01 JDBC快速入门

JDBC

一：JDBC描述

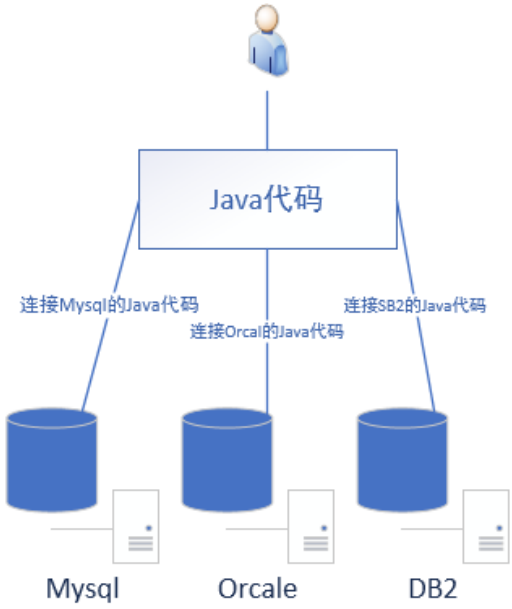
1、JDBC（Java DataBase Connectivity，java数据库连接）是一种用于执行SQL语句的Java API，是 Java 访问数据库的标准规范

2、可以为多种关系数据库提供统一访问，它由一组用Java语言编写的类和接口组成，真正怎么操作数据库还需要具体的实现类，也就是数据库驱动

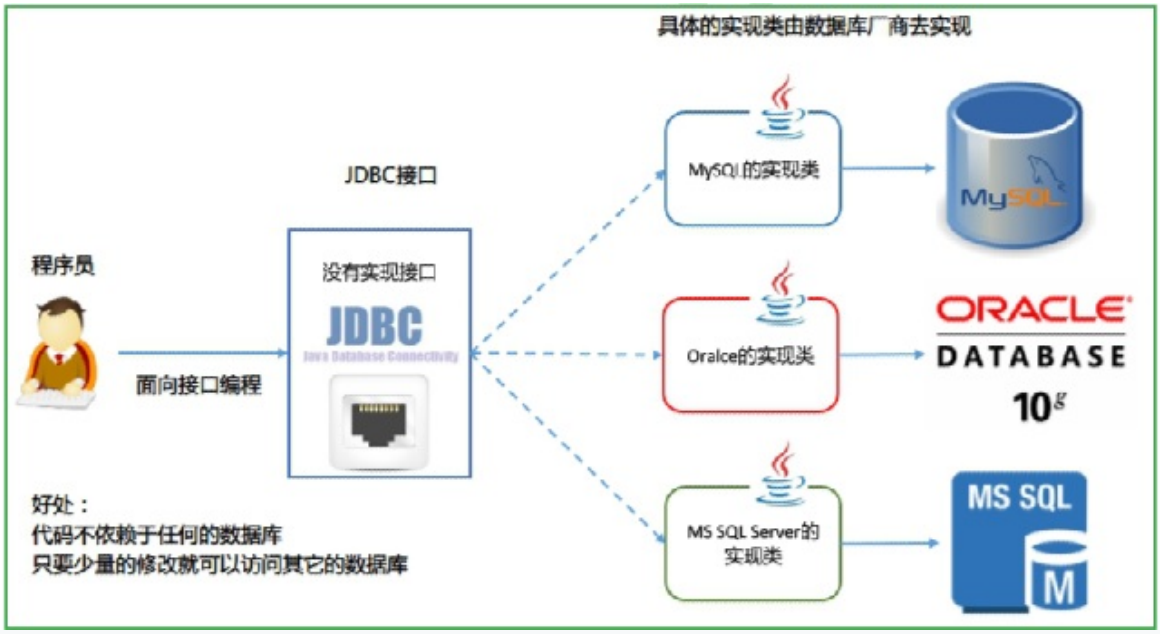
3、JDBC提供了一种基准，据此可以构建更高级的工具和接口，使数据库开发人员能够编写数据库应用程序，同时，JDBC也是个商标名

4、每个 数据库厂商根据自家数据库的通信格式编写好自己数据库的驱动。所以我们只需要会调用 JDBC 接口中的方法即可

5、之前使用JAVA代码调用数据库方式



6、使用了JDBC之后



1、使用 JDBC 的好处

> 1、程序员如果要开发访问数据库的程序，只需要会调用 JDBC 接口中的方法即可，不用关注类是如何实现的。
>
> 2、使用同一套 Java 代码，进行少量的修改就可以访问其他 JDBC 支持的数据库

二：JDBC 入门

> 1、导入对应的jar包
> mysql-connector-java-5.1.38.jar

1、JDBC API

> 1、JDBC 的核心 API

接口或类	作用
DriverManager 类	1) 管理和注册数据库驱动 2) 得到数据库连接对象
Connection 接口	一个连接对象，可用于创建 Statement 和 PreparedStatement 对象
Statement 接口	一个 SQL 语句对象，用于将 SQL 语句发送给数据库服务器。
PreparedStatemen 接口	一个 SQL 语句对象，是 Statement 的子接口
ResultSet 接口	用于封装数据库查询的结果集，返回给客户端 Java 程序

