

# **LẬP TRÌNH CSDL VỚI JDBC (JAVA DATABASE CONNECTIVITY)**

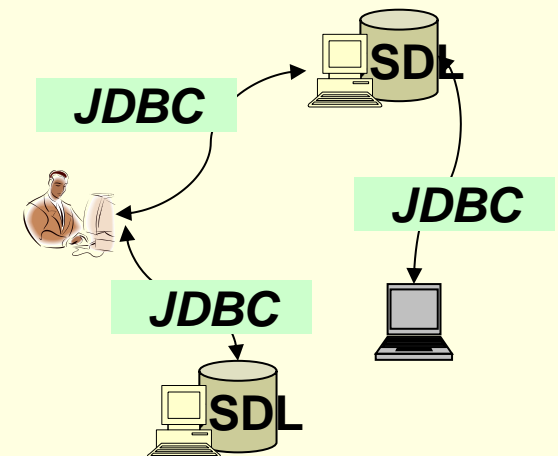
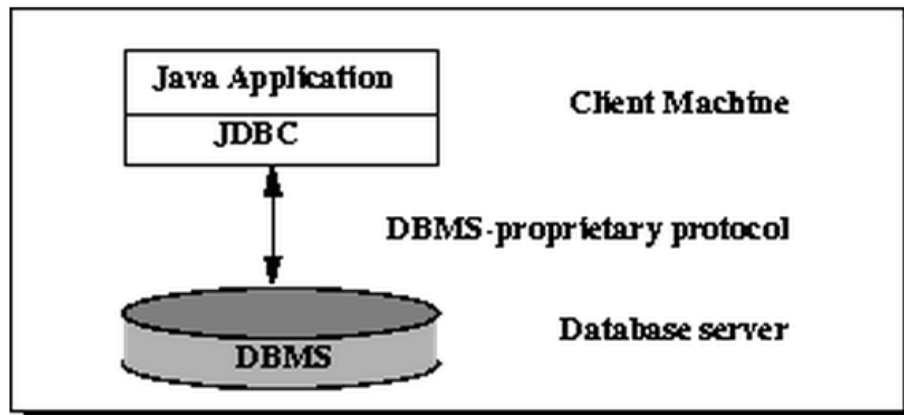
# NỘI DUNG

---

- ❖ Giới thiệu JDBC.
- ❖ Kiến trúc JDBC.
- ❖ Các JDBC Component.
- ❖ JDBC Drivers.
- ❖ Các bước làm việc với Database dùng JDBC.
- ❖ Đối tượng ResultSet & Cursor.
- ❖ Statement & PreparedStatement.
- ❖ Quản lý Transaction.
- ❖ Chương trình thao tác CSDL theo kiến trúc MVC.

# Giới thiệu về JDBC

- ❑ JDBC (Java DataBase Connectivity) API là Java API để truy xuất các nguồn dữ liệu dạng bảng, đặc biệt là dữ liệu lưu trong các CSDL quan hệ như MS Access, SQL Server, Oracle,... (JDBC API độc lập database)
- ❑ JDBC API cung cấp API truy xuất CSDL dựa trên SQL.
- ❑ `java.sql.*`, `javax.sql.*`.



