CODEGYM

Raising the bar

JDBC & CRUD

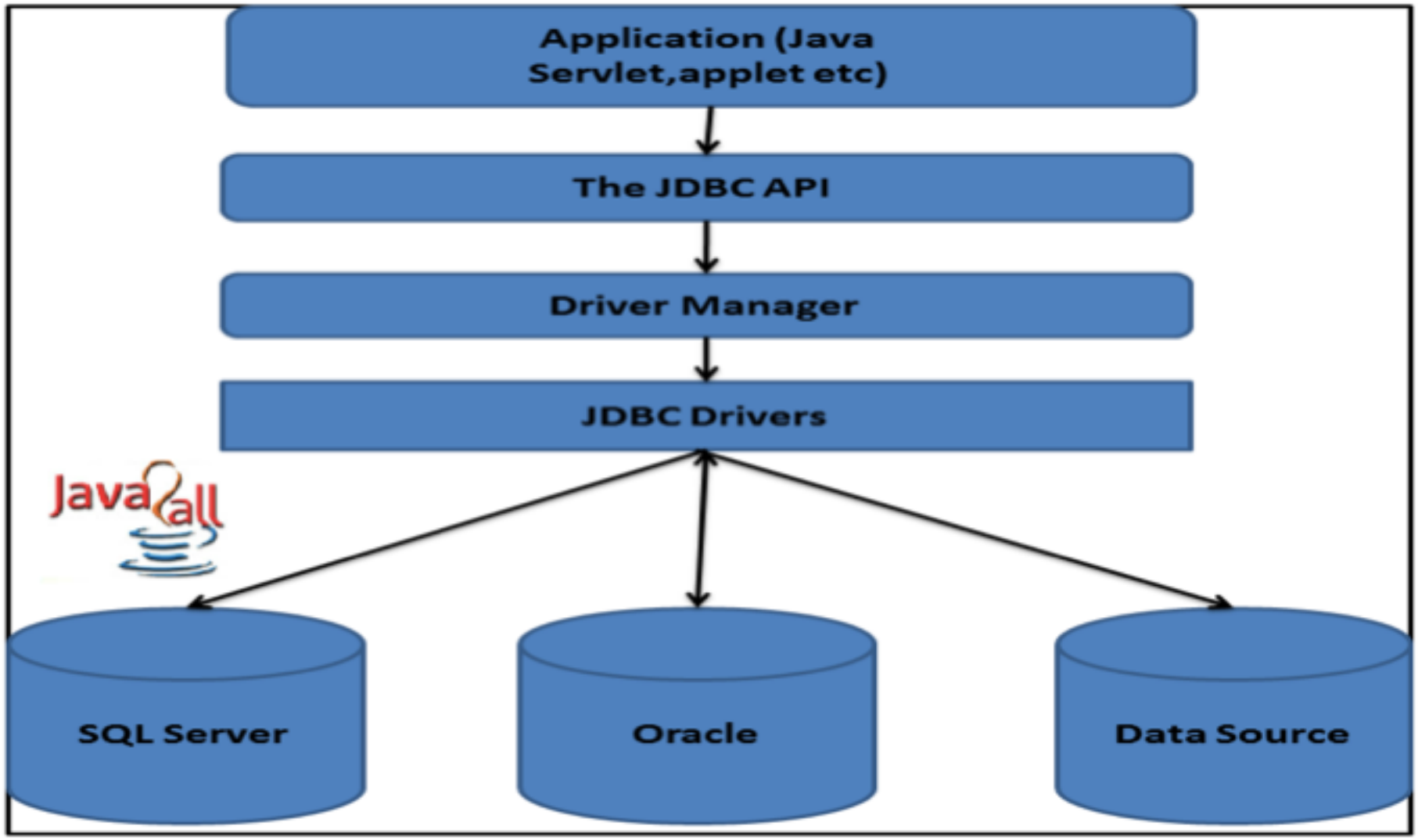
Mục tiêu

- Sử dụng được JDBC kết nối cơ sở dữ liệu
- Sử dụng được các thao tác CRUD với JDBC

JDBC là gì ?

- JDBC (Java Database Connectivity) là một chuẩn API (Application Program Interface) cho phép kết nối các chương trình viết bởi Java với các hệ quản trị cơ sở dữ liệu (MySQL, MS SQL, Postgre SQL, Oracle, DB2...)
- JDBC chỉ là một tập các interface, các định nghĩa, thông báo lỗi, đặc tả chứ không phải là thư viện.
- Với mỗi hệ quản trị cơ sở dữ liệu ta sẽ có một cài đặt JDBC riêng cho nó, ví dụ JDBC cho MySQL, JDBC cho MS SQL ...

