## 一、基础篇：

1、HashMap和HashTable区别？

（1）hashMap线程不安全，HashTable线程安全

（2）HashMap只允许一条记录的key为null，多条记录的value为null

HashTable不允许key和value插入null

2、ArrayList和LinkedList区别

（1）ArrayList是数组结构，linkList是链表结构

（2）ArrayList适合查找和遍历，LinkList适合动态插入和删除

## 二、框架篇

Spring篇

1、spring ioc、aop原理

aop是通过动态代理实现，如果目标类是个接口或者是个代理类，则使用jdk动态代理，否则使用cglib动态代理

2、mybatis的优势

1. 易于上手和掌握。  
2. sql写在xml里，便于统一管理和优化。  
3. 解除sql与程序代码的耦合。  
4. 提供映射标签，支持对象与数据库的orm字段关系映射  
5. 提供对象关系映射标签，支持对象关系组建维护  
6. 提供xml标签，支持编写动态sql。

3、spring设计模式

1.简单工厂模式

2.工厂方法模式

3.单例模式

4.适配器模式

5.包装器模式

6.代理模式

7.观察者模式

8.策略模式

9.模板方法模式

4、spring如何解决循环依赖

spring的单例对象初始化主要分为调用对象的实例化、填充属性、初始化。循环依赖主要发生在第一、第二步，也就是构造器循环依赖和field循环依赖。

spring为了解决单例的循环依赖问题，使用了三级缓存，这三个缓存实质上是3个map，分别是singletonFactories单例对象工厂的cache、earlySingletonObjects提前曝光的单例对象cache和singletonObjects单例对象的cache。Spring首先从一级缓存singletonObjects中获取。如果获取不到，并且对象正在创建中，就再从二级缓存earlySingletonObjects中获取。如果还是获取不到且允许singletonFactories通过getObject()获取，就从三级缓存singletonFactory.getObject()(三级缓存)获取，如果获取到了则从singletonFactories中移除，并放入earlySingletonObjects中。其实也就是从三级缓存移动到了二级缓存。

5、Spring事务隔离级别：

TransactionDefinition 接口中定义了五个表示隔离级别的常量：

TransactionDefinition.ISOLATION\_DEFAULT: 使用后端数据库默认的隔离级别，Mysql 默认采用的REPEATABLE\_READ隔离级别 Oracle 默认采用的 READ\_COMMITTED隔离级别.  
TransactionDefinition.ISOLATION\_READ\_UNCOMMITTED: 最低的隔离级别，允许读取尚未提交的数据变更，可能会导致脏读、幻读或不可重复读  
TransactionDefinition.ISOLATION\_READ\_COMMITTED: 允许读取并发事务已经提交的数据，可以阻止脏读，但是幻读或不可重复读仍有可能发生  
TransactionDefinition.ISOLATION\_REPEATABLE\_READ: 对同一字段的多次读取结果都是一致的，除非数据是被本身事务自己所修改，可以阻止脏读和不可重复读，但幻读仍有可能发生。  
TransactionDefinition.ISOLATION\_SERIALIZABLE: 最高的隔离级别，完全服从ACID的隔离级别。所有的事务依次逐个执行，这样事务之间就完全不可能产生干扰，也就是说，该级别可以防止脏读、不可重复读以及幻读。但是这将严重影响程序的性能。通常情况下也不会用到该级别。

6、Spring事务中的事务传播行为：

支持当前事务的情况：

TransactionDefinition.PROPAGATION\_REQUIRED： 如果当前存在事务，则加入该事务；如果当前没有事务，则创建一个新的事务。

TransactionDefinition.PROPAGATION\_SUPPORTS： 如果当前存在事务，则加入该事务；如果当前没有事务，则以非事务的方式继续运行。

TransactionDefinition.PROPAGATION\_MANDATORY： 如果当前存在事务，则加入该事务；如果当前没有事务，则抛出异常。（mandatory：强制性）

不支持当前事务的情况：

TransactionDefinition.PROPAGATION\_REQUIRES\_NEW： 创建一个新的事务，如果当前存在事务，则把当前事务挂起。

TransactionDefinition.PROPAGATION\_NOT\_SUPPORTED： 以非事务方式运行，如果当前存在事务，则把当前事务挂起。

TransactionDefinition.PROPAGATION\_NEVER： 以非事务方式运行，如果当前存在事务，则抛出异常。

其他情况：

TransactionDefinition.PROPAGATION\_NESTED： 如果当前存在事务，则创建一个事务作为当前事务的嵌套事务来运行；如果当前没有事务，则该取值等价于TransactionDefinition.PROPAGATION\_REQUIRED。

7、请解释Spring Bean的生命周期？

 首先说一下Servlet的生命周期：实例化，初始init，接收请求service，销毁destroy

 Spring上下文中的Bean生命周期也类似，如下：

（1）实例化Bean：

对于BeanFactory容器，当客户向容器请求一个尚未初始化的bean时，或初始化bean的时候需要注入另一个尚未初始化的依赖时，容器就会调用createBean进行实例化。对于ApplicationContext容器，当容器启动结束后，通过获取BeanDefinition对象中的信息，实例化所有的bean。

（2）设置对象属性（依赖注入）：

实例化后的对象被封装在BeanWrapper对象中，紧接着，Spring根据BeanDefinition中的信息以及通过BeanWrapper提供的设置属性的接口完成依赖注入。