# Giới thiệu về JDBC

---

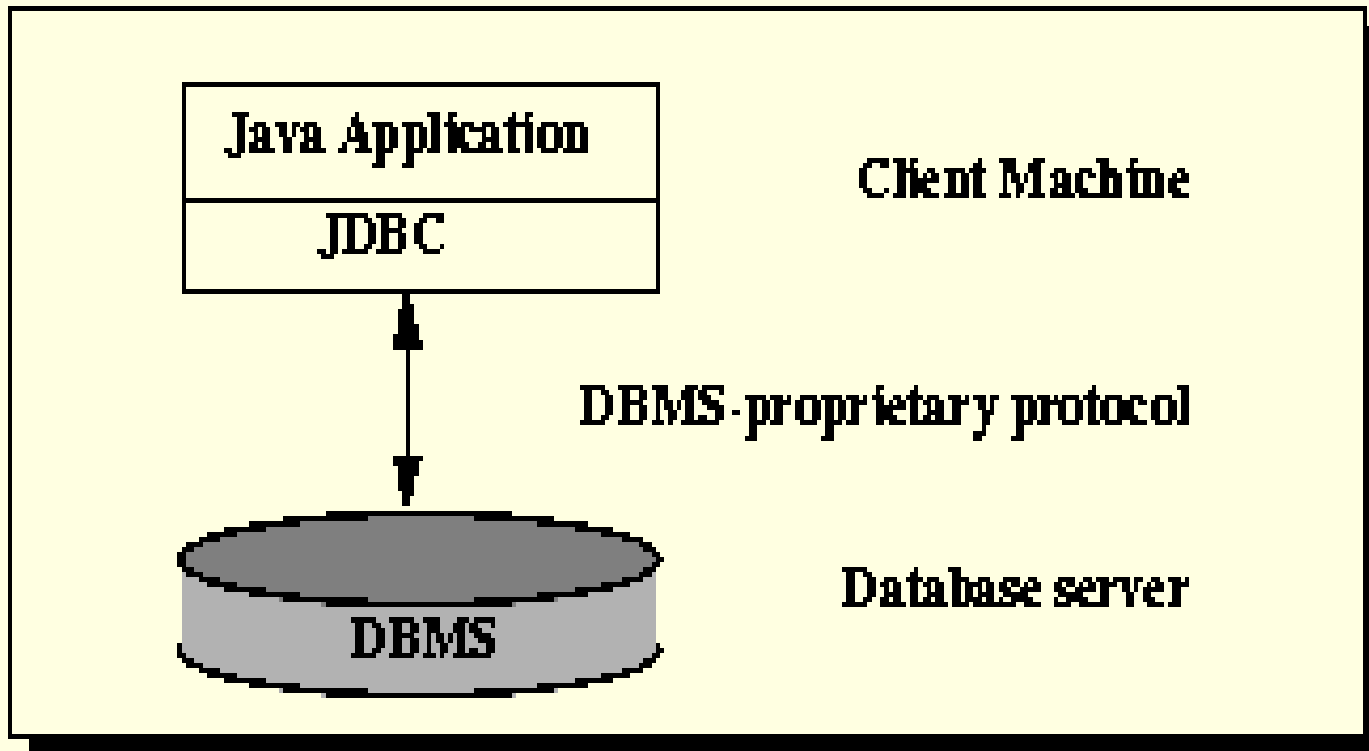
- ❑ JDBC giúp viết các Java Application quản lý các hoạt động lập trình.
  - ❑ *Kết nối nguồn dữ liệu, CSDL.*
  - ❑ *Gửi các câu lệnh truy vấn đến CSDL.*
  - ❑ *Truy vấn và xử lý kết quả trả về từ CSDL.*

