量前要加上static关键字，而实例变量前则不加

**28.Mybatis的SQL映射中#和$的区别**

* #相当与PreparedStatement，占位符，能防止sql注入
* $相当于Statement，相对于PreparedStatement来说不安全

**29.对比一下Mybatis和Hibernate框架，它们的主要区别是什么？**

* Hibernate是完整的ORM(对象关系映射)框架(全自动)，映射的是完整的对象关系模型，自动加载关联，它不用自己写sql语句，修改比较困难
* Mybatis不是完整的ORM框架(半自动)，它映射的是sql语句，由于自己编写sql语句所以修改优化起来也比较简单
* 相比于Mybatis，Hibernate有更好的二级缓存机制

1. Spring支持的有哪几种bean的作用域

五种作用域

* singleton 单例 --默认
* propertype 原型(多例)
* request
* session
* global-session

1. 列举一下SpringMVC的常用注解

* @Controller
* @RequestMapping
* @ResponseBody
* @PathVariable (Restful风格)
* @RequestParam
* @ModelAttribute （数据回显）

1. **Socket编程有哪些,TCP和UDP有什么区别，什么叫UDP**

TCP和UDP都是网络7层架构中传输层的通信协议

* TCP--三次握手（基于连接） UDP--传输不用建立在连接

三次握手的原理图：



* 流模式（TCP） 数据报模式（UDP）；
* TCP保证数据正确性，UDP可能丢包，TCP保证数据顺序，UDP不保证
* 对系统资源的要求（TCP较多，UDP少）

1. **栈（stack）和堆（heap）的区别**

* 各司其职
* 最主要的区别就是栈内存用来存储局部变量和方法调用。
* 堆内存用来存储Java中的对象。无论是成员变量，局部变量，还是类变量，它们指向的对象都存储在堆内存中
* 独有还是共享
* 栈内存归属于单个线程，每个线程都会有一个栈内存，其存储的变量只能在其所属线程中可见，即栈内存可以理解成线程的私有内存
* 堆内存中的对象对所有线程可见，堆内存中的对象可以被所有线程访问
* 异常错误
* 如果栈内存没有可用的空间存储方法调用和局部变量，JVM会抛出java.lang.StackOverFlowError
* 而如果是堆内存没有可用的空间存储生成的对象，JVM会抛出java.lang.OutOfMemoryError
* 空间大小

栈的内存要远远小于堆内存，如果你使用递归的话，那么你的栈很快就会充满。如果递归没有及时跳出，很可能发生StackOverFlowError错误

1. **什么叫ORM框架**

* ORM即通过类与数据库表的映射关系，将对象持久化到数据库中，映射如
* 数据库中的表和Java中的实体类相映射
* 数据库中的表记录和Java中的对象相映射
* 数据库中的表字段和Java实体类中的属性相映射
* ORM框架就是对象关系映射框架，如Hibernate、Mybatis

1. **说一下Mybatis的缓存，解释一下它的缓存**

Mybatis缓存分为一级缓存和二级缓存

* 一级缓存是SqlSession级别的，默认就是一级缓存
* 二级缓存是mappper级别的，多个SqlSession共享，要去配置文件中开启(cacheEnabled)

1. **Spring框架怎么管理事务，用到什么技术**

Spring支持编程式事务管理和声明式事务管理两种方式，Spring事务管理的核心接口是PlatformTransactionManager

* 编程式事务管理使用TransactionTemplate或者直接使用底层的PlatformTransactionManager。对于编程式事务管理，spring推荐使用TransactionTemplate
* 声明式事务管理建立在AOP面向切面编程的技术之上的,声明式事务管理也有两种常用的方式
* 一种是基于tx和aop名字空间的xml配置文件
* 另一种就是基于@Transactional注解
* 事务一般是用于对项目业务层的管理

1. **在Spring中怎么去注入Java集合**

Spring提供了四种类型的如下集合的配置元素：



1. **Spring自动装配的方式有哪些，什么叫ByName，什么叫ByType**

* 方式有byName和byType
* byName：根据属性名自动装配。此选项将检查容器并根据名字查找与属性完全一致的bean，并将其与属性自动装配
* byType：根据类型自动装配。如果容器中存在一个与指定属性类型相同的bean，那么将与该属性自动装配；如果存在多个该类型bean，那么抛出异常，并指出不能使用byType方式进行自动装配；如果没有找到相匹配的bean，则什么事都不发生

