TRƯỜNG ĐẠI HỌC
SƯ PHẠM KỸ THUẬT TP. HỒ CHÍ MINH
HCMC University of Technology and Education
HCMUTE

KHOA CÔNG NGHỆ THÔNG TIN
BỘ MÔN CÔNG NGHỆ PHẦN MỀM

LẬP TRÌNH WEB (WEBPR330479)

Java Database Connectivity (JDBC)

THS. NGUYỄN HỮU TRUNG

PR33 WEB 0479

Thông tin Giảng viên

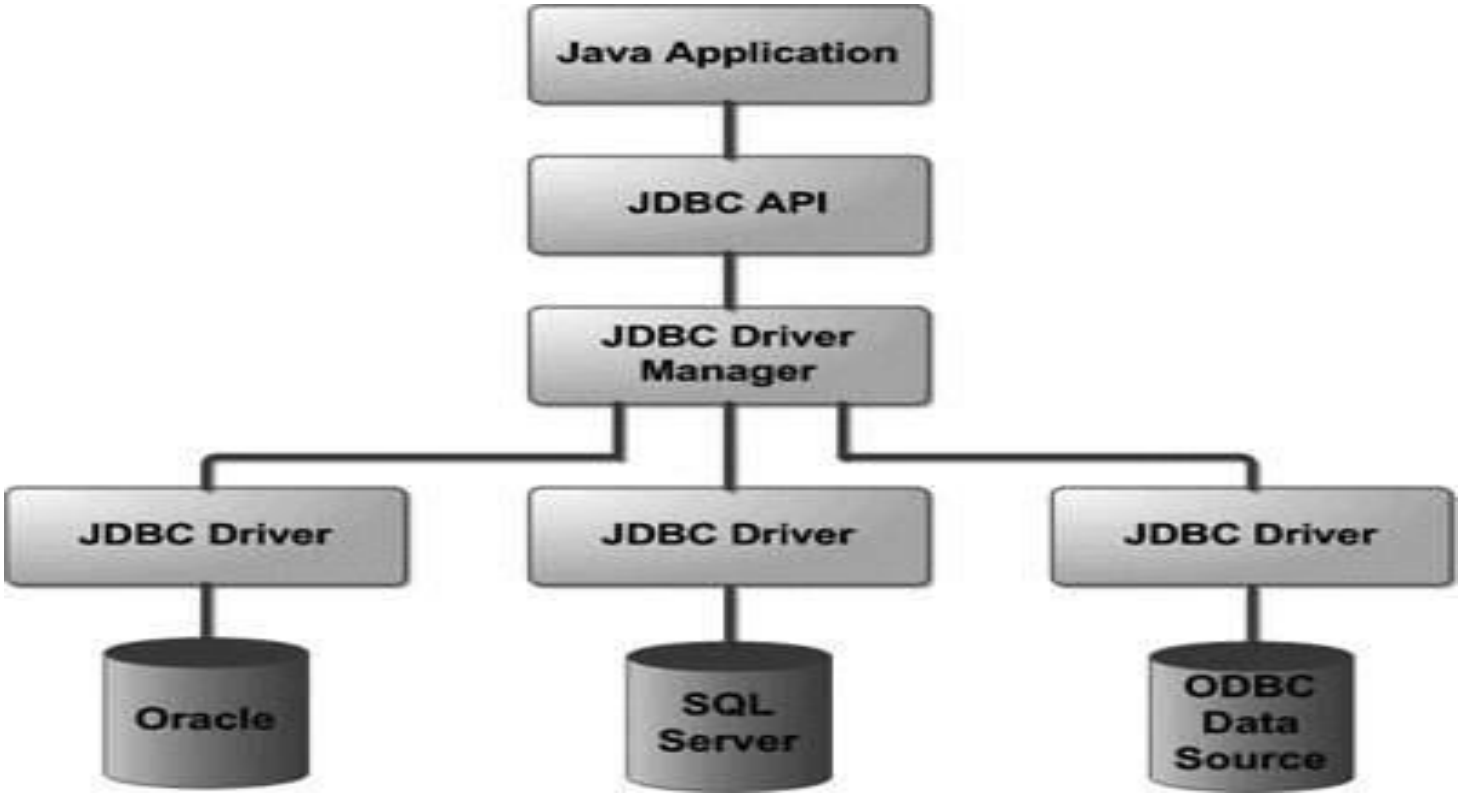
- Ths. Nguyễn Hữu Trung
- Khoa Công Nghệ Thông Tin
- Trường Đại học Sư Phạm Kỹ Thuật TP.HCM
- 090.861.7108
- trungnh@hcmute.edu.vn
- https://www.youtube.com/@baigiai

Nội dung

- JDBC là gì?
- Connection trong JDBC
- Statement trong JDBC
- ResultSet trong JDBC
- PreparedStatement trong JDBC
- ResultSetMetaData trong JDBC
- DatabaseMetaData trong JDBC

Java Database Connectivity (JDBC) là gì?

- JDBC là một Java API được sử dụng để kết nối và thực hiện truy vấn với cơ sở dữ liệu.
- JDBC API cung cấp: **DriverManager, Driver, Connection, Statement**, **ResultSet**, **SQLException**



Connection trong JDBC

- **Connection trong JDBC** là phiên làm việc giữa ứng dụng Java và cơ sở dữ liệu.
- Đối tượng **Connection** được sử dụng để tạo Statement, PreparedStatement, và DatabaseMetaData.
- Giao diện **Connection** cung cấp nhiều phương thức quản lý transaction như commit(), rollback(), ...

Connection trong JDBC

Phương thức	Mô tả
1) public Statement createStatement():	Tạo một đối tượng Statement được sử dụng để thực thi các câu truy vấn SQL.
2) public Statement createStatement(int resultSetType, int resultSetConcurrency**):**	Tạo ra một đối tượng Statement sẽ tạo các đối tượng ResultSet với kiểu đã cho và concurrency.
3) public void setAutoCommit(boolean status):	Được sử dụng để thiết lập trạng thái commit. Mặc định là true.
4) public void commit():	Lưu các thay đổi được thực hiện từ khi commit/rollback trước đó.
5) public void rollback():	Loại bỏ tất cả các thay đổi được thực hiện kể từ lần commit/rollback trước đó.
6) public void close():	Đóng kết nối và giải phóng tài nguyên JDBC ngay lập tức.

Ví dụ kết nối MySQL


```
public class DBConnectionMySQL {

    private static String DB_URL = "jdbc:mysql://localhost:3306/mvcservletltweb2021";
    private static String USER_NAME = "root";
    private static String PASSWORD = "123456@a";
    private static Connection con;
    public static Connection getConnection() throws IOException {
        con = null;
        try {
            // driver register
            DriverManager.registerDriver(new com.mysql.cj.jdbc.Driver());
            con = (Connection) DriverManager.getConnection(DB_URL, USER_NAME, PASSWORD);
        } catch (SQLException ex) {
            Logger.getLogger(DBConnectionMySQL.class.getName()).log(Level.SEVERE, null, ex);
        }
        return (con);
    }
    public static void main(String[] args) {
        try {

            Connection c = getConnection();
            if (c == null) {
                System.out.println("something wrong");
            } else {
                System.out.println("ok");
            }

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

Statement trong JDBC

- **Statement interface** trong **JDBC** cung cấp các phương thức để thực thi các câu lệnh truy vấn với cơ sở dữ liệu SQL.
- **Statement** interface cung cấp phương thức để tạo ra đối tượng ResultSet.

Phương thức	Mô tả
public ResultSet executeQuery(String sql**)**	Được sử dụng để thực hiện truy vấn SELECT. Nó trả về đối tượng của ResultSet.
public int executeUpdate(String sql)	Được sử dụng để thực thi câu truy vấn được chỉ định, nó có thể là create, drop, insert, update, delete, ...
boolean execute(String sql)	Được sử dụng để thực thi các câu truy vấn trả về nhiều kết quả.
int[] executeBatch()	Được sử dụng để thực thi một tập các lệnh.

Ví dụ: Statement