# Ví dụ minh họa

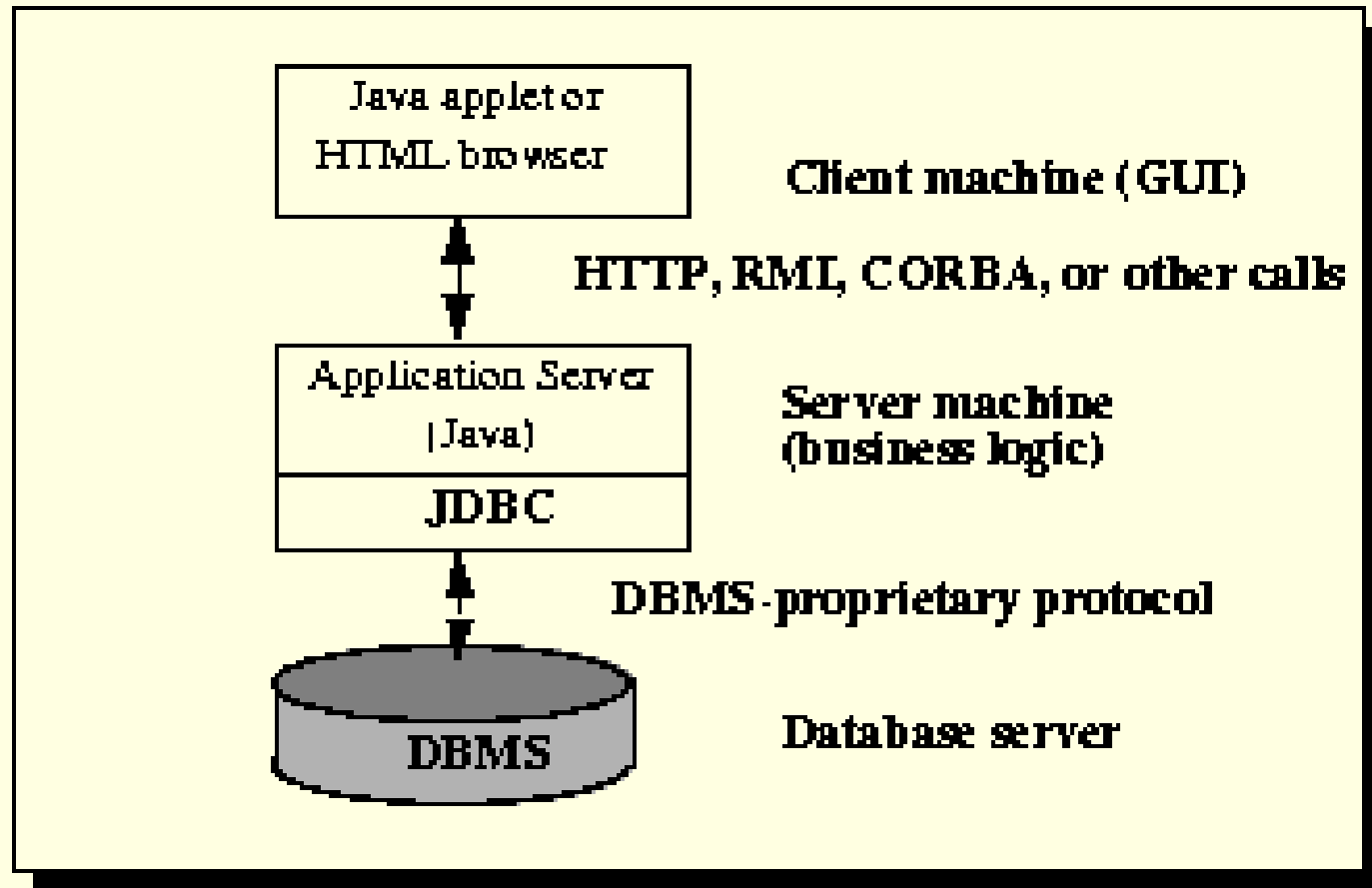
```
public void connectToAndQueryDatabase(String username, String password) {  
  
    Connection con = DriverManager.getConnection(  
        "jdbc:mysql:myDatabase",  
        username,  
        password);  
  
    Statement stmt = con.createStatement();  
    ResultSet rs = stmt.executeQuery("SELECT a, b, c FROM Table1");  
  
    while (rs.next()) {  
        int x = rs.getInt("a");  
        String s = rs.getString("b");  
        float f = rs.getFloat("c");  
    }  
}
```

