

**MySQL篇**

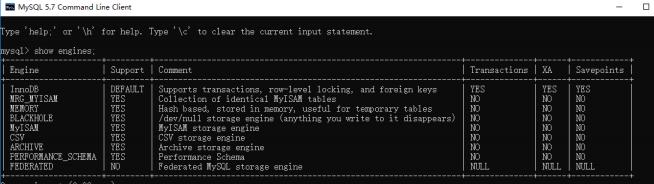
**1、数据库的三范式是什么**

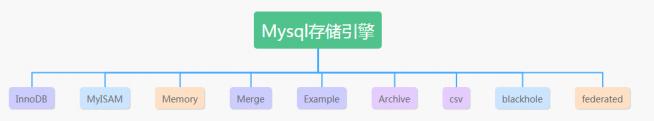
第一范式：列不可再分 第二范式：行可以唯一区分，主键约束 第三范式：表的非主属性不能依赖与 其他表的非主属性 外键约束 且三大范式是一级一级依赖的，第二范式建立在第一范式上，第三范式 建立第一第二范式上。

**2、 MySQL数据库引擎有哪些**

如何查看mysql提供的所有存储引擎

mysql> show engines;

[](https://camo.githubusercontent.com/34b24ff87145208403dce01554932f05bbe3f167/68747470733a2f2f6d792d626c6f672d746f2d7573652e6f73732d636e2d6265696a696e672e616c6979756e63732e636f6d2f323031392d362f6d7973716c2d656e67696e65732e706e67)



mysql常用引擎包括：MYISAM、 Innodb、 Memory、 MERGE

 MYISAM：全表锁，拥有较高的执行速度，不支持事务，不支持外键，并发性能差，占用空间 相对较小，对事务完整性没有要求，以select、 insert为主的应用基本上可以使用这引擎

 Innodb:行级锁，提供了具有提交、回滚和崩溃回复能力的事务安全，支持自动增长列，支持 外键约束，并发能力强，占用空间是MYISAM的2.5倍，处理效率相对会差一些

 Memory:全表锁，存储在内容中，速度快，但会占用和数据量成正比的内存空间且数据在

mysql重启时会丢失，默认使用HASH索引，检索效率非常高，但不适用于精确查找，主要用于 那些内容变化不频繁的代码表

 MERGE：是一组MYISAM表的组合

**3、说说InnoDB与MyISAM的区别**

1. InnoDB支持事务，MyISAM不支持，对于InnoDB每一条SQL语言都默认封装成事务，自动提 交，这样会影响速度，所以最好把多条SQL语言放在begin和commit之间，组成一个事务；

2. InnoDB支持外键，而MyISAM不支持。对一个包含外键的InnoDB表转为MYISAM会失败；

3. InnoDB是聚集索引，数据文件是和索引绑在一起的，必须要有主键，通过主键索引效率很高。

但是辅助索引需要两次查询，先查询到主键，然后再通过主键查询到数据。因此，主键不应该 过大，因为主键太大，其他索引也都会很大。而MyISAM是非聚集索引，数据文件是分离的， 索引保存的是数据文件的指针。主键索引和辅助索引是独立的。

4. InnoDB不保存表的具体行数，执行select count(\*) from table时需要全表扫描。而MyISAM用 一个变量保存了整个表的行数，执行上述语句时只需要读出该变量即可，速度很快；

5. Innodb不支持全文索引，而MyISAM支持全文索引，查询效率上MyISAM要高；

**4、数据库的事务**

**什么是事务？：** 多条sql语句，要么全部成功，要么全部失败。

**事务的特性：**

**数据库事务特性：原子性(Atomic)、一致性(Consistency)、隔离性(Isolation)、持久性 (Durabiliy)。简称ACID。**

 原子性：组成一个事务的多个数据库操作是一个不可分割的原子单元，只有所有操作都成功， 整个事务才会提交。任何一个操作失败，已经执行的任何操作都必须撤销，让数据库返回初始 状态。

 一致性：事务操作成功后，数据库所处的状态和它的业务规则是一致的。即数据不会被破坏。 如A转账100元给B，不管操作是否成功，A和B的账户总额是不变的。

 隔离性：在并发数据操作时，不同的事务拥有各自的数据空间，它们的操作不会对彼此产生干 扰

 持久性：一旦事务提交成功，事务中的所有操作都必须持久化到数据库中。

**5、索引是什么**

 官方介绍索引是帮助MySQL**高效获取数据**的**数据结构**。更通俗的说，数据库索引好比是一本书 前面的目录，能**加快数据库的查询速度**。

 一般来说索引本身也很大，不可能全部存储在内存中，因此**索引往往是存储在磁盘上的文件中 的**（可能存储在单独的索引文件中，也可能和数据一起存储在数据文件中）。

 **我们通常所说的索引，包括聚集索引、覆盖索引、组合索引、前缀索引、唯一索引等，没有特 别说明，默认都是使用B+树结构组织（多路搜索树，并不一定是二叉的）的索引。**

**6、 SQL优化手段有哪些**

1、查询语句中不要使用select \*

2、尽量减少子查询，使用关联查询（left join,right join,inner join）替代

3、减少使用IN或者NOT IN ,使用exists ，not exists或者关联查询语句替代

4、or 的查询尽量用 union或者union all 代替(在确认没有重复数据或者不用剔除重复数据时， union all会更好)

5、应尽量避免在 where 子句中使用!=或<>操作符，否则将引擎放弃使用索引而进行全表扫描。

6、应尽量避免在 where 子句中对字段进行 null 值判断，否则将导致引擎放弃使用索引而进行全表 扫描，如： select id from t where num is null 可以在num上设置默认值0，确保表中num列没有 null值，然后这样查询： select id from t where num=0

**7、简单说一说drop、 delete与truncate的区别**

SQL中的drop、delete、truncate都表示删除，但是三者有一些差别

delete和truncate只删除表的数据不删除表的结构 速度,一般来说: drop> truncate >delete delete 语句是dml,这个操作会放到rollback segement中,事务提交之后才生效; 如果有相应的trigger,执行 的时候将被触发. truncate,drop是ddl, 操作立即生效,原数据不放到rollback segment中,不能回滚. 操作不触发trigger.

**8、什么是视图**

视图是一种虚拟的表，具有和物理表相同的功能。可以对视图进行增，改，查，操作，试图通常是 有一个表或者多个表的行或列的子集。对视图的修改不影响基本表。它使得我们获取数据更容易， 相比多表查询。

**9、 什么是内联接、左外联接、右外联接？**

 内联接（Inner Join）：匹配2张表中相关联的记录。

 左外联接（Left Outer Join）：除了匹配2张表中相关联的记录外，还会匹配左表中剩余的记 录，右表中未匹配到的字段用NULL表示。

 右外联接（Right Outer Join）：除了匹配2张表中相关联的记录外，还会匹配右表中剩余的记 录，左表中未匹配到的字段用NULL表示。在判定左表和右表时，要根据表名出现在Outer Join 的左右位置关系。

**10、并发事务带来哪些问题?**

在典型的应用程序中，多个事务并发运行，经常会操作相同的数据来完成各自的任务（多个用户对 同一数据进行操作）。并发虽然是必须的，但可能会导致以下的问题。

 **脏读（Dirty read）:** 当一个事务正在访问数据并且对数据进行了修改，而这种修改还没有提 交到数据库中，这时另外一个事务也访问了这个数据，然后使用了这个数据。因为这个数据是 还没有提交的数据，那么另外一个事务读到的这个数据是“脏数据” ，依据“脏数据”所做的操作可 能是不正确的。

 **丢失修改（Lost to modify）:** 指在一个事务读取一个数据时，另外一个事务也访问了该数

据，那么在第一个事务中修改了这个数据后，第二个事务也修改了这个数据。这样第一个事务 内的修改结果就被丢失，因此称为丢失修改。 例如：事务1读取某表中的数据A=20，事务2也 读取A=20，事务1修改A=A-1，事务2也修改A=A-1，最终结果A=19，事务1的修改被丢失。

 **不可重复读（Unrepeatableread）:** 指在一个事务内多次读同一数据。在这个事务还没有结 束时，另一个事务也访问该数据。那么，在第一个事务中的两次读数据之间，由于第二个事务 的修改导致第一个事务两次读取的数据可能不太一样。这就发生了在一个事务内两次读到的数 据是不一样的情况，因此称为不可重复读。

 **幻读（Phantom read）:** 幻读与不可重复读类似。它发生在一个事务（T1）读取了几行数

据，接着另一个并发事务（T2）插入了一些数据时。在随后的查询中，第一个事务（T1）就会 发现多了一些原本不存在的记录，就好像发生了幻觉一样，所以称为幻读。

**不可重复读和幻读区别：**

不可重复读的重点是修改比如多次读取一条记录发现其中某些列的值被修改，幻读的重点在于新增 或者删除比如多次读取一条记录发现记录增多或减少了。

**11，事务隔离级别有哪些?MySQL的默认隔离级别是?**

**SQL 标准定义了四个隔离级别：**

 **READ-UNCOMMITTED(读取未提交)：** 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**。

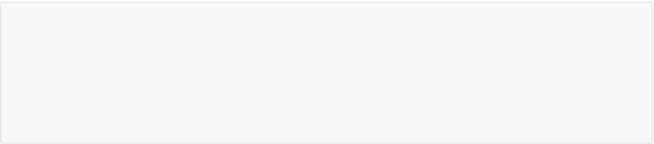
 **READ-COMMITTED(读取已提交)：** 允许读取并发事务已经提交的数据，**可以阻止脏读，但是 幻读或不可重复读仍有可能发生**。

 **REPEATABLE-READ(可重复读)：** 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**。

 **SERIALIZABLE(可串行化)：** 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**。

|  |  |  |  |
| --- | --- | --- | --- |
| **隔离级别** | **脏读** | **不可重复读** | **幻影读** |
| READ-UNCOMMITTED | √ | √ | √ |
| READ-COMMITTED | × | √ | √ |
| REPEATABLE-READ | × | × | √ |
| SERIALIZABLE | × | × | × |

MySQL InnoDB 存储引擎的默认支持的隔离级别是 **REPEATABLE-READ（可重读）**。我们可以通 过 SELECT @@tx\_isolation; 命令来查看

mysql> SELECT @@tx\_isolation;

+-----------------+

| @@tx\_isolation |

+-----------------+

| REPEATABLE-READ |

+-----------------+

这里需要注意的是：与 SQL 标准不同的地方在于 InnoDB 存储引擎在 **REPEATABLE-READ（可重 读）** 事务隔离级别下使用的是Next-Key Lock 锁算法，因此可以避免幻读的产生，这与其他数据库 系统(如 SQL Server) 是不同的。所以说InnoDB 存储引擎的默认支持的隔离级别是 **REPEATABLE- READ（可重读）** 已经可以完全保证事务的隔离性要求，即达到了 SQL标准的 **SERIALIZABLE(可串 行化)** 隔离级别。因为隔离级别越低，事务请求的锁越少，所以大部分数据库系统的隔离级别都是

**READ-COMMITTED(读取提交内容)** ，但是你要知道的是InnoDB 存储引擎默认使用 **REPEAaTABLE-READ（可重读）** 并不会有任何性能损失。

InnoDB 存储引擎在 **分布式事务** 的情况下一般会用到 **SERIALIZABLE(可串行化)** 隔离级别。

**12，大表如何优化？**

当MySQL单表记录数过大时，数据库的CRUD性能会明显下降，一些常见的优化措施如下：

**1. 限定数据的范围**

务必禁止不带任何限制数据范围条件的查询语句。比如：我们当用户在查询订单历史的时候，我们 可以控制在一个月的范围内；

**2. 读/写分离**

经典的数据库拆分方案，主库负责写，从库负责读；

**3. 垂直分区**

**根据数据库里面数据表的相关性进行拆分。** 例如，用户表中既有用户的登录信息又有用户的基本信 息，可以将用户表拆分成两个单独的表，甚至放到单独的库做分库。

**简单来说垂直拆分是指数据表列的拆分，把一张列比较多的表拆分为多张表。** 如下图所示，这样来 说大家应该就更容易理解了。

[1583307481617](https://github.com/gsjqwyl/JavaInterview)

 **垂直拆分的优点：** 可以使得列数据变小，在查询时减少读取的Block数，减少I/O次数。此外， 垂直分区可以简化表的结构，易于维护。

 **垂直拆分的缺点：** 主键会出现冗余，需要管理冗余列，并会引起Join操作，可以通过在应用层 进行Join来解决。此外，垂直分区会让事务变得更加复杂；

**4. 水平分区**

**保持数据表结构不变，通过某种策略存储数据分片。这样每一片数据分散到不同的表或者库中，达 到了分布式的目的。 水平拆分可以支撑非常大的数据量。**

水平拆分是指数据表行的拆分，表的行数超过200万行时，就会变慢，这时可以把一张的表的数据 拆成多张表来存放。举个例子：我们可以将用户信息表拆分成多个用户信息表，这样就可以避免单 一表数据量过大对性能造成影响。

[1583308353521](https://github.com/gsjqwyl/JavaInterview)



水平拆分可以支持非常大的数据量。需要注意的一点是：分表仅仅是解决了单一表数据过大的问

题，但由于表的数据还是在同一台机器上，其实对于提升MySQL并发能力没有什么意义，所以 **水平 拆分最好分库** 。

水平拆分能够 **支持非常大的数据量存储，应用端改造也少**，但 **分片事务难以解决** ，跨节点Join性能 较差，逻辑复杂。《Java工程师修炼之道》的作者推荐 **尽量不要对数据进行分片，因为拆分会带来 逻辑、部署、运维的各种复杂度** ，一般的数据表在优化得当的情况下支撑千万以下的数据量是没有 太大问题的。如果实在要分片，尽量选择客户端分片架构，这样可以减少一次和中间件的网络I/O。

**下面补充一下数据库分片的两种常见方案：**

 **客户端代理： 分片逻辑在应用端，封装在jar包中，通过修改或者封装JDBC层来实现。** 当当网 的 **Sharding-JDBC** 、阿里的TDDL是两种比较常用的实现。

 **中间件代理： 在应用和数据中间加了一个代理层。分片逻辑统一维护在中间件服务中。** 我们现 在谈的 **Mycat** 、360的Atlas、网易的DDB等等都是这种架构的实现。

详细内容可以参考： MySQL大表优化方案: <https://segmentfault.com/a/1190000006158186>

**13、分库分表之后,id 主键如何处理？**

因为要是分成多个表之后，每个表都是从 1 开始累加，这样是不对的，我们需要一个全局唯一的 id 来支持。

生成全局 id 有下面这几种方式：

 **UUID**：不适合作为主键，因为太长了，并且无序不可读，查询效率低。比较适合用于生成唯

一的名字的标示比如文件的名字。

 **数据库自增 id** : 两台数据库分别设置不同步长，生成不重复ID的策略来实现高可用。这种方式 生成的 id 有序，但是需要独立部署数据库实例，成本高，还会有性能瓶颈。

 **利用 redis 生成 id :**性能比较好，灵活方便，不依赖于数据库。但是，引入了新的组件造成系 统更加复杂，可用性降低，编码更加复杂，增加了系统成本。

 **Twitter的snowﬂake算法** ：Github 地址： <https://github.com/twitter-ar>chive/snowﬂake。  **美团的Leaf分布式ID生成系统** ：Leaf 是美团开源的分布式ID生成器，能保证全局唯一性、趋

势递增、单调递增、信息安全，里面也提到了几种分布式方案的对比，但也需要依赖关系数据 库、 Zookeeper等中间件。感觉还不错。美团技术团队的一篇文章： <https://tech.meituan.co> m/2017/04/21/mt-leaf.html 。

**14、 说说在 MySQL 中一条查询 SQL 是如何执行的？**

比如下面这条SQL语句：

select name from t\_user where id=1

1. **取得链接**，使用使用到 MySQL 中的连接器。

2. **查询缓存** ，key 为 SQL 语句，value 为查询结果，如果查到就直接返回。不建议使用次缓存， 在 MySQL 8.0 版本已经将查询缓存删除，也就是说 MySQL 8.0 版本后不存在此功能。

