

# 简答题

## 数据库部分

### 1、JDBC 连接数据库 7 步

#### 1、加载 JDBC 驱动程序：

在连接数据库之前，首先要加载想要连接的数据库的驱动到 JVM（Java 虚拟机），这通过 `java.lang.Class` 类的静态方法 `forName(String className)` 实现。成功加载后，会将 `Driver` 类的实例注册到 `DriverManager` 类中。

#### 2、提供 JDBC 连接的 URL

- 连接 URL 定义了连接数据库时的协议、子协议、数据源标识。

- 书写形式：协议：子协议：数据源标识

协议：在 JDBC 中总是以 `jdbc` 开始

子协议：是桥连接的驱动程序或是数据库管理系统名称。

数据源标识：标记找到数据库来源的地址与连接端口。

#### 3、创建数据库的连接

- 要连接数据库，需要向 `java.sql.DriverManager` 请求并获得 `Connection` 对象，该对象就代表一个数据库的连接。

- 使用 `DriverManager` 的 `getConnection(String url, String username, String password)` 方法传入指定的欲连接的数据库的路径、数据库的用户名和密码来获得。

#### 4、创建一个 Statement

- 要执行 SQL 语句，必须获得 `java.sql.Statement` 实例，`Statement` 实例分为以下 3 种类型：

- 1、执行静态 SQL 语句。通常通过 `Statement` 实例实现。

- 2、执行动态 SQL 语句。通常通过 `PreparedStatement` 实例实现。

- 3、执行数据库存储过程。通常通过 `CallableStatement` 实例实现。

具体的实现方式：

```
Statement stmt = con.createStatement();
```

```
PreparedStatement pstmt = con.prepareStatement(sql);
```

```
CallableStatement cstmt =
```

```
con.prepareCall("{CALL demoSp(?, ?)}");
```

#### 5、执行 SQL 语句

`Statement` 接口提供了三种执行 SQL 语句的方法：`executeQuery`、`executeUpdate` 和 `execute`

- 1、`ResultSet executeQuery(String sqlString)`：执行查询数据库的 SQL 语句，返回一个结果集（`ResultSet`）对象。

- 2、`int executeUpdate(String sqlString)`：用于执行 `INSERT`、`UPDATE` 或 `DELETE` 语句以及 SQL DDL 语句，如：`CREATE TABLE` 和 `DROP TABLE` 等

- 3、`execute(sqlString)`：用于执行返回多个结果集、多个更新计数或二者组合的

语句。

具体实现的代码：

```
ResultSet rs = stmt.executeQuery("SELECT * FROM ...") ;  
int rows = stmt.executeUpdate("INSERT INTO ...") ;  
boolean flag = stmt.execute(String sql) ;
```

#### 6、处理结果

两种情况：

- 1、执行更新返回的是本次操作影响到的记录数。
  - 2、执行查询返回的结果是一个 ResultSet 对象。
- ResultSet 包含符合 SQL 语句中条件的所有行，并且它通过一套 get 方法提供了对这些

些

行中数据的访问。

- 使用结果集 (ResultSet) 对象的访问方法获取数据：

```
while(rs.next()) {  
    String name = rs.getString("name") ;  
    String pass = rs.getString(1) ; // 此方法比较高效  
}
```

(列是从左到右编号的，并且从列 1 开始)

#### 7、关闭 JDBC 对象

操作完成以后要把所有使用的 JDBC 对象全都关闭，以释放 JDBC 资源，关闭顺序和声明顺序相反：

- 1、关闭记录集
- 2、关闭声明
- 3、关闭连接对象

## 2、SQL 语言共分为四大类：

DML (数据操纵语言) 语句 insert      update      delete      select  
DDL(数据定义语言)语句      creat      alter      drop  
TCL(事物控制语言) commit      savepoint      roallback  
DCL (数据控制语言)      grant      revoke

## 3、Having Where group by 的正确执行顺序是？

--根据部门分组，且最大工资大于10000的部门

```
select e.department_id,max(e.salary) from employees e group by  
e.department_id having max(e.salary)>10000
```

```
SELECT      column, group_function
FROM        table
[WHERE      condition]
[GROUP BY   group_by_expression]
[HAVING     group_condition]
[ORDER BY   column];
```

#### 4、列级别的约束和表级别的约束

列级约束:列级约束是行定义的一部分，只能应用于一列上。

表级约束:表级约束是独立于列的定义，可以应用在一个表中的多列上

只能定义在列级别的约束是： not null

一张表只能建立一个主键约束。

建表约束：NOT NULL 只能在列级定义；其它 4 种既可以在列级定义，也可以在表级定义。

复合主键约束只能在表级定义。

```
create table table1(xx int,no int,name varchar2(10) primary key)
```

```
create table table2(xx int,no int,name varchar2(10) ,primary key (no,name))
```

#### 5、数据库连接池的实现及原理

数据库连接池：

- 为解决传统开发中的数据库连接问题，可以采用数据库连接池技术。
- 数据库连接池的**基本思想**就是为数据库连接建立一个“缓冲池”。预先在缓冲池中放入一定数量的连接，当需要建立数据库连接时，只需从“缓冲池”中取出一个，使用完毕之后再放回去。
- **数据库连接池**负责分配、管理和释放数据库连接，它**允许应用程序重复使用一个现有的数据库连接，而不是重新建立一个**。
- 数据库连接池在初始化时将创建一定数量的数据库连接放到连接池中，这些数据库连接的数量是由**最小数据库连接数**来设定的。无论这些数据库连接是否被使用，连接池都将一直保证至少拥有这么多的连接数量。连接池的**最大数据库连接数量**限定了这个连接池能占有的最大连接数，当应用程序向连接池请求的连接数超过最大连接数量时，这些请求将被加入到等待队列中。

**优势：**连接复用。通过建立一个数据库连接池以及一套连接使用管理策略，使得一个数据库连接可以得到高效、安全的复用，避免了数据库连接频繁建立、关闭的开销。连接池技术尽可能多地重用了消耗内存地资源，大大节省了内存，提高了服务器地服务效率，能够支持更多的客户服务。通过使用连接池，将大大提高程序运行效率，同时，我们可以通过其自身的管理机制来监视数据库连接的数量、使用情况等

- 资源重用：
  - 由于数据库连接得以重用，避免了频繁创建，释放连接引起的大量性能开销。在减少系统消耗的基础上，另一方面也增加了系统运行环境的平稳性。
- 更快的系统反应速度

- 数据库连接池在初始化过程中，往往已经创建了若干数据库连接置于连接池中备用。此时连接的初始化工作均已完成。对于业务请求处理而言，直接利用现有可用连接，避免了数据库连接初始化和释放过程的时间开销，从而减少了系统的响应时间
- 新的资源分配手段
  - 对于多应用共享同一数据库的系统而言，可在应用层通过数据库连接池的配置，实现某一应用最大可用数据库连接数的限制，避免某一应用独占所有的数据库资源
- 统一的连接管理，避免数据库连接泄露
  - 在较为完善的数据库连接池实现中，可根据预先的占用超时设定，强制回收被占用连接，从而避免了常规数据库连接操作中可能出现的资源泄露

## 6 数据库建表的约束的类型

- ◆ NOT NULL
- ◆ UNIQUE
- ◆ PRIMARY KEY
- ◆ FOREIGN KEY
- ◆ CHECK

## 7.什么是视图

### 视图的概念

视图是一种虚表。是建立在已有表的基础上，视图赖以建立的这些表称为基表。向视图提供数据内容的语句为 `SELECT` 语句，可以将视图理解为存储起来的 `SELECT` 语句。视图向用户提供基表数据的另一种表现形式通过一个视图透视一个数据，dba 的时候一条 sql 语句上千行，为了方便把查询的数据保存在视图中，下次直接从视图中查询数据

### 视图的优缺点：

- 简单性。视图不仅可以简化用户对数据的理解，也可以简化他们的操作。那些被经常使用的查询可以被定义为视图，从而使用户不必为以后的操作每次都指定全部的条件。
- 安全性。通过视图用户只能查询和修改他们所能见到的数据。数据库中的其他数据则既看不见也取不到。数据库授权命令可以使每个用户对数据库的检索限制到特定的数据库对象上，但不能授权到数据库特定行和特定的列上。通过视图，用户可以被限制在数据的不同子集上。
- 逻辑数据独立性。视图可以使应用程序和数据库表在一定程度上独立。如果没有视图，应用一定是建立在表上的。有了视图之后，程序可以建立在视图之上，从而程序与数据库表被视图分割开来。

视图也存在一些缺点，主要如下。

- 性能：数据库必须把视图的查询转化成对基本表的查询，如果这个视图是由一个复杂的多表查询所定义，那么，即使是视图的一个简单查询，数据库也把它变成一个复杂的结合体，需要花费一定的时间。
- 修改限制：当用户试图修改视图的某些行时，数据库必须把它转化为对基本表的某些行的修改。对于简单视图来说，这是很方便的，但是，对于比较复杂的视图，可能是不可修改

的。

## 8.什么是索引，怎样创建一个索引,索引使用的原则,有什么优点和缺点

### 索引的概念和优势

索引是用来提高查询效率的数据库对象 使用索引可以快速定位数据 减少磁盘操作

- ◆ 一种独立于表模式的对象 可以存储在于表不同的磁盘表空间
- ◆ 索引被删除或者被破坏不会对表产生影响，只会影响查询速度
- ◆ 索引一旦创建完毕 oracle 管理系统会对其自动维护，而且 oracle 管理系统决定何时使用什么索引 用户不需要在查询语句中指定使用哪个索引
- ◆ 在删除一个表的时候所有基于该表的索引都会被自动删除
- ◆ 通过指针加速 oracle 服务器的查询速度
- ◆ 通过快速定位的方法检索磁盘 I/O

### 索引的分类

- 1) 唯一性索引和非唯一性索引  
按照索引字段是否允许出现重复
- 2) 单字段索引和联合索引  
按照字段基于索引的数目
- 3) 普通索引和函数索引  
按照索引所基于的是普通字段还是复合表达式
- 4) B 树索引和位图索引  
按照索引的数据结构划分 B 树索引(B\*Tree) 和位图索引(Bitmap)

### 索引创建原则：

- 1、表的主键、外键必须有索引；
- 2、数据量超过 300 的表应该有索引；
- 3、经常与其他表进行连接的表,在连接字段上应该建立索引；
- 4、经常出现在 Where 子句中的字段,特别是大表的字段,应该建立索引；
- 5、索引应该建在选择性高的字段上；
- 6、索引应该建在小字段上,对于大的文本字段甚至超长字段,不要建索引；
- 7、复合索引的建立需要进行仔细分析； 尽量考虑用单字段索引代替：
  - A、正确选择复合索引中的主列字段,一般是选择性较好的字段；
  - B、复合索引的几个字段是否经常同时以 AND 方式出现在 Where 子句中?单字段查询是否极少甚至没有?如果是,则可以建立复合索引； 否则考虑单字段索引；
  - C、如果复合索引中包含的字段经常单独出现在 Where 子句中,则分解为多个单字段索引；
  - D、如果复合索引所包含的字段超过 3 个,那么仔细考虑其必要性,考虑减少复合的字段；
  - E、如果既有单字段索引,又有这几个字段上的复合索引,一般可以删除复合索引；
- 8、频繁进行数据操作的表,不要建立太多的索引；
- 9、删除无用的索引,避免对执行计划造成负面影响；

## 9.数据库优化方面的经验

- 1) 用 PreparedStatement 一般来说比 Statement 性能高：一个 sql 发给服务器去执行，涉及步骤：语法检查、语义分析，编译，缓存
- 2) 有外键约束会影响插入和删除性能，如果程序能够保证数据的完整性，那在设计数据库时就去掉外键。
- 3) 表中允许适当冗余，譬如，主题帖的回复数量和最后回复时间等
- 4) 使用子查询代替关联查询，下面子查询语句要比第二条关联查询的效率高

```
1.select e.name,e.salarywhere e.managerid=(select id from employee where name='hzgg');
```

```
2. select e.name,e.salary,m.name,m.salary fromemployees e,employees m where
```

```
e.managerid = m.id andm.name='hzgg';
```

- 5) sql 语句全部大写，特别是列名和表名都大写。特别是 sql 命令的缓存功能，更加需要统一大小写。
- 6) 合理建立索引改善查询性能。ORACLE 使用了一个复杂的自平衡 B-tree 结构。通常,通过索引查询数据比全表扫描要快。当 ORACLE 找出执行查询和 Update 语句的最佳路径时, ORACLE 优化器将使用索引

## 10.什么是存储过程？什么是函数？怎样创建存储过程和函数？有什么优势？

过程用于执行特定操作。如果在应用程序中经常需要执行特定的操作，可以基于这些操作建立一个特定的过程。通过使用过程，不仅可以简化客户端应用程序的开发和维护，而且还可以提高应用程序的运行性能。建立过程的语法如下所示：

```
create or replace procedure proc_name  
arg1 mode datatype,  
arg2 mode datatype  
is/as  
pl/sql
```

函数用于返回特定数据。如果在应用程序中经常需要通过执行 SQL 语句来返回特定数据，那么可以基于这些操作建立特定的函数。通过使用函数，不仅可以简化客户端应用程序的开发和维护，而且还可以提高应用程序的执行性能。建立函数的语法如下所示：

```
create or replace function fun_name  
return datatype  
is/as  
pl/sql
```



从参数的返回情况来看：

如果返回多个参数值最好使用存储过程，如果只有一个返回值的话可以使用函数。

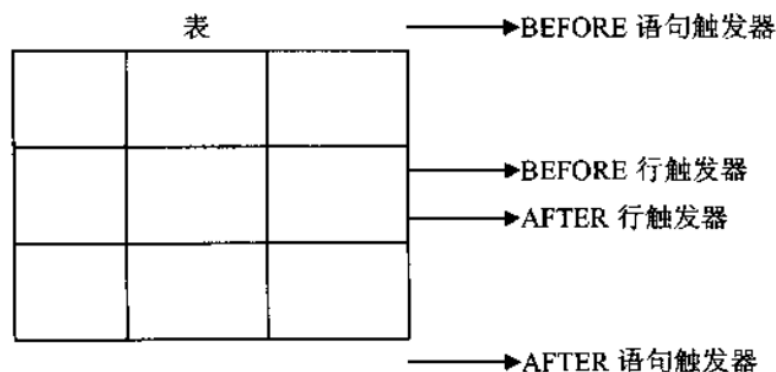
从调用情况来看：

如果在 SQL 语句（DML 或 SELECT）中调用的话一定是存储函数或存储的封装函数不可以是存储过程，但调用存储函数的时候还有好多限制以及函数的纯度等级的问题，如果是在过程化语句中调用的话，就要看你要实现什么样的功能。函数一般情况下是用来计算并返回一个计算结果而存储过程一般是用来完成特定的数据操作（比如修改、插入数据库表或执行某些 DDL 语句等等），所以虽然他们的语法上很相似但用户在使用他们的时候所需要完成的功能大部分情况下是不同的。

## 11.什么是触发器，有何作用？

触发器的概念和触发器的四类：

触发器是指被隐含执行的存储过程，它可以使用 PL/SQL, Java 和 C 进行开发。当发生特定事件（例如修改表、建立对象、登录到数据库）时，Oracle 会自动执行触发器的相应代码。触发器由触发事件、触发条件和触发操作三部分组成。



相对于外部程序、存储过程，触发器可以更快更高效的维护数据

## 12.select count (\*) from student 和 select count (id) from student 之间的区别。

select count(\*) 统计所有学生的记录个数，包括空记录。关心总记录数

Select count(Id) 统计所有学生的记录个数，不包括 null 记录。关心该字段当中有多少个非空记录

## 13 创建 CUSTOMERS 表，字段为：ID:（非空，主键）bigint, NAME:（非空）varchar, AGE: int 类型; 创建 ORDERS 表，字段为：ID:（非空，主键，）bigint, ORDER\_NUMBER:

(非空) **varchar**, **PRICE: double**, **CUSTOMER\_ID : (外键) bigint**, 设置级连删除;

```
create table CUSTOMERS(  
    ID bigint not null,  
    NAME varchar(15),  
    AGE int,  
    primary key (ID)  
);  
  
create table ORDERS(  
    ID bigint not null,  
    ORDER_NUMBER varchar(15) not null,  
    PRICE double precision,  
    CUSTOMER_ID bigint,  
    primary key (ID),  
);  
alter table ORDERS add constraint FK_CUSTOMER foreign key (CUSTOMER_ID)  
references CUSTOMERS(ID) on delete cascade;
```

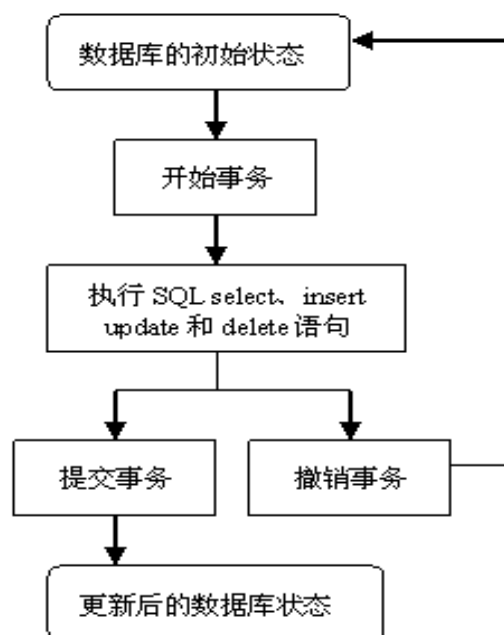
#### 14.delete from table & truncate table table, drop table 的区别

truncate 属于DDL，对整张表进行操作，删除后不可以进行回滚；

delete 属于DML只是将被删除的记录标志为不可用，可以选择性地删除表中的记录，删除后可以回滚。

drop table 属于 DDL 语言，用途是将表从数据库中删除，删除之后不可以恢复。

#### 15.简述数据库事务的生命周期？（可画流程图）





## 16. 如何理解数据库事务

- 在数据库中,所谓事务是指一组逻辑操作单元,使数据从一种状态变换到另一种状态。
- 为确保数据库中数据的一致性,数据的操纵应当是离散的成组的逻辑单元:当它全部完成时,数据的一致性可以保持,而当这个单元中的一部分操作失败,整个事务应全部视为错误,所有从起始点以后的操作应全部回退到开始状态。
- 事务的操作:先定义开始一个事务,然后对数据作修改操作,这时如果提交(COMMIT),这些修改就永久地保存下来,如果回退(ROLLBACK),数据库管理系统将放弃所作的所有修改而回到开始事务时的状态。

## 17. 数据库事务的四大特性

1. 原子性 (Atomicity) 原子性是指事务是一个不可分割的工作单位,事务中的操作要么都发生,要么都不发生。
2. 一致性 (Consistency) 事务必须使数据库从一个一致性状态变换到另外一个一致性状态。
3. 隔离性 (Isolation) 事务的隔离性是指一个事务的执行不能被其他事务干扰,即一个事务内部的操作及使用的数据对并发的其他事务是隔离的,并发执行的各个事务之间不能互相干扰。
4. 持久性 (Durability) 持久性是指一个事务一旦被提交,它对数据库中数据的改变就是永久性的,接下来的其他操作和数据库故障不应该对其有任何影响

## 18. 事务的隔离级别

由隔离性引发的并发问题:

- 对于同时运行的多个事务,当这些事务访问数据库中相同的数据时,如果没有采取必要的隔离机制,就会导致各种并发问题:
  - **脏读:** 对于两个事物 T1, T2, T1 读取了已经被 T2 更新但还没有被提交的字段. 之后,若 T2 回滚, T1 读取的内容就是临时且无效的.
  - **不可重复读:** 对于两个事物 T1, T2, T1 读取了一个字段, 然后 T2 更新了该字段. 之后, T1 再次读取同一个字段, 值就不同了.
  - **幻读:** 对于两个事物 T1, T2, T1 从一个表中读取了一个字段, 然后 T2 在该表中插入了一些新的行. 之后, 如果 T1 再次读取同一个表, 就会多出几行.
- 数据库事务的隔离性: 数据库系统必须具有隔离并发运行各个事务的能力,使它们不会相互影响,避免各种并发问题.
- 一个事务与其他事务隔离的程度称为隔离级别. 数据库规定了多种事务隔离级别,不同隔离级别对应不同的干扰程度,隔离级别越高,数据一致性就越好,但并发性越弱

为解决事务并发性引发的问题数据库定义了四种隔离级别:

隔离级别	描述
READ UNCOMMITTED (读未提交数据)	允许事务读取未被其他事物提交的变更. 脏读, 不可重复读和幻读的问题都会出现
READ COMMITTED (读已提交数据)	只允许事务读取已经被其它事务提交的变更. 可以避免脏读, 但不可重复读和幻读问题仍然可能出现
REPEATABLE READ (可重复读)	确保事务可以多次从一个字段中读取相同的值. 在这个事务持续期间, 禁止其他事物对这个字段进行更新. 可以避免脏读和不可重复读, 但幻读的问题仍然存在.
SERIALIZABLE(串行化)	确保事务可以从一个表中读取相同的行. 在这个事务持续期间, 禁止其他事务对该表执行插入, 更新和删除操作. 所有并发问题都可以避免, 但性能十分低下.

- Oracle 支持的 2 种事务隔离级别: **READ COMMITED**, **SERIALIZABLE**. Oracle 默认的事务隔离级别为: **READ COMMITED**
- Mysql 支持 4 中事务隔离级别. Mysql 默认的事务隔离级别为: **REPEATABLE READ**

## 19. 数据库中如何实现分页简述 mysql 和 oracle 两种数据库

Oracle

```
--每页显示10条数据, 查看第3页的数据 31-40条
select rn ,t2.first_name,t2.salary from
(select rownum rn,t1.* from
(select *
from employees e
order by e.salary desc) t1)t2
where rn>=31 and rn<=40
```

Mysql

```
select * from students order by id limit ?,?
```

## 20. 大数据量下的分页解决方法。

答: 最好的办法是利用 sql 语句进行分页, 这样每次查询出的结果集中就只包含某页的数据内容。再 sql 语句无法实现分页的情况下, 可以考虑对大的结果集通过游标定位方式来获取某页的数据。

## 21. 项目中如何设计数据库? 需要考虑哪些情况?

数据库的设计步骤:

- 1、需求分析: 了解用户的数据需求、处理需求、安全性及完整性要求;
- 2、概念设计: 通过数据抽象, 设计系统概念模型, 一般为 E-R 模型;

- 3、逻辑结构设计：设计系统的模式和外模式,对于关系模型主要是基本表和视图；
- 4、物理结构设计：设计数据的存储结构和存取方法,如索引的设计；
- 5、系统实施：组织数据入库、编制应用程序、试运行；
- 6、运行维护：系统投入运行,长期的维护工作.

**设计时应注意的问题和考虑的情况：**

- 1、规范命名.所有的库名、表名、域名必须遵循统一的命名规则,并进行必要说明,以方便设计、维护、查询.
- 2、控制字段的引用.在设计时,可以选择适当的数据库设计管理工具,以方便开发人员的分布式设计和数据小组的集中审核管理.

3. 数据库设计时就要考虑到效率和优化问题

一开始就要分析哪些表会存储较多的数据量，对于数据量较大的表的设计往往是粗粒度的，也会冗余一些必要的字段，已达到尽量用最少的表、最弱的表关系去存储海量的数据。并且在设计表时，一般都会对主键建立聚集索引，含有大数据量的表更是要建立索引以提高查询性能。对于含有计算、数据交互、统计这类需求时，还要考虑是否有必要采用存储过程。

4. 添加必要的（冗余）字段

像“创建时间”、“修改时间”、“备注”、“操作用户 IP”和一些用于其他需求（如统计）的字段等，在每张表中必须都要有，不是说只有系统中用到的数据才会存到数据库中，一些冗余字段是为了便于日后维护、分析、拓展而添加的

5. 设计合理的表关联

若多张表之间的关系复杂，建议采用第三张映射表来关联维护两张表之间的关系，以降低表之间的直接耦合度。若多张表涉及到大数据量的问题，表结构尽量简单，关联也要尽可能避免。

6. 选择合适的主键生成策略

**22.什么是笛卡尔积？如何产生的？**

**EMPLOYEES (20行)**

EMPLOYEE_ID	LAST_NAME	DEPARTMENT_ID
100	King	90
101	Kochhar	90
...		
202	Fay	20
205	Higgins	110
206	Gietz	110

20 rows selected.

**DEPARTMENTS (8行)**

DEPARTMENT_ID	DEPARTMENT_NAME	LOCATION_ID
10	Administration	1700
20	Marketing	1800
50	Shipping	1500
60	IT	1400
80	Sales	2500
90	Executive	1700
110	Accounting	1700
190	Contracting	1700

8 rows selected.

EMPLOYEE_ID	DEPARTMENT_ID	LOCATION_ID
100	90	1700
101	90	1700
102	90	1700
103	60	1700
104	60	1700
107	60	1700
...		

**笛卡尔集:**  
**20x8=160行**

笛卡尔积 会在下面的条件产生

- 省略连接条件
- 连接条件无效
- 所有表中的所有行相互连接

### 23.数据库中的 BLOB 和 CLOB 是什么

BLOB 全称为二进制大型对象 (Binary Large Object)。它用于存储数据库中的大型二进制对象。可存储的最大大小为 4G 字节

CLOB CLOB 全称为字符大型对象 (Character Large Object)。它与 LONG 数据类型类似，只不过 CLOB 用于存储数据库中的大型单字节字符数据块，不支持宽度不等的字符集。可存储的最大大小为 4G 字节

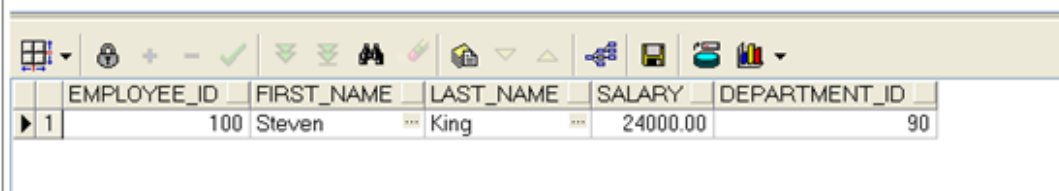
BLOB 和 CLOB 都是大字段类型，BLOB 是按二进制来存储的，而 CLOB 是可以直接存储文字的。其实两个是可以互换的，或者可以直接用 LOB 字段代替这两个。但是为了更好的管理 ORACLE 数据库，通常像图片、文件、音乐等信息就用 BLOB 字段来存储，先将文件转为二进制再存储进去。而像文章或者是较长的文字，就用 CLOB 存储，这样对以后的查询更新存储等操作都提供很大的方便。

当某个字段，要保存的长度太长时，varchar2 最大 4000，但是如果需要的字段长度大于 4000 应该怎么办，采取什么办法呢，这时候就用 clob。

比如某个字段保存的内容是，类似于 word 的文本，那么该字段一般用 clob 来存储。

### 24 如何判断数据库中某个字段是空值

```
select employee_id,first_name, last_name, salary,department_id
from employees
where manager_id is null
```



	EMPLOYEE_ID	FIRST_NAME	LAST_NAME	SALARY	DEPARTMENT_ID
1	100	Steven	King	24000.00	90

### 25.SQL 注入攻击

### 26.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。

## Js&xml

### 1、Js 中有没有数据类型

有：包括以下内容

字符串(String)、数字(Number)、布尔(Boolean)、数组(Array)、对象(Object)、Null、Undefined

### 2、xml 有哪些解析技术?区别是什么?

答:有 DOM, SAX, dom4j 等

DOM:处理大型文件时其性能下降的非常厉害。这个问题是由 DOM 的树结构所造成的，这种结构占用的内存较多，而且 DOM 必须在解析文件之前把整个文档装入内存，适合对 XML 的随机