# Kiến trúc JDBC

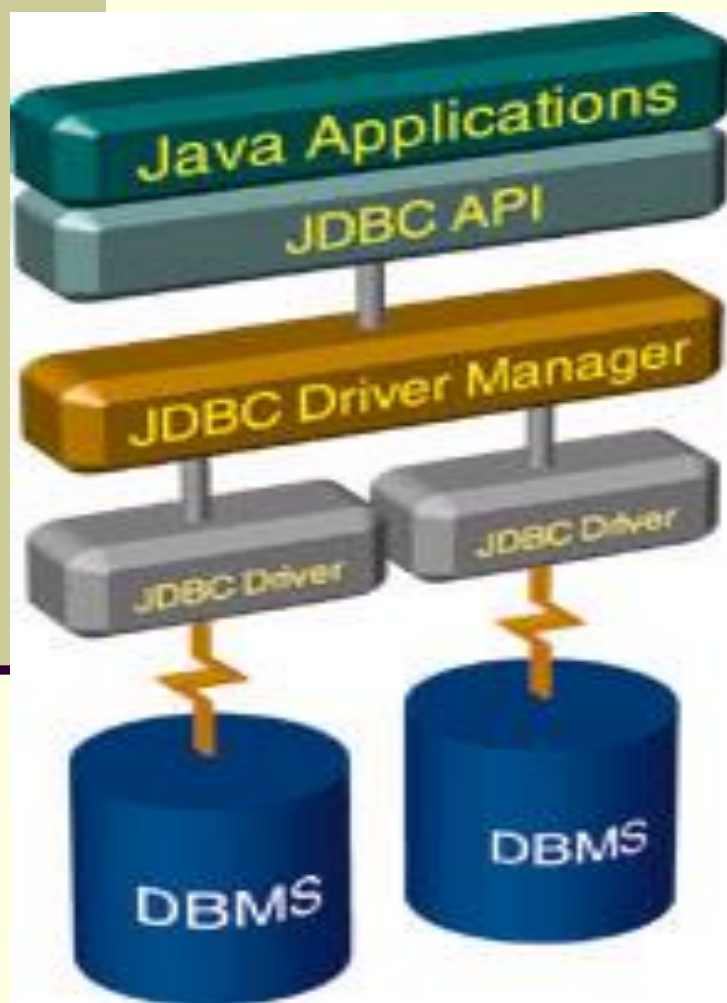
- ❑ Mô hình xử lý 2-tier và 3-tier: JDBC API hỗ trợ truy xuất database cho cả các mô hình xử lý 2-tier và 3-tier.



# Kiến trúc JDBC (tt)



# Các thành phần JDBC



- **JDBC API:** là một API hoàn toàn dựa trên Java, giúp truy xuất CSDL từ NNLT Java.
- **JDBC Diver Manager:** một class giúp kết nối giữa Java Application đến các JDBC Driver.
- **JDBC Driver:** phần mềm cho phép Java Application tương tác với các CSDL khác nhau.



# JDBC Drivers

---

- Sun cung cấp đặc tả là các JDBC interface (*Connection*, *PreparedStatement*, *Statement*, *ResultSet*, ...). Tập hợp các class hiện thực các JDBC interface đối với một database engine cụ thể được gọi là JDBC driver.
- Do các hãng xây dựng DBMS hoặc một đơn vị thứ 3 khác cung cấp.

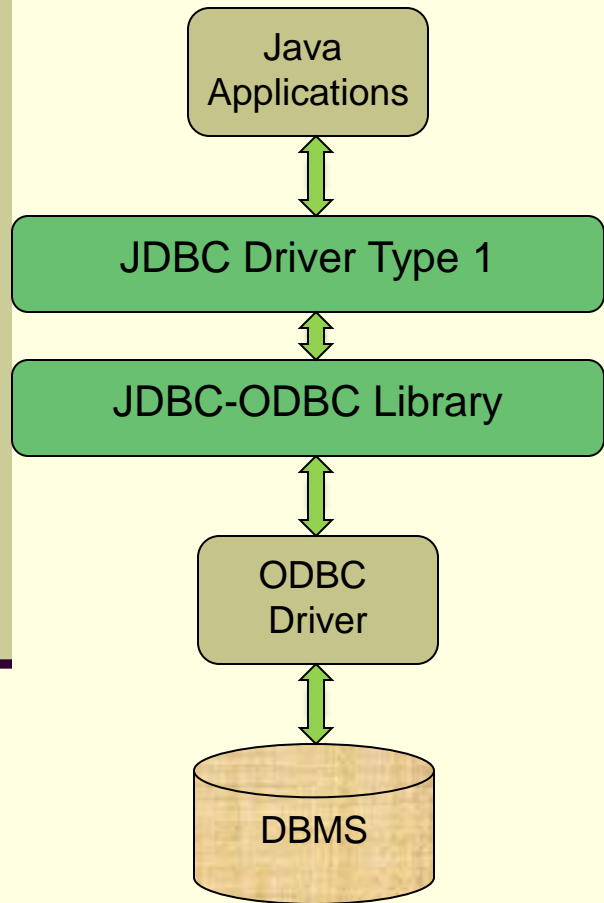
(Tham khảo: <http://industry.java.sun.com/products/jdbc/drivers>, truy cập lần cuối 01/2008)