JDBC Architecture



Các thành phần của JDBC

- DriverManager: Dùng để quản lý danh sách các Driver (database drivers).
- Driver: Dùng để liên kết các kết nối tới cơ sở dữ liệu, điều khiển các liên kết.
- Connection: Biểu thị kết nối tới cơ sở dữ liệu. Dùng để tạo ra Statement, PreparedStatement và CallableStatement.
- Statement, PreparedStatement, CallableStatement: Chứa lệnh SQL gửi tới cơ sở dữ liệu để thực thi.
- ResultSet – biểu diễn một tập kết quả trong cơ sở dữ liệu tạo ra bởi việc sử dụng một câu lệnh SQL là SELECT.
- SQLException – một lớp xử lý lỗi ngoại lệ chứa các lỗi truy cập cơ sở dữ liệu.

Trình tự làm việc với CSDL sử dụng JDBC

1. Tạo kết nối đến database
2. Gửi SQL query đến database sử dụng JDBC driver tương ứng
3. JDBC driver kết nối đến database
4. Thực thi câu lệnh query để lấy kết quả trả về (số bản ghi lấy được, số bản ghi được update/delete)
5. Gửi dữ liệu đến ứng dụng thông qua Driver Manager
6. Xử lý dữ liệu trả về
7. Đóng (giải phóng) kế nối đến database

Kiểu dữ liệu trong JDBC

- JDBC Driver chuyển đổi kiểu dữ liệu của Java thành kiểu dữ liệu của JDBC tương ứng trước khi gửi giá trị dữ liệu tới Database. Ví dụ, một double trong Java được chuyển đổi thành một SQL DOUBLE.

Kiểu dữ liệu trong JDBC – Ví dụ

SQL	JDBC/Java	setXXX	updateXXX
VARCHAR	java.lang.String	setString	updateString
CHAR	java.lang.String	setString	updateString
LONGVARCHAR	java.lang.String	setString	updateString
BIT	boolean	setBoolean	updateBoolean
NUMERIC	java.math.BigDecimal	setBigDecimal	updateBigDecimal
TINYINT	byte	setByte	updateByte
SMALLINT	short	setShort	updateShort
INTEGER	int	setInt	updateInt
BIGINT	long	setLong	updateLong
REAL	float	setFloat	updateFloat
FLOAT	float	setFloat	updateFloat
DOUBLE	double	setDouble	updateDouble
VARBINARY	byte[ ]	setBytes	updateBytes
BINARY	byte[ ]	setBytes	updateBytes
DATE	java.sql.Date	setDate	updateDate
TIME	java.sql.Time	setTime	updateTime
TIMESTAMP	java.sql.Timestamp	setTimestamp	updateTimestamp
CLOB	java.sql.Clob	setClob	updateClob

Kết nối JDBC với CSDL

- Bước 1: Import các package
- Bước 2: Đăng ký JDBC driver
- Bước 3: Tạo địa chỉ Database URL chính xác
- Bước 4: Tạo đối tượng Connection

Lớp DriverManager trong JDBC

- Lớp DriverManager hoạt động như một giao diện giữa người dùng và các driver. Nó theo dõi các driver có sẵn và xử lý việc thiết lập kết nối giữa một Database và Driver thích hợp. Lớp DriverManager duy trì một danh sách các lớp Driver mà đã được đăng ký bởi chính chúng bằng cách gọi phương thức DriverManager.registerDriver().
- Các phương thức:
  - **public static void registerDriver(Driver driver) throws SQLException**
  - **public static void deregisterDriver(Driver driver)**
  - **public static Connection getConnection(String url)**
  - **public static Connection getConnection(String url,String userName,String password)**
  - **public static void setLoginTimeout(int second)**
  - **public static int getLoginTimeout()**

Connection interface trong JDBC

- Đối tượng Connection biểu diễn ngữ cảnh giao tiếp. Đối tượng của Connection có thể được sử dụng để lấy đối tượng của Statement và DatabaseMetaData. Connection Interface cung cấp nhiều phương thức để quản lý giao tác như commit(), rollback(), …
- Một số phương thức của Connection Interface:
  - **public Statement createStatement()**
  - **public Statement createStatement(int resultSetType,int resultSetConcurrency) throws SQLException**
  - **public void setAutoCommit(boolean autoCommit) throws SQLException**
  - **public void commit() throws SQLException**
  - **public void rollback()**
  - **public void close()**
  - **setSavepoint (String ten) throws SQLException**