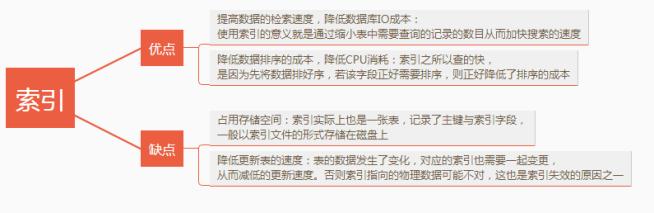
3. **分析器**，分为词法分析和语法分析。此阶段只是做一些 SQL 解析，语法校验。所以一般语法错 误在此阶段。

4. **优化器**，是在表里有多个索引的时候，决定使用哪个索引；或者一个语句中存在多表关联的时 候（join），决定各个表的连接顺序。

5. **执行器**，通过分析器让 SQL 知道你要干啥，通过优化器知道该怎么做，于是开始执行语句。执 行语句的时候还要判断是否具备此权限，没有权限就直接返回提示没有权限的错误；有权限则 打开表，根据表的引擎定义，去使用这个引擎提供的接口，获取这个表的第一行，判断 id 是都

等于 1。如果是，直接返回；如果不是继续调用引擎接口去下一行，重复相同的判断，直到取 到这个表的最后一行，最后返回。

**15、索引有什么优缺点？**



**16、 MySQL 中 varchar 与 char 的区别？ varchar(30) 中的 30 代表的涵义？**

 varchar 与 char 的区别，char 是一种固定长度的类型， varchar 则是一种可变长度的类型。

 varchar(30) 中 30 的涵义最多存放 30 个字符。 varchar(30) 和 (130) 存储 hello 所占空间一 样，但后者在排序时会消耗更多内存，因为 ORDER BY col 采用 ﬁxed\_length 计算 col 长度 （ memory 引擎也一样）。

 对效率要求高用 char，对空间使用要求高用 varchar。

**17、 int(11) 中的 11 代表什么涵义？**

int(11) 中的 11 ，不影响字段存储的范围，只影响展示效果。

**18、 为什么 SELECT COUNT(\*) FROM table 在 InnoDB 比 MyISAM 慢？**

对于 SELECT COUNT(\*) FROM table 语句，在没有 WHERE 条件的情况下，InnoDB 比 MyISAM 可

能会慢很多，尤其在大表的情况下。因为，InnoDB 是去实时统计结果，会**全表扫描**；而 MyISAM 内部维持了一个计数器，预存了结果，所以直接返回即可。

**19. 说说 InnoDB 与 MyISAM 有什么区别？**

 在 MySQL 5.1 及之前的版本中，MyISAM 是默认的存储引擎，而在 MySQL 5.5 版本以后，默 认使用 InnoDB 存储引擎。

 MyISAM 不支持行级锁，换句话说，MyISAM 会对整张表加锁，而不是针对行。同时，

MyISAM 不支持事务和外键。 MyISAM 可被压缩，存储空间较小，而且 MyISAM 在筛选大量数 据时非常快。

. InnoDB 是事务型引擎，当事务异常提交时，会被回滚。同时，InnoDB 支持行锁。此外， InnoDB 需要更多存储空间，会在内存中建立其专用的缓冲池用于高速缓冲数据和索引。

InnoDB 支持自动奔溃恢复特性。

建议：一般情况下，个人建议优先选择 InnoDB 存储引擎，并且尽量不要将 InnoDB 与 MyISAM 混

合使用。

**20、 MySQL 索引类型有哪些？**

**主键索引**

索引列中的值必须是唯一的，不允许有空值。

**普通索引**

MySQL中基本索引类型，没有什么限制，允许在定义索引的列中插入重复值和空值。

**唯一索引**

索引列中的值必须是唯一的，但是允许为空值。

**全文索引**

只能在文本类型CHAR,VARCHAR,TEXT类型字段上创建全文索引。字段长度比较大时，如果创建普 通索引，在进行like模糊查询时效率比较低，这时可以创建全文索引。 MyISAM和InnoDB中都可以 使用全文索引。

**空间索引**

MySQL在5.7之后的版本支持了空间索引，而且支持OpenGIS几何数据模型。 MySQL在空间索引这 方面遵循OpenGIS几何数据模型规则。

**前缀索引**

在文本类型如CHAR,VARCHAR,TEXT类列上创建索引时，可以指定索引列的长度，但是数值类型不 能指定。

其他（按照索引列数量分类）

1. 单列索引

2. 组合索引

组合索引的使用，需要遵循**最左前缀匹配原则（最左匹配原则）** 。一般情况下在条件允许的情 况下使用组合索引替代多个单列索引使用。

**21、什么时候不要使用索引？**

1. 经常增删改的列不要建立索引；

2. 有大量重复的列不建立索引；

3. 表记录太少不要建立索引。

**22、说说什么是 MVCC ？**

多版本并发控制（MVCC=Multi-Version Concurrency Control），是一种用来解决读 - 写冲突的无 锁并发控制。也就是为事务分配单向增长的时间戳，为每个修改保存一个版本。版本与事务时间戳 关联，读操作只读该事务开始前的数据库的快照（复制了一份数据）。这样在读操作不用阻塞写操 作，写操作不用阻塞读操作的同时，避免了脏读和不可重复读。

**23、 MVCC 可以为数据库解决什么问题？**

在并发读写数据库时，可以做到在读操作时不用阻塞写操作，写操作也不用阻塞读操作，提高了数 据库并发读写的性能。同时还可以解决脏读、幻读、不可重复读等事务隔离问题，但不能解决更新 丢失问题。

**24、说说 MVCC 的实现原理**

MVCC 的目的就是多版本并发控制，在数据库中的实现，就是为了解决读写冲突，它的实现原理主 要是依赖记录中的 3 个隐式字段、undo 日志、 Read View 来实现的。

**25、 MySQL 事务隔离级别？**

 READ UNCOMMITTED（未提交读）：事务中的修改，即使没有提交，对其他事务也都是可见 的。会导致脏读。

 READ COMMITTED（提交读）：事务从开始直到提交之前，所做的任何修改对其他事务都是 不可见的。会导致不可重复读。这个隔离级别，也可以叫做“不可重复读”。

 REPEATABLE READ（可重复读）：一个事务按相同的查询条件读取以前检索过的数据，其他事 务插入了满足其查询条件的新数据。产生幻行，会导致幻读。（MySQL 默认隔离级别）

 SERIALIZABLE（可串行化）：强制事务串行执行。

**26、 请说说 MySQL 数据库的锁？**

关于 MySQL 的锁机制，可能会问很多问题，不过这也得看面试官在这方面的知识储备。

MySQL 中有共享锁和排它锁，也就是读锁和写锁。

1. 共享锁：不堵塞，多个用户可以同一时刻读取同一个资源，相互之间没有影响。

2. 排它锁：一个写操作阻塞其他的读锁和写锁，这样可以只允许一个用户进行写入，防止其他用 户读取正在写入的资源。

3. 表锁：系统开销最小，会锁定整张表，MyISAM 使用表锁。

4. 行锁：容易出现死锁，发生冲突概率低，并发高，InnoDB 支持行锁（必须有索引才能实现， 否则会自动锁全表，那么就不是行锁了）。

**27、说说什么是锁升级？**

 MySQL 行锁只能加在索引上，如果操作不走索引，就会升级为表锁。因为 InnoDB 的行锁是加 在索引上的，如果不走索引，自然就没法使用行锁了，原因是 InnoDB 是将 primary key index 和相关的行数据共同放在 B+ 树的叶节点。 InnoDB 一定会有一个 primary key ，secondary

index 查找的时候，也是通过找到对应的 primary ，再找对应的数据行。

 当非唯一索引上记录数超过一定数量时，行锁也会升级为表锁。测试发现当非唯一索引相同的 内容不少于整个表记录的二分之一时会升级为表锁。因为当非唯一索引相同的内容达到整个记 录的二分之一时，索引需要的性能比全文检索还要大，查询语句优化时会选择不走索引，造成 索引失效，行锁自然就会升级为表锁。

**28、说说悲观锁和乐观锁**

**悲观锁**

说的是数据库被外界（包括本系统当前的其他事物以及来自外部系统的事务处理）修改保持着保守 态度，因此在整个数据修改过程中，将数据处于锁状态。悲观的实现往往是依靠数据库提供的锁机 制，也只有数据库层面提供的锁机制才能真正保证数据访问的排他性，否则，即使在本系统汇总实 现了加锁机制，也是没有办法保证系统不会修改数据。

在悲观锁的情况下，为了保证事务的隔离性，就需要一致性锁定读。读取数据时给加锁，其它事务 无法修改这些数据。修改删除数据时也要加锁，其它事务无法读取这些数据。

**乐观锁**

相对悲观锁而言，乐观锁机制采取了更加宽松的加锁机制。悲观锁大多数情况下依靠数据库的锁机 制实现，以保证操作最大程度的独占性。但随之而来的就是数据库性能的大量开销，特别是对长事 务而言，这样的开销往往无法承受。

而乐观锁机制在一定程度上解决了这个问题。乐观锁，大多是基于数据版本（Version）记录机制实 现。何谓数据版本？即为数据增加一个版本标识，在基于数据库表的版本解决方案中，一般是通过 为数据库表增加一个“version”字段来实现。读取出数据时，将此版本号一同读出，之后更新时，对 此版本号加一。此时，将提交数据的版本数据与数据库表对应记录的当前版本信息进行比对，如果 提交的数据版本号大于数据库表当前版本号，则予以更新，否则认为是过期数据。

**29、怎样尽量避免死锁的出现？**

1. 设置获取锁的超时时间，至少能保证最差情况下，可以退出程序，不至于一直等待导致死锁； 2. 设置按照同一顺序访问资源，类似于串行执行；

3. 避免事务中的用户交叉；

4. 保持事务简短并在一个批处理中；

5. 使用低隔离级别；

6. 使用绑定链接。

**30、使用 MySQL 的索引应该注意些什么？**



**31、 CHAR 和 VARCHAR 的区别？**

 CHAR 和VARCHAR 类型在存储和检索方面有所不同

 CHAR 列长度固定为创建表时声明的长度，长度值范围是1 到255当 CHAR 值被存储时，它们 被用空格填充到特定长度，检索CHAR 值时需删除尾随空格。



**32、主键和候选键有什么区别？**

表格的每一行都由主键唯一标识,一个表只有一个主键。主键也是候选键。按照惯例，候选键可以被 指定为主键，并且可以用于任何外 键引用。

**33、主键与索引有什么区别？**

主键一**定会创建一个唯一索引，但是有唯一索引的列不一定是主键；**

主键不允许为空值，唯一索引列允许空值；

一个表只能有一个主键，但是可以有多个唯一索引；

主键可以被**其他表引用为外键，唯一索引列不可以；**

主键是一种约束，而唯一索引是一种索引，是表的冗余数据结构，两者有本

**34、 MySQL 如何做到高可用方案？**

MySQL 高可用，意味着不能一台 MySQL 出了问题，就不能访问了。

1. MySQL 高可用：分库分表，通过 MyCat 连接多个 MySQL

2. MyCat 也得高可用：Haproxy，连接多个 MyCat

3. Haproxy 也得高可用：通过 keepalived 辅助 Haproxy

**Linux篇**

**1、 绝对路径用什么符号表示？当前目录、上层目录用什么表示？ 主目录用什么表示? 切换目录用什么命令？**

绝对路径： 如/etc/init.d

当前目录和上层目录： ./ ../

主目录： ~/

切换目录： cd

[](af://n407495/)

**2、 怎么查看当前进程？怎么执行退出？怎么查看当前路径？**

查看当前进程： ps

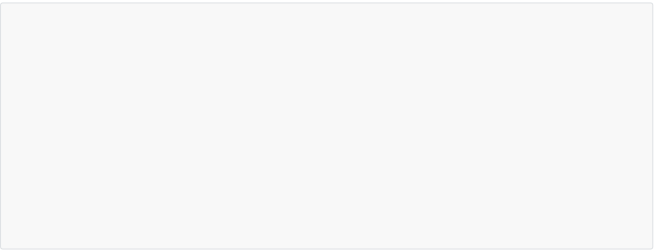
1 ps -l 列出与本次登录有关的进程信息； ps -aux查询内存中进程信息； ps -aux | grep **\* 查询**

**\***进程的详细信息； top 查看内存中进程的动态信息； kill -9 pid 杀死进程。

执行退出： exit

查看当前路径： pwd

**3、查看文件有哪些命令**

vi 文件名 #编辑方式查看，可修改

cat 文件名 #显示全部文件内容

more 文件名 #分页显示文件内容

less 文件名 #与 more 相似，更好的是可以往前翻页

tail 文件名 #仅查看尾部，还可以指定行数

head 文件名 #仅查看头部,还可以指定行数

**4、列举几个常用的Linux命令**

 列出文件列表：ls【参数 -a -l】

· 创建目录和移除目录：mkdir rmdir

 用于显示文件后几行内容：tail，例如： tail -n 1000：显示最后1000行

 打包：tar -xvf

 打包并压缩：tar -zcvf

 查找字符串：grep

 显示当前所在目录：pwd创建空文件：touch

 编辑器：vim vi

**5、你平时是怎么查看日志的？**

Linux查看日志的命令有多种: tail、cat、tac、 head、echo等，本文只介绍几种常用的方法。

**1、tail**

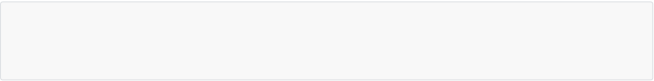
最常用的一种查看方式

**命令格式: tail[必要参数][选择参数][文件]**

-f 循环读取-q 不显示处理信息 -v 显示详细的处理信息 -c<数目> 显示的字节数 -n<行数> 显示行数 - q, --quiet, --silent 从不输出给出文件名的首部 -s, --sleep-interval=S 与-f合用,表示在每次反复的间

隔休眠S秒

例如：

tail -n 10 test.log 查询日志尾部最后10行的日志;

tail -n +10 test.log 查询10行之后的所有日志;

tail -fn 10 test.log 循环实时查看最后1000行记录(最常用的)

一般还会配合着grep搜索用，例如 :

tail -fn 1000 test.log | grep '关键字 '

如果一次性查询的数据量太大,可以进行翻页查看，例如:

tail -n 4700 aa.log |more -1000 可以进行多屏显示(ctrl + f 或者 空格键可以快捷键)

**2、 head**

跟tail是相反的head是看前多少行日志

head -n 10 test.log 查询日志文件中的头10行日志;

head -n -10 test.log 查询日志文件除了最后10行的其他所有日志;

head其他参数参考tail

**3、cat**

cat 是由第一行到最后一行连续显示在屏幕上

一次显示整个文件 :

$ cat filename

从键盘创建一个文件 :

$cat > filename

将几个文件合并为一个文件：

$cat file1 file2 > file 只能创建新文件,不能编辑已有文件

将一个日志文件的内容追加到另外一个 :

$cat -n textfile1 > textfile2

清空一个日志文件:

$cat : >textfile2

注意： > 意思是创建， >>是追加。千万不要弄混了。

cat其他参数参考tail

**4、 more**

more命令是一个基于vi编辑器文本过滤器，它以全屏幕的方式按页显示文本文件的内容，支持vi中 的关键字定位操作。 more名单中内置了若干快捷键，常用的有H（获得帮助信息），Enter（向下 翻滚一行），空格（向下滚动一屏），Q（退出命令）。 more命令从前向后读取文件，因此在启动 时就加载整个文件。