# Các loại JDBC Drivers

---

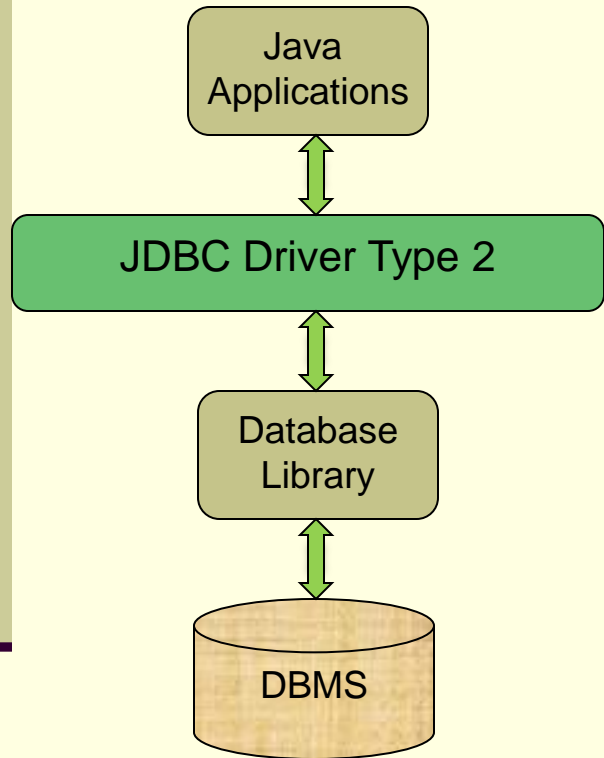
- 1) *JDBC-ODBC Bridge plus ODBC Driver*
- 2) *A native API partly Java technology-enabled driver*
- 3) *Pure Java Driver for Database Middleware*
- 4) *Direct-to-Database Pure Java Driver*

# Type 1: JDBC-ODBC Bridge



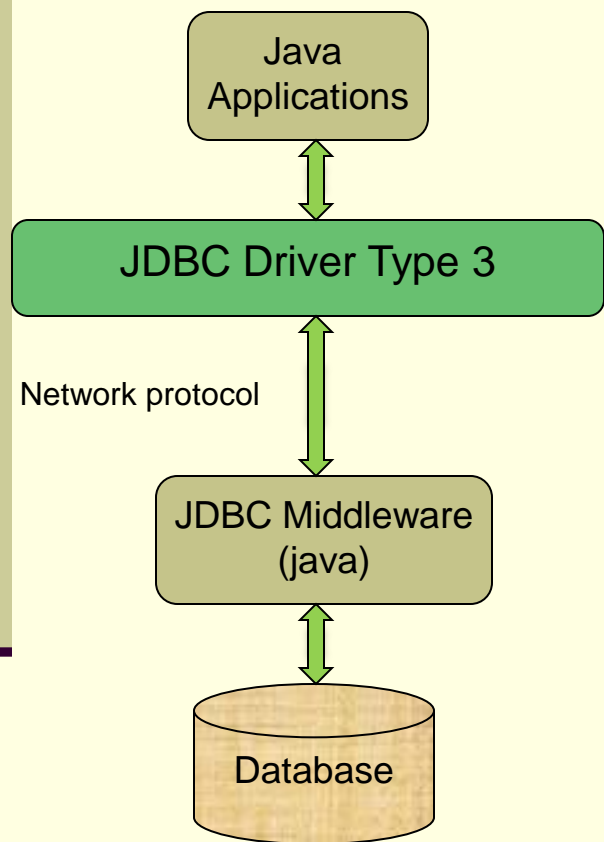
- Dùng truy cập ODBC driver trên máy client. Cần cấu hình trên máy client 1 Data Source Name (DNS) thể hiện database đích.
- **Thuận lợi:**
  - Dễ dùng,
  - Hầu hết các DBMS hỗ trợ ODBC, nên Type1 rất hữu ích khi JAVA mới ra.
- **Hạn chế:** chậm, do phải chuyển từ lời gọi JDBC sang lời gọi ODBC.

