**J2SE**

**String类为什么是final**

主要是为了“效率” 和 “安全性” 的缘故

**重写equals为何要重写hashCode**

抽象类和接口的区别

**线程**

**ConcurrentHashMap**

同步的数据结构，例如concurrentHashMap的源码理解以及内部实现原理，为什么他是同步的且效率高

atomicinteger和Volatile等线程安全操作的关键字的理解和使用

了解可重入锁的含义，以及ReentrantLock 和synchronized的区别

2. 常用的线程池模式以及不同线程池的使用场景

3. newFixedThreadPool此种线程池如果线程数达到最大值后会怎么办，底层原理。

ThreadPool的深入考察； BlockingQueue的使用

原子类的实现volatile的理解

多线程同步怎么做

定时线程的使用

**NIO**

1.非阻塞: 避免创建大量线程，相对于多线程开销较小。

2.基于事件驱动: 线程上下文的切换都是有意义的。

3.复用单一长连接，并使用线程池并发处理请求，减少握手和提高并发效率，性能较好。长连接双向异步推送，比轮询、阻塞（线程无效切换的开销）效率高。

Netty: 异步高性能的通信框架,RPC工具，基础通信组件。

**Map或者HashMap的存储原理**

由数组+链表的一个结构组成

**反射**

反射中，Class.forName和classloader的区别

**JVM**

1. Java的内存模型以及GC算法

2. jvm性能调优都做了什么

3. 介绍JVM中7个区域，然后把每个区域可能造成内存的溢出的情况说明

4. 介绍GC 和GC Root不正常引用。

5. 自己从classload 加载方式，加载机制说开去，从程序运行时数据区，讲到内存分配，讲到String常量池，讲到JVM垃圾回收机制，算法，hotspot。反正就是各种扩展

6. jvm 如何分配直接内存， new 对象如何不分配在堆而是栈上，常量池解析

7. 数组多大放在 JVM 老年代（不只是设置 PretenureSizeThreshold ，问通常多大，没做过一问便知）

8. 老年代中数组的访问方式

9. GC 算法，永久代对象如何 GC ， GC 有环怎么处理

10. 谁会被 GC ，什么时候 GC

11. 如果想不被 GC 怎么办

12. 如果想在 GC 中生存 1 次怎么办

**Web**

**HTTP协议，GET和POST 的区别**

1.根据HTTP规范，GET用于信息获取，而且应该是安全的和幂等的。POST表示可能修改变服务器上的资源的请求

2.GET请求的数据会附在URL之后,如：login.action?name=hyddd。POST把提交的数据则放置在是HTTP包body中

3.GET方式提交的数据最多只能是1024字节，理论上POST没有限制，可传较大量的数据

**Listener vs Filter**

filter:过滤请求。过滤字符编码，逻辑判断，如是否登录，有没有权限访问页面

listener：它也是随web应用的启动而启动，只初始化一次，随web应用的停止而销毁。主要作用是： 做一些初始化

**J2EE**

**分布式、集群环境中，缓存如何刷新，如何保持同步？**

缓存如何刷新？

1)定时刷新  2)主动刷新覆盖   每个缓存框架都有自带的刷新机制，或者说缓存失效机制，就拿Redis和 Ehcache举例， 他们都有自带的过期机制，另外主动刷新覆盖时，只需获取对应的key进行数据的覆盖即可

缓存如何保持同步？

这个redis有自带的集群同步机制，即复制功能，具体参考：[基于Redis分布式缓存实现](http://blog.csdn.net/pi9nc/article/details/17719737" \t "_blank)      Ehcache也有分布式缓存同步的配置，只需要配置不同服务器地址即可，参照：[Ehcache分布式缓存同步](http://blog.csdn.net/bjyfb/article/details/7939672" \t "_blank)

**Spring**

spring如何实现事物管理的

**1.MVC**

springMVC的原理

**2.IOC:** spring作为第三方管理资源对象

1)资源集中管理，实现资源的可配置和易管理。**(面向接口编程,扩展性高)**。

2)降低了资源双方的耦合度。让开发人员更多的关注业务代码

**3.AOP**

1)aspectj静态代理

2)CGLIB动态代理/JDK动态代理: JDK针对接口

**freemarker**

在java领域，表现层技术主要有三种：jsp、freemarker、velocity。

优点：1.不能写java代码，实现严格的mvc分离2.可使用表达式语言

**DB**

**一条sql执行过长时间，如何优化**

联表性能差：

1、涉及多表的联表或者子查询

1）看是否能进行业务拆分2）相关字段冗余或者3）创建汇总表。4)分解关联查询，单表查询之后的结果进行字段整合  
2.如果以上两种都不能操作，非要联表查询

1)那么考虑对相对应的查询条件做索引。加快查询速度  
3、针对数量大的表进行历史表分离（如交易流水表）  
4、读写分离，降低主库压力

**NOSQL**

现代互联网应用，关系数据库的很多主要特性却往往无用武之地

1.数据库事务一致性需求

2.数据库的写实时性和读实时性需求

3.对复杂的SQL查询，特别是多表关联查询的需求

去掉关系型数据库的两大重要基础：以关系代数为基础的结构化查询语句（SQL）和事务一致性保证（ACID）

1. 高并发读写2)海量数据的高效率存储和访问3)高可用性和可伸缩性

**Redis**

redis是一款开源的高性能key-value数据库，拥有丰富的键值储存类型，并提供多种语言的API。与一般数据库不同，redis是使用内存作为主存，周期性的将数据写到硬盘上。因为redis读写数据都使用内存，所以它的速度是非常快的，很适合我们来存放一些临时性的数据。

**Redis vs Memcached**

1.使用简单的key-value存储的话，Memcached的内存利用率更高

2. 由于Redis只使用单核，而Memcached可以使用多核， Memcached相对性能更高

3.拥有更多的数据结构和并支持更丰富的数据操作。

4.集群管理的不同

**服务器**

**Apache和tomcat**

apache是web服务器，专门提供HTTP服务。处理静态内容，比如HTML/图片。

tomcat是java应用服务器，servlet容器，是apache的扩展

**Nginx与apache**

轻量级，异步非阻塞，高并发环境保持低资源低消耗的高性能。而apache是阻塞的。但比较稳定。

1.反向代理，缓存静态文件 2.负载均衡

**设计原则**

**工厂方法模式的优点**

低耦合、高内聚，开放封闭原则

1开闭原则2单一职责 2依赖倒转