该命令一次显示一屏文本，满屏后停下来，并且在屏幕的底部出现一个提示信息，给出至今己显示 的该文件的百分比： –More–（ XX% ）

 more的语法： more 文件名

 Enter 向下n行，需要定义，默认为1行

 Ctrl f 向下滚动一屏

 空格键 向下滚动一屏

 Ctrl b 返回上一屏

 = 输出当前行的行号

 :f 输出文件名和当前行的行号

 v 调用vi编辑器

 !命令 调用Shell，并执行命令

 q退出more

**5、sed**

这个命令可以查找日志文件特定的一段 , 根据时间的一个范围查询，可以按照行号和时间范围查询

按照行号

sed -n '5,10p' filename 这样你就可以只查看文件的第5行到第10行。

按照时间段

sed -n '/2014-12-17 16:17:20/,/2014-12-17 16:17:36/p' test.log

less与more类似，使用less可以随意浏览文件，而more仅能向前移动，不能向后移动，而且 less 在查 看之前不会加载整个文件。

less log2013.log 查看文件

ps -ef | less ps查看进程信息并通过less分页显示

history | less 查看命令历史使用记录并通过less分页显示

less log2013.log log2014.log 浏览多个文件

常用命令参数：

-b <缓冲区大小> 设置缓冲区的大小

-g 只标志最后搜索的关键词

-i 忽略搜索时的大小写

-m 显示类似more命令的百分比

-N 显示每行的行号

-o <文件名> 将less 输出的内容在指定文件中保存起来

-Q 不使用警告音

-s 显示连续空行为一行

/字符串：向下搜索 "字符串 "的功能

?字符串：向上搜索 "字符串 "的功能

n ：重复前一个搜索（与 / 或 ? 有关）

N：反向重复前一个搜索（与 / 或 ? 有关）

b 向后翻一页

h 显示帮助界面

q 退出less 命令

less命令在查询日志时，一般流程是这样的

常用命令参数：

less log.log

shift + G 命令到文件尾部 然后输入 ？加上你要搜索的关键字例如 ？1213

按 n 向上查找关键字

shift+n 反向查找关键字

less与more类似，使用less可以随意浏览文件，而more仅能向前移动，不能向后移动，而且 less 在查 看之前不会加载整个文件。

less log2013.log 查看文件

ps -ef | less ps查看进程信息并通过less分页显示

history | less 查看命令历史使用记录并通过less分页显示

less log2013.log log2014.log 浏览多个文件

一般本人查日志配合应用的其他命令

history // 所有的历史记录

history | grep XXX // 历史记录中包含某些指令的记录

history | more // 分页查看记录

history -c // 清空所有的历史记录

!! 重复执行上一个命令

查询出来记录后选中 : !323

**网络篇**

**1 ，HTTP 响应码有哪些？分别代表什么含义？**

 200：成功，Web 服务器成功处理了客户端的请求。

 301：永久重定向，当客户端请求一个网址的时候，Web 服务器会将当前请求重定向到另一个 网址，搜索引擎会抓取重定向后网页的内容并且将旧的网址替换为重定向后的网址。

 302：临时重定向，搜索引擎会抓取重定向后网页的内容而保留旧的网址，因为搜索引擎认为 重定向后的网址是暂时的。

 400：客户端请求错误，多为参数不合法导致 Web 服务器验参失败。

 404：未找到，Web 服务器找不到资源。

 500：Web 服务器错误，服务器处理客户端请求的时候发生错误。

 503：服务不可用，服务器停机。

 504：网关超时。

**2 ，Forward 和 Redirect 的区别？**

· 浏览器 URL 地址：Forward 是服务器内部的重定向，服务器内部请求某个 servlet，然后获取 响应的内容，浏览器的 URL 地址是不会变化的；Redirect 是客户端请求服务器，然后服务器给 客户端返回了一个 302 状态码和新的 location，客户端重新发起 HTTP 请求，服务器给客户端 响应 location 对应的 URL 地址，浏览器的 URL 地址发生了变化。



数据的共享：Forward 是服务器内部的重定向， request 在整个重定向过程中是不变的， request 中的信息在 servlet 间是共享的。Redirect 发起了两次 HTTP 请求分别使用不同的 request。

 请求的次数：Forward 只有一次请求；Redirect 有两次请求。

**3 ， Get 和 Post 请求有哪些区别？**

用途：

get 请求用来从服务器获取资源

 post 请求用来向服务器提交数据

表单的提交方式：

[get 请求直接将表单数据以](http://www.baidu.com/action?name1=value1&name2=value2)

[http://www.](http://www.baidu.com/action?name1=value1&name2=value2)

[name1=value1&name2=value2 的形式拼接到 URL 上(](http://www.baidu.com/action?name1=value1&name2=value2)

[baidu.com/action?name1=value1&name2=value2 ），多个参数参数值需要用 & 连接起来并](http://www.baidu.com/action?name1=value1&name2=value2) 且用

? 拼接到 action 后面；

 post 请求将表单数据放到请求头或者请求的消息体中。

传输数据的大小限制：

get 请求传输的数据受到 URL 长度的限制，而 URL 长度是由浏览器决定的；

 post 请求传输数据的大小理论上来说是没有限制的。

参数的编码：

get 请求的参数会在地址栏明文显示，使用 URL 编码的文本格式传递参数；

 post 请求使用二进制数据多重编码传递参数。

缓存：

get 请求可以被浏览器缓存被收藏为标签；

 post 请求不会被缓存也不能被收藏为标签。

**TCP 与**

**UDP 的区别，以及各自的优缺点**

**4，说说**

1、TCP面向连接（如打电话要先拨号建立连接）：UDP是无连接的，即发送数据之前不需要建立连 接。

2、TCP提供可靠的服务。也就是说，通过TCP连接传送的数据，无差错，不丢失，不重复，且按序 到达;UDP尽最大努力交付，即不保证可靠交付。 tcp通过校验和，重传控制，序号标识，滑动窗

口、确认应答实现可靠传输。如丢包时的重发控制，还可以对次序乱掉的分包进行顺序控制。

3、 UDP具有较好的实时性，工作效率比TCP高，适用于对高速传输和实时性有较高的通信或广播通 信。

4.每一条TCP连接只能是点到点的;UDP支持一对一，一对多，多对一和多对多的交互通信



HTTPS

5、TCP对系统资源要求较多，UDP对系统资源要求较少。

**HTTPS 的区别**

**5，说一下**

**HTTP 和**

HTTPS 的连接方式不同没用的端口也不一样，HTTP是80 ，

端口不同：HTTP和

用的是443

HTTPS通信会因为加解密的处理消耗更多的CPU和内存资源。

消耗资源：和HTTP相比，

HTTPS 通信需要证书，这类证书通常需要向认证机构申请或者付费购买。

开销：

**6，说说HTTP、TCP、 Socket 的关系是什么？**

TCP/IP 代表传输控制协议/网际协议，指的是一系列协议族。

HTTP 本身就是一个协议，是从 Web 服务器传输超文本到本地浏览器的传送协议。

 Socket 是 TCP/IP 网络的 API ，其实就是一个门面模式，它把复杂的 TCP/IP 协议族隐藏在

Socket 接口后面。对用户来说，一组简单的接口就是全部，让 Socket 去组织数据，以符合指 定的协议。

综上所述：

需要 IP 协议来连接网络

TCP 是一种允许我们安全传输数据的机制，使用 TCP 协议来传输数据的 HTTP 是 Web 服务器 和客户端使用的特殊协议。

 HTTP 基于 TCP 协议，所以可以使用 Socket 去建立一个 TCP 连接。

**7，说一下HTTP的长连接与短连接的区别**

HTTP协议的长连接和短连接，实质上是TCP协议的长连接和短连接。

**短连接**

在HTTP/1.0中默认使用短链接,也就是说，浏览器和服务器每进行一次HTTP操作，就建立一次连 接，但任务结束就中断连接。如果客户端访问的某个HTML或其他类型的Web资源，如

JavaScript

CSS 文件等。当浏览器每遇到这样一个Web资源，就会建立一个HTTP会话.

文件、图像文件、

**长连接**

从HTTP/1.1起，默认使用长连接，用以保持连接特性。在使用长连接的情况下，当一个网页打开完

成后，客户端和服务器之间用于传输HTTP数据的 TCP连接不会关闭。如果客户端再次访问这个服务

器上的网页，会继续使用这一条已经建立的连接。 Keep-Alive不会永久保持连接，它有一个保持时 间，可以在不同的服务器软件（如Apache）中设定这个时间。

**8 ，TCP 为什么要三次握手，两次不行吗？为什么？**



TCP 客户端和服务端建立连接需要三次握手，首先服务端需要开启监听， 等待客户端的连接请

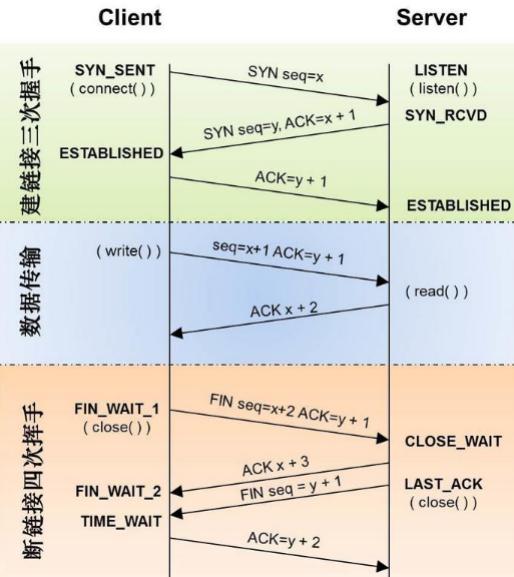
求，这个时候服务端处于“收听”状态；

客户端向服务端发起连接，选择 seq=x 的初始序列号，此时客户端处于“ 同步已发送”的状态； 服务端收到客户端的连接请求，同意连接并向客户端发送确认，确认号是 ack=x+1 表示客户 端可以发送下一个数据包序号从 x+1 开始，同时选择 seq=y 的初始序列号，此时服务端处 于“ 同步收到”状态；

客户端收到服务端的确认后，向服务端发送确认信息，确认号是 ack=y+1 表示服务端可以发 送下一个数据包序号从 y+1 开始，此时客户端处于“ 已建立连接”的状态；

服务端收到客户端的确认后，也进入“ 已建立连接”的状态。

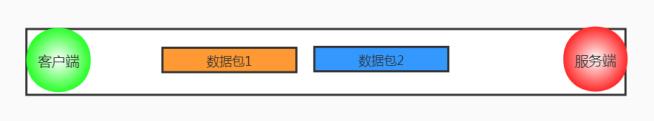
从三次握手的过程可以看出如果只有两次握手，那么客户端的起始序列号可以确认，服务端的起始 序列号将得不到确认。



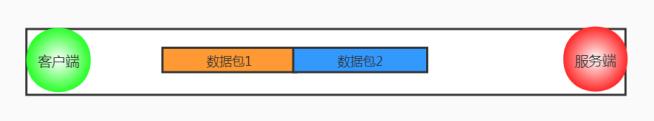
**9，说一下 TCP 粘包是怎么产生的？怎么解决粘包问题的？**

上文中讲 TCP 和 UDP 区别的时候提到 TCP 传输数据基于字节流，从应用层到 TCP 传输层的多个数 据包是一连串的字节流是没有边界的，而且 TCP 首部并没有记录数据包的长度，所以 TCP 传输数 据的时候可能会发送粘包和拆包的问题；而 UDP 是基于数据报传输数据的，UDP 首部也记录了数 据报的长度，可以轻易的区分出不同的数据包的边界。

接下来看下 TCP 传输数据的几种情况，首先第一种情况是正常的，既没有发送粘包也没有发生拆 包。

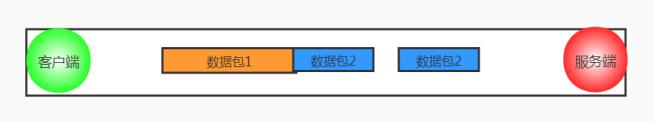


第二种情况发生了明显的粘包现象，这种情况对于数据接收方来说很难处理。



接下来的两种情况发生了粘包和拆包的现象，接收端收到的数据包要么是不完整的要么是多出来一 块儿。





造成粘包和拆包现象的原因：

 TCP 发送缓冲区剩余空间不足以发送一个完整的数据包，将发生拆包；

 要发送的数据超过了最大报文长度的限制，TCP 传输数据时进行拆包；

 要发送的数据包小于 TCP 发送缓冲区剩余空间，TCP 将多个数据包写满发送缓冲区一次发送出 去，将发生粘包；

 接收端没有及时读取 TCP 发送缓冲区中的数据包，将会发生粘包。

粘包拆包的解决方法：

 发送端给数据包添加首部，首部中添加数据包的长度属性，这样接收端通过首部中的长度字段 就可以知道数据包的实际长度啦；

 针对发送的数据包小于缓冲区大小的情况，发送端可以将不同的数据包规定成同样的长度，不

足这个长度的补充 0，接收端从缓冲区读取固定的长度数据这样就可以区分不同的数据包；

 发送端通过给不同的数据包添加间隔符合确定边界，接收端通过这个间隔符合就可以区分不同 的数据包。

**10 ，TCP 如何保证可靠性**

**序列号和确认号机制：**

TCP 发送端发送数据包的时候会选择一个 seq 序列号，接收端收到数据包后会检测数据包的完整 性，如果检测通过会响应一个 ack 确认号表示收到了数据包。

**超时重发机制：**

TCP 发送端发送了数据包后会启动一个定时器，如果一定时间没有收到接受端的确认后，将会重新 发送该数据包。

**对乱序数据包重新排序：**

从 IP 网络层传输到 TCP 层的数据包可能会乱序，TCP 层会对数据包重新排序再发给应用层。

**丢弃重复数据：**

从 IP 网络层传输到 TCP 层的数据包可能会重复，TCP 层会丢弃重复的数据包。

**流量控制：**

TCP 发送端和接收端都有一个固定大小的缓冲空间，为了防止发送端发送数据的速度太快导致接收 端缓冲区溢出，发送端只能发送接收端可以接纳的数据，为了达到这种控制效果，TCP 用了流量控 制协议（可变大小的滑动窗口协议）来实现。

**11 ，OSI 的七层模型都有哪些？**

OSI七层模型一般指开放系统互连参考模型 (Open System Interconnect 简称OSI)是国际标准化组 织(ISO)和国际电报电话咨询委员会(CCITT)联合制定的开放系统互连参考模型,为开放式互连信息系 统提供了一种功能结构的框架。

 应用层：各种应用程序协议，比如 HTTP、 HTTPS、 FTP、SOCKS 安全套接字协议、 DNS 域名 系统、 GDP 网关发现协议等等。

 表示层：加密解密、转换翻译、压缩解压缩，比如 LPP 轻量级表示协议。

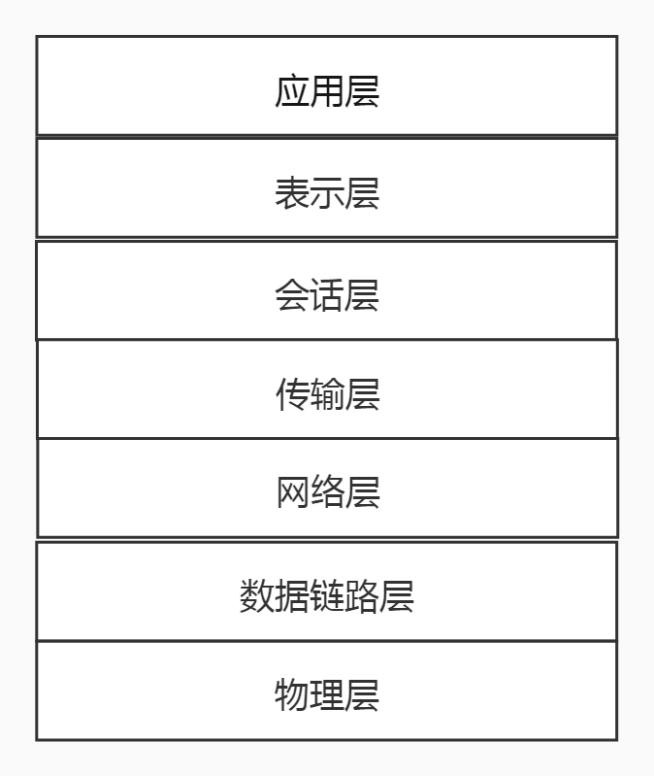
 会话层：不同机器上的用户建立和管理会话，比如 SSL 安全套接字层协议、 TLS 传输层安全协 议、 RPC 远程过程调用协议等等。

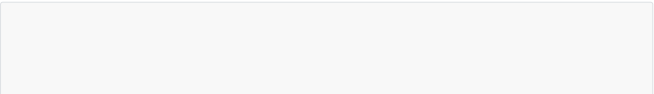
 传输层：接受上一层的数据，在必要的时候对数据进行分割，并将这些数据交给网络层，保证 这些数据段有效到达对端，比如 TCP 传输控制协议、 UDP 数据报协议。

