你必须知道的 261 个 Java 语言问题

1、 Java 语言的运行机制：

Java 既不是编译型语言也不是解释型语言，它是编译型和解释型语言的结合体。首先采用通用的 java 编译器将 Java 源程序编译成为与平台无关的字节码文件（class 文件），然后由 Java 虚拟机对字节文件解释执行

2、 什么是 JVM？有什么作用？工作机制如何？

JVM 是一个虚构出来的计算机，可在实际的计算机上模拟各种计算机功能，JVM 有自己完善的硬件架构，如：处理器、堆栈、寄存器等，还有相应的指令系统。

JVM 是由 Java 字节码执行的引擎，为 java 程序的执行提供必要的支持，它还能优化 java 字节码，使之转换成效率更高的机器指令。JVM 屏蔽了与具体操作系统平台相关的信息，从而实现了 Java 程序只需要生成在 JVM 上运行的字节码文件，就可以在多种平台上不加修改地运行。JVM 中类的装载是由类加载器（CLassLoader）和它的子类来实现的。CLassLoader 是 java 运行时一个重要的系统组件，负责在运行时查找和装入类文件的类。

操作系统装入 JVM 是通过 JDK 中的 java.exe 来实现，主要通过以下几个步骤完成：

a、 创建 JVM 装载环境和配置
b、 装载 jvm.dll
c、 初始化 jvm.dll
d、 调用 JNIEnv 实例装载并处理 class 类
e、 运行 Java 程序

3、 为何在 JDK 安装路径下存在两个 JRE

第一个：C:\Program Files\Java\jdk1.6.0_13\jre；第二个：C:\Program Files\Java\jre6；第一个 JRE 用于为 JDK 自带的开发工具提供运行环境，第二 JRE 用于为开发者编写的代码提供运行环境。

4、 this：用于解决变量的命名冲突和不确定性问题而引入的关键字。使用情况：

a、 返回调用当前方法的对象的引用
b、 在构造方法中调用当前类中的其他构造方法
c、 当方法参数名和成员变量名（字段更专业，java 专有名词）相同时，用于区分参数名和成员变量名

5、 super:代表父类的实例，在子类中，使用 super 可以调用其父类的方法、属性和构造方法。使用情况：

a、 调用父类中的构造方法；
b、 调用父类中的方法和属性：super.xxx();

6、 static：可修饰方法、属性、自由块、内部类。使用 static 修饰这些成员时，可以理解成这些成员与类相关，通过“类名.成”的形式调用；没有 static 修饰可以理解成这些成员与对象相关，需要通过“对象名.成员”的形式调用。

a、 static 不能修饰构造方法
b、 在 static 修饰的方法中，不能调用没有 static 修饰的方法和属性，也不能使用 this 和 super 关键字。

c、 static 修饰属性时，这个静态属性还具有一个特性，那就是该属性被多个当前类对象共享。

d、 static 修饰自由块，只要类被加载，即使没有创建对象，也将被执行。此外静态自由块无论创建多少个对象，仅执行一次。

7、 成员变量和局部变量区别：局部变量指方法体内部定义的变量，作用域只在方法块内部有效。局部变量在使用时，必须初始化。成员变量指在类中定义的变量，也就是属性，作用域是在整个类中有效。成员变量在定义时可以不指定初始值，系统可以按默认原则初始化。以下几个方面：

- public protect private static 等修饰符可用于修饰成员变量，但不能修饰局部变量。两者都可以使用 final 修饰。
- 成员变量存储在堆内存中，局部变量存储在栈内存中。
- 作用域看上面
- 初始化看上面

8、 简单类型变量和引用类型变量存储机制：简单类型变量是直接在栈内存中开辟存储空间存储变量值。引用类型变量是由引用空间和存储空间两部分构成，引用空间在栈内存中，存储空间在堆内存中，存储空间负责存储变量值，引用空间负责存放存储空间的首地址。引用变量中存放的是地址值，通过地址值可以定义存储位置并修改存储信息。当变量与变量之间赋值时，引用类型变量和简单变量都属于值传递，不同的是简单变量传递的是内容本身，而引用变量传递的却是引用地址。

9、 新式样 for 循环：

```
String[] arr={“ddd”,” dda”,” dds”};
for(String s:arr){
    System.out.println(s);
}(type to from)
```