## Type 2: Java to Native API



- Type-2 dùng thư viện client-side của Database.
- Driver Type-2 có nhiệm vụ chuyển lời gọi JDBC thành các lời gọi Database API.
- Driver Type-2 không viết hoàn toàn bằng Java. Thường được cung cấp bởi Database vendors.
- **Ưu điểm:** tốt hơn JDBC-ODBC
- **Hạn chế:** Driver Type-2 và thư viện Client-side của database cần cài đặt trên máy client.

# Type 3: Network Protocol Driver



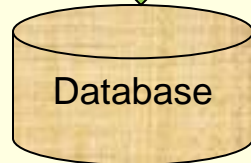
- Dùng Network protocol để giao tiếp với JDBC middleware trên server.
- Chuyển đổi các lời gọi JDBC thành giao thức mạng. Sau đó, một phần mềm trung gian (middleware) chạy trên máy server chuyển đổi giao thức mạng thành giao thức DBMS đặc thù. Sự chuyển này đặt ở phía server mà không đòi hỏi cài đặt trên máy tính client.
- **Ưu điểm:** không cần thư viện phía client

# Type 4: Java to Database Protocol

Java  
Applications



JDBC Driver Type 4



Database

- Drivers thuần java, dùng database protocol để giao tiếp trực tiếp với database.
- Chuyển lời gọi JDBC thành các lời gọi giao thức database đặc thù.
- Hầu hết do Database vendor cung cấp.
- **Ưu điểm:**
  - Nhanh nhất so với các loại khác.
  - Không cần thư viện phía client, phía server
- **Hạn chế:** Phụ thuộc Database

# Các loại JDBC Drivers

**A List of JDBC Driver Vendors**

| <b>Vendor</b>                     | <b>Type</b> | <b>Supported Databases</b>   |
|-----------------------------------|-------------|--|
| Agave Software Design             | 3           | Oracle, Sybase, Informix, ODBC supported databases                                   |
| Asgard Software                   | 3           | Unisys A series DMSII database   |
| Borland                           | 4           | InterBase 4.0  |
| Caribou Lake Software             | 3           | CA-Ingres  |
| Center for Imaginary Environments | 4           | mSQL   |
| Connect Software                  | 4           | Sybase, MS SQL Server  |
| DataRamp                          | 3           | ODBC supported databases   |
| IBM                               | 2/3         | IBM DB2 Version 2  |
| IDS Software                      | 3           | Oracle, Sybase, MS SQL Server, MS Access, Informix, Watcom, ODBC supported databases |
| InterSoft                         | 3           | Essentia   |
| Intersolv                         | 2           | Oracle, Sybase   |
| JavaSoft                          | 1           | ODBC supported databases   |
| OpenLink                          | 3           | CA-Ingres, Postgres95, Progress, Unify   |
| SAS                               | 4           | SAS, and via SAS/ACCESS, Oracle, Informix, Ingres, and ADABAS                        |
| SCO                               | 3           | Informix, Oracle, Ingres, Sybase, Interbase  |
| StormCloud Development            | 3           | ODBC supported databases   |
| Sybase                            | 3/4         | Sybase SQL Server, SQL Anywhere, Sybase IQ, Replication Server                       |
| Symantec                          | 3           | Oracle, Sybase, MS SQL Server, MS Access, Watcom, ODBC supported databases           |
| Visigenic                         | 3           | ODBC supported databases   |
| WebLogic                          | 2/3         | Oracle, Sybase, MS SQL Server/ODBC supported databases                               |

# JDBC API

The screenshot displays the Java Platform Standard Ed. 6 API documentation in Mozilla Firefox. The browser window shows the file path `file:///C:/Sun/SDK/jdk/docs/api/index.html`. The page title is "Java™ Platform Standard Ed. 6". The navigation bar includes links for Overview, Package, Class, Use, Tree, Deprecated, Index, and Help. The main content area shows the "Package javax.sql" page, which provides the API for server side data source access and processing from the Java™ programming language. The page includes a "See: Description" link and an "Interface Summary" table.