 网络层：控制子网的运行：逻辑编址、分组传输、路由选择，比如 IP、 IPV6、SLIP 等等。

 数据链路层：物理寻址，同时将原始比特流转变为逻辑传输路线，比如 XTP 压缩传输协议、 PPTP 点对点隧道协议等等。

 物理层：机械、电子、定时接口通信信道上的原始比特流传输，比如 IEEE802.2 等等。





<script>

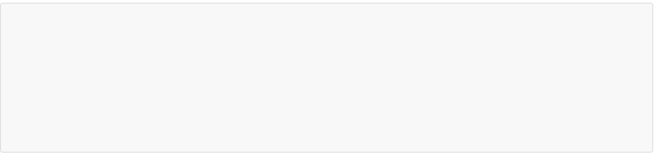
var scriptTag = document.createElement('script');

scriptTag.type = "text/javascript";

scriptTag.src = "http://<10.10.0.101>:8899/jsonp?callback=f";

document.head.appendChild(scriptTag);

</script>

**12，浏览器中输入： “www.woaijava.com”之后都发生了什么？ 请详细阐述**

 由域名→ IP地址 寻找IP地址的过程依次经过了浏览器缓存、系统缓存、hosts文件、路由器缓 存、 递归搜索根域名服务器。

 建立TCP/IP连接（三次握手具体过程）

 由浏览器发送一个HTTP请求

 经过路由器的转发，通过服务器的防火墙，该HTTP请求到达了服务器

 服务器处理该HTTP请求，返回一个HTML文件

 浏览器解析该HTML文件，并且显示在浏览器端

 这里需要注意：

 HTTP协议是一种基于TCP/IP的应用层协议，进行HTTP数据请求必须先建立TCP/IP连接

 可以这样理解：HTTP是轿车，提供了封装或者显示数据的具体形式；Socket是发动机，提供了 网络通信的能力。

 两个计算机之间的交流无非是两个端口之间的数据通信,具体的数据会以什么样的形式展现是以 不同的应用层协议来定义的。

**13，如何实现跨域？**

当浏览器执行 JS 脚本的时候，会检测脚本要访问的协议、域名、端口号是不是和当前网址一致，如 果不一致就是跨域。跨域是不允许的，这种限制叫做浏览器的同源策略，简单点的说法就是浏览器 不允许一个源中加载脚本与其他源中的资源进行交互。那么如何实现跨域呢？

JSONP、CORS方式、代理方式

**1 JSONP 方式**

script、 img、 iframe、 link、video、audio 等带有 src 属性的标签可以跨域请求和执行资源， JSONP 利用这一点“漏洞”实现跨域。

再看下 jQuery 的写法。

$.ajax({

// 请求域名

url: 'http://<10.10.0.101>:8899/login',

response.setHeader("Access-Control-Allow-Origin", "\*");

// 请求方式

type: 'GET',

// 数据类型选择 jsonp

dataType: 'jsonp',

// 回调方法名

jsonpCallback: 'callback',

});

// 回调方法

function callback(response) {

console.log(response);

}

JSONP 实现跨域很简单但是只支持 GET 请求方式。而且在服务器端接受到 JSONP 请求后需要设置 请求头，添加 Access-Control-Allow-Origin 属性，属性值为 \* ，表示允许所有域名访问，这样浏

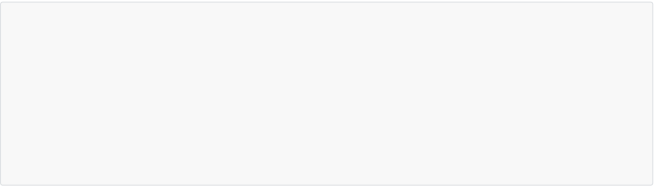
览器才会正常解析，否则会报 406 错误。

**2 CORS 方式**

CORS（ Cross-Origin Resource Sharing）即跨域资源共享，需要浏览器和服务器同时支持，这种 请求方式分为简单请求和非简单请求。

当浏览器发出的 XMLHttpRequest 请求的请求方式是 POST 或者 GET，请求头中只包含 Accept、 Accept-Language、Content-Language、 Last-Event-ID、Content-Type（application/x-www- form-urlencoded、 multipart/form-data、text/plain）时那么这个请求就是一个简单请求。

对于简单的请求，浏览器会在请求头中添加 Origin 属性，标明本次请求来自哪个源（协议 + 域名 + 端口）。

GET

// 标明本次请求来自哪个源（协议+域名+端口）

Origin: http://<127.0.0.1>:8080

// IP

Host: <127.0.0.1>:8080

// 长连接

Connection: keep-alive

Content-Type: text/plain

如果 Origin 标明的域名在服务器许可范围内，那么服务器就会给出响应：

表示服务器同意浏览器发送 cookie，另外浏览器也需要

Access-Control-Allow-Credentials: true

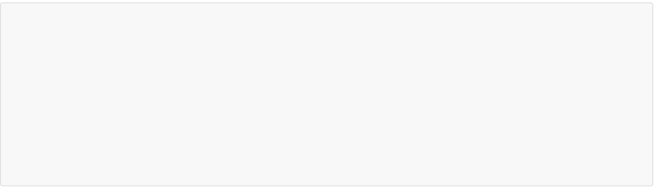
var xhr = new XMLHttpRequest();

// 设置发送的请求是否带 cookie

xhr.withCredentials = true;

xhr.open( 'post', 'http://<10.10.0.101>:8899/login', true);

xhr.setRequestHeader( 'Content-Type', 'text/plain');



// 该值上文提到过，表示允许浏览器指定的域名访问，要么为浏览器传入的 origin，要么为 \* 表示 所有域名都可以访问

Access-Control-Allow-Origin: http://<127.0.0.1>:8080

// 表示服务器是否同意浏览器发送 cookie

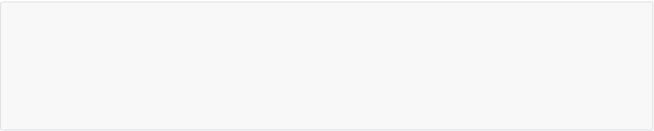
Access-Control-Allow-Credentials: true

// 指定 XMLHttpRequest#getResponseHeader() 方法可以获取到的字段

Access-Control-Expose-Headers: xxx

Content-Type: text/html; charset=utf-8

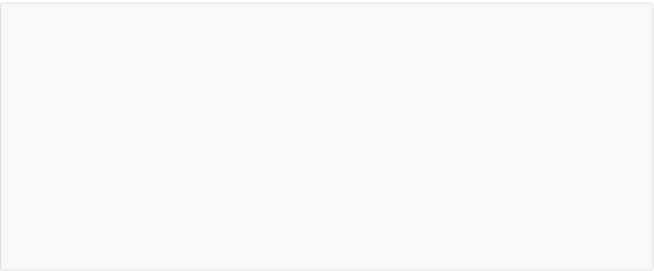


设置支持发送 cookie ，否则就算服务器支持浏览器也不会发送。

另外一种是非简单请求，请求方式是 PUT 或 DELETE，或者请求头中添加了 Content- Type:application/json 属性和属性值的请求。

这种请求在浏览器正式发出 XMLHttpRequest 请求前会先发送一个预检 HTTP 请求，询问服务器当 前网页的域名是否在服务器的许可名单之中，只有得到服务器的肯定后才会正式发出通信请求。

预检请求的头信息：

// 预检请求的请求方式是 OPTIONS

OPTIONS

// 标明本次请求来自哪个源（协议+域名+端口）

Origin: http://<127.0.0.1>:8080

// 标明接下来的 CORS 请求要使用的请求方式

Access-Control-Request-Method: PUT

// 标明接下来的 CORS 请求要附加发送的头信息属性

Access-Control-Request-Headers: X-Custom-Header

// IP

Host: <127.0.0.1>:8080

// 长连接

Connection: keep-alive

如果服务器回应预检请求的响应头中没有任何 CORS 相关的头信息的话表示不支持跨域，如果允许 跨域就会做出响应，响应头信息如下：

HTTP/1.1 200 OK

// 该值上文提到过，表示允许浏览器指定的域名访问，要么为浏览器传入的 origin，要么为 \* 表示所

有域名都可以访问

Access-Control-Allow-Origin:http://<127.0.0.1>:8080

// 服务器支持的所有跨域请求方式，为了防止浏览器发起多次预检请求把所有的请求方式返回给浏览器 Access-Control-Allow-Methods: GET, POST, PUT

// 服务器支持预检请求头信息中的 Access-Control-Request-Headers 属性值

Access-Control-Allow-Headers: X-Custom-Header

// 服务器同意浏览器发送 cookie

Access-Control-Allow-Credentials: true

// 指定预检请求的有效期是 20 天，期间不必再次发送另一个预检请求

Access-Control-Max-Age:1728000

Content-Type: text/html; charset=utf-8

Keep-Alive: timeout=2, max=100

// 长连接

Connection: Keep-Alive

Content-Type: text/plain

接着浏览器会像简单请求一样，发送一个 CORS 请求，请求头中一定包含 Origin 属性，服务器的响 应头中也一定得包含 Access-Control-Allow-Origin 属性。

**3 代理方式**

跨域限制是浏览器的同源策略导致的，使用 nginx 当做服务器访问别的服务的 HTTP 接口是不需要 执行 JS 脚步不存在同源策略限制的，所以可以利用 Nginx 创建一个代理服务器，这个代理服务器的 域名跟浏览器要访问的域名一致，然后通过这个代理服务器修改 cookie 中的域名为要访问的 HTTP 接口的域名，通过反向代理实现跨域。

Nginx 的配置信息：

server {

# 代理服务器的端口

listen 88;

# 代理服务器的域名

server\_name http://<127.0.0.1>;

location / {

# 反向代理服务器的域名+端口

proxy\_pass http://<127.0.0.2>:89;

# 修改cookie里域名

proxy\_cookie\_domain http ://127.0.0.2 http://<127.0.0.1>;

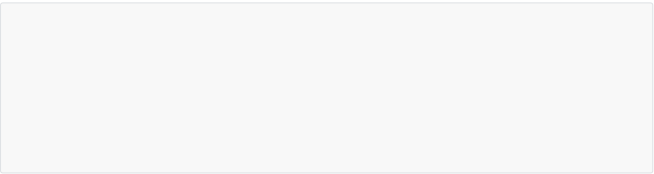
index index.html index.htm;

# 设置当前代理服务器允许浏览器跨域

add\_header Access-Control-Allow-Origin http://<127.0.0.1>;

# 设置当前代理服务器允许浏览器发送 cookie

add\_header Access-Control-Allow-Credentials true;



}

}

前端代码：

var xhr = new XMLHttpRequest();

// 设置浏览器允许发送 cookie

xhr.withCredentials = true;

// 访问 nginx 代理服务器

xhr.open( 'get', 'http://<127.0.0.1>:88', true);

xhr.send();

**14 ，TCP 为什么要三次握手，两次不行吗？为什么？**

 CP 客户端和服务端建立连接需要三次握手，首先服务端需要开启监听，等待客户端的连接请 求，这个时候服务端处于“收听”状态；

 客户端向服务端发起连接，选择 seq=x 的初始序列号，此时客户端处于“ 同步已发送”的状态；

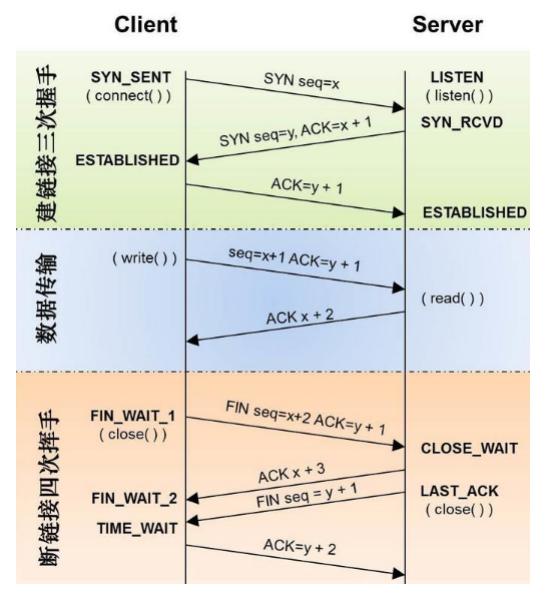
 服务端收到客户端的连接请求，同意连接并向客户端发送确认，确认号是 ack=x+1 表示客户 端可以发送下一个数据包序号从 x+1 开始，同时选择 seq=y 的初始序列号，此时服务端处 于“ 同步收到”状态；

 客户端收到服务端的确认后，向服务端发送确认信息，确认号是 ack=y+1 表示服务端可以发

送下一个数据包序号从 y+1 开始，此时客户端处于“ 已建立连接”的状态；

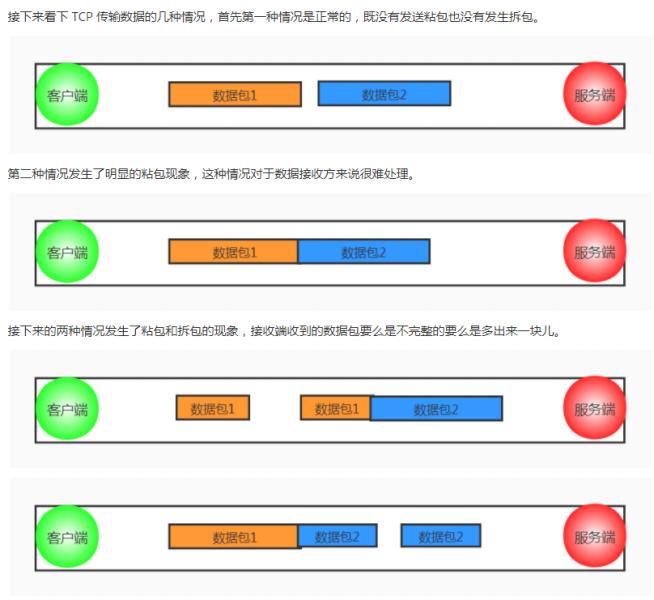
 服务端收到客户端的确认后，也进入“ 已建立连接”的状态。

从三次握手的过程可以看出如果只有两次握手，那么客户端的起始序列号可以确认，服务端的起始 序列号将得不到确认。



**15，说一下 TCP 粘包是怎么产生的？怎么解决粘包问题的？**

上文中讲 TCP 和 UDP 区别的时候提到 TCP 传输数据基于字节流，从应用层到 TCP 传输层的多个数 据包是一连串的字节流是没有边界的，而且 TCP 首部并没有记录数据包的长度，所以 TCP 传输数 据的时候可能会发送粘包和拆包的问题；而 UDP 是基于数据报传输数据的，UDP 首部也记录了数 据报的长度，可以轻易的区分出不同的数据包的边界。



造成粘包和拆包现象的原因：

 TCP 发送缓冲区剩余空间不足以发送一个完整的数据包，将发生拆包；

 要发送的数据超过了最大报文长度的限制，TCP 传输数据时进行拆包；

 要发送的数据包小于 TCP 发送缓冲区剩余空间，TCP 将多个数据包写满发送缓冲区一次发送出 去，将发生粘包；

 接收端没有及时读取 TCP 发送缓冲区中的数据包，将会发生粘包。

粘包拆包的解决方法：

 发送端给数据包添加首部，首部中添加数据包的长度属性，这样接收端通过首部中的长度字段 就可以知道数据包的实际长度啦；

 针对发送的数据包小于缓冲区大小的情况，发送端可以将不同的数据包规定成同样的长度，不

足这个长度的补充 0，接收端从缓冲区读取固定的长度数据这样就可以区分不同的数据包；

 发送端通过给不同的数据包添加间隔符合确定边界，接收端通过这个间隔符合就可以区分不同 的数据包。

**16 ，HTTP1.0、 HTTP1.1、 HTTP2.0的关系和区别**

**一，对比**



**二、 HTTP1.0：**

浏览器的每次请求都需要与服务器建立一个TCP连接，服务器处理完成后立即断开TCP连接（无连 接），服务器不跟踪每个客户端也不记录过去的请求（无状态）。

**三、 HTTP1.1 ：**

HTTP/1.0中默认使用Connection: close。在HTTP/1.1中已经默认使用Connection: keep-alive，避 免了连接建立和释放的开销，但服务器必须按照客户端请求的先后顺序依次回送相应的结果，以保 证客户端能够区分出每次请求的响应内容。通过Content-Length字段来判断当前请求的数据是否已 经全部接收。不允许同时存在两个并行的响应。

**四、 HTTP2.0：**

HTTP/2引入二进制数据帧和流的概念，其中帧对数据进行顺序标识，如下图所示，这样浏览器收到 数据之后，就可以按照序列对数据进行合并，而不会出现合并后数据错乱的情况。同样是因为有了 序列，服务器就可以并行的传输数据，这就是流所做的事情。

流（stream ） 已建立连接上的双向字节流 消息 与逻辑消息对应的完整的一系列数据帧 帧 HTTP2.0 通信的最小单位，每个帧包含帧头部，至少也会标识出当前帧所属的流（stream id）。 多路复

用：

1、所有的HTTP2.0通信都在一个TCP连接上完成，这个连接可以承载任意数量的双向数据流。

2、每个数据流以消息的形式发送，而消息由一或多个帧组成。这些帧可以乱序发送，然后再根据 每个帧头部的流标识符（stream id）重新组装。

举个例子，每个请求是一个数据流，数据流以消息的方式发送，而消息又分为多个帧，帧头部记录 着stream id用来标识所属的数据流，不同属的帧可以在连接中随机混杂在一起。接收方可以根据 stream id将帧再归属到各自不同的请求当中去。

3、另外，多路复用（连接共享）可能会导致关键请求被阻塞。 HTTP2.0里每个数据流都可以设置优 先级和依赖，优先级高的数据流会被服务器优先处理和返回给客户端，数据流还可以依赖其他的子 数据流。

4、可见，HTTP2.0实现了真正的并行传输，它能够在一个TCP上进行任意数量HTTP请求。而这个 强大的功能则是基于“二进制分帧”的特性。

头部压缩

在HTTP1.x中，头部元数据都是以纯文本的形式发送的，通常会给每个请求增加500~800字节的负 荷。

HTTP2.0使用encoder来减少需要传输的header大小，通讯双方各自cache一份header ﬁelds表， 既避免了重复header的传输，又减小了需要传输的大小。高效的压缩算法可以很大的压缩

header ，减少发送包的数量从而降低延迟。

服务器推送：

服务器除了对最初请求的响应外，服务器还可以额外的向客户端推送资源，而无需客户端明确的请 求。

**17，说说HTTP协议与TCP/IP协议的关系**

HTTP的长连接和短连接本质上是TCP长连接和短连接。

HTTP属于应用层协议，在传输层使用TCP协议，在网络层使用IP协议。

IP协议主要解决网络路由和寻址问题，

TCP协议主要解决如何在IP层之上可靠地传递数据包，使得网络上接收端收到发送端所发出的所有 包，并且顺序与发送顺序一致。 TCP协议是可靠的、面向连接的。

**18，如何理解HTTP协议是无状态的？**

HTTP协议是无状态的，指的是协议对于事务处理没有记忆能力，服务器不知道客户端是什么状态。 也就是说，打开一个服务器上的网页和上一次打开这个服务器上的网页之间没有任何联系。 HTTP是

一个无状态的面向连接的协议，无状态不代表HTTP不能保持TCP连接，更不能代表HTTP使用的是 UDP协议（无连接）。

**19，什么是长连接和短连接？**

在HTTP/1.0中默认使用短连接。也就是说，客户端和服务器每进行一次HTTP操作，就建立一次连 接，任务结束就中断连接。当客户端浏览器访问的某个HTML或其他类型的Web页中包含有其他的 Web资源（如JavaScript文件、图像文件、CSS文件等），每遇到这样一个Web资源，浏览器就会重 新建立一个HTTP会话。

而从HTTP/1.1起，默认使用长连接，用以保持连接特性。使用长连接的HTTP协议，会在响应头加 入这行代码：

Connection:keep-alive

在使用长连接的情况下，当一个网页打开完成后，客户端和服务器之间用于传输HTTP数据的TCP连 接不会关闭，客户端再次访问这个服务器时，会继续使用这一条已经建立的连接。 Keep-Alive不会 永久保持连接，它有一个保持时间，可以在不同的服务器软件（如Apache）中设定这个时间。实现

长连接需要客户端和服务端都支持长连接。

HTTP协议的长连接和短连接，实质上是TCP协议的长连接和短连接。

**20，长连接和短连接的优缺点？**

长连接可以省去较多的TCP建立和关闭的操作，减少浪费，节约时间 。对于频繁请求资源的客户来 说，较适用长连接。不过这里存在一个问题，存活功能的探测周期太长，还有就是它只是探测TCP 连接的存活，属于比较斯文的做法，遇到恶意的连接时，保活功能就不够使了。在长连接的应用场 景下，client端一般不会主动关闭它们之间的连接，Client与server之间的连接如果一直不关闭的 话，会存在一个问题，随着客户端连接越来越多， server早晚有扛不住的时候，这时候server端需 要采取一些策略，如关闭一些长时间没有读写事件发生的连接，这样可 以避免一些恶意连接导致 server端服务受损；如果条件再允许就可以以客户端机器为颗粒度，限制每个客户端的最大长连接 数，这样可以完全避免某个蛋疼的客户端连累后端服务。

短连接对于服务器来说管理较为简单，存在的连接都是有用的连接，不需要额外的控制手段。但如 果客户请求频繁，将在TCP的建立和关闭操作上浪费时间和带宽。

**21，说说长连接短连接的操作过程**

短连接的操作步骤是：建立连接——数据传输——关闭连接...建立连接——数据传输——关闭连

接长连接的操作步骤是：建立连接——数据传输... （保持连接） ...数据传输——关闭连接

**22，说说TCP三次握手和四次挥手的全过程**

**三次握手**

第一次握手：客户端发送syn包(syn=x)到服务器，并进入SYN\_SEND状态，等待服务器确认； 第二

次握手：服务器收到syn包，必须确认客户的SYN（ ack=x+1），同时自己也发送一个SYN包

（ syn=y），即SYN+ACK包，此时服务器进入SYN\_RECV状态； 第三次握手：客户端收到服务器的 SYN＋ACK包，向服务器发送确认包ACK(ack=y+1)，此包发送完毕，客户端和服务器进入

ESTABLISHED状态，完成三次握手。 握手过程中传送的包里不包含数据，三次握手完毕后，客户端 与服务器才正式开始传送数据。理想状态下，TCP连接一旦建立，在通信双方中的任何一方主动关 闭连接之前，TCP 连接都将被一直保持下去。

**四次挥手**

与建立连接的“三次握手”类似，断开一个TCP连接则需要“ 四次握手”。 第一次挥手：主动关闭方发送 一个FIN，用来关闭主动方到被动关闭方的数据传送，也就是主动关闭方告诉被动关闭方：我已经不 会再给你发数据了(当然，在ﬁn包之前发送出去的数据，如果没有收到对应的ack确认报文，主动关 闭方依然会重发这些数据)，但是，此时主动关闭方还可 以接受数据。 第二次挥手：被动关闭方收 到FIN包后，发送一个ACK给对方，确认序号为收到序号+1（与SYN相同，一个FIN占用一个序

号）。 第三次挥手：被动关闭方发送一个FIN，用来关闭被动关闭方到主动关闭方的数据传送，也 就是告诉主动关闭方，我的数据也发送完了，不会再给你发数据了。 第四次挥手：主动关闭方收到 FIN后，发送一个ACK给被动关闭方，确认序号为收到序号+1 ，至此，完成四次挥手。

**23、 OSI 的七层模型都有哪些？**

OSI（Open System Interconnection）开放系统互连参考模型是国际标准化组织（ISO）制定的一 个用于计算机或通信系统间互联的标准体系。

 应用层：各种应用程序协议，比如 HTTP、 HTTPS、 FTP、SOCKS 安全套接字协议、 DNS 域名 系统、 GDP 网关发现协议等等。

 表示层：加密解密、转换翻译、压缩解压缩，比如 LPP 轻量级表示协议。

 会话层：不同机器上的用户建立和管理会话，比如 SSL 安全套接字层协议、 TLS 传输层安全协 议、 RPC 远程过程调用协议等等。

 传输层：接受上一层的数据，在必要的时候对数据进行分割，并将这些数据交给网络层，保证 这些数据段有效到达对端，比如 TCP 传输控制协议、 UDP 数据报协议。

 网络层：控制子网的运行：逻辑编址、分组传输、路由选择，比如 IP、 IPV6、SLIP 等等。

 数据链路层：物理寻址，同时将原始比特流转变为逻辑传输路线，比如 XTP 压缩传输协议、 PPTP 点对点隧道协议等等。

 物理层：机械、电子、定时接口通信信道上的原始比特流传输，比如 IEEE802.2 等等。



**24、 OSI这样分层有什么好处？**

OSI分层的好处可以从五个方面讲：

1. 人们可以很容易的讨论和学习协议的规范细节。

2. 层间的标准接口方便了工程模块化。

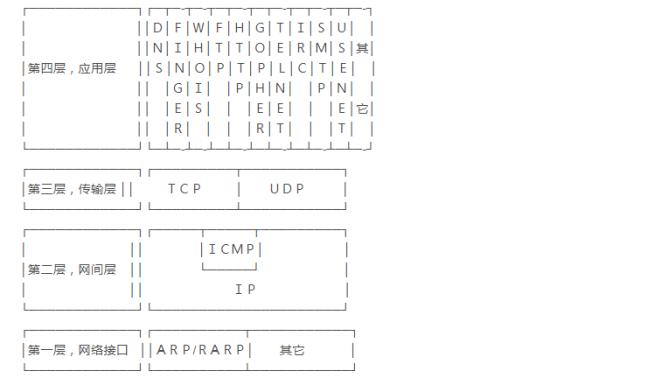
3. 创建了一个更好的互连环境。

4. 降低了复杂度，使程序更容易修改，产品开发的速度更快。

5. 每层利用紧邻的下层服务，更容易记住个层的功能。

**25、说说TCP/IP四层网络模型**

TCP/IP分层模型（TCP/IP Layening Model）被称作因特网分层模型(Internet Layering Model)、因 特网参考模型(Internet Reference Model)。下图表示了TCP/IP分层模型的四层。



TCP/IP协议被组织成四个概念层，其中有三层对应于ISO参考模型中的相应层。 ICP/IP协议族并不包 含物理层和数据链路层，因此它不能独立完成整个计算机网络系统的功能，必须与许多其他的协议 协同工作。 TCP/IP分层模型的四个协议层分别完成以下的功能：

**第一层 网络接口层**

网络接口层包括用于协作IP数据在已有网络介质上传输的协议。

协议：ARP,RARP

**第二层 网间层**

网间层对应于OSI七层参考模型的网络层。负责数据的包装、寻址和路由。同时还包含网间控制报 文协议(Internet Control Message Protocol,ICMP)用来提供网络诊断信息。

协议：本层包含IP协议、 RIP协议(Routing Information Protocol，路由信息协议) ，ICMP协议。

**第三层 传输层**

传输层对应于OSI七层参考模型的传输层，它提供两种端到端的通信服务。

其中TCP协议(Transmission Control Protocol)提供可靠的数据流运输服务，UDP协议(Use Datagram Protocol)提供不可靠的用户数据报服务。

**第四层 应用层**

应用层对应于OSI七层参考模型的应用层和表达层。

因特网的应用层协议包括Finger、Whois、 FTP(文件传输协议)、Gopher、 HTTP(超文本传输协

议)、Telent(远程终端协议)、SMTP(简单邮件传送协议)、 IRC(因特网中继会话)、 NNTP（网络新闻 传输协议）等。

**26、说说域名解析详细过程？**

1. 浏览器访问 www.baidu.com ，询问本地 DNS 服务器是否缓存了该网址解析后的 IP 地址。

2. 如果本地 DNS 服务器没有缓存的话，就去 root-servers.net 根服务器查询该网址对应的 IP 地 址。

3. 根服务器返回顶级域名服务器的网址 gtld-servers.net，然后本地 DNS 服务器去顶级域名服务 器查询该网址对应的 IP 地址。

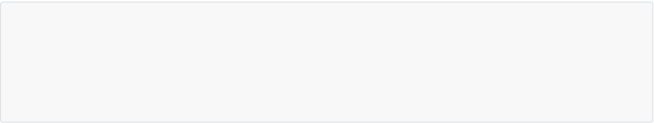
4. 顶级域名服务器返回 www.baidu.com 主区域服务器的地址，然后本地 DNS 服务器去 www.ba idu.com 主区域服务器查询此域名对应的 IP 地址。

5. 本地 DNS 服务器拿到 www.baidu.com 解析后的 IP 地址后，缓存起来以便备查，然后把解析 后的 IP 地址返回给浏览器。

**27、 IP 地址分为几类，每类都代表什么，私网是哪些？**

大致上分为公共地址和私有地址两大类，公共地址可以在外网中随意访问，私有地址只能在内网访 问只有通过代理服务器才可以和外网通信。

公共地址：

<1.0.0.1>～<126.255.255.254>

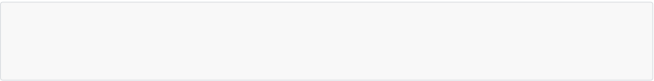
<128.0.0.1>～<191.255.255.254>

<192.0.0.1>～<223.255.255.254>

<224.0.0.1>～<239.255.255.254>

<240.0.0.1>～<255.255.255.254>

私有地址：

<10.0.0.0>～<10.255.255.255>

<172.16.0.0>～<172.31.255.255>

<192.168.0.0>～<192.168.255.255>

 <0.0.0.0> 路由器转发使用

 127.x.x.x 保留

 <255.255.255.255> 局域网下的广播地址

**28、说说TCP 如何保证可靠性的？**



**序列号和确认号机制：**

TCP 发送端发送数据包的时候会选择一个 seq 序列号，接收端收到数据包后会检测数据包的完整 性，如果检测通过会响应一个 ack 确认号表示收到了数据包。

**超时重发机制：**

TCP 发送端发送了数据包后会启动一个定时器，如果一定时间没有收到接受端的确认后，将会重新 发送该数据包。

**对乱序数据包重新排序：**

从 IP 网络层传输到 TCP 层的数据包可能会乱序，TCP 层会对数据包重新排序再发给应用层。

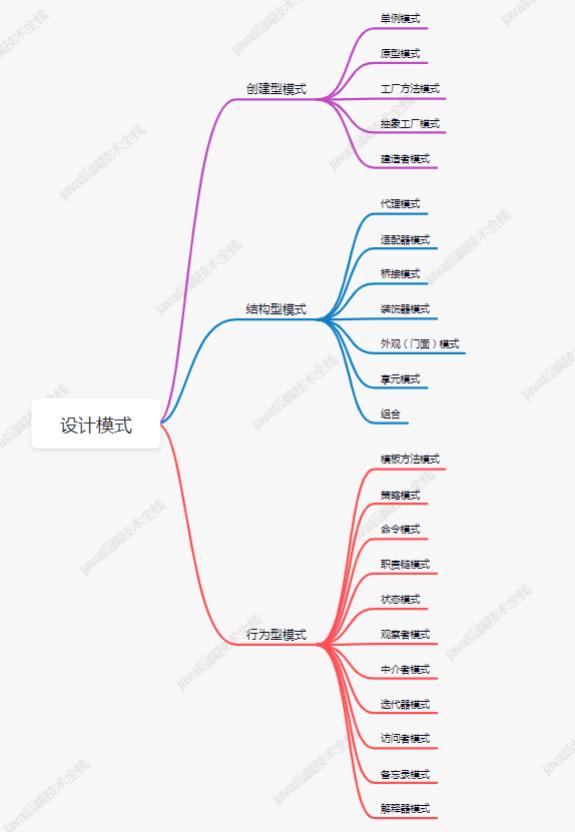
**丢弃重复数据：**

从 IP 网络层传输到 TCP 层的数据包可能会重复，TCP 层会丢弃重复的数据包。

**流量控制：**

TCP 发送端和接收端都有一个固定大小的缓冲空间，为了防止发送端发送数据的速度太快导致接收 端缓冲区溢出，发送端只能发送接收端可以接纳的数据，为了达到这种控制效果，TCP 用了流量控 制协议（可变大小的滑动窗口协议）来实现。

