4-5-3.SQLの実行

SQLの実行

はじめに

なぜ、プログラミングでSQLを実行する必要があるのか...

郵便局のホームページを想像してみてください。
郵便番号を入力するだけで、途中まで表示されますよね。
WEB画面で入力された値を元に、プログラミングがデータベースでSQLを実行し、
データベースから住所の情報が返ってきて、
それをプログラミングが受けとってWEB画面に表示しています。
WEB画面の裏側には必ずと言っていいほど何かしらのデータベースがあって、
それを **プログラムで制御** しているのです。
なんとなくイメージできたでしょうか？

このあたりから、少しずつ難しくなってきますが、
エンジニアになるための必須科目なので、めげずに頑張って行きましょう！

Step1：主要インタフェース

> **例題 1**

下記に示すプログラムを作成し、実行しなさい。
「Javaアプリケーション」で実行しましょう。

?

```
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;

public class Example5_1 {

    /* 定数 */
    /** ドライバーのクラス名 */
    private static final String POSTGRES_DRIVER = "org.postgresql.Driver";
    /** ・JDBC接続先情報 */
    private static final String JDBC_CONNECTION = "jdbc:postgresql://localhost:5432/lesson_db";
    /** ・ユーザー名 */
    private static final String USER = "postgres";
    /** ・パスワード */
    private static final String PASS = "postgres";

    public static void main(String[] args) {

       Connection connection = null;
       Statement statement = null;
       ResultSet resultSet = null;

       try {
           Class.forName(POSTGRES_DRIVER);
           connection = DriverManager.getConnection(JDBC_CONNECTION, USER, PASS);

           statement = connection.createStatement();
           String SQL = "SELECT * FROM Goods";
           resultSet = statement.executeQuery(SQL);

           while (resultSet.next()) {
               String column1 = resultSet.getString("GoodsName");
               String column2 = resultSet.getString("UnitPrice");
               String column3 = resultSet.getString("UpdateDate");

               System.out.print(column1 + ",");
               System.out.print(column2 + ",");
               System.out.println(column3);
           }

       // forName()で例外発生
       } catch (ClassNotFoundException e) {
           e.printStackTrace();

       // getConnection()、createStatement()、executeQuery()で例外発生
       } catch (SQLException e) {
           e.printStackTrace();

       } finally {
           try {
               if (resultSet != null) {
                   resultSet.close();
               }
               if (statement != null) {
                   statement.close();
               }
               if (connection != null) {
                   connection.close();
               }

           } catch (SQLException e) {
               e.printStackTrace();
           }
       }
    }
}
```

解説

```
Statement statement = connection.createStatement();
String SQL = "SELECT * FROM テーブル名";
ResultSet resultSet = statement.executeQuery(SQL);
```

SQLの実行には、 **Statementインタフェース** を使用します。
このクラスはConnectionクラスの`createStatement()`メソッドで取得します。

Statementインタフェースの`executeQuery()`メソッドは、
引数で指定されたSQLをデータベースで実行するメソッドです。
何のエラーもなく処理が完了すると、
SQLの実行結果を格納したResultSetインタフェース型のオブジェクトを返します。
つまり、ResultSetはSQLインターフェースは **実行結果を格納** し、
その **情報も取得できる** メソッドも備えているということです。

```
while (resultSet.next()) {
    String column1 = resultSet.getString("列名");
    String column2 = resultSet.getString("列名");
    int column3 = resultSet.getInt("列名");
}
```

ResultSetインタフェースのresultSetオブジェクトは初期状態では、
最初の行の1つ前に位置していますので、
必ず **next()メソッド** （次の行に移動する）で最初の行に移動してやる必要があります。

また、 `next()メソッド` は、該当する行が存在する場合にのみ `true` を返します。
下記のようにwhile文と組み合わせて使うことにより、
実行結果を格納した情報を先頭から最後尾まで順に追っていくことができます。

該当する行の任意の列の取得には、データのタイプに応じて「getXXX()」メソッドを使用します。
文字列の場合は`getString()`メソッド、
整数の場合は`getInt()`メソッド、`getLong()`メソッド、
日付の場合には `getDate()` メソッドを使用します。

Step2 : INSERT文、UPDATE文、DELETE文の実行

例題２

下記に示すプログラムを作成し、実行しなさい。

```
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.SQLException;
import java.sql.Statement;

public class Example5_2 {

     /* 定数 */
     /** ドライバーのクラス名 */
     private static final String POSTGRES_DRIVER = "org.postgresql.Driver";
     /** ・JDMC接続先情報 */
     private static final String JDBC_CONNECTION = "jdbc:postgresql://localhost:5432/lesson_db";
     /** ・ユーザー名 */
     private static final String USER = "postgres";
     /** ・パスワード */
     private static final String PASS = "postgres";

     public static void main(String[] args) {

        Connection connection = null;
        Statement statement = null;

        try {
            Class.forName(POSTGRES_DRIVER);
            connection = DriverManager.getConnection(JDBC_CONNECTION,USER,PASS);
            statement = connection.createStatement();

            String SQL = "INSERT INTO TB_SHOHIN( SHOHIN_ID, SHOHIN_NAME, TANKA)"
                    + " VALUES('021', 'SHOHIN021', 2100) ";
            statement.executeUpdate(SQL);

        } catch (ClassNotFoundException e) {
            e.printStackTrace();

        } catch (SQLException e) {
            e.printStackTrace();

        } finally {
            try {
                if (statement != null) {
                    statement.close();
                }
                if (connection != null) {
                    connection.close();
                }

            } catch (SQLException e) {
                e.printStackTrace();
            }
        }
    }
}
```

解説

```
Statement statement = connection.createStatement();
String SQL = "INSERT INTO テーブル名(列名, 列名, …) VALUES( 値, 値, …) ";
statement.executeUpdate(SQL);
```

SQLの **INSERT文、UPDATE文、DELETE文の実行** でも、Statementインタフェースを使用します。
SELECT文のSQLの実行には `executeQuery()` メソッドを使用しましたが、
INSERT文、UPDATE文、DELETE文の場合は、 `executeUpdate()` **メソッドを使用します。**

課題

提出課題はありませんので、一通り学習が終わったら次の章に進んで下さい。

最終更新日時: 2024年 06月 16日(日曜日) 14:19