访问 SAX: 不现于 DOM, SAX 是事件驱动型的 XML 解析方式。它顺序读取 XML 文件, 不需要一次全部装载整个文件。当遇到像文件开头, 文档结束, 或者标签开头与标签结束时, 它会触发一个事件, 用户通过在其回调事件中写入处理代码来处理 XML 文件, 适合对 XML 的顺序访问

### 3、你在项目中用到了 xml 技术的哪些方面?如何实现的?

答:用到了数据存贮, 信息配置两方面。在做数据交换平台时, 将不能数据源的数据组装成 XML 文件, 然后将 XML 文件压缩打包加密后通过网络传送给接收者, 接收解密与解压缩后再同 XML 文件中还原相关信息进行处理。在做软件配置时, 利用 XML 可以很方便的进行, 软件的各种配置参数都存贮在 XML 文件中。

### 4.XML 文档定义有几种形式? 它们之间有何本质区别? 解析 XML 文档有哪几种方式?

a: 两种形式 dtd schema, b:本质区别:schema 本身是 xml 的, 可以被 XML 解析器解析(这也是从 DTD 上发展 schema 的根本目的), c:有 DOM, SAX, STAX 等

DOM:处理大型文件时其性能下降的非常厉害。这个问题是由 DOM 的树结构所造成的, 这种结构占用的内存较多, 而且 DOM 必须在解析文件之前把整个文档装入内存, 适合对 XML 的随机访问

SAX:不现于 DOM, SAX 是事件驱动型的 XML 解析方式。它顺序读取 XML 文件, 不需要一次全部装载整个文件。当遇到像文件开头, 文档结束, 或者标签开头与标签结束时, 它会触发一个事件, 用户通过在其回调事件中写入处理代码来处理 XML 文件, 适合对 XML 的顺序访问

## Jsp&servlet

### 1、三个 Statment 区别, 用法

**statement**对象作为最基本的数据操作对象, 可以应用于几乎所有的数据库, 但是由于运行时使用的是字符串连接技术, 所以存在安全隐患。使用 Statement 对象。在对数据库只执行一次性存取的时候, 用 Statement 对象进行处理。PreparedStatement 对象的开销比 Statement 大, 对于一次性操作并不会带来额外的好处。statement 每次执行 sql 语句, 相关数据库都要执行 sql 语句的编译

**preparedstatement:** 是预编译的, preparedstatement 支持批处理, 对于批量处理可以大大提高效率。也叫 JDBC 存储过程, 在语句执行之前, 向数据库发送类似于公式一样的模板, 其中使用了替换变量, 从而提高了数据存储的安全性, 但这个数据操作对象不是效率最高的。可以应用于绝大多数数据库

**callablestatement:** 效率和安全性最高的数据操作对象, 但是兼容性是最差的。因为这个对象是用来调用数据库当中的存储过程的, 不是所有的数据库都支持存储过程。



## 2 什么是 Cookie

答: cookie 是用来将信息永久或临时保存数据到客户端。在使用的时候要设定有效期和有效路径。如果不设置有效期, 这个 cookie 就是会话性的 cookie, 会话结束后 cookie 就被删除。如果不设置有效路径, 那么 cookie 只在当前 (创建 cookie) 路径有效。

## 3.什么是 Session

Session 是存放在服务器端的, 类似于 Session 结构来存放用户数据, 当浏览器 第一次发送请求时, 服务器自动生成了一个 Session 和一个 Session ID 用来唯一标识这个 Session, 并将其通过响应发送到浏览器。当浏览器第二次发送请求, 会将前一次服务器响应中的 Session ID 放在请求中一并发送到服务器上, 服务器从请求中提取出 Session ID, 并和保存的所有 Session ID 进行对比, 找到这个用户对应的 Session。

一般情况下, 服务器会在一定时间内 (默认 30 分钟) 保存这个 Session, 过了时间限制, 就会销毁这个 Session。在销毁之前, 程序员可以将用户的一些数据以 Key 和 Value 的形式暂时存放在这个 Session 中。当然, 也有使用数据库将这个 Session 序列化后保存起来的, 这样的好处是没了时间的限制, 坏处是随着时间的增加, 这个数据 库会急速膨胀, 特别是访问量增加的时候。一般还是采取前一种方式, 以减轻服务器压力

## 4.Cookie 与 session 的区别是:

首先要介绍 cookie 和 session 是什么。

然后是区别:

- a) cookie 数据存放在客户的浏览器上, session 数据放在服务器上。
- b) cookie 不是很安全, 别人可以分析存放在本地的 COOKIE 并进行 COOKIE 欺骗考虑到安全应当使用 session。
- c) session 会在一定时间内保存在服务器上。当访问增多, 会比较占用你服务器的性能  
考虑到减轻服务器性能方面, 应当使用 COOKIE。
- d) 单个 cookie 保存的数据不能超过 4K, 很多浏览器都限制一个站点最多保存 20 个 cookie。

## 5.Get 请求和 Post 请求区别

- 1、Get 是用来从服务器上获得数据, 而 Post 是用来向服务器上传递数据。
- 2、Get 将表单中数据的按照 variable=value 的形式, 添加到 action 所指向的 URL 后面, 并且两者使用 “?” 连接, 而各个变量之间使用 “&” 连接; Post 是将表单中的数据放在 form 的数据体中, 按照变量和值相对应的方式, 传递到 action 所指向 URL。
- 3、Get 是不安全的, 因为在传输过程, 数据被放在请求的 URL 中, 而如今现有的很多服务器、代理服务器或者用户代理都会将请求 URL 记录到日志文件中, 然后放在某个地方, 这样就可能会有一些隐私的信息被第三方看到。另外, 用户也可以在浏览器上直接看到提交的数据, 一些系统内部消息将会一同显示在用户面前。Post 的所有操作对用户来说都是不可见的。
- 4、Get 传输的数据量小, 这主要是因为受 URL 长度限制; 而 Post 可以传输大量的数据, 所以在上传文件只能使用 Post (当然还有一个原因, 将在后面的提到)。
- 5、Get 限制 Form 表单的数据集的值必须为 ASCII 字符; 而 Post 支持整个 ISO10646 字符集。
- 6、Get 是 Form 的默认方法。

## 6. servlet 生命周期

Servlet 生命周期可以分为三个阶段：1、初始化阶段 2、响应客户阶段 3、终止阶段  
在 `javax.servlet.Servlet` 接口中定义了三个方法 `init()`,`service()`,`destroy()`,它们将分别在 Servlet 不同的阶段被调用。

### Servlet 的初始化阶段

在下列时刻 Servlet 容器装载 Servlet：1、servlet 容器启动后会自动装载某些实现它只需要在 web.XML 文件中的 `<Servlet></Servlet>` 之间添加如下代码：

```
<loadon-startup>1</loadon-startup> Servlet
```

2、在 servlet 容器启动后，客户首次向 Servlet 容器发出请求。3、Servlet 的文件被更新后，重新装载 Servlet。

Servlet 被装载后，Servlet 容器创建一个 Servlet 实例并且调用 Servlet 的 `init()` 方法进行初始化，在 Servlet 生命周期中 `init()` 方法只会被调用一次。