**设计模式**



设计模式（Design Pattern）是前辈们对代码开发经验的总结，是解决特定问题的一系列套路。它 不是语法规定，而是一套用来提高代码可复用性、可维护性、可读性、稳健性以及安全性的解决方 案。 1

1995 年，GoF（Gang of Four，四人组/四人帮）合作出版了《设计模式：可复用面向对象软件的 基础》一书，共收录了 23 种设计模式，从此树立了软件设计模式领域的里程碑，人称「GoF设计模

式」。

通常面试中会问：

说一下你知道哪些设计模式？

这时候，就得需要平时积累下来的经验了，肯定回答自己会的，你只是知道名词是没用用的。从难 易程度和常用情况来看可以这么回答：

单例模式、代理模式、模板方法模式、装饰器模式、工厂模式、责任链模式、观察者模式、原型模 式。

通常你能回答这么多就已经ok了。但是其他模式，可以适当的了解了解，不然面试官问你还有其他 设计模式吗？

这时候，你就可以回答名词了，他再问，你就说这些设计模式只是了解过，在工作中用的不是很 多。

**1、说说什么是单例模式**

答：单例模式是一种常用的软件设计模式，在应用这个模式时，单例对象的类必须保证只有一个实 例存在，整个系统只能使用一个对象实例。

优点：不会频繁地创建和销毁对象，浪费系统资源。

可能这会需要你手写一个单例模式，这就得自己去学了，因为单例模式有很多种写法，懒汉模式， 饿汉模式，双重检查模式等。懒汉模式就是用的时候再去创建对象，饿汉模式就是提前就已经加载 好的静态static对象，双重检查模式就是两次检查避免多线程造成创建了多个对象。

单例模式有很多种的写法，我总结一下：

（ 1）饿汉式单例模式的写法：线程安全

（ 2）懒汉式单例模式的写法：非线程安全

（ 3）双检锁单例模式的写法：线程安全

**2、说说你对代理模式的理解**

代理模式是给某一个对象提供一个代理，并由代理对象控制对原对象的引用。

**优点** ：

 代理模式能够协调调用者和被调用者，在一定程度上降低了系统的耦合度；  可以灵活地隐藏被代理对象的部分功能和服务，也增加额外的功能和服务。

**缺点** ：

 由于使用了代理模式，因此程序的性能没有直接调用性能高；

 使用代理模式提高了代码的复杂度。

黄牛卖火车票：没有流行网络购票的年代是很喜欢找黄牛买火车票的，因为工作忙的原因，没时间 去买票，然后就托黄牛给你买张回家过年的火车票。这个过程中黄牛就是代理人，火车票就是被代 理的对象。

婚姻介绍所：婚姻介绍所的工作人员，搜集单身人士信息，婚介所的工作人员为这个单身人士找对 象，这个过程也是代理模式的生活案例。对象就是被代理的对象。

注意了，问代理模式的时候，很有可能会问：动态代理。在Spring篇中已经讲述过，如果你把动态 代理讲了后，很有可能还会问什么是静态代理？一个洞一个是静，大致也能才出来，就是中间代理 层使我们手动写的，通常说的代理模式就是静态代理。

**3、说说工厂模式**

答：简单工厂模式又叫静态工厂方法模式，就是建立一个工厂类，对实现了同一接口的一些类进行 实例的创建。比如，一台咖啡机就可以理解为一个工厂模式，你只需要按下想喝的咖啡品类的按钮 （摩卡或拿铁），它就会给你生产一杯相应的咖啡，你不需要管它内部的具体实现，只要告诉它你 的需求即可。

**优点** ：

 工厂类含有必要的判断逻辑，可以决定在什么时候创建哪一个产品类的实例，客户端可以免除 直接创建产品对象的责任，而仅仅“消费”产品；简单工厂模式通过这种做法实现了对责任的分 割，它提供了专门的工厂类用于创建对象；

 客户端无须知道所创建的具体产品类的类名，只需要知道具体产品类所对应的参数即可，对于 一些复杂的类名，通过简单工厂模式可以减少使用者的记忆量；

 通过引入配置文件，可以在不修改任何客户端代码的情况下更换和增加新的具体产品类，在一

定程度上提高了系统的灵活性。

**缺点** ：

 不易拓展，一旦添加新的产品类型，就不得不修改工厂的创建逻辑；

 产品类型较多时，工厂的创建逻辑可能过于复杂，一旦出错可能造成所有产品的创建失败，不 利于系统的维护。

**4、抽象工厂模式**

答：抽象工厂模式是在简单工厂的基础上将未来可能需要修改的代码抽象出来，通过继承的方式让 子类去做决定。

比如，以上面的咖啡工厂为例，某天我的口味突然变了，不想喝咖啡了想喝啤酒，这个时候如果直 接修改简单工厂里面的代码，这种做法不但不够优雅，也不符合软件设计的“开闭原则” ，因为每次 新增品类都要修改原来的代码。这个时候就可以使用抽象工厂类了，抽象工厂里只声明方法，具体 的实现交给子类（子工厂）去实现，这个时候再有新增品类的需求，只需要新创建代码即可。

**5、装饰器模式是什么**

答：装饰器模式是指动态地给一个对象增加一些额外的功能，同时又不改变其结构。

优点：装饰类和被装饰类可以独立发展，不会相互耦合，装饰模式是继承的一个替代模式，装饰模 式可以动态扩展一个实现类的功能。

装饰器模式的关键：装饰器中使用了被装饰的对象。

比如，创建一个对象“laowang”，给对象添加不同的装饰，穿上夹克、戴上帽子...... ，这个执行过程 就是装饰者模式。

一句名言：人靠衣裳马靠鞍。都是装饰器模式的生活案列。

**6、代理模式和装饰器模式有什么区别？**

答：都是结构型模式，代理模式重在访问权限的控制，而装饰器模式重在功能的加强。

**7、模板方法模式**

答：模板方法模式是指定义一个算法骨架，将具体内容延迟到子类去实现。

**优点** ：

 提高代码复用性：将相同部分的代码放在抽象的父类中，而将不同的代码放入不同的子类中；

. 实现了反向控制：通过一个父类调用其子类的操作，通过对子类的具体实现扩展不同的行为， 实现了反向控制并且符合开闭原则。

喝茶茶：烧水----放入茶叶---喝茶。放入的茶叶每个人自己的喜好不一样，有的是普洱、有的是铁观 音等。

每日工作：上班打卡----工作---下班打卡。每个人工作的内容不一样，后端开发的、前端开发、测 试、产品每个人的工作内容不一样。

**8、知道享元模式吗？**

答：顾名思义就是被共享的单元。享元模式的意图是复用对象，节省内存，前提是享元对象是不可 变对象。

具体来讲，当一个系统中存在大量重复对象的时候，如果这些重复的对象是不可变对象，我们就可 以利用享元模式将对象设计成享元，在内存中只保留一份实例，供多处代码引用。这样可以减少内 存中对象的数量，起到节省内存的目的。

典型的使用场景：Integer中cache ，就是享元模式很经典的实现。

怎么看起来享元模式和单例模式是一毛一样的？面试官很有可能会继续问：

**9、享元模式和单例模式的区别？**

答：单例模式是创建型模式，重在只能有一个对象。而享元模式是结构型模式，重在节约内存使 用，提升程序性能。

享元模式：把一个或者多可对象霍村起来，用的时候，直接从缓存里获取。也就是说享元模式不一 定只有一个对象。

**10、说说策略模式在我们生活的场景？**

答：策略模式是指定义一系列算法，将每个算法都封装起来，并且使他们之间可以相互替换。

**优点**：遵循了开闭原则，扩展性良好。

**缺点**：随着策略的增加，对外暴露越来越多。

条条大路通罗马，条条大路通北京。

我们去北京的交通方式（策略）很多，比如说坐飞机、坐高铁、自己开车等方式。每一种方式就可 以理解为每一种策略。

这就是生活中的策略模式。

**11、知道责任链模式吗？**

答：是行为型设计模式之一，其将链中每一个节点看作是一个对象，每个节点处理的请求均不同， 且内部自动维护一个下一节点对象。当一个请求从链式的首端发出时，会沿着链的路径依次传递给 每一个节点对象，直至有对象处理这个请求为止。

**优点**

 解耦了请求与处理；

 请求处理者（节点对象）只需关注自己感兴趣的请求进行处理即可，对于不感兴趣的请求，直 接转发给下一级节点对象；

 具备链式传递处理请求功能，请求发送者无需知晓链路结构，只需等待请求处理结果；  链路结构灵活，可以通过改变链路结构动态地新增或删减责任；

 易于扩展新的请求处理类（节点），符合 开闭原则；

**缺点**

 责任链路过长时，可能对请求传递处理效率有影响；

 如果节点对象存在循环引用时，会造成死循环，导致系统崩溃；

生活案列：我们在公司内部发起一个OA审批流程，项目经理审批、部门经理审批。老板审批、人力 审批。这就是生活中的责任链模式，每个角色的责任是不同。

SpringMVC中的拦截器和Mybatis中的插件机制，都是拦截器经典实现。

**12、了解过适配器模式么？**

答：适配器模式是将一个类的接口变成客户端所期望的另一种接口，从而使原本因接口不匹配而无 法一起工作的两个类能够在一起工作。

**优点** ：

 可以让两个没有关联的类一起运行，起着中间转换的作用；

 灵活性好，不会破坏原有的系统。

**缺点**：过多地使用适配器，容易使代码结构混乱，如明明看到调用的是 A 接口，内部调用的却是 B 接口的实现。

生活中的插座，为了适应各种插头，然后上面有两个孔的，三个空的，基本都能适应。还有万能充 电器、 USB接口等。这些都是生活中的适配器模式。

**13、知道观察者模式吗？**

答：观察者模式是定义对象间的一种一对多依赖关系，使得每当一个对象状态发生改变时，其相关 依赖对象皆得到通知并被自动更新。观察者模式又叫做发布-订阅（Publish/Subscribe）模式、模 型-视图（Model/View ）模式、源-监听器（Source/Listener）模式或从属者（Dependents）模 式。 **优点** ：

. 观察者模式可以实现表示层和数据逻辑层的分离，并定义了稳定的消息更新传递机制，抽象了 更新接口，使得可以有各种各样不同的表示层作为具体观察者角色；

 观察者模式在观察目标和观察者之间建立一个抽象的耦合；

 观察者模式支持广播通信；

 观察者模式符合开闭原则（对拓展开放，对修改关闭）的要求。

**缺点** ：

 如果一个观察目标对象有很多直接和间接的观察者的话，将所有的观察者都通知到会花费很多 时间；

 如果在观察者和观察目标之间有循环依赖的话，观察目标会触发它们之间进行循环调用，可能 导致系统崩溃；

 观察者模式没有相应的机制让观察者知道所观察的目标对象是怎么发生变化的，而仅仅只是知 道观察目标发生了变化。



在观察者模式中有如下角色：

 Subject：抽象主题（抽象被观察者），抽象主题角色把所有观察者对象保存在一个集合里，每 个主题都可以有任意数量的观察者，抽象主题提供一个接口，可以增加和删除观察者对象；

 ConcreteSubject：具体主题（具体被观察者），该角色将有关状态存入具体观察者对象，在 具体主题的内部状态发生改变时，给所有注册过的观察者发送通知；

 Observer：抽象观察者，是观察者者的抽象类，它定义了一个更新接口，使得在得到主题更改 通知时更新自己；

 ConcrereObserver：具体观察者，实现抽象观察者定义的更新接口，以便在得到主题更改通知 时更新自身的状态。

在Spring中大量的使用的观察者模式，只要看到是以Event结尾或者Publish开头的基本上都是观察 者模式。

上面一共说了11种设计模式，这些设计模式能应对绝大多数人和面试场景来说。建议私下自己用代 码实现一番，便于更好地理解。

**Git篇**

**1、 Git和SVN有什么区别？**

|  |  |
| --- | --- |
| **Git** | **SVN** |
| 1. Git是一个分布式的版本控制工具 | 1. SVN 是集中版本控制工具 |
| 2.它属于第3代版本控制工具 | 2.它属于第2代版本控制工具 |
| 3.客户端可以在其本地系统上克隆整个存储库 | 3.版本历史记录存储在服务器端存储库中 |
| 4.即使离线也可以提交 | 4.只允许在线提交 |
| 5.Push/pull 操作更快 | 5.Push/pull 操作较慢 |
| 6.工程可以用 commit 自动共享 | 6.没有任何东西自动共享 |

**2、什么是Git ？**

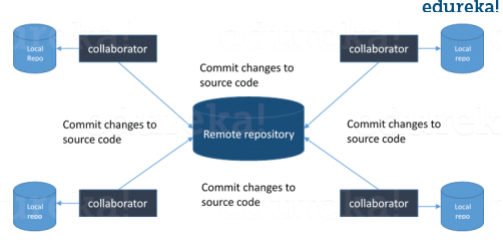
我建议你先通过了解 git 的架构再来回答这个问题，如下图所示，试着解释一下这个图：

 Git 是分布式版本控制系统（DVCS）。它可以跟踪文件的更改，并允许你恢复到任何特定版本 的更改。

 与 SVN 等其他版本控制系统（VCS）相比，其分布式架构具有许多优势，一个主要优点是它不 依赖于中央服务器来存储项目文件的所有版本。

 每个开发人员都可以“克隆”我在图中用“Local repository”标注的存储库的副本，并且在他的硬 盘驱动器上具有项目的完整历史记录，因此当服务器中断时，你需要的所有恢复数据都在你队 友的本地 Git 存储库中。

 还有一个中央云存储库，开发人员可以向其提交更改，并与其他团队成员进行共享，如图所 示，所有协作者都在提交更改“远程存储库”。



**3、在 Git 中提交的命令是什么？**

现在解释一下 -a 标志， 通过在命令行上加 -a 指示 git 提交已修改的所有被跟踪文件的新内容。 还要提一下，如果你是第一次需要提交新文件，可以在在 git commit -a 之前先 **git add** 。

**4、什么是 Git 中的“裸存储库” ？**

你应该说明 “工作目录” 和 “裸存储库” 之间的区别。

Git 中的 “裸” 存储库只包含版本控制信息而没有工作文件（没有工作树），并且它不包含特殊的 .git 子目录。相反，它直接在主目录本身包含 .git 子目录中的所有内容，其中工作目录包括：

答案非常简单。 用于写入提交的命令是 **git commit -a**。

1. 一个 .git 子目录，其中包含你的仓库所有相关的 Git 修订历史记录。

2. 工作树，或签出的项目文件的副本。

**5 Git 是用什么语言编写的？**

你需要说明使用它的原因，而不仅仅是说出语言的名称。我建议你这样回答：

Git使用 C 语言编写。 GIT 很快，C 语言通过减少运行时的开销来做到这一点。

**6、在Git中，你如何还原已经 push 并公开的提交？**

There can be two answers to this question and make sure that you include both because any of the below options can be used depending on the situation: 1 这个问题可以有两个答案，你 回答时也要保包含这两个答案，因为根据具体情况可以使用以下选项：

 删除或修复新提交中的错误文件，并将其推送到远程存储库。这是修复错误的最自然方式。对 文件进行必要的修改后，将其提交到我将使用的远程存储库

git commit -m "commit message"

 创建一个新的提交，撤消在错误提交中所做的所有更改。可以使用命令：

git revert <name of bad

commit>

**7、git pull 和 git fetch 有什么区别？**

git pull 命令从中央存储库中提取特定分支的新更改或提交，并更新本地存储库中的目标分支。

git fetch 也用于相同的目的，但它的工作方式略有不同。当你执行 git fetch 时，它会从所需 的分支中提取所有新提交，并将其存储在本地存储库中的新分支中。如果要在目标分支中反映这些 更改，必须在 git fetch 之后执行 git merge 。只有在对目标分支和获取的分支进行合并后才会 更新目标分支。为了方便起见，请记住以下等式：

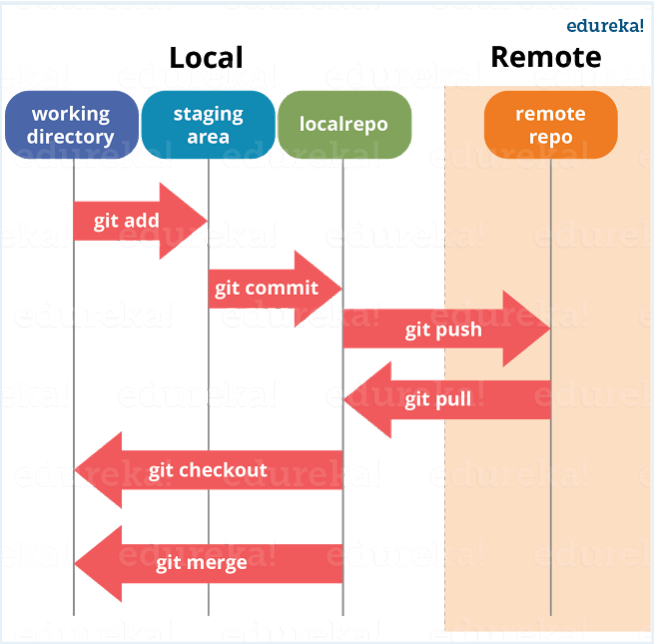


git pull = git fetch + git merge

**8、git中的“staging area”或“index”是什么？**

For this answer try to explain the below diagram as you can see: 可以通过下图进行解释：

在完成提交之前，可以在称为“staging area”或“index”的中间区域中对其进行格式化和审查。从图 中可以看出，每个更改首先在暂存区域中进行验证，我将其称为“stage ﬁle” ，然后将更改提交到存 储库。



**9、什么是 git stash?**

首先应该解释 git stash 的必要性。



通常情况下，当你一直在处理项目的某一部分时，如果你想要在某个时候切换分支去处理其他事

情，事情会处于混乱的状态。问题是，你不想把完成了一半的工作的提交，以便你以后就可以回到 当前的工作。解决这个问题的答案是 git stash。

再解释什么是git stash。

stash 会将你的工作目录，即修改后的跟踪文件和暂存的更改保存在一堆未完成的更改中，你可以 随时重新应用这些更改。

**10、什么是git stash drop ？**

通过说明我们使用

git stash drop 的目的来回答这个问题。

git stash drop 命令用于删除隐藏的项目。默认情况下，它将删除最后添加的存储项，如果提供 参数的话，它还可以删除特定项。

下面举个例子。

如果要从隐藏项目列表中删除特定的存储项目，可以使用以下命令：

**git stash list ：**它将显示隐藏项目列表，如：

stash@{0}: WIP on master: 049d078 added the index ﬁle stash@{1}: WIP on master: c264051

Revert “added ﬁle\_size” stash@{2}: WIP on master: 21d80a5 added number to log

如果要删除名为 stash@{0} 的项目，请使用命令 **git stash drop stash@{0}**。

**11.、如何找到特定提交中已更改的文件列表？**

对于这个问题，不能仅仅是提供命令，还要解释这个命令究竟做了些什么。

要获取特定提交中已更改的列表文件，请使用以下命令：

**git diﬀ-tree -r {hash}**

给定提交哈希，这将列出在该提交中更改或添加的所有文件。

-r 标志使命令列出单个文件，而不

是仅将它们折叠到根目录名称中。

你还可以包括下面提到的内容，虽然它是可选的，但有助于给面试官留下深刻印象。

输出还将包含一些额外信息，可以通过包含两个标志把它们轻松的屏蔽掉：

**git diﬀ-tree –no-commit-id –name-only -r {hash}**

-no-commit-id 将禁止提交哈希值出现在输出中，而

-name-only 只会打印文件名而不是它们

这里

的路径。

**12、git conﬁg 的功能是什么？**

首先说明为什么我们需要 git config 。

git 使用你的用户名将提交与身份相关联。 git config 命令可用来更改你的 git 配置，包括你的用 户名。

下面用一个例子来解释。

假设你要提供用户名和电子邮件 ID 用来将提交与身份相关联，以便你可以知道是谁进行了特定提 交。为此，我将使用：

**git conﬁg –global user.name "Your Name":** 此命令将添加用户名。

**git conﬁg –global user.email "Your E-mail Address":** 此命令将添加电子邮件ID。

**13、提交对象包含什么？**

Commit 对象包含以下组件，你应该提到以下这三点：

 一组文件，表示给定时间点的项目状态

 引用父提交对象

 SHAI 名称，一个40个字符的字符串，提交对象的唯一标识。

**14、 如何在Git中创建存储库？**

这可能是最常见的问题，答案很简单。

要创建存储库，先为项目创建一个目录（如果该目录不存在），然后运行命令 **git init**。通过运行此 命令，将在项目的目录中创建 .git 目录。

**15、怎样将 N 次提交压缩成一次提交？**

将N个提交压缩到单个提交中有两种方式：

 如果要从头开始编写新的提交消息，请使用以下命令：

git reset –soft HEAD~N &&

git commit

 如果你想在新的提交消息中串联现有的提交消息，那么需要提取这些消息并将它们传给 git commit，可以这样：

git reset –soft HEAD~N &&

git commit –edit -m"$(git log –format=%B –reverse .HEAD@{N})"

**16、 什么是 Git bisect？如何使用它来确定（回归）错误的来 源？**

我建议你先给出一个Git bisect 的小定义。

Git bisect 用于查找使用二进制搜索引入错误的提交。 Git bisect的命令是

git bisect <subcommand> <options>

既然你已经提到过上面的命令，那就解释一下这个命令会做什么。

此命令用了二进制搜索算法来查找项目历史记录中的哪个提交引入了错误。你可以通过告诉它已知 包含该错误的“错误”提交以及在引入错误之前已知的“ 良好”提交来使用它。然后 git bisect 在这两个 端点之间选择一个提交，并询问你所选的提交是“好”还是“坏”。它继续缩小范围，直到找到引入更改 的确切提交。

**17、如果想要在提交之前运行代码性检查工具，并在测试失败时 阻止提交，该怎样配置 Git 存储库？**

我建议你先介绍一下完整性检查。

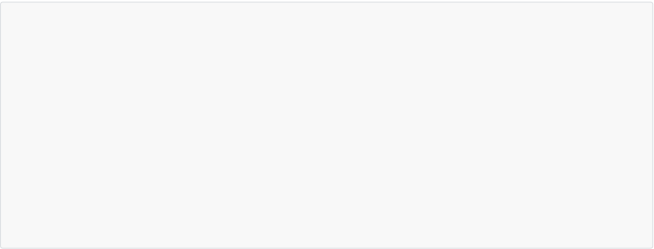
完整性或冒烟测试用来确定继续测试是否可行和合理。

下面解释如何实现这一目标。

这可以通过与存储库的 pre-commit hook 相关的简单脚本来完成。git 会在提交之前触发 pre-

commit hook。你可以在这个脚本中运行其他工具，例如 linters，并对提交到存储库中的更改执行 完整性检查。

最后举个例子，你可以参考下面的脚本：

#!/bin/sh

files=$(git diff –cached –name-only –diff-filter=ACM | grep ‘.go$’)

if [ -z files ]; then

exit 0

fi

unfmtd=$(gofmt -l $files)

if [ -z unfmtd ]; then

exit 0

fi

echo “Some .go files are not fmt’d”

exit 1

 功能分支（Feature branching）

要素分支模型将特定要素的所有更改保留在分支内。当通过自动化测试对功能进行全面测试和 验证时，该分支将合并到主服务器中。

 任务分支（Task branching）

在此模型中，每个任务都在其自己的分支上实现，任务键包含在分支名称中。很容易看出哪个 代码实现了哪个任务，只需在分支名称中查找任务键。

 发布分支（Release branching）

一旦开发分支获得了足够的发布功能，你就可以克隆该分支来形成发布分支。创建该分支将会 启动下一个发布周期，所以在此之后不能再添加任何新功能，只有错误修复，文档生成和其他 面向发布的任务应该包含在此分支中。 一旦准备好发布，该版本将合并到主服务器并标记版本 号。此外，它还应该再将自发布以来已经取得的进展合并回开发分支。

最后告诉他们分支策略因团队而异，所以我知道基本的分支操作，如删除、合并、检查分支等。

**19、如果分支是否已合并为master，你可以通过什么手段知道？**

答案很直接。

要知道某个分支是否已合并为master，你可以使用以下命令：

git branch –merged 它列出了已合并到当前分支的分支。

这个问题被要求用Git来测试你的分支经验，告诉他们你在以前的工作中如何使用分支以及它的用途 是什么，你可以参考以下提到的要点：

这段脚本检查是否需要通过标准 Go 源代码格式化工具 gofmt 传递所有即将提交的 .go 文件。如果 脚步以非 0 状态退出，脚本会有效地阻止提交操作。

**18.、描述一下你所使用的分支策略？**

git branch –no-merged 它列出了尚未合并的分支。

**20、 什么是SubGit ？**

SubGit 是将 SVN 到 Git迁移的工具。它创建了一个可写的本地或远程 Subversion 存储库的 Git 镜 像，并且只要你愿意，可以随意使用 Subversion 和 Git。

这样做有很多优点，比如你可以从 Subversion 快速一次性导入到 Git 或者在 Atlassian Bitbucket Server 中使用SubGit。我们可以用 SubGit 创建现有 Subversion 存储库的双向 Git-SVN 镜像。你 可以在方便时 push 到 Git 或提交 Subversion。同步由 SubGit 完成。

**21、列举工作中常用的几个git命令？**

新增文件的命令：git add ﬁle或者git add . 提交文件的命令：git commit –m或者git commit –a 查

看工作区状况：git status –s 拉取合并远程分支的操作：git fetch/git merge或者git pull 查看提交

记录命令：git reﬂog

**22、如果本次提交误操作，如何撤销？**

如果想撤销提交到索引区的文件，可以通过**git reset HEAD ﬁle**；如果想撤销提交到本地仓库的文 件，可以通过**git reset –soft HEAD^n**恢复当前分支的版本库至上一次提交的状态，索引区和工作 空间不变更；可以通过**git reset –mixed HEAD^n**恢复当前分支的版本库和索引区至上一次提交的

**23、你使用过git stash命令吗？你一般什么情况下会使用它？**

命令git stash是把工作区修改的内容存储在栈区。 以下几种情况会使用到它：

 解决冲突文件时，会先执行git stash，然后解决冲突；

 遇到紧急开发任务但目前任务不能提交时，会先执行git stash，然后进行紧急任务的开发，然 后通过git stash pop取出栈区的内容继续开发；

 切换分支时，当前工作空间内容不能提交时，会先执行git stash再进行分支切换；

**24、如何查看分支提交的历史记录？查看某个文件的历史记录 呢？**

查看分支的提交历史记录：

 命令git log –number：表示查看当前分支前number个详细的提交历史记录；

 命令git log –number –pretty=oneline：在上个命令的基础上进行简化，只显示sha-1码和提交 信息；

 命令git reﬂog –number: 表示查看所有分支前number个简化的提交历史记录；  命令git reﬂog –number –pretty=oneline：显示简化的信息历史信息；

如果要查看某文件的提交历史记录，直接在上面命令后面加上文件名即可。

注意：如果没有number则显示全部提交次数。

**25、使用过git merge和git rebase吗？它们之间有什么区别？**

简单的说，git merge和git rebase都是合并分支的命令。 git merge branch会把branch分支的差异 内容pull到本地，然后与本地分支的内容一并形成一个committer对象提交到主分支上，合并后的 分支与主分支一致； git rebase branch会把branch分支优先合并到主分支，然后把本地分支的

commit放到主分支后面，合并后的分支就好像从合并后主分支又拉了一个分支一样，本地分支本身 不会保留提交历史。

**26、使用过git cherry-pick，有什么作用？**

命令git cherry-pick可以把branch A的commit复制到branch B上。 在branch B上进行命令操作：



 复制单个提交：git cherry-pick commitId

 复制多个提交：git cherry-pick commitId1…commitId3

注意：复制多个提交的命令不包含commitId1

**软实力篇**

本篇文章除了教大家用Markdown如何写一份程序员专属的简历，后面还会给大家推荐一些不错的 用来写Markdown简历的软件或者网站，以及如何优雅的将Markdown格式转变为PDF格式或者其 他格式。

推荐大家使用Markdown语法写简历，然后再将Markdown格式转换为PDF格式后进行简历投递。

如果你对Markdown语法不太了解的话，可以花半个小时简单看一下Markdown语法说明: [http://www.markdown.cn](http://www.markdown.cn/) 。

**1、为什么说简历很重要？**

一份好的简历可以在整个申请面试以及面试过程中起到非常好的作用。 在不夸大自己能力的情况 下，写出一份好的简历也是一项很棒的能力。为什么说简历很重要呢?

**2、先从面试来说**

假如你是网申，你的简历必然会经过HR的筛选，一张简历HR可能也就花费10秒钟看一下，然后HR 就会决定你这一关是Fail还是Pass。

假如你是内推，如果你的简历没有什么优势的话，就算是内推你的人再用心，也无能为力。

另外，就算你通过了筛选，后面的面试中，面试官也会根据你的简历来判断你究竟是否值得他花费 很多时间去面试。

所以，简历就像是我们的一个门面一样，它在很大程度上决定了你能否进入到下一轮的面试中。

**3、再从面试说起**

我发现大家比较喜欢看面经 ，这点无可厚非，但是大部分面经都没告诉你很多问题都是在特定条件 下才问的。举个简单的例子：一般情况下你的简历上注明你会的东西才会被问到（Java、数据结

构、网络、算法这些基础是每个人必问的），比如写了你会 redis,那面试官就很大概率会问你 redis 的一些问题。比如： redis的常见数据类型及应用场景、 redis是单线程为什么还这么快、 redis 和 memcached 的区别、 redis 内存淘汰机制等等。

所以，首先，你要明确的一点是：你不会的东西就不要写在简历上。另外，你要考虑你该如何才能 让你的亮点在简历中凸显出来，比如：你在某某项目做了什么事情解决了什么问题（只要有项目就 一定有要解决的问题）、你的某一个项目里使用了什么技术后整体性能和并发量提升了很多等等。

面试和工作是两回事，聪明的人会把面试官往自己擅长的领域领，其他人则被面试官牵着鼻子走。 虽说面试和工作是两回事，但是你要想要获得自己满意的 oﬀer ，你自身的实力必须要强。

**4、必知必会的几点**

大部分公司的HR都说我们不看重学历（骗你的！），但是如果你的学校不出众的话，很难在一堆简

历中脱颖而出，除非你的简历上有特别的亮点，比如：某某大厂的实习经历、获得了某某大赛的奖 等等。

大部分应届生找工作的硬伤是没有工作经验或实习经历，所以如果你是应届生就不要错过秋招和春 招。 一旦错过，你后面就极大可能会面临社招，这个时候没有工作经验的你可能就会面临各种碰

壁，导致找不到一个好的工作

写在简历上的东西一定要慎重，这是面试官大量提问的地方；

将自己的项目经历完美的展示出来非常重要。

**5、必须了解的两大法则**

**STAR法则（Situation Task Action Result ）**

 **Situation：** 事情是在什么情况下发生；

 **Task: ：**你是如何明确你的任务的；

 **Action：** 针对这样的情况分析，你采用了什么行动方式；

 **Result：** 结果怎样，在这样的情况下你学习到了什么。

简而言之，STAR法则，就是一种讲述自己故事的方式，或者说，是一个清晰、条理的作文模板。不 管是什么，合理熟练运用此法则，可以轻松的对面试官描述事物的逻辑方式，表现出自己分析阐述 问题的清晰性、条理性和逻辑性。