**Package javax.sql**

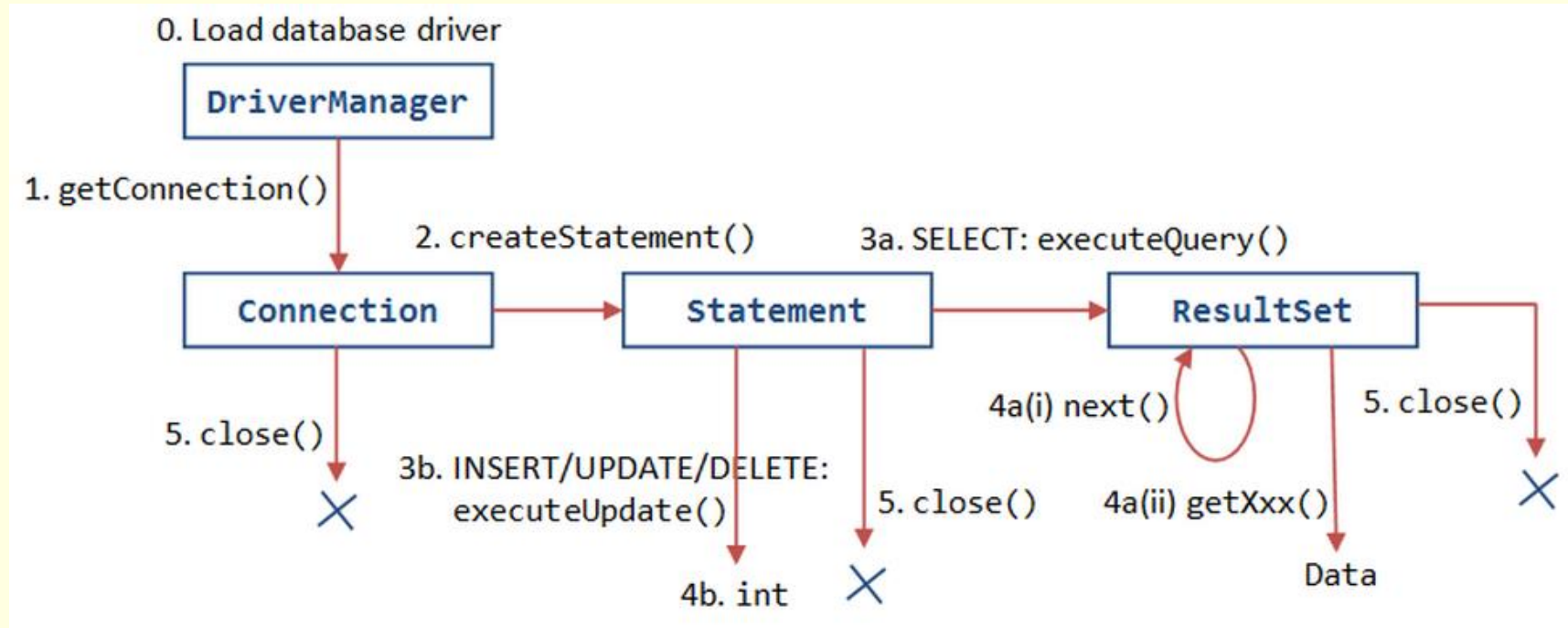
Provides the API for server side data source access and processing from the Java™ programming language.

See: [Description](#)

| Interface Summary                        |  |
|--|--|
| <a href="#">CommonDataSource</a>         | Interface that defines the methods which are common between DataSource, XADataSource and ConnectionPoolDataSource. |
| <a href="#">ConnectionEventListener</a>  | An object that registers to be notified of events generated by a PooledConnection object.                          |
| <a href="#">ConnectionPoolDataSource</a> | A factory for PooledConnection objects.  |
| <a href="#">DataSource</a>               | A factory for connections to the physical data source that this  |



# Các bước làm việc với CSDL



Nguồn hình vẽ: [http://www.ntu.edu.sg/home/ehchua/programming/java/JDBC\\_Basic.html](http://www.ntu.edu.sg/home/ehchua/programming/java/JDBC_Basic.html)

# Đăng ký/nạp JDBC driver

---

- ❑ Dùng phương thức `forName()` của lớp `Class` để đăng ký.