### Servlet 的响应客户请求阶段

对于到达 Servlet 容器的客户请求，Servlet 会创建针对这个请求的 `HttpRequest` 对象和 `HttpResponse` 对象，然后调用 Servlet 的 `service` 方法。`Service` 方法从 `ServletRequest` 对象获得客户请求信息、处理该请求，并通过 `ServletResponse` 对象向客户端返回响应结果。

### Servlet 的终止阶段

当 Web 应用被终止，或 Servlet 容器终止运行，或 Servlet 容器重新装载 Servlet 的新实例时，Servlet 容器会调用 Servlet 的 `destory` 方法。在 `destory` 方法中，可以释放 Servlet 所占用的资源。

## 7、Servlet 和 Jsp 的区别？

答：Servlet 是直接执行的文件。

Jsp 是要被通过编译形成 Servlet 后才执行。

在 MVC 框架当中，jsp 是用来显示和取得数据的，而 servlet 是作为控制器工作的。JSP 在显示上有很大的优势，但是在控制和业务处理上有缺陷。而 servlet 在程序控制上有很大的优势，但是在显示和取得数据方面远远不如 jsp。

## 8. JSP 九大隐含对象 （也叫做内置对象、内建对象等等）

1) request 表示 `HttpServletRequest` 对象。用户端请求，此请求会包含来自 GET/POST 请求的参数。它包含了有关浏览器请求的信息，并且提供了几个用于获取 cookie, header, 和 session 数据的有用的方法。

2) response 表示 `HttpServletResponse` 对象，网页传回用户端的回应，并提供了几个用于设置送回浏览器的响应的方法（如 cookies, 头信息等）

3) out 对象是 `javax.jsp.JspWriter` 的一个实例，用来传送回应的输出，并提供了几个方法使你能用于向浏览器回送输出结果。

4) pageContext 表示一个 `javax.servlet.jsp.PageContext` 对象。它是用于方便存取各种范围的

名字空间、servlet 相关的对象的 API，并且包装了通用的 servlet 相关功能的方法。

5) session 表示一个请求的 javax.servlet.http.HttpSession 对象。Session 可以存贮用户的状态信息

6) applicaton 表示一个 javax.servle.ServletContext 对象。这有助于查找有关 servlet 引擎和 servlet 环境的信息

7) config 表示一个 javax.servlet.ServletConfig 对象。该对象用于存取 servlet 实例的初始化参数。

8) page 表示从该页面产生的一个 servlet 实例

9) exception 针对错误网页，未捕捉的例外（异常）

### 9.jsp 中的四大共享范围（或四大作用域）

答：a、page 当前页面。

b、session 在当前用户的会话范围内有效。

c、request 从上一页到下一个页面。

d、application 在整个服务器运行期间，在服务器内有效，所有的访问者都可以使用到 application 范围内的对象。

会话作用域 ServletsJSP 页面描述

### 10.MVC——如何理解 MVC

MVC 是 Model—View—Controller 的简写。

"Model" 代表的是应用的业务逻辑（通过 JavaBean，EJB 组件实现），

"View" 是应用的表示面（由 JSP 页面产生），

"Controller" 是提供应用的处理过程控制（一般是一个 Servlet），通过这种设计模型把应用逻辑，处理过程和显示逻辑分成不同的组件实现。这些组件可以进行交互和重用。

Mvc 的优势：

MVC 式的出现不仅实现了功能模块和显示模块的分离，同时它还提高了应用系统的可维护性、可扩展性、可移植性和组件的可复用性

效率提高了，三层是严格的调用关系，传输或是读取数据的时候，需要经过层层调用，才能获取到数据。效率大大降低了。而 View 和 Model 可以进行通信，传输速度加快。

解耦合，在 Controller 的统一管理下，用户和数据操作完全隔离。

灵活性大大提高，Controller 可以管理多个 View，这样当 View 发生改变时，可以随意的更换。

### 11、JSP 中动态 INCLUDE 与静态 INCLUDE 的区别？

1) 动态 INCLUDE（也叫做 include 动作标签）用<jsp:include>动作实现，它总是会检查所含文件中的变化，适合用于包含动态页面，并且可以带参数。它引用的运行的结果，在编译过程中不参与当前页的编译，比如 a.jsp 引用了 b.jsp，那么我们首次访问 a.jsp 的时候两个 jsp 都将被编译。如果在程序运行期间，我们修改了 b.jsp，那么只有 b.jsp 会被编译。

2) 静态 INCLUDE (也叫做 include 指令标签) 用 include 伪码实现, 定不会检查所含文件的变化, 适用于包含静态页面引用的被引用文件的代码。如果使用 include 指令引入了其它 JSP 页面, 那么 JSP 引擎将把这两个 JSP 翻译成一个 servlet。

## 12. Forward 和 SendRedirect 区别 (请求转发和重定向的区别)

- 1) 范围不同: RequestDispatcher.forward 方法只能将请求转发给同一个 WEB 应用中的组件; 而 HttpServletResponse.sendRedirect 方法还可以重定向到同一个站点上的其他应用程序中的资源, 甚至是使用绝对 URL 重定向到其他站点的资源。
- 2) "/" 含义不同: 如果传递给 HttpServletResponse.sendRedirect 方法的相对 URL 以 "/" 开头, 它是相对于整个 WEB 站点的根目录; 如果创建 RequestDispatcher 对象时指定的相对 URL 以 "/" 开头, 它是相对于当前 WEB 应用程序的根目录。
- 3) 地址栏的变化: 调用 HttpServletResponse.sendRedirect 方法重定向的访问过程结束后, 浏览器地址栏中显示的 URL 会发生改变, 由初始的 URL 地址变成重定向的目标 URL; 调用 RequestDispatcher.forward 方法的请求转发过程结束后, 浏览器地址栏保持初始的 URL 地址不变。
- 4) 请求次数的不同: 请求转发是一次请求, 一次响应, 重定向是两次请求, 两次响应。HttpServletResponse.sendRedirect 方法对浏览器的请求直接作出响应, 响应的结果就是告诉浏览器去重新发出对另外一个 URL 的访问请求; RequestDispatcher.forward 方法在服务器端内部将请求转发给另外一个资源, 浏览器只知道发出了请求并得到了响应结果, 并不知道在服务器程序内部发生了转发行为。

## 13. jsp 有哪些动作? 作用分别是什么?

答: JSP 共有以下 6 种基本动作

jsp:include: 在页面被请求的时候引入一个文件。

jsp:useBean: 寻找或者实例化一个 JavaBean。

jsp:setProperty: 设置 JavaBean 的属性。

jsp:getProperty: 输出某个 JavaBean 的属性。

jsp:forward: 把请求转到一个新的页面。

jsp:plugin: 根据浏览器类型为 Java 插件生成 OBJECT 或 EMBED 标记

## 14. 如何在 jsp 页面中设置/取得 JavaBean 中的属性值?

答: 设置属性值 <jsp:setProperty name="haha", property="haha 的属性" value="变量值"/>

<jsp:setProperty name="haha", property="\*" /> 获取从上一个表单中提交过来的, 与 Bean 中变量名字相同的所有属性。

取得属性值: <jsp:getProperty name="haha" property="bean 的属性"/>

相当于: <%=getA()%>

## 15. JSP 和 Servlet 有哪些相同点和不同点, 他们之间的联系是什么?

JSP 是 Servlet 技术的扩展, 本质上是 Servlet 的简易方式, 更强调应用的外表表达。JSP 编

译后是"类 servlet"。

Servlet 和 JSP 最主要的不同点在于,Servlet 的应用逻辑是在 Java 文件中,并且完全从表示层中的 HTML 里分离开来。而 JSP 的情况是 Java 和 HTML 可以组合成一个扩展名为.jsp 的文件。JSP 侧重于视图,Servlet 主要用于控制逻辑。

## 16、四种会话跟踪技术

**page** 是代表与一个页面相关的对象和属性。一个页面由一个编译好的 Java servlet 类(可以带有任何的 include 指令,但是没有 include 动作)表示。这既包括 servlet 又包括被编译成 servlet 的 JSP 页面

**request** 是代表与 Web 客户机发出的一个请求相关的对象和属性。一个请求可能跨越多个页面,涉及多个 Web 组件(由于 forward 指令和 include 动作的关系)

**session** 是代表与用于某个 Web 客户机的一个用户体验相关的对象和属性。一个 Web 会话可以也经常跨越多个客户机请求

**application** 是代表与整个 Web 应用程序相关的对象和属性。这实质上是跨越整个 Web 应用程序,包括多个页面、请求和会话的一个全局作用域

## 17.如何理解 Servlet 中的过滤器

Servlet 过滤器的基本原理

在 Servlet 作为过滤器使用时,它可以对客户请求进行处理。处理完成后,它会交给下一个过滤器处理,这样,客户的请求在过滤链里逐个处理,直到请求发送到目标为止。例如,某网站里有提交“修改的注册信息”的网页,当用户填写完修改信息并提交后,服务器在进行处理时需要做两项工作:判断客户端的会话是否有效;对提交的数据进行统一编码。这两项工作可以在由两个过滤器组成的过滤链里进行处理。当过滤器处理成功后,把提交的数据发送到最终目标;如果过滤器处理不成功,将把视图派发到指定的错误页面。

## 18、ServletConfig 对象和 ServletContext 对象作用分别是什么?有什么区别?

- WEB 容器在启动时,它会为每个 WEB 应用程序都创建一个对应的 ServletContext 对象,它代表当前 web 应用。
- ServletConfig 对象中维护了 ServletContext 对象的引用,开发人员在编写 servlet 时,可以通过 ServletConfig.getServletContext 方法获得 ServletContext 对象。
- 由于一个 WEB 应用中的所有 Servlet 共享同一个 ServletContext 对象,因此 Servlet 对象之间可以通过 ServletContext 对象来实现通讯。ServletContext 对象通常也被称之为 context 域对象。可以实现多个 servlet 共享数据、获取获取 WEB 应用的初始化参数。实现 Servlet 的转发。利用 ServletContext 对象读取资源文件。

## 19.过滤器和拦截器的区别?他们之间分别创建几个实例?



**拦截器**：拦截器就是拦截 Action 在 Action 执行先执行拦截器 在 Action 执行之后再回到拦截器 中执行相关的代码，是在面向切面编程的就是在你的 service 或者一个方法，前调用一个方法，或者在方法后调用一个方法比如动态代理就是拦截器的简单实现，在你调用方法前打印出字符串（或者做其它业务逻辑的操作），也可以在你调用方法后打印出字符串，甚至在你抛出异常的时候做业务逻辑的操作。

**过滤器**：是在 java web 中，你传入的 request, response 提前过滤掉一些信息，或者提前设置一些参数，然后再传入 servlet 或者 struts 的 action 进行业务逻辑，比如过滤掉非法 url（不是 login.action 的地址请求，如果用户没有登陆都过滤掉），或者在传入 servlet 或者 struts 的 action 前统一设置字符集，或者去除掉一些非法字符。

区别：

- 1、拦截器是基于 java 的反射机制的，而过滤器是基于函数回调
- 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的请求起作用
- 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，拦截器可以多次被调用，而过滤器只在容器初始化时调用一次

## 20、项目中如何解决编码集和乱码的？

### 一、表单提交时出现乱码：

**1、客户端的 get 请求：** get 提交时， 容器以容器的编码来编码 如果用的 tomcat 默认的编码是 iso-8859-1 在 server.xml 里面设置编码 或者 下面代码如

```
String name = request.getParamter("name");  
String strName = new String(name.getBytes("iso-8859-1"), "UTF-8");
```

### 2、客户端的 post 请求

对于客户端的 post 请求来说，处理乱码的问题就比较简单了，因为请求的数据时作为请求体的一部分传递给服务器的，所以只要修改请求内的编码就行了。只要在服务器端的最开始处将请求的数据设置为“UTF-8”就行了，

输入如下语句： request.setCharacterEncoding("UTF-8");

### 二、超链接时出现乱码

对于超链接来说，它实际上是向服务器端发送了一个请求，而它发出的请求是属于 get 请求所以对于超链接的乱码来说，它处理乱码的方式和表单的 get 请求出现乱码的方式是一样的。

### 三、重定向时出现乱码

有时写上 response 的 sendRedirect 方法进行重定向时也会出现乱码，重定向时实际上也是向服务器发送了一个请求，所以解决乱码的方法和和上面是一样的。

### 四、浏览器版本低导致的乱码

在 Java.NET 包中提供了 URLEncoder 类和 URLDecoder 类，这两个类又分别提供了 encode 和 decode 两个静态方法，分别用于进行编码和解码。我们将要传递的中文参数进行编码之后，在传递给服务器，服务器解码之后，就可以显示中文了。



进行编码: `URLEncoder.encode(stuname," UTF-8" )`

传递给服务器: `<a href="/1.jsp?stuname<%=stuname%>" >传递</a>`

进行解码: `URLDecoder.decode(stuname," UTF-8" )`

### 五、返回浏览器显示的乱码

`setCharacterEncoding("UTF-8")`、`setContentType("text/html;charset=UTF-8")`等方法来指定 `getWriter` 方法返回的 `PrintWriter` 对象所使用的字符集编码, 所以我们在写 `Servlet` 程序中, 在调用 `getWriter` 方法之前设置这些方法的值。

### 六、修改 Tomcat 的编码

在 Tomcat 的配置文件 `server.xml` 中找到修改 Tomcat 的端口的地方, 在其内部加入 `URIEncoding` 属性, 设置为和你的项目中所设的编码一样的值, 这里全部都是 UTF-8。

### 七、通过过滤器解决全站乱码

## 21 什么是 ajax, 为什么要使用 Ajax (请谈一下你对 Ajax 的认识)

什么是 ajax:

AJAX 是 “Asynchronous JavaScript and XML” 的缩写。他是指一种创建交互式网页应用的网页开发技术。 Ajax 包含下列技术:

- 1) 基于 web 标准 (standards-based presentation) XHTML+CSS 的表示;
- 2) 使用 DOM (Document Object Model) 进行动态显示及交互;
- 3) 使用 XML 和 XSLT 进行数据交换及相关操作;
- 4) 使用 XMLHttpRequest 进行异步数据查询、检索;
- 5) 使用 JavaScript 将所有的东西绑定在一起。

为什么要用 ajax: Ajax 应用程序的优势在于:

1. 通过异步模式, 提升了用户体验
2. 优化了浏览器和服务器之间的传输, 减少不必要的数据往返, 减少了带宽占用
3. Ajax 引擎在客户端运行, 承担了一部分本来由服务器承担的工作, 从而减少了大用户量下的服务器负载。

## 22. Ajax 的最大的特点是什么。

Ajax 可以实现动态不刷新 (局部刷新), 就是能在不更新整个页面的前提下维护数据。这使得 Web 应用程序更为迅捷地回应用户动作, 并避免了在网络上发送那些没有改变过的信息。

Ajax 提供与服务器异步通信的能力, 从而使用户从请求/响应的循环中解脱出来。借助于 Ajax, 可以在用户单击按钮时, 使用 JavaScript DHTML 立即更新 UI, 并向服务器发出异步请求, 以执行更新或查询数据库。当请求返回时, 就可以使用 JavaScript 和 CSS 来相应地更新 UI, 而不是刷新整个页面。

### 23. Ajax 有几种返回值

Ajax 请求总共有八种 Callback

onUninitialized → (未初始化) 还没有调用 send() 方法

onLoading → (载入) 已调用 send() 方法, 正在发送请求

onLoaded → (载入完成) send() 方法执行完成, 已经接收到全部响应内容

onInteractive → (交互) 正在解析响应内容

onComplete → (完成) 响应内容解析完成, 可以在客户端调用了

onSuccess → 数据成功返回到客户端

onFailure → 失败

onException → 异常

### 24. Ajax 由几部分组成? 其核心是什么?

### 25. 在项目中是否有运用 ajax, 说说你用过那些 ajax 技术

### 26. JQuery 的优势?

### 27. web 中常见的常见 HTTP 状态码

200 OK

301 Moved Permanently

302 Found

304 Not Modified

307 Temporary Redirect

400 Bad Request

401 Unauthorized

403 Forbidden

404 Not Found

410 Gone

500 Internal Server Error

501 Not Implemented

200 OK

一切正常, 对 GET 和 POST 请求的应答文档跟在后面

301 Moved Permanently

客户请求的文档在其他地方, 新的 URL 在 Location 头中给出, 浏览器应该自动地访问新的 URL。

302 Found

类似于 301, 但新的 URL 应该被视为临时性的替代, 而不是永久性的。

### 304 Not Modified

客户端有缓冲的文档并发出了一个条件性的请求(一般是提供 **If-Modified-Since** 头表示客户只想比指定日期更新的文档)。服务器告诉客户, 原来缓冲的文档还可以继续使用。

### 307 Temporary Redirect

和 302 (**Found**) 相同。许多浏览器会错误地响应 302 应答进行重定向, 即使原来的请求是 **POST**, 即使它实际上只能在 **POST** 请求的应答是 303 时才能重定向。由于这个原因, **HTTP 1.1** 新增了 307, 以便更加清除地区分几个状态代码: 当出现 303 应答时, 浏览器可以跟随重定向的 **GET** 和 **POST** 请求; 如果是 307 应答, 则浏览器只能跟随对 **GET** 请求的重定向。

### 400 Bad Request

请求出现语法错误。

### 401 Unauthorized

客户试图未经授权访问受密码保护的页面。应答中会包含一个 **WWW-Authenticate** 头, 浏览器据此显示用户名/密码对话框, 然后在填写合适的 **Authorization** 头后再次发出请求。

### 403 Forbidden

资源不可用。

### 404 Not Found

无法找到指定位置的资源

### 410 Gone

所请求的文档已经不再可用, 而且服务器不知道应该重定向到哪一个地址。它和 404 的不同在于, 返回 407 表示文档永久地离开了指定的位置, 而 404 表示由于未知的原因文档不可用。

### 500 Internal Server Error

服务器遇到了意料不到的情况, 不能完成客户的请求

### 501 Not Implemented

服务器不支持实现请求所需要的功能。例如, 客户发出了一个服务器不支持的 **PUT** 请求

## 28. 关于 web 页面性能的优化

我们专注的 web 性能指标有以下

#### 1、页面加载时间

从页面开始加载到页面 **onload** 事件触发的时间。一般来说 **onload** 触发代表着直接通过 **HTML** 引用的 **CSS**, **JS**, 图片资源已经完全加载完毕。

#### 2、全部页面加载时间

全部页面载入时间指从最初启动浏览开始, 直到所有元素都被加载完成后, 在 2 秒后仍然没有网络活动的时间。

#### 3、首字节时间

从开始加载到收到服务器返回数据的第一字节的时间

#### 4、使用长连接

连接视图展现了页面加载过程中创建的(**keepalive**)连接, 以及通过每个连接所加载的资源。

- 5、DNS 时间 域名解析所需要的时间
- 6、TCP 时间 端建立连接的时间
- 7、HTTP 网页打分 渲染、下载速度、页面流畅度
8. 并发用户数
- 9 吞吐量 指的是在一次性能测试过程中网络上传输的数据量的总和  
吞吐量/传输时间,就是吞吐率.
- 10.资源利用率

### 优化的建议:

#### 1. 管理“页面膨胀

#### 2.进行图像优化

进行图像优化是提升性能最简单的一种方法,它可以使页面加载更快。为了更有效的完成图像渲染,图像必须经过压缩和整合、图像的尺寸和格式必须经过仔细调整,图像质量也必须经过优化,这样才可以依据图像的重要性进行区别化的加载处理。

#### 3.控制第三方脚本

解决办法是延迟第三方脚本的加载,将其放在关键页面内容之后进行加载,更为理想的情况是放在页面 onLoad 事件之后加载,

#### 4.在进行响应式 Web 设计时兼顾性能

响应式设计建立在样式表和 JavaScript 之上。然而,低效的 CSS 和 JS 所带来的性能问题远远大于其设计优势给我们带来的好处。样式表应当放在 HEAD 文档中,用以实现页面的逐步渲染。然而,样式表却经常出现在页面其它位置,这就阻碍了页面的渲染速度。换句话说,JavaScript 文件应当放在页面底部或在关键内容加载完成之后再被加载才是合理的处理方式。

#### 5.实时监控性能

实时用户监控(RUM)工具可以从真实用户的角度实时获取、分析并记录网站的性能和可用性。

## 软件测试

### 1.测试生命周期,测试过程分为几个阶段,以及各个阶段的含义

解答：软件测试生命周期一般包括 6 个阶段：1) 计划 2) 分析，3) 设计，4) 构建，5) 测试周期，6) 最后测试和实施，

- 1) 计划: 产品定义阶段
- 2). 分析:外部文档阶段
- 3). 设计:文档架构阶段
- 4). 构建:单元测试阶段
- 5). 测试周期:错误修正,重复系统测试阶段
- 6). 最后的测试和实施:代码冻结阶段

**2.项目完成后做过哪些方便的测试？**