```
try {
        // connnect to database 'testdb'
        Connection conn = DBContext().Connection();
        // crate statement
        Statement stmt = conn.createStatement();
        // insert ‘GiaoVien'
        stmt.executeUpdate("INSERT INTO GiaoVien(id, name, address) "
                + "VALUES (1, ‘Trung', ‘HCM')");
        // get data from table ‘GiaoVien'
        ResultSet rs = stmt.executeQuery("SELECT * FROM GiaoVien");
        // show data
        while (rs.next()) {
           System.out.println(rs.getInt(“id”) + "  " + rs.getString(2)
                   + "  " + rs.getString(3));
        }
        conn.close(); // close connection
    } catch (Exception ex) {
        ex.printStackTrace();
    }
```

- **ResultSet** duy trì một con trỏ trỏ đến một hàng của một bảng. Ban đầu, con trỏ trỏ đến hàng đầu tiên.
- Nhưng chúng ta có thể làm cho đối tượng này di chuyển hướng chuyển tiếp và ngược lại bằng cách truyền một trong hai kiểu TYPE_SCROLL_INSENSITIVE hoặc TYPE_SCROLL_SENSITIVE trong phương thức createStatement(int, int) :

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

ResultSet trong java

Phương thức	Mô tả
public boolean next()	được sử dụng để di chuyển con trỏ đến một hàng tiếp theo từ vị trí hiện tại.
public boolean previous()	được sử dụng để di chuyển con trỏ đến một hàng trước đó từ vị trí hiện tại.
public boolean first()	được sử dụng để di chuyển con trỏ đến hàng đầu tiên trong đối tượng thiết lập kết quả.
public boolean last()	được sử dụng để di chuyển con trỏ đến hàng cuối cùng trong đối tượng thiết lập kết quả.
public boolean absolute(int row)	được sử dụng để di chuyển con trỏ đến số hàng được chỉ định trong đối tượng ResultSet.
public boolean relative(int row)	được sử dụng để di chuyển con trỏ đến số hàng tương đối trong đối tượng ResultSet, nó có thể là dương hoặc âm.
public int getInt(int columnIndex)	được sử dụng để trả về dữ liệu của chỉ mục cột được chỉ định của hàng hiện tại như int.
public int getInt(String columnName)	được sử dụng để trả lại dữ liệu của tên cột được chỉ định của hàng hiện tại như int.
public String getString(int columnIndex)	được sử dụng để trả về dữ liệu của chỉ mục cột được chỉ định của dòng hiện tại như String.
public String getString(String columnName)	được sử dụng để trả lại dữ liệu của tên cột được chỉ định của hàng hiện tại như String.

Ví dụ 1: ResultSet


```
// crate statement
        Statement stmt =
conn.createStatement(ResultSet.TYPE_SCROLL_SENSITIVE,
            ResultSet.CONCUR_UPDATABLE);
       // get data from table ‘GiaoVien'
        ResultSet rs = stmt.executeQuery("SELECT * FROM GiaoVien");
        // getting the record of 3rd row
        rs.absolute(3);
```

Ví dụ 2: ResultSet


```
try {
        // connnect to database 'testdb'
        Connection conn = DBContext().getConnection();
        // crate statement
        Statement stmt = conn.createStatement();
        // insert ‘GiaoVien'
        stmt.executeUpdate("INSERT INTO GiaoVien(id, name, address) "
              + "VALUES (1, ‘Trung', ‘HCM')");
        // get data from table ‘GiaoVien'
        ResultSet rs = stmt.executeQuery("SELECT * FROM GiaoVien");
        // show data
        while (rs.next()) {
           System.out.println(rs.getInt(1) + "  " + rs.getString(2)  + "  " +
rs.getString(3));
        }
        conn.close(); // close connection
     } catch (Exception ex) {
        ex.printStackTrace();
     }
```

PreparedStatement trong JDBC

- **PreparedStatement** là một sub-interface của Statement. Nó được sử dụng để thực hiện truy vấn tham số.