2、加载和注册驱动

1、mysql的注册加载驱动，获取数据的连接

加载和注册驱动的方法	描述
Class.forName(数据库驱动实现类)	加载和注册数据库驱动，数据库驱动由 mysql 厂商 "com.mysql.jdbc.Driver"

```
1.      @Test
2.      public void test01() throws Exception {
3.           //注册驱动
4.           //Mysql jar包5.1以后可以不用写 Class.forName，默认自动就有
5.          Class.forName("com.mysql.jdbc.Driver");
6.          //获取数据库连接对象
7.      Connection conn =
    DriverManager.getConnection("jdbc:mysql://192.168.0.115:3306/cost", "r
    oot", "root");
8.          //连接本地
9.          //Connection conn =
    DriverManager.getConnection("jdbc:mysql:://localhost:3306/cost", "root
    ", "root")
10.         // Connection conn =
    DriverManager.getConnection("jdbc:mysql:///cost", "root", "root");
11.             System.out.println("conn:"+conn);
12.     }
```

2、Oracle连接:注意导入jar包ojdbc14.jar

```
1.                  //注册驱动
2.                   Class.forName("oracle.jdbc.driver.OracleDriver");
3.                   //获取连接
4.                   String url =
```

```
"jdbc:oracle:thin:@192.168.99.129:1521:orcl";
                String username = "tomcat";
                String password = "root";
                Connection conn = DriverManager.getConnection(url, user
name, password);
```

3、结果信息

```
conn:com.mysql.jdbc.JDBC4Connection@70beb599
```

4、乱码的处理

可以指定参数: ?characterEncoding=utf8，表示让数据库以 UTF-8 编码来处理数据

```
jdbc:mysql://192.168.0.115:3306/cost?characterEncoding=utf8
```

三：DriverManager 类

1、DriverManager 作用

1) 管理和注册驱动
2) 创建数据库的连接

2、类中的方法

1、静态方法

静态方法	描述
Connection getConnection (String url, String user, String password)	通过连接字符串，用户名，密码来得到数据 库的连接对象
Connection getConnection (String url, Properties info)	通过连接字符串，属性对象来得到连接对象

2、连接数据库的 URL 地址格式

协议名:子协议://服务器名或 IP 地址:端口号/数据库名?参数=参数值

四：Connection 接口

> 1、Connection 接口，具体的实现类由数据库的厂商实现，代表一个连接对象

1、Connection 方法

方法名	描述
Statement createStatement()	创建一条 SQL 语句对象

五：Statement 接口

> 1、 Statement 对象执行 SQL 语句

方法名	描述
int executeUpdate(String sql)	用于发送 DML 语句，增删改的操作，insert、update、delete 参数：SQL 语句 返回值：返回对数据库影响的行数
ResultSet executeQuery(String sql)	用于发送 DQL 语句，执行查询的操作。select 参数：SQL 语句

返回值：查询的结果集 |

1、DML 语句简单增删改

> 1、进行对数据表的添加数据

```
1.      @Test
```

```
    public void test03() throws Exception {
        Class.forName("com.mysql.jdbc.Driver");
    Connection conn = DriverManager.getConnection("jdbc:mysql://192.168.0.115:3306/cost?characterEncoding=utf8", "root", "root");
            String sql = "insert into t_cost values(null,'火车票','不能超出2W','0')";
            //获取执行sql的对象 Statement
            Statement stmt = conn.createStatement();
            //执行sql
            int count = stmt.executeUpdate(sql);
            //处理结果
            System.out.println(count);
            //释放资源
            stmt.close();
            conn.close();
    }
```

2、进行对数据表的修改

```
    @Test
    public void test04() throws Exception {
        Class.forName("com.mysql.jdbc.Driver");
    Connection conn = DriverManager.getConnection("jdbc:mysql://192.168.0.115:3306/cost?characterEncoding=utf8", "root", "root");
                String sql  = "update t_cost set costName = '轮船票' ,costDesc='不超过3w' where costId = 3";
            //获取执行sql的对象 Statement
            Statement stmt = conn.createStatement();
            //执行sql
            int count = stmt.executeUpdate(sql);
            //处理结果
            System.out.println(count);
            //释放资源
            stmt.close();
            conn.close();
    }
```

3、进行对数据表的删除一条记录

```java
@Test
    public void test05() throws Exception {
        Class.forName("com.mysql.jdbc.Driver");
    Connection conn = DriverManager.getConnection("jdbc:mysql://192.168.0.115:3306/cost?characterEncoding=utf8", "root", "root");
                    String sql  = "delete from t_cost where costId = 8";
                   //获取执行sql的对象 Statement
              Statement stmt = conn.createStatement();
              //执行sql
              int count = stmt.executeUpdate(sql);
              //处理结果
              System.out.println(count);
              //释放资源
              stmt.close();
              conn.close();
    }
```

2、DDL 语句

1、创建一张表

```java
@Test
    public void test06() throws Exception {
        Class.forName("com.mysql.jdbc.Driver");
    Connection conn = DriverManager.getConnection("jdbc:mysql://192.168.0.115:3306/cost?characterEncoding=utf8", "root", "root");
              String sql  = "create table student (id int  primary key , name varchar(20), adder varchar(20))";
              //获取执行sql的对象 Statement
              Statement stmt = conn.createStatement();
              //获取执行sql对象
              stmt = conn.createStatement();//可不写
              //执行sql
              int count = stmt.executeUpdate(sql);
              //处理结果
              System.out.println(count);
              //释放资源
              stmt.close();
              conn.close();
    }
```