1. **什么是事务，事务的特性，什么叫ACID**

事务：逻辑单元内的一系列操作，要么全部完成执行，要么全部不执行。

* Atomicity（原子性）
  + 原子性很容易理解，也就是说事务里的所有操作要么全部做完，要么都不做，事务成功的条件是事务里的所有操作都成功，只要有一个操作失败，整个事务就失败，需要回滚
* Consistency（一致性）
  + 一致性也比较容易理解，也就是说数据库要一直处于一致的状态，事务开始前是一个一致状态，事务结束后是另一个一致状态，事务将数据库从一个一致状态转移到另一个一致状态
* Isolation（独立性/隔离性）
  + 从字面上来说，独立性是其中最难理解的一点，但如果结合Oracle中的undo，也就不难理解了。所谓的独立性就是指并发的事务之间不会互相影响，如果一个事务要访问的数据正在被另外一个事务修改，只要另外一个事务还未提交，它所访问的数据就不受未提交事务的影响。换句话说，一个事务的影响在该事务提交前对其它事务是不可见的
* Durability(持久性)
  + 持久性也不难理解，是指一旦事务提交后，它所做的修改将会永久的保存在数据库上，即使出现宕机也不会丢失

1. **做项目中怎么去处理事务问题**

* 数据库级别的事务,数据库管理员处理的是数据库级别的事务
* 应用程序级别的事务,Java程序员处理的是应用程序级别的事务（一般应用在业务层上）

1. **Spring事务管理的传播途径**

Spring定义了七种传播行为：

****

1. **Hibernate 查询方式有哪些**

* QID检索
* Query(hql检索)
* QBC(criteria)
* 离线检索DetachedCriteria
* 本地sql检索

1. **spring常用注解**

* @Service
* @Component
* @Autowired
* @ComponentScan
* @Configuration
* @Transactional
* @EnableWebMvc
* @EnableTransactionManagement

1. **Spring怎么管理事务，用到什么原理**

* 事务管理可以帮助我们保证数据的一致性，对应企业的实际应用很重要
* Spring的事务机制包括(xml配置或注解配置)声明式事务和编程式事务
* 编程式事务管理：Spring推荐使用TransactionTemplate，实际开发中使用声明式事务较多
* 声明式事务管理：将我们从复杂的事务处理中解脱出来，获取连接，关闭连接、事务提交、回滚、异常处理等这些操作都不用我们处理了，Spring都会帮我们处理
* 声明式事务管理使用了AOP面向切面编程实现的，本质就是在目标方法执行前后进行拦截。在目标方法执行前加入或创建一个事务，在执行方法执行后，根据实际情况选择提交或是回滚事务
* Spring事务管理主要包括3个接口，Spring的事务主要是由他们三个共同完成的
* PlatformTransactionManager：事务管理器--主要用于平台相关事务的管理
* TransactionDefinition：事务定义信息--用来定义事务相关的属性，给事务管理器PlatformTransactionManager使用
* TransactionStatus：事务具体运行状态--事务管理过程中，每个时间点事务的状态信息

1. **请你说一下shiro框架的作用，它怎么实现权限控制**

* **Apache的强大灵活的开源安全框架**
* **作用：认证、授权、会话管理、安全加密**
* **shiro通过Security Manager(核心)提供安全服务**
* **实现权限控制四种方式：**
* URL拦截方式：在spring配置文件中配置过滤器链：配置项目请求对应拦截规则
* 注解方式：1、开启注解扫描  2、service方法使用shiro注解
* 页面标签实现权限控制
* 代码级别权限控制

1. **JDK和JRE的区别**

* JDK中包含JRE
* JDK：开发环境
* JRE：运行环境

1. **请你说一下Ajax应用和传统web应用有什么不同**

* 传统的页面更新数据一般是请求转发或者重定向即刷新整个页面
* Ajax是异步发送请求，客户端动态加载服务器返回的更新数据，服务器响应客户端请求的只是更新的数据，而不是整个页面（局部刷新），它与传统技术相比提高了网页运行效率，减轻了服务器的负担

1. **MySQL和Oracle分页有什么区别**

* MySQL 分页 (利用LIMIT关键字） 计算参数为 开始序号(startNum)，要查的总条数 (totalNum)
* Oracle 分页 (利用自带的rownum) 计算参数为 开始序号(startNum) ， 结束序号 (endNum)

1. **你了解的spring框架用到了哪些设计模式**

* 工厂设计模式 : Spring使用工厂模式通过 BeanFactory、ApplicationContext 创建 bean 对象
* 代理设计模式 : Spring AOP 功能的实现。 单例设计模式 : Spring 中的 Bean 默认都是单例的
* 模板方法模式 :Spring 中 jdbcTemplate、hibernateTemplate 等以 Template结尾的对数据库操作的类，它们就使用到了模板模式
* 包装器设计模式 :我们的项目需要连接多个数据库，而且不同的客户在每次访问中根据需要会去访问不同的数据库。这种模式让我们可以根据客户的需求能够动态切换不同的数据源
* 观察者模式: Spring 事件驱动模型就是观察者模式很经典的一个应用
* 适配器模式 :Spring AOP 的增强或通知(Advice)使用到了适配器模式、spring MVC 中也是用到了适配器模式适配Controller

1. **Http响应头、Http请求头信息**

* Request Headers
* Accept
* Accept-Encoding
* Accept-Language
* Cookie
* Host
* User-Agent
* Response Header
* Content-Encoding
* Content-Language
* Content-Length
* Content-Location
* Content-MD5
* Date
* Accept-Ranges
* Age
* Refresh
* Allow

1. **Heap(堆)和Stack(栈)的区别**

* 栈是用来存放基本类型的变量和引用类型的变量，堆用来存放new出来的对象和数组
* 栈的存取速度快，但不灵活。堆的存取速度慢，但是存取灵活，空间动态分配
* 栈在建立在连续的物理位置上，而堆只需要逻辑位置连续即可
* 堆是JVM运行时最大的内存区域

1. **请你说一下Spring、Spring MVC、Spring boot的区别**

* SpringMVC是Spring框架的一个模块，是开源的MVC框架
* Spring Boot是一个全新的搭建JavaEE应用程序的高级框架，Spring Boot 延续了Spring框架的核心思想IOC和AOP，简化了应用的开发和部署，主要用于开发微服务后台应用,它和传统的JavaEE应用的对比：
* 配置方式
  + Spring Boot

1. 去除繁琐的xml配置，仅保留少数配置
2. 以Java的方式配置
3. 自动配置

* 传统应用

1. 大量的xml配置，配置方式繁琐

* Web容器
  + Spring Boot

每一个springboot应用都有一个内嵌的web容器

* 打包方式
* Spring Boot ：jar包
* 传统 ：war包

1. **说一下Hibernate的缓存机制**

* 一级缓存**（session的缓存）：默认是开启的**
* 二级缓存**（sessionFactory的缓存):默认不会开启（存在可配置插件，需要进行配置）**

1. **请你说一下activiti和shiro框架有什么作用**

* **Activiti 是业务流程管理框架，它能处理多人共同参与的复杂业务，多用于OA审批系统.**
* **Shiro作用参照 题45.**

1. **请你对比一下oop和aop编程**

* **OOP：面向对象编程  
   核心思想：将客观存在的不同事物抽象成相互独立的类，然后把与事物相关的属性和行为封装到类里，并通过继承与多态来定义类彼此之间的关系，通过操作类的实例，来完成业务逻辑的功能需求**
* **AOP：面向切面的编程,是OOP的延续和补充  
   核心思想：将业务逻辑中与类不相关的通用功能切面式的提取分离出来，让多个类共享一个行为，一旦这个行为发生改变，不必修改类，而只需要修改这个行为即可**
* **区别：**
* **面向目标：OOP面向名词领域，AOP面向动词领域**
* **思想结构：OOP是纵向结构，AOP是横向结构**
* **注重方面：OOP注重业务逻辑单元的划分，AOP偏重业务处理过程的某个步骤或阶段**

1. **Spring 的通知是什么？有哪几种类型？**

* **前置通知[Before advice]：在连接点前面执行，前置通知不会影响连接点的执行，除非此处抛出异常**
* **后置通知[After returning advice]：在连接点正常执行完成后执行，如果连接点抛出异常，则不会执行**
* **异常通知[After throwing advice]：在连接点抛出异常后执行。**
* **返回(最终)通知[After (finally) advice]：在连接点执行完成后执行，不管是正常执行完成，还是抛出异常，都会执行返回通知中的内容**
* **环绕通知[Around advice]：环绕通知围绕在连接点前后，比如一个方法调用的前后。这是最强大的通知类型，能在方法调用前后自定义一些操作**