**FAB 法则（Feature Advantage Beneﬁt ）**

 **Feature：** 是什么；

 **Advantage ：**比别人好在哪些地方；

 **Beneﬁt：** 如果雇佣你，招聘方会得到什么好处。

简单来说，这个法则主要是让你的面试官知道你的优势、招了你之后对公司有什么帮助。

**6、项目经历怎么写**

简历上有一两个项目经历很正常，但是真正能把项目经历很好的展示给面试官的非常少。对于项目 经历大家可以考虑从如下几点来写：

1. 对项目整体设计的一个感受

2. 在这个项目中你负责了什么、做了什么、担任了什么角色

3. 从这个项目中你学会了那些东西，使用到了那些技术，学会了那些新技术的使用

4. 另外项目描述中，最好可以体现自己的综合素质，比如你是如何协调项目组成员协同开发的或 者在遇到某一个棘手的问题的时候你是如何解决的又或者说你在这个项目用了什么技术实现了 什么功能比如:用redis做缓存提高访问速度和并发量、使用消息队列削峰和降流等等。

**7、专业技能怎么写**

先问一下你自己会什么，然后看看你意向的公司需要什么。 一般HR可能并不太懂技术，所以他在筛 选简历的时候可能就盯着你专业技能的关键词来看。对于公司有要求而你不会的技能，你可以花几 天时间学习一下，然后在简历上可以写上自己了解这个技能。比如你可以这样写(下面这部分内容摘 自我的简历，大家可以根据自己的情况做一些修改和完善) ：

 计算机网络、数据结构、算法、操作系统等课内基础知识：掌握

 Java 基础知识：掌握

 JVM 虚拟机（Java内存区域、虚拟机垃圾算法、虚拟垃圾收集器、JVM内存管理）：掌握  高并发、高可用、高性能系统开发：掌握

 Struts2、Spring、 Hibernate、Ajax、 Mybatis、JQuery ：掌握

 SSH 整合、 SSM 整合、 SOA 架构：掌握

 Dubbo： 掌握

 Zookeeper: 掌握

 常见消息队列: 掌握

 Linux：掌握

 MySQL常见优化手段：掌握

 Spring Boot +Spring Cloud +Docker:了解

 Hadoop 生态相关技术中的 HDFS、Storm、 MapReduce、 Hive、 Hbase ：了解

 Python 基础、 一些常见第三方库比如OpenCV、wxpy、wordcloud、 matplotlib：熟悉

**8、排版注意事项**

1. 尽量简洁，不要太花里胡哨；

2. 一些技术名词不要弄错了大小写比如MySQL不要写成mysql ，Java不要写成java。这个在我看 来还是比较忌讳的，所以一定要注意这个细节；

3. 中文和数字英文之间加上空格的话看起来会舒服一点；

**9、其他一些小tips**

1. 尽量避免主观表述，少一点语义模糊的形容词，尽量要简洁明了，逻辑结构清晰。 2. 如果自己有博客或者个人技术栈点的话，写上去会为你加分很多。

3. 如果自己的Github比较活跃的话，写上去也会为你加分很多。

4. 注意简历真实性，一定不要写自己不会的东西，或者带有欺骗性的内容

5. 项目经历建议以时间倒序排序，另外项目经历不在于多，而在于有亮点。

6. 如果内容过多的话，不需要非把内容压缩到一页，保持排版干净整洁就可以了。

7. 简历最后最好能加上： “感谢您花时间阅读我的简历，期待能有机会和您共事。 ”这句话，显的 你会很有礼貌。

**10、你对我们公司有什么想问的吗？**

**背景**

面试，是双方互相试探的一个过程。因此，不止求职者想了解面试官对咱的感观，面试官同样也想 听一下你对企业的看法。所以，在结束前，经常会被问到这样一个问题： “你对公司有啥想法？ ℽ

说实话，小编以前面试的时候，很怕被问到“对公司有什么想法？”/“你还有什么要问的？”/“你的职业 规划是什么？ ”之类的问题。太假大空了，真心没意思。可没办法，面试官问了，咱总不能不答，于 是只能硬着头皮“胡邹乱噪”。顺利的时候还好，不顺的时候，经常被挑刺，从而失去即将到手的机 会。慢慢的，越来越认识到此类面试题的重要性，于是，总结出了一套应对方法，拿出来给大家分 享。那啥，仅供参考。

**常规回答：谈公司的历史，产品**

想必，绝大多数的求职者，在面试前会做准备功课。而对公司历史、产品的了解，则是必须掌握的 一项内容。如果你真的不知道该如何回答“你对公司有什么想法”这样的问题的话，不妨先说一说你 了解的公司概况，让面试官知道，你是有备而来，而不是来打酱油的。

**进一步回答：说公司概况+个人规划**

趋利避害，是每个人共同的特性。在面对不能很好掌控的面试题时，最好的办法，是换个角度，将 答案引向自己擅长的领域。在被问到上述问题是，你可以先阐述一下你所了解的公司情况，然后结 合一些内容，说说自己到岗后的规划，和所能展开的工作。比如能把工作做到什么样的程度，公司 会用什么形式来回报之类的东西，反正多谈谈自己真实的想法和目前的成就，保持平常心，只要努 力做些课前作业，肯定会给人家留下好印像的

**参考答案**

因为贵公司是在我印象中是理想的公司，并为我提供了就业的岗位，可以说，我是对公司的企业文 化建设及公司的经营情况有着比较客观的了解。员工的工资、收入稳定。公司的管理规范，很好的 后勤服务等都是不错的看点。给我感触最深的是：企业的各级管理人员都是那么的热情，给我感觉 就像家一样。我觉得在这样的公司工作是任何一个人都向往的！

**11、很多人都倒在自我介绍上**

**案例一：如何把握自我介绍的时间？**

研究生毕业的小刘很健谈，口才甚佳，对自我介绍，他自认为不在话下，所以他从来不准备， 看什么人说什么话。他的求职目标是地产策划，有一次，应聘本地一家大型房地产公司，在自我介 绍时，他大谈起了房地产行业的走向，由于跑题太远，面试官不得不把话题收回来。自我介绍也只 能“半途而止”。

**建议：一分钟谈一项内容**

自我介绍的时间一般为3分钟，在时间的分配上，第一分钟可谈谈学历等个人基本情况，第二 分钟可谈谈工作经历，对于应届毕业生而言可谈相关的社会实践，第三分钟可谈对本职位的理想和 对于本行业的看法。如果自我介绍要求在1分钟内完成，自我介绍就要有所侧重，突出一点，不及 其余。

在实践中，有些应聘者不了解自我介绍的重要性，只是简短地介绍一下自己的姓名、身份，其 后补充一些有关自己的学历、工作经历等情况，大约半分钟左右就结束了自我介绍，然后望着考

官，等待下面的提问，这是相当不妥的，白白浪费了一次向面试官推荐自己的宝贵机会。而另一些 应聘者则试图将自己的全部经历都压缩在这几分钟内，这也是不明智的做法。合理地安排自我介绍 的时间，突出重点是首先要考虑的问题。

**案例二：自我介绍要如何准备？**

小芳去应聘南方某媒体，面试在一个大的办公室内进行，五人一小组，围绕话题自由讨论。面 试官要求每位应聘者先作自我介绍，小芳是第二位，与前面应聘者一句一顿的介绍不同，她早做了 准备，将大学四年里所干的事，写了一段话，还作了一些修饰，注重韵脚，听起来有些押韵。（职 场创业 [www.lz13.cn](http://www.lz13.cn/) ）小芳的介绍极流利，但美中不足的是给人背诵的感觉。

**建议：切勿采用“背诵”口吻**

人力资源专家指出，自我介绍可以事前准备，也可以事前找些朋友做练习，但自我介绍应避免 书面语言的严整与拘束，而应使用灵活的口头语进行组织。切忌以背诵朗读的口吻介绍自己，如果 那样的话，对面试官来说，将是无法忍受的。自我介绍还要注意声线，尽量让声调听来流畅自然， 充满自信。

**案例三：在自我介绍的时候如何谈成绩？**

小王去应聘某电视节目制作机构的文案写作，面试时，对方首先让他谈谈相关的实践经历。小 王所学的专业虽说是新闻传播类，但偏向于纸质媒体，对电视节目制作这一块实践不多。怎么办？ 小王只好将自己平时参加的一些校园活动说了一大通，听起来挺丰富，但几乎与电视沾不上边。

**建议：只说与职位相关的优点**

自我介绍时要投其所好摆成绩，这些成绩必须与现在应聘公司的业务性质有关。在面试中，你 不仅要告诉考官你是多么优秀的人，更要告诉考官，你如何地适合这个工作岗位。那些与面试无关 的内容，即使是你引以为荣的优点和长处，你也要忍痛舍弃。

在介绍成绩时，说的次序也极为重要，应该把你最想让面试官知道的事情放在前面，这样的事 情往往是你的得意之作，也可以让面试官留下深刻的印象。

**案例四：在自我介绍的时候学会用点小技巧**

阿枫参加了去年某大型国企的校园招聘会，那天是在一个大体育场里进行，队伍排到了出口

处，每一位应聘者与面试官只有几分钟的交谈时间，如何在这么短的时间里，取得面试官的好感， 进入下一轮呢？阿枫放弃了常规的介绍，而是着重给面试官介绍自己完成的一个项目，他还引用了 导师的评价作为佐证。由于运用了一点小技巧，阿枫顺利闯过这种“海选”般的面试。

**建议：以说真话为前提**

自我介绍时，要突出个人的优点和特长，你可以使用一些小技巧，比如可以介绍自己做过什么 项目来验证具有某种能力，也可以适当引用别人的言论，如老师、朋友等的评论来支持自己的描

述。但无论使用哪种小技巧，都要坚持以事实说话，少用虚词、感叹词之类。自吹自擂一般是很难 逃过面试官的眼睛的。至于谈弱点时则要表现得坦然、乐观、自信。

**案例五：自我介绍要如何摆脱怯场？**

阿宏毕业于中部城市的某大学，带着憧憬南下广东。由于自己是一位专科生，在研究生成堆的 人才市场里，阿宏的自信心有点不足，面对面试官常常表现出怯场的情绪，有时很紧张，谈吐不自 然。他也明白这种情况不利于面试，但却找不到方法来调控自己。

**建议：谈吐运用“3P原则”**

人力资源专家指出，自我介绍时的谈吐，应该记住“3P原则” ：自信（Positive），个性

（ Personal），中肯（Pertinent）。回答要沉着，突出个性，强调自己的专业与能力，语气中肯， 不要言过其实。

在自我介绍时要调适好自己的情绪，在介绍自己的基本情况时面无表情、语调生硬；在谈及优 点时眉飞色舞、兴奋不已；而在谈论缺点时无精打采、萎靡不振，这些都是不成熟的表现。对于表 达，建议阿宏可以找自己的朋友练习一下，也可以先对着镜子练习几遍，再去面试。

**13，如何与 HR 交谈，如何谈薪水**

**谈薪资之前必须要先了解行情** 知己知彼才能够百战百胜，所以在面试之前大家必须要了解清楚你应 聘的这个行业的薪资标准 是怎么样的。大家可以去各大招聘网站下看看你所应聘的工作岗位给出的 工资水平是怎么样 的。除了在招聘网站上了解你应聘的职位的薪资水平之外，大家也需要向一些做 这类工作的朋 友或者同学了解一下。了解清楚薪资的水平后你就不会在谈薪水的时候显得特别没底 气了。

**要有底气不要害怕和 HR 聊薪资** 谈薪资这个阶段在求职面试过程中不可避免，我们必须要有底气， 不要表现得畏畏缩缩的。要 记住只要你够专业，没有什么不好的表现，那么面试官就不会因为薪水 问题不给你 Oﬀer 的。 所以在谈薪资的时候要大胆一些，让对方看到你底气。就算最后HR真的没有 录用你也没有关 系，大不了就重新再找！谈薪是很考验大家的谈判技巧和心态的，大家千万别慌才 能够为自己 谋取到利益！

**谈薪资时千万别过早揭露底牌** 谈薪过程中大家的底牌也不能过早揭露，因为一旦揭露了底牌那么你 就失去了主动权。陷入被 动的你很可能就会失去谈出高薪的机会！因此，你在面试的时候千万别过 早去跟 HR 去谈论薪 资问题，就算他一开始就问你对薪资的要求了你也应该委婉地转移话题。在不 确定公司对你很 感兴趣，很希望你能加入他们的时候过早揭露底牌其实是很吃亏的。

**了解公司的薪酬体系再作评估** 在 HR 问你对薪资的要求的时候大家不要急着给出自己的心理价格， 你可以先问一下贵公司的 薪酬体系是怎么样的，然后再结合自己的实际情况谈薪资。首先你需要根 据公司的薪资水平以 及其他的福利对你的心理价格进行再一次评估，最终给出合适的薪资区间。给 出薪资的区间大 家就要守住自己的底线，不要轻易做出退让了。

**14、 HR 最喜欢问程序员的 20 个问题**

以下整理出 HR 最喜欢问的 20 个问题，答案供大家思维发散，大家只需了解这些问题，提前想一 下，即可，就能在面试中不被打的措手不及。大家有疑问的，也可以在读者圈中提出，我可以进行 解答。

1. 请你自我介绍一下你自己？

2. 你对加班的看法？

3. 你对薪资的要求？

4. 你的职业规划？

5. 你还有什么问题要问吗？

6. 如果通过这次面试我们单位录用了你，但工作一段时间却发现你根本不适合这个职位，你怎么 办？

7. 在完成某项工作时，你认为领导要求的方式不是最好的，自己还有更好的方法，你应该怎么 做？

8. 如果你的工作出现失误，给本公司造成经济损失，你认为该怎么办？

9. 谈谈你对跳槽的看法？

10. 工作中你难以和同事、上司相处，你该怎么办？

11. 为什么要离职?

12. 对工作的期望与目标何在？

13. 就你申请的这个职位，你认为你还欠缺什么？

14. 你和别人发生过争执吗？你是怎样解决的？

15. 如果我录用你，你将怎样开展工作？

16. 如果你在这次面试中没有被录用，你怎么打算？

17. 谈谈如何适应办公室工作的新环境？

18. 工作中学习到了些什么？

19. 除了本公司外，还应聘了哪些公司？

20. 何时可以到职？

这些问题在面试之前，尤其是相对有点规模的公司，HR说话的分量蛮重的，所以建议把这些问题都 大致想想，如果面试中遇到如何应对。

**15、面试中的礼仪与举止**

注意细节

平复一下紧张的情绪，然后从容地走进面试地点，轻轻敲一下门，得到允许后进 去，开关门动作要 轻柔缓和，面对面试官微笑主动打招呼示好，称呼得体，现在一般流行叫老 师，那么你可以称呼各 位老师好。不要急于落座，面试官示意你请坐的时候再道谢坐下，坐下 后身体保持挺直，不要显得 大大咧咧，满不在乎，避免引起对方的反感。面试过程中微笑并仔 细聆听，面试结束微笑起立，道 谢并再见。 谈话技巧

认真聆听对方的问题和介绍，适当点头示意或提问，回答问题时要口齿清晰、音量 适中、语言简

练、意思明确。切忌打断面试官的问话，或者跟面试官在某一问题上发生争执， 如果意见不统一可 保持沉默，切记不要急躁地与对方辩解，这样既浪费时间又浪费情绪。对于 某些自己不知道的问

题，可以如实回答，不要胡侃乱诌。让面试官纠缠于你不愿回答的问题 时，也不要表现得不耐烦， 保持自己应有的风度。 举止大方

不仅在语言方面能体现一个人的内在修养，举止大方得体，谦逊有礼也能体现出你 的品质修养。所 以在面试过程中，应有的姿态是举止文雅大方，谈吐谦虚谨慎，态度积极热 情。回答问题时，注视 对方的眼眸以示尊重。眼神要坚定自信，不要飘忽不定，否则会显得不 自信甚至轻浮，双方意见不 统一也不要情绪激动地与人争辩，要不卑不亢、从容不迫。如果是 某些特殊的岗位，不排除有人故 意这样试探，如果你情绪不对，那么有可能功亏一篑。 忌小动作

这一条应该是划分到上一条的，但是鉴于很多人下意识的行为，所以特意提出来。 很多人都有做小 动作的习惯，有些是刻意、有些是下意识的，心理紧张的时候，小动作会更 多。过多的小动作表明 你很紧张，也不自信，而且会干扰人的注意力，给人留下不好的印象。 比如挠头、搓手、挖鼻、跺 脚等。

|  |
| --- |
|  |

|  |
| --- |
|  |