*public static void **forName**(String string) throws ClassNotFoundException*

*Ví dụ:*

*Class.forName("oracle.jdbc.driver.**OracleDriver**");*

*Class.forName("com.mysql.jdbc.**Driver**");*

*Class.forName("com.microsoft.jdbc.sqlserver.**SQLServerDriver**");*

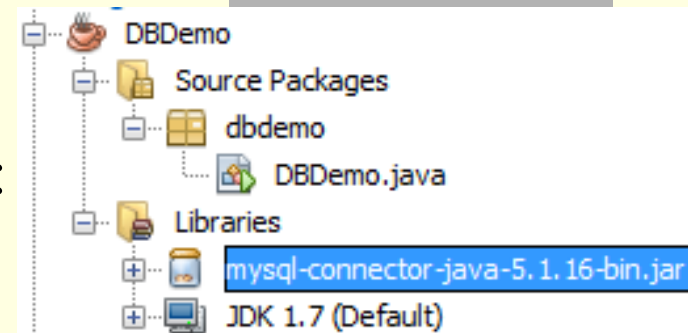
- ❑ **Lưu ý:** Driver kiểu 3 tự động nạp

# Đăng ký/nạp JDBC driver

Driver được cài đặt trong **JAR file**.

JAR phải được khai báo trong **classpath**:

1. Thêm *jar file* to vào IDE project
2. Thêm JAR file vào CLASSPATH  
**CLASSPATH = /my/path/mysql-connector.jar;.**
3. Thêm JAR sử dụng Java command line:  
**java -cp /my/path/mysql-connector.jar ...**
4. Đặt JAR file vào **JRE/lib/ext** directory:  
**C:/java/jre1.6.0/lib/ext/mysql-connector.jar**



# Thiết lập kết nối

❑ Phương thức getConnection() của lớp DriverManager dùng để thiết lập kết nối với database.

❑ Ví dụ:

```
Connection con=DriverManager.getConnection(  
"jdbc:oracle:thin:@localhost:1521:DBName","user","password");
```

```
Connection con=DriverManager.getConnection(  
"jdbc:mysql://localhost:3306/DBName?useUnicode=true",  
"user","password");
```

```
Connection con=DriverManager.getConnection(  
"jdbc:microsoft:sqlserver://localhost:1433; DatabaseName=DBName;  
User=sa; Password=sa");
```

# Thiết lập kết nối

| RDBMS                | Database URL format   |
|----------------------|---|
| MySQL                | <code>jdbc:mysql://hostname:portNumber/databaseName</code>  |
| ORACLE               | <code>jdbc:oracle:thin:@hostname:portNumber:databaseName</code>   |
| DB2                  | <code>jdbc:db2:hostname:portNumber/databaseName</code>  |
| Java DB/Apache Derby | <code>jdbc:derby:databaseName</code> (embedded)<br><code>jdbc:derby://hostname:portNumber/databaseName</code> (network) |
| Microsoft SQL Server | <code>jdbc:sqlserver://hostname:portNumber;databaseName=databaseName</code>   |
| Sybase               | <code>jdbc:sybase:Tds:hostname:portNumber/databaseName</code>   |

# Tạo đối tượng thực hiện câu lệnh

---

- ❑ Phương thức *createStatement()* của *Connection* interface được dùng để tạo đối tượng *Statement*.
- ❑ Ví dụ:

```
Connection con=DriverManager.getConnection(...);  
Statement stmt=con.createStatement();
```

# Thực hiện câu truy vấn SQL & Xử lý kết quả trả về

- ❑ Dùng phương thức *executeQuery()* của *Statement* interface.

*public ResultSet executeQuery(String sql) throws SQLException*

- ❑ Ví dụ:

```
ResultSet rs = stmt.executeQuery("select * from emp");  
while(rs.next()) {  
    System.out.println(rs.getInt(1) + " " + rs.getString(2));  
}
```

- ❑ *executeUpdate()*: dùng cho câu lệnh SQL liên quan UPDATE, INSERT, DELETE (không trả về ResