JDBC 调用存储过程

如果想要使用 JDBC 执行存储过程，应该使用 CallableStatement 接口。

CallableStatement 接口继承自 PreparedStatement 接口。所以 CallableStatement 接口包含有 Statement 接口和 PreparedStatement 接口定义的全部方法，但是并不是所有的方法我们都要使用，主要使用的方法如表-1 所示。

表-1 CallableStatement 接口的常用方法

方法	说明
boolean execute()	执行 SQL 语句，如果第一个结果是 ResultSet 对象，则返回 true；如果第一个结果是更新计数或者没有结果，则返回 false。
void registerOutParameter(int parameterIndex, int sqlType)	按顺序位置 parameterIndex 将 OUT 参数注册为 SQL 类型 sqlType，sqlType 为 Types 类中的常量。
Type getType(int parameterIndex)	根据参数的序号获取指定的 JDBC 参数的值。第一个参数是 1，第二个参数是 2，依此类推。

execute()方法可以用来执行存储过程。CallableStatement 为所有的数据库提供了一种统一的标准形式调用存储过程。所以，你将会看到我们使用 execute()调用存储过程的语法与在 SQLServer 中会所有不同。

为了获得存储过程或函数的返回值，我们需要使用 registerOutParameter()方法将返回的参数注册为 SQL 的类型。 registerOutParameter()方法的第一个参数是参数的序号，第一个为 1，第二个为 2，以此类推。第二个参数需要一个 int 值，用来标记 SQL 的类型，我们可以使用 java.sql.Types 类中的常量来设置这个参数。比如 VARCHAR、DOUBLE 等类型。如果类型不够用，也可以从具体数据库的驱动中寻找合适的类型常量。如果存储过程或函数有返回值，这个方法是必须要调用的，否则无法得到返回值，甚至会发生异常。

CallableStatement 接口中定义了很多 get 方法，表格中的“Type”是指各种基本数据

类型和 String、Date、Time、Timestamp、Object 等类型，用于获取存储过程返回的值，根据值的类型不同，你可以使用不同 get 方法，比如 getInt()、getString()、getDouble()等等。

我们看一下使用 CallableStatement 接口执行存储过程和函数的语法格式。

语法：

存储过程：{call <procedure-name>[(<arg1>,<arg2>, ...)]}

函数：{?= call <procedure-name>[(<arg1>,<arg2>, ...)]}

- 如果要调用存储过程，则使用第一种语法，就是开头不带问号的语法，call 后面是过程名，如果没有参数，可以省略小括号。
- 如果要调用函数，则使用第二种语法，开头带有一个问号加等号，实际上这个问号就是一个占位符，这个问号总是调用函数的第一个占位符。其它部分与过程的语法相同。

假设我们已经编写好了一个存储过程，如下所示。

```
CREATE PROCEDURE find_book
    @id int,
    @bName varchar(50) output
AS
     SELECT @bName=BOOK_NAME FROM  book WHERE  BOOK_ID=@id
GO
```

这个存储过程的功能是根据传入的图书编号 id 查询图书的名字，第一个参数是要查询的图书编号 id，第二个是图书的名字，并将其设置为 OUT，这样就可以在存储过程执行之后获取它的值了。我们使用 JDBC 来执行这个存储过程，代码如示例 1 所示。

示例 1：

```
public class BookDAO {
   public String findById(int id) {
        Connection con = null;
        CallableStatement cstmt = null;
        String sql = "{call find_book(?,?)}";
        String bookName=null;
        try {
             Class.forName("com.microsoft.sqlserver.jdbc.SQLServerDriver");
             con = DriverManager.getConnection(
                  "jdbc:sqlserver://localhost:1433;DatabaseName=employee", "sa","sa");
             cstmt = con.prepareCall(sql);
             //设置操作对象的相关参数，第一个参数是图书编号
             cstmt.setInt(1, 1);
             //注册操作对象的相关参数，调用存储过程时有输出参数时要注册输出参数的数据类型。
             cstmt.registerOutParameter(2, java.sql.Types.VARCHAR);
             cstat.execute();
             //接收存储过程输出参数的值
```

```
            bookName =(String) cstat.getObject(2);
        } catch  {
            //省略
        } finally {
            //省略
        }
        return bookName;
    }
    public static void main(String[] args) {
        BookDAO bookDAO = new BookDAO();
        String bookName = bookDAO.findById(1);
        System.out.println(bookName);
    }
}
```

示例 1 代码中，sql 是调用存储过程的字符串语句。两个参数都使用了占位符。使用 Connection 对象的 prepareCall()方法创建一个 CallableStatement 对象，在创建对象时就传入了调用存储过程的语句。

然后使用 CallableStatement 对象的 set 方法设置参数。第二个参数是用来读取的，所以我们要使用 registerOutParameter()方法注册它的 SQl 类型，使用的类型是 Types 中的常量 VARCHAR。

最后使用 execute()方法来执行存储过程。执行之后，我们可以通过 CallableStatement 对象的 get 方法获得执行的结果。

执行函数和执行存储过程是差不多的。只不过使用的调用语句的语法不同，设置参数的时候，不要忘记注册第一个参数的 SQL 类型，其他都跟执行存储过程是一样的，代码如示例 2 所示。

示例 2：

```
String sql = "{?=call find_emp2(?)}";
Connection con = null;
CallableStatement cstmt = null;
try {
    Class.forName("com.microsoft.sqlserver.jdbc.SQLServerDriver ");
    con = DriverManager.getConnection(
          "jdbc:sqlserver://localhost:1433;DatabaseName=employee", "sa","sa");
    cstmt = con.prepareCall(sql);
    cstmt.setInt(2, 1);
    cstmt.registerOutParameter(1, java.sql.Types.VARCHAR);
    cstmt.execute();
    System.out.println(cstmt.getString(1));
} catch  {
     //省略
} finally {
     //省略
}
```

这段代码执行的函数与刚才的存储过程的功能是一样的，也是根据 id 返回姓名。