（3）处理Aware接口：

接着，Spring会检测该对象是否实现了xxxAware接口，并将相关的xxxAware实例注入给Bean：

①如果这个Bean已经实现了BeanNameAware接口，会调用它实现的setBeanName(String beanId)方法，此处传递的就是Spring配置文件中Bean的id值；

②如果这个Bean已经实现了BeanFactoryAware接口，会调用它实现的setBeanFactory()方法，传递的是Spring工厂自身。

③如果这个Bean已经实现了ApplicationContextAware接口，会调用setApplicationContext(ApplicationContext)方法，传入Spring上下文；

（4）BeanPostProcessor：

如果想对Bean进行一些自定义的处理，那么可以让Bean实现了BeanPostProcessor接口，那将会调用postProcessBeforeInitialization(Object obj, String s)方法。由于这个方法是在Bean初始化结束时调用的，所以可以被应用于内存或缓存技术；

（5）InitializingBean 与 init-method：

如果Bean在Spring配置文件中配置了 init-method 属性，则会自动调用其配置的初始化方法。

（6）如果这个Bean实现了BeanPostProcessor接口，将会调用postProcessAfterInitialization(Object obj, String s)方法；

以上几个步骤完成后，Bean就已经被正确创建了，之后就可以使用这个Bean了。

（7）DisposableBean：

当Bean不再需要时，会经过清理阶段，如果Bean实现了DisposableBean这个接口，会调用其实现的destroy()方法；

（8）destroy-method：

最后，如果这个Bean的Spring配置中配置了destroy-method属性，会自动调用其配置的销毁方法。

8、 解释Spring支持的几种bean的作用域。

Spring容器中的bean可以分为5个范围：

（1）singleton：默认，每个容器中只有一个bean的实例，单例的模式由BeanFactory自身来维护。

（2）prototype：为每一个bean请求提供一个实例。

（3）request：为每一个网络请求创建一个实例，在请求完成以后，bean会失效并被垃圾回收器回收。

（4）session：与request范围类似，确保每个session中有一个bean的实例，在session过期后，bean会随之失效。

（5）global-session：全局作用域，global-session和Portlet应用相关。当你的应用部署在Portlet容器中工作时，它包含很多portlet。如果你想要声明让所有的portlet共用全局的存储变量的话，那么这全局变量需要存储在global-session中。全局作用域与Servlet中的session作用域效果相同。

SpringMVC篇

1、SpringMVC的流程？

（1）用户发送请求至前端控制器DispatcherServlet；

（2） DispatcherServlet收到请求后，调用HandlerMapping处理器映射器，请求获取Handle；

（3）处理器映射器根据请求url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 具体处理器(Handler，也叫后端控制器)；

（6）Handler执行完成返回ModelAndView；

（7）HandlerAdapter将Handler执行结果ModelAndView返回给DispatcherServlet；

（8）DispatcherServlet将ModelAndView传给ViewResolver视图解析器进行解析；

（9）ViewResolver解析后返回具体View；

（10）DispatcherServlet对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。

MyBatis篇

1、Mybatis是如何进行分页的？分页插件的原理是什么？

        Mybatis使用RowBounds对象进行分页，它是针对ResultSet结果集执行的内存分页，而非物理分页。可以在sql内直接书写带有物理分页的参数来完成物理分页功能，也可以使用分页插件来完成物理分页。

       分页插件的基本原理是使用Mybatis提供的插件接口，实现自定义插件，在插件的拦截方法内拦截待执行的sql，然后重写sql，根据dialect方言，添加对应的物理分页语句和物理分页参数。

2、Mybatis的配置内容？

外层：configuration标签作为mybatis配置的根元素，

内部：

environment配置mybatis环境，即数据库连接

transationManager：配置mybatis事务管理

dataSource：数据源

mappers:映射器，告诉mybatis去哪找映射文件

## 三、数据库篇：

（一）关系型数据库

1、mysql索引匹配规则

最左匹配原则

2、聊聊mysql存储引擎

3、说说mysql锁

4、mysql怎么解决不可重复读的

5、数据库三范式：

（1）数据库表的每一列都是不可分割的原子数据项；

（2）数据库表中的每个实例或行必须可以被唯一地区分

**（3）**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

（二）NOSQL数据库

1、redis有几种存储结构

hash、String、ZSet、Set、List

2、redis单线程为什么效率还很高

1.redis是基于内存的，内存的读写速度非常快；  
2.redis是单线程的，省去了很多上下文切换线程的时间；  
3.redis使用多路复用技术，可以处理并发的连接；

3、redis的使用场景

**数据库、缓存和消息中间件**

## 四、线程篇：

1、线程的生命周期

新建（new Thread）、就绪（Runnable）、运行（Running）、阻塞（Block）、死亡（Dead）

2、volatile的作用

变量可见性、禁止重排序

3、说说synchronized和ReentrantLock的区别

1.共同点：

i.都是用来协调多线程对共享对象、变量的访问

ii.都是可重入锁，同一线程可以多次获得同一个锁

iii.都保证了可见性和互斥性

2.不同点：

1.ReentrantLock通过lock()与unlock()来进行加锁与解锁操作，与synchronized会被JVM自动解锁机制不同，ReentrantLock加锁后需要手动解锁操作

2. ReentrantLock是可中断、公平锁、多个锁

4、java内存模型

5、++i操作线程安全吗？