```
String sql = "INSERT INTO GiaoVien VALUES(?, ?, ?)";
```

- Chúng ta truyền tham số (?) cho các giá trị. Giá trị của nó sẽ được cài đặt bằng cách gọi các phương thức **Setter** của PreparedStatement.
- Hiệu suất của ứng dụng sẽ nhanh hơn nếu sử dụng giao diện PreparedStatement vì truy vấn được biên dịch chỉ một lần.
- Phương thức prepareStatement() của giao diện Connection được sử dụng để trả về đối tượng PreparedStatement. Cú pháp:

```
public PreparedStatement prepareStatement(String query) throws SQLException;
```

PreparedStatement trong java

Phương thức	Mô tả
public void setInt(int paramIndex, int value)	đặt giá trị số nguyên cho chỉ số tham số đã cho.
public void setString(int paramIndex, String value)	đặt giá trị String cho chỉ số tham số đã cho.
public void setFloat(int paramIndex, float value)	đặt giá trị float vào chỉ số tham số đã cho.
public void setDouble(int paramIndex, double value)	đặt giá trị double vào chỉ số tham số đã cho.
public int executeUpdate()	thực hiện truy vấn. Nó được sử dụng để create, drop, insert, update, delete, vv.
public ResultSet executeQuery()	thực hiện truy vấn chọn. Nó trả về một thể hiện của ResultSet.

Ví dụ: PreparedStatement


```
String sqlInsert = "INSERT INTO GiaoVien VALUES(?, ?, ?)";
String selectAll = "SELECT * FROM GiaoVien";
try {
   // connect to database
   Connection conn = DBContext().getConnection();
   // crate statement to insert GiaoVien
   PreparedStatement stmt = conn.prepareStatement(sqlInsert);
   stmt.setInt(1, 1);
   stmt.setString(2, “Trung");
   stmt.setString(3, “HCM");
   stmt.execute();
   // select all GiaoVien
   stmt = conn.prepareStatement(selectAll);
   // get data from table GiaoVien
   ResultSet rs = stmt.executeQuery();
   // show data
   while (rs.next()) {
      System.out.println(rs.getInt(1) + "  " + rs.getString(2) + "  " + rs.getString(3));
   }
   stmt.close();
   conn.close(); }
```

ResultSetMetaData trong JDBC

- **ResultSetMetaData** được sử dụng để lấy ra các metadata từ đối tượng ResultSet.
- Siêu dữ liệu (metadata) là các thông tin của dữ liệu. Ví dụ, một metadata của file là ngày tháng tạo file, dung lượng của file, ...
- Metadata của một bảng trong CSDL là các thông tin về bảng như tên bảng, tên cột, kiểu giá trị của cột, ..
- Phương thức getMetaData() của giao diện ResultSet trả về đối tượng ResultSetMetaData. Cú pháp:

```
ResultSetMetaData getMetaData() throws SQLException;
```

ResultSetMetaData trong JDBC

Method	Description
public int getColumnCount()throws SQLException	trả về tổng số các cột trong đối tượng ResultSet.
public String getColumnName(int index)throws SQLException	trả về tên cột của chỉ số cột được chỉ định.
public String getColumnTypeName(int index)throws SQLException	trả về tên cột cho chỉ số đã chỉ định.
public String getTableName(int index)throws SQLException	trả về tên bảng cho chỉ số cột được chỉ định.

DatabaseMetaData trong JDBC

- **DatabaseMetaData** cung cấp các phương thức để lấy metadata của cơ sở dữ liệu như tên sản phẩm cơ sở dữ liệu, phiên bản sản phẩm cơ sở dữ liệu, tên driver, tên của tổng số bảng, tên của tổng số các view, ...
- Phương thức getMetaData() của Giao diện Connection trả về đối tượng DatabaseMetaData. Cú pháp:

```
public DatabaseMetaData getMetaData() throws SQLException;
```