10、 内部类：

优点：

- 内部类对象能访问其所处类的私有属性和方法
- 内部类能够隐藏起来，不被同一个包中的其他类访问。
- 匿名内部类可以方便地用在回调方法中

特征：

a、 内部类可以声明为抽象类，因此可以被其他的内部类继承，也可以声明为 final 的

b、 和外部类不同，内部类可以声明为 private 和 protected，外部类只能用 public 和 default

c、 内部类可以声明为 static 的，但此时就不能再使用外层封装类的非 static 的成员变量

d、 非 static 的内部类中的成员不能声明为 static 的，只有在顶层类或 static 的内部类中才可以声明 static 成员。

11、 抽象类和接口的不同：定义格式不同、使用方式不同、设计理念不同（is a ,like a）、使用关系

12、 Java 中动态绑定：

将一个方法调用同一个方法主体连接到一起称为“绑定”。如果在程序运行之前执行绑定，由编译器决定方法调用的程序，称为“早期绑定”或“静态绑定”。

如果绑定过程在程序运行期间进行，以对象的类型为基础，则称为“后期绑定”或“动态绑定”。

如果一种语言实现了后期绑定，同时必须提供一些机制，可以在运行期间判断对象的实际类型，并分别调用适当的方法，即编译器此时依然不知道对象的类型，但方法调用机制能够自己去调查，找到正确的方法主体。Java 方法的执行主要采用动态绑定技术，在程序运行时，虚拟机将调用对象实际类型所限定的方法。

Java 方法在调用过程中主要经历了以下过程。

- 编译器查看对象变量的声明类型和方法名，通过声明类型找到方法列表。
- 编译器查看调用方法时提供的参数类型。
- 如果方法由 private,static,final 修饰或者是构造器，编译器就可以确定调用哪一种方法，即采用静态绑定技术。如果不是上述情况，就使用动态绑定技术，执行后续过程。
- 虚拟机提取对象的实际类型的方法表。
- 虚拟机搜索方法签名。
- 调用方法。

13、 创建类的对象时，类中各成员的执行顺序：

属性、方法、构造方法和自由块都是类中的成员，在创建对象时，各成员的执行顺序如下：

- 父类静态成员和静态初始化块，按在代码中出现的顺序依次执行。
- 子类静态成员和静态初始化块，按在代码中出现的顺序依次执行。
- 父类实例成员和实例初始化块，按在代码中出现的顺序依次执行。
- 执行父类构造方法
- 子类实例成员和实例初始化块，按在代码中出现的顺序依次执行。
- 执行子类构造方法

14、 静态初始化块和非静态初始化块：

静态初始化块比非静态初始化块执行要早，而且静态初始化块只执行一次，非静态的初始化块可执行多次。静态初始化块的的执行时机需要注意，它是在类加载器第一次加载该类时调用，不一定非要创建对象才触发，如果使用“类.静态方法”也会执行静态方法。

15、 Java 中异常处理的方式：try-catch-finally；throws

Java 数据库操作

16、 JDBC：为开发人员提供了一套标准的 API，都是由 Java 语言编写的类和接口，可用于连接数据库和执行 SQL 语句，主要有：

1、 DriverManager:管理一组 JDBC 驱动程序的基本程序
2、 Connection:Java 程序与特定数据库的连接
3、 Statement:用于执行静态 SQL 语句并返回它所生成结果的对象
4、 PreparedStatement:表示预编译的 SQL 语句的对象（是 Statement 接口的子接口，能完成 Statement 所能实现的功能，PreparedStatement 表示预编译的 SQL 语句的对象，SQL 语句被预编译并且存储在 PreparedStatement 对象中，然后可以使用此对象高效地多次执行该语句。）

5、 CallableStatement:用于执行 SQL 存储过程的接口

6、 ResultSet:表示数据库查询的结果集

17、 Java 与数据库的连接方式：

在 Java 访问数据库时，Java 程序首先使用 DriverManager 类载入指定的驱动程序，然后使用 JDBC API 与 DriverManger 类交互，完成数据库的增加、删除、修改和查询操作。

1、 JDBC-ODBC bridge plus ODBC driver：桥驱动

2、 Native-API partly-Java driver：本地 API 驱动

3、 Pure Java Driver for Database Middleware：网络协议驱动

4、 Direct-to-Database Pure Java Driver：本地协议驱动

18、 JDBC 在对各种不同数据库进行连接时，只需要使用不同的驱动包，传递不同的参数。

1、 注册驱动：ClassName(“”)