Statement interface trong JDBC

- Interface này cung cấp nhiều phương thức để thực thi các truy vấn với cơ sở dữ liệu và trả về kết quả mà nó tạo ra.
- Một số phương thức của Statement interface:
  - **public ResultSet executeQuery(String sql)**
  - **public int executeUpdate(String sql)**
  - **public boolean execute(String sql)**
  - **public int[] executeBatch()**
  - **void close() throws SQLException**

PreparedStatement interface trong JDBC

- PreparedStatement Interface là một interface con của Statement. Nó được sử dụng để thực thi các truy vấn được tham số hóa.
- Một số phương thức của PreparedStatement interface:
  - **public void setInt(int paramIndex, int giaTri)**
  - **public void setString(int paramIndex, String giaTri)**
  - **public void setFloat(int paramIndex, float giaTri)**
  - **public void setDouble(int paramIndex, double giaTri)**
  - **public int executeUpdate()**
  - **public ResultSet executeQuery() throws SQLException**

CallableStatement interface trong JDBC

- CallableStatement Interface được sử dụng để thực thi Stored Procedure

ResultSet interface trong JDBC

- ResultSet là một bảng dữ liệu mà biểu diễn tập kết quả từ cơ sở dữ liệu mà được trả về bởi các lệnh SQL.

Tạo ứng dụng CRUD với JDBC

Kết nối CSDL

```
private String jdbcURL =
"jdbc:mysql://localhost:3306/demo?useSSL=false";
private String jdbcUsername = "root";
private String jdbcPassword = ”123456";

...
Connection connection = null;
try {
    Class.forName("com.mysql.jdbc.Driver");
    connection = DriverManager.getConnection(jdbcURL,
jdbcUsername, jdbcPassword);
} catch (SQLException e) {...}
```

Thao tác đọc

```
private static final String SELECT_ALL_USERS = "select * from users";
…

try (Connection connection = getConnection();
     PreparedStatement preparedStatement =
connection.prepareStatement(SELECT_ALL_USERS);) {
    ResultSet rs = preparedStatement.executeQuery();
    while (rs.next()) {
…
}
} catch (SQLException e) {…}
```

Thao tác thêm mới

```
private static final String INSERT_USERS_SQL = "INSERT INTO users" +
"  (name, email, country) VALUES " + " (?, ?, ?);";

...

try (Connection connection = getConnection(); PreparedStatement
preparedStatement =
connection.prepareStatement(INSERT_USERS_SQL)) {
            preparedStatement.setString(1, user.getName());
            preparedStatement.setString(2, user.getEmail());
            preparedStatement.setString(3, user.getCountry());
            preparedStatement.executeUpdate();
        } catch (SQLException e) {...}
```

Thao tác sửa

```
private static final String UPDATE_USERS_SQL = "update users set
name = ?,email= ?, country =? where id = ?;";
...

try (Connection connection = getConnection(); PreparedStatement
statement = connection.prepareStatement(UPDATE_USERS_SQL);) {
        statement.setString(1, user.getName());
        statement.setString(2, user.getEmail());
        statement.setString(3, user.getCountry());
        statement.setInt(4, user.getId());
        rowUpdated = statement.executeUpdate() > 0;
    }
```

Thao tác xoá

```
private static final String DELETE_USERS_SQL = "delete from
users where id = ?;";

…

try (Connection connection = getConnection();
PreparedStatement statement =
connection.prepareStatement(DELETE_USERS_SQL);) {
        statement.setInt(1, id);
        rowDeleted = statement.executeUpdate() > 0;
    }
```

Tổng kết

- JDBC (Java Database Connectivity) là một chuẩn API (Application Program Interface) cho phép kết nối các chương trình viết bởi Java với các hệ quản trị cơ sở dữ liệu (MySQL, MS SQL, Postgre SQL, Oracle, DB2...)
- JDBC chỉ là một tập các interface, các định nghĩa, thông báo lỗi, đặc tả chứ không phải là thư viện.
- Với mỗi hệ quản trị cơ sở dữ liệu ta sẽ có một cài đặt JDBC riêng cho nó, ví dụ JDBC cho MySQL, JDBC cho MS SQL ...
- JDBC Driver chuyển đổi kiểu dữ liệu của Java thành kiểu dữ liệu của JDBC tương ứng trước khi gửi giá trị dữ liệu tới Database.

Raising the bar