2、 获取连接:Conncetion conn=DriverManager.getConnection(url,”username”,”password”)

3、 获取 SQL 执行器:Statement stmt=conn.createStatement();

4、 执行查询的 SQL 语句:ResultSet rs=stmt.executeQuery(“”);

5、 遍历结果集中的每一行记录:while(rs.next());

6、 获取结果中的 XXX 字段的值:rs.getInt(“”);

19、 Statement 和 PreparedStatement 区别：

1、 都可以执行 SQL 语句实现对数据表的操作。

2、 Statement 用于执行静态 SQL 语句并返回它所生成结果的对象。Statement 在执行 SQL 语句时，必须指一一个事先准备好的 SQL 语句。

3、 PreparedStatement 表示预编译的 SQL 语句的对象，SQL 语句被编译并存储在对象中。被封装的 SQL 语句代表某一类操作，SQL 语句中允许包含动态参数“？”，在执行时可以为“？”动态设置参数值。

4、 在使用 PreparedStatement 对象执行 SQL 命令时，SQL 命令被数据库进行解析和编译，然后被放到命令缓冲区。然后每当执行同事 PreparedStatement 对象时，它就会被再解析一次，但是不会被再次编译。在缓冲区可以发现预编译的命令，并且可以重新使用。

20、 存储过程：存储在数据库服务器中的一组 SQL 操作的单元。CallableStatement 接口继承自 PreparedStatement，可用于执行 SQL 存储过程。JDBC API 提供了一个存储过程 SQL 转义语法，

21、 连接 Oracle 数据库时 thin 和 oci 方式区别：

1、 jdbc:oracle:thin@<主机名或 IP>:1521:<数据 SID 名>

2、 java:oracle:oci@<本地服务器名>

3、 这两种不同的连接类型，thin 属于 Direct-to-Database Pure Java Driver（本地协议驱动）类型，只要有数据库驱动包就可以直接通过网络接口访问数据库；而 oci 是 Oracle Call Interface 的缩写，属于 Native-API partly-Java driver(网

络协议驱动）类型，需要在客户端安装 Oracle 的客户端软件，并注册一个本地服务名。在理论上 oci 性能要好于 thin

22、 获取 ResultSet 中含有的记录数量：

在已获取 ResultSet 结果集的情况下，可以使用该对象的 last()和 getRow()方法取得记录数量。last()用于将 ResultSet 指针指向到最后一行记录，getRow()用于返回当前指针所在的位置。

ResultSet 默认情况下，只能使用 next()方法向前逐行移动指计，不支持 last(),absolute()，first()等，如果要使用 last()和 absolut()方法，由 Connection 生成 Statement 时需要指定参数，格式：

Statement stmt=conn.createStatement(游标类型，记录更新权限)；

游标类型：

ResultSet.TYPE_FORWORD_ONLY:只可以向前移动

ResultSet.TYPE_SCROLL_INSENSITIVE：可滚动，不受其他用户对数据库更改的影响

ResultSet.TYPE_SCROLL_SENSITIVE：可滚动，当其他用户更改数据库时这个记录也会改变。

记录更新权限：

ResultSet.CONCUR_READ_ONLY:只读

ResultSet.CONCUR_UPDATABLE:可更新

Statement stmt=conn.createStatement(ResultSet.TYPE_SCROLL_INSENSITIVE , ResultSet.CONCUR_READ_ONLY)；

在数据量很大时会出现内存溢出异常，不推荐。

可使用 SQL 统计函数获取符合查询条件的记录数量：

```
int count=0;
if(rs.next()){
    count=rs.getInt(1);
}
```

23、 获取 ResultSet 中 n-m 位置区间的记录：

在分页操作时，经常要将某一页显示的记录获取，然后在界面上显示，从 ResultSet 中获取指定区间的记录，主要是使用 absolute()方法将指针定位到参数指定的位置，然后通过 getter()方法获取指针指定记录的字段值。

```
Statement stmt=conn.createStatement(ResultSet.TYPE_SCROLL_INSENSITIVE , ResultSet.CONCUR_READ_ONLY);
ResulteSet rs=stmt.executeQuery(“”);
for(int i=begin;i<end;i++){
    if(!rs.absolute(i)){
        break;
    }
    rs.getXXX();
}
```

24、 使用 LIKE 关键字实现模糊查询：

1、%：代表 0 个或多个字符；_：代表任意单一字符；[]：代表在指定区

域或集合中的任意单一字符；[^]：代表不在指定区域或集合中的任意单一字符。

2、使用模糊查询对数据文件进行预处理可以过滤掉大量的无用数据，缩小操作对象的范围，从而提高后续处理的效率，避免对无用数据进行操作造成资源的浪费。

3、模糊查询不适用于数值类型的字段。

25、 实现查询的分组统计和排序：

在进行数据处理时，为了方便操作，往往希望先将一字段内具有相同值的数据归为一组，然后再针对每组做出统一的处理。

SQL 语言中提供了 GROUP BY 子句和分组函数来执行分组操作。GROUP BY 子句使用格式：

```
select  字段，分组函数
from table
[where 查询条件] //指定过滤条件
[group by 分组字段] //指定分组字段
[having 分组函数] //按分组函数的结果排序
[order by 排序字段]； //按普通的字段排序
```

Example:

```
select deptno as no,avg(sal) as a,sum(sal) as a,from emp
group by deptno
having avg(sal)
```

如果需要按照分组函数的结果排序，必须使用 Having 子句，不能使用 order by

在使用分组查询的 SQL 语句时，需要注意一个重要的原则：select 关键字后出现的字段，除分组函数使用的参数外，其他的都要在 group by 子句中出现。

26、 实现多表联合查询：

1、 等值连接：作用是查询结果是由两个表的记录共同决定的，只有两个表的连接字段值相等的记录，才会作为结果返回。格式：

```
select t1.column,t1.column
from table1 t1, table2 t2
where t1.column=t2.column;
```

或：

```
select t1.column,t1.column
from table1 t1
join table2 t2 on(t1.column=t2.column);
```

2、 外连接：作用是查询结果是由某一主表的记录决定的，即使另一方没有对应记录，主表记录也要作为结果返回，另一方记录的字段值为 NULL，外连接分为外连接右连接。

左连接格式：

```
select t1.column,t2.column
from table1 t1
left outer join table2 t2(t1.column=t2.column);
```

右连接格式：

```
select t1.column,t2.column
from table1 t1
right outer join table2 t2(t1.column=t2.column);
```

3、 自然连接：作用与等值连接类拟，只不过连接双方都是同一个表，一般是同一个表的两个不同字段做等值。格式：

```
select t1.column,t2.colum
from table1 t1
join table1 t2(t1.column1=t2.column2);
```

27、 JDBC 的排处理操作：Statement 或 PrepareStatement 都可以执行批处理功能，可以使用 addBatch(“”)方法处理中追加 SQL 语句，然后使用 executeBatch()执行批处理中的 SQL 语句。批处理不能用于查询语句

28、 字段值递增：

Mysql 和 SQLServe,Sybase 都支持字段值自动递增功能，但在 Oracle、DB2、PostgreSQl 中却不支持，在 Oracle 中可以使用 Sequence 序列，实现字段值的自动递增。格式：

```
create sequence 序列名
[increment by 递增量]:设置递增量
[start with 起始值]：设置起始值
[maxvalue|nomaxvalue]：设置最大值
[nocycle]：设置累加，不循环
[cache 数量]：设置一次生成多少个序列值存入缓存
```

删除序列：

```
drop sequence 序列名
```

序列的使用：

可以使用 nextval 和 currentval 两个属性。currentval 表示序列当前值；而 nextvat 表示在当前值基础之上递增之后的值。

Sequence 是数据库系统按照一定规则自动增加的数字序列，这个序列一般作为主键代理，因为其不会重复。

29、 to_date(‘2010-10-10’,yyyy-mm-dd)Oracle 数据库的转换函数，MySQL 数据库中，‘2010-10-10’格式的字符串会自动转换为 Date 类型，

30、 向表中插入含有特殊字符的信息，使用 PrepareStatement 操作数据。使用 BLOB 类型的字段，使用 PrepareStatement 操作数据。通过 setBinaryStream()方法可实现将 BLOB 类型数据写入数据库。使用 CLOB 类型的字段，使用 PrepareStatement 操作数据。通过 setAsciiStream()方法可实现将 CLOB 类型数据写入数据库。

31、 获取数据表的结构信息：

1、 DatabaseMetaData

通过 Connection 的 getMetaData()方法可以获取包含数据库元的 DatabaseMetaData 类的对象。DatabaseMetaData 提供了非常丰富的方法，用于获取数据库的整体信息。如：版本号、产品名称、驱动名称和列名称允许的最大字符等。

2、 ResultSetMetaData

3、 通过 ResultSet 的 getMetaData()方法可以获取包含数据的 ResultSetMetaData 对象。ResultSetMetaData 提供了获取表名称、字段名称、字段类型和字段个数等信息的方法，

32、 获取数据库中的所有表名：

通过 Connection 的 getMetaData()方法可以获取包含数据库元的

DatabaseMetaData 类的对象。DatabaseMetaData 提供了非常丰富的方法，用于获取数据库的整体信息。利用 DatabaseMetaData 的 getTable()方法可以获取数据库中所的表名。

33、 程序备份和恢复数据库：

实现用 Java 程序备份和恢复数据库，最通用的方法是利用 Runtime 类的 exec()方法执行备份和恢复的命令语句。MySQL 数据库备份和恢复的命令分别为（CMD 转到 C:\Program Files\MySQL\MySQL Server 5.0\bin 目录下执行下面的命令）：

```
mysqldump -u用户名 -p密码 -opt 库名 >备份文件路径
mysql -u用户名 -p密码 库名 <恢复文件路径
```

34、 使用事务可以将一组 SQL 操作当作一个整体进行控制，保障逻辑和数据的完整性，在数据库中使用 commit 和 rollback 命令也可以实现事务的提交和回滚操作。

35、 JTA 事务与 JDBC 事务区别：

JTA（Java Transaction API）是一种高层的、与实现无关的、与协议无关的 API，应用程序和应用服务器可以使用 JTA 实现事务管理。

JTA 主要用于分布式的多个数据源的事务控制，而 JDBC 的 Connection 提供的是单个数据源的事务。JDBC 事务因为只涉及一个数据源，所以其事务可以由数据库自己单独实现，而 JTA 事务因为其分布式和多数据源的特性，不能由任何一个数据源实现事务管理，因此 JTA 中的事务由事务管理器实现，它会在多个数据源之间管理事务。一般 JTA 事务都用于 EJB 中，因此一般的应用服务器都有自己的事务管理器用来管理 JTA 事务。注意：如果使用 Tomcat 应用服务器，是不能使用 JTA 事务的。JTA 在使用时，一般会选用 Weblogic、JBoss、Websphere 等服务器。JTA 也是用于管理事务的一套 API，与 JDBC 相比，JTA 主要用于管理分布式多个数据源的事务操作，而 JDBC 主要用于管理单个数据源的事务操作。

36、 JTA 实现分布式事务控制：在分布系统中，一个逻辑单元可能会涉及若干数据源的数据，JTA 能够实现在网络环境中多个数据库在一个事务中进行操作，而 JDBC 事务只能在一个数据库中进行，因为 JDBC 中的事务是与连接相关的。

与 JTA 相关的 API 都在 javax.transaction 包中，

37、 数据库连接池：连接池用于创建和管理数据库连接的缓冲池技术，缓冲池中的连接可以被任何需要它们的线程使用。当一个线程需要用 JDBC 对一个数据库操作时，将从池中请求一个连接。当这个连接使用完毕后，将返回到连接池中，等待为其他的线程服务，优点

- 减少连接创建时间
- 简化的编程模式
- 控制资源的使用

连接池原理：

连接池技术的核心思想是连接复用，通过建立一个数据库连接池以及一套连接使用、分配和管理策略，使得该连接池中的连接可以得到高效、安全的复用，避免了数据库连接频繁建立、关闭的开销。

连接池的工作原理主要由三部组成，分别为连接池的建立、连接池中连接的使用管理、连接池的关闭。

a、 连接池的建立

一般在系统初始化时，连接池会根据系统配置建立，并在池中创建了几个连接对象，以便使用时能从连接池中获取。连接池中的连接不能随意创建和关闭，这样避免了连接随意建立和关闭造成的系统开销。Java 中提供了很多容器类可以方便的构建连接池，如：Vector、Stack

b、 连接池的管理

连接池管理策略是连接池机制的核心，连接池内连接的分配和释放对系统的性能有很大的影响。其管理策略如下：

当客户请求数据库连接时，首先查看连接池中是否有空闭连接，如果存在空闲连接，则将连接分配给客户使用；如果没有空闲连接，则查看当前所开的连接数是否已经达到最连接数，如果没有达到就重新创建一个连接给请求的客户；如果达到就按设定的最大等待时间进行等待，如果超出最大等待时间，则抛出异常给客户。

当客户释放数据库连接时，先判断该连接的引用次数是否超过了规定值，如果超过就从连接池中删除该连接，否则保留为其他客户服务。

c、 连接池的关闭

当应用程序退出时，关闭连接池中所有的连接，释放连接池相关的资源，该过程正好与创建相反。

38、 提升 SQL 语句的查询性能：

数据库设计与规划：

- Primary Key 字段的长度尽量小，能用 small integer 就不要用 integer
- 字符字段如果长度固定，就不要用 varchar、nvarchar 类型
- 设计字段时，如果其值可有可无，最好给一个默认值，并设成“不允许 NULL”

适当地创建索引：

a、 Primary Key 字段可以自动创建索引，而 Foreign Key 字段不可以。

b、 为经常被查询或排序的字段创建索引

c、 创建索引字段的长度不宜过长，不要用超过 20 个字符。

d、 不要为内容重复性高的字段创建索引

e、 不要为使用率低的字段建立索引

f、 不宜为过多字段建立索引，否则影响到 insert update delete 语句的性能

g、 如果说数据表存放的数据很少，就不必刻意使用权索引。

使用索引功能：

在查询数据表时，使用索引查询可以极大提升查询速度，但是如果 where 子句书写不当。即使某些列存在索引，也不能使用该索引查询，而同样会使用全表扫描，这就造成了查询速度的降低。在 where 语句中避免使用以下关键词：NOT、! =、<>、!>、!<、Exists、In、Like、||。使用 LIKE 关键字做模糊查询时，即使已经为某个字段建立索引，但需要以常量字符开头才会使用到索引，如果以“%”开头则不会使用索引。例如“name Like ‘%To’ ”不启用 name 字段上的索引；而“name LIKE ‘TO %’ ”会启用 name 字段上的索引。

避免在 where 子句中对字段使用函数：

对字段使用函数，也等于对字段做运算或连接的动作，调用函数的次数与数据表的记录成正比。如果数据表内记录很多时，会严重影响查询性能。

在 AND 与 OR 的使用：

在 AND 运算中，只要有一个条件使用到索引，即可大幅提升查询速度。但在 OR 运算中，则要所有的条件都有使用到索引才能提升查询速度，因此使用 OR 运算符时需要特别小心

JOIN 与子查询：

相对于子查询，如果能使用 JOIN 完成的查询，一般建议使用后者。原因除了 JOIN 的语法较容易理解外，在多数的情况下，JOIN 的性能也会比子查询高。

其他查询技巧：

DISTINCT、ORDER BY 语法，会让数据库做额外的计算。如果没有要过滤重复记录的需求，使用 Union All 会比 Union 更好，因为后者会加入类似 DIStinct 的算法。

尽可能使用存储过程(Store Procedure)：

Store Procedure 除了经过事先编译、性能较好以外，也可减少 SQL 语句在网络中的传递，方便商业逻辑的重复使用。

尽可能在数据源过滤数据

使用 Select 语法时，尽量先用 SQL 条件或 Store Procedure 过滤所要的信息，避免将大量冗余数据返回给程序，然后由程序处理。

39、 解决 MySQL 数据库插入乱码：

- 设置连接字符串编码：

  在数据库连接字符串后面追加参数，指明 MySQL 服务器发送 SQL 语句的编码格式，格式如下：

  jdbc:mysql://localhost:3306/test?useUnicode==true&characterEncoding=utf-8

- 设置数据表及其字段的编码：

  将数据表的存储类型、表中字符字段的存储型都设置成与连接字符串一致的编码。依据上述连接字符串示例，数据表的存储编码应该设置成 UTF-8

- 设置其他编码：

  如果是从 JSP 页面取值，然后使用 SQL 写入数据库，那么还要保障从 JSP 页面取值正常。具体步骤：

  a、 在 JSP 页面中设置以下代码：

  <%@page language=" java" pageEncoding =" utf-8" %>

  <%@page contentType=" text/html;charset =utf-8" %>

  b、 在使用 request.getParameter()方法获取 JSP 页面值之前，设置 request.seCharacterEncoding(“UTF-8”);

Java 常用功能

40、 过滤字符串前后以及中音出现的空格：

分为两种情况：

- 只需要过滤字符串前后的空格，而中间的空格不需要过滤

用 String 类中提供的 trim()方法即可实现。

- 字符串的前后和中间可能出现的空格，都需要过滤

41、 String、StringBuffer、StringBuilder 区别：

String 类代表定长字符串，其内容在创建之后是不可更改的

StringBuffer 类与 String 类相似，代表的是可变长的字符串缓冲区，通过特定的方法可以改变字符串序列的长度和内容，并且对于多线程操作是安全的。在字符的连接操作上提供了性能和效率都优于 String 类的”+”的 append（）方法，因此如果需要大量频繁地进行字符连接操作时，优先采用 StringBuffer 类的 append()方法。

StringBuilder 类是 StringBuffer 类的一个等价类，该类与 StringBuffer 类具有相同的方法，且同样代表的是可变长的字符串缓冲区，不同的地方在于 StringBuilder 类是非线程安全的。但是也正是因为少了很多的同步操作，在效率上会高于 StringBuffer 类。因此如果不涉及多线程操作，可以优先考虑使用 StringBuilder 类来提高方法的执行效率。

42、 List、Set、Map 是否继承自 Collection 接口：

在 Java 体系中，容器类库分为两大类，即 Collection(集合)和 Map(映像)。Collection 中存放的是一组各自独立的对象，而 Map 中存放的是“键——值”对象。

List 和 Set 都是 Collection 的子接口，List 是一个有序可重复列表，Set 是一个无序重复集。

43、 遍历 Map 和 Vector 集合：

Map:

- Iterator 迭代器遍历 :Map.entrySet().iterator(); XX.hasNext();
- 新式 for 循环启遍历:for(String key:Map.keySet());

Vector:

a、 Enumeration 枚举器遍历 :Enumeration em=Vector.element();em.hasMoreElement();

b、 for 循环遍历:for(int i=0;i<Vector.size();i++)

44、 反射机机制及作用：

- 定义：

反射是指程序可以访问、检测和修改其本身状态或行为的一种能力，在 Java 环境中，反射机制允许程序在执行时获取某个类的自身的定义信息，例如属性和方法等也可以实现动态创建类的对象、变更属性的内容或执行特定的方法的功能。从而使 Java 具有动态语言的特性，增强了程序的灵活性和可移植性

- 反射机制的作用：

Java 反射机制主要用于实现以下功能（在运行时时环境中）。

I、在运行时判断任意一个对象所属的类型

II、在运行时构造任意一个类的对象

III、在运行时判断任意一个类所具有的成员变量和方法

IV、在运行时调用任意一个对象的方法，甚至可以调用 private 方法

- Java 反射机制 API：

实现 Java 反射机制的 API 在 java.lang.reflect 包下，具有以下几

点：

（1）、Class 类：代表一个类

（2）、Field 类：代表类的成员变量

（3）、Method 类：代表类的方法

（4）、Constructor 类：代表类的构造方法

（5）、Array 类：提供了动态创建数组及访问数据的元素的静态方法。该类中的所有方法都是静态的。

- 难点：

  反射机制是 Java 中非常重要的一项功能，应用也非常广泛，在现在流行的 Structs、Hibernate、Spring 等各种框架都是基于反射机制实现的，首先需要将 XML 配置文件的配置信息读取，然后利用反射机制创建对象、执行方法等。

45、 如何使用 Java 调用系统的 exe 文件

通过 Runtime 类可以方便调用外部的 exe 文件。

```
Runtime rm = new Runtime.getRuntime();
rm.exec(“notepad.exe”);
```

46、 如何使用 Java 调用系统的 CMD

```
Process rt;
    try {
        rt = Runtime.getRuntime().exec("ping 127.0.0.1");
        BufferedReader br=new BufferedReader(new
InputStreamReader(rt.getInputStream()));
        while(true){
            String s = br.readLine();
            if(s==null){
                break;
            }
            System.out.println(s);
        }
        br.close();
        rt.waitFor();
        if(rt.exitValue()==0){

            System.out.println("运行成功！");

        }


    } catch (Exception e) {
        // TODO Auto-generated catch block
        e.printStackTrace();
    }
```

47、 使用 MD5 和 SHA 算法加密信息：

Java Web 程序设计

48、 jsp、java、JavaScript 区别：

Jsp 是由 Sun 公司建立一种动态网页技术标准，用于编写动态网站程序。JSP

技术以 Java 语言作为脚本，嵌入到 JSP 页面中，由服务器负责解释运行。JSP 运行环境是 JRE 和服务器

JAVA 程序需要由 JRE 运行环境才能解释运行，面向对象的语言。

JavaScript 由 Netscape 公司基于 Java 的语法开发的。一种基于对象的脚本语言。可以在浏览器里直接运行，不需服务器的支持。是基于对象的函数式的语言，在客户端用于实现表单验证和网页特效功能。

49、 Page、request、session、application 区别：

类型不同分别： Object 、 HttpServletRequest 、 HttpSession 、 ServletContext；

作用范围不同分别：全局作用范围，整个应用程序共享，生命周期为从应用程序启动到停止；会话作用域，当用户首次访问，产生一个新的会话，以后服务器就可以记住这个会话状态。生命周期为会话超时或服务器端强制制使会话失效；请求作用域，客户端的一次请求，生命周期为一次请求或使用 forward 方式执行请求转发；一个页面有效。

50、 forward 和 redirect 区别：

forward 和 redirect 都可以实现页面的跳转，但是跳转时的工作原理不现，使用 forward 时，浏览器请求 URL 不会改变，request 对象不会被销毁；使用 redirect 时，浏览器请求的 URL 会改变，request 对象会被销毁并重新创建。在使用 request 对象在页面传值时，需要使用 forward 方式，而不能使用 redirect 方式。

51、 多个 JSP 页面之间传递信息：

使用 URL；使用 request 对象；使用 Session 对象；使用 application 对象

52、 <jsp:include>和<%@include%>都可以实现在当前 JSP 页面中引入另一个页面，前者主要用于引入动态变化的 JSP 页面；后者主要用于引入 HTML 静态页面和共通的 JSP 源代码。

53、 JavaBean 就是一个符合 JavaBean 规范的 JAVA 类，可用于封装一些共通的业务逻辑，从而实现重复利用。特点：

- 放在一个包中
- JavaBean 类必须要提供一个无参的构造方法。在 JSP 中使用<jsp:getuseBean/>创建 JavaBean 对象时会使用无参的构造方法。
- JavaBean 类不要定义公共类型的属性，避免外界直接访问实例变量，变量名称首字母必须小写
- JavaBean 类通过 gettero/setter()方法来读写属性的值，并且将对应的属性首字母改成大写。注意使用 setter()时 value 属性的类型要匹配。

54、 Cookie：

Cookie 是指存储在客户浏览器目录下的文本文件，文件信息由 Web 服务器发送到客户浏览器并存储，下次该客户再次访问该 web 服务器时，可从浏览器读回此信息。使用 Cookie,Web 服务器可以将一些客户的特定信息存储在客户计算机中，例如上次访问的位置、花费时间或用户密码等不建议使用 Cookie 保存。

- JSP/Servlet 操作 Cookie：

  写入 Cookie 的示代码如下：

  ```
  Cookie c = new Cookie(“username”,” tom”);
  c.setMaxAge(120);
  ```

response.addCookie(c);

上述代码中，如果不使用 setMaxAge()方法设置有效期，Cookie 信息将在客户关闭浏览器之后删除。

读取 Cookie 的示例代码如下：

```
Cookie[] c =request.get Cookies();
for(int i=0;i<c.length;i++){
    if(c[i].getName().equals(“username”){
        username=c[i].getValue();
    }
}
```

在客户端的 Cookie 文件可以存储若干个 Cookie 对象的信息，在读取时,request.get Cookies()返回一个 Cookie 数组，可在该数组中遍历寻找指定的 Cookie 对象。

删除 Cookie 示例代码如下：

```
Cookie c = new Cookie(“mycookie”,null);
c.setMaxAge(0);
c.setPath(“/”);
response.addCookie(c);
```

- JavaScript 操作 Cookie

写入 Cookie 的示例代码如下：

略

55、 Servlet 生命周期：

加载和实例化、初始化、处理请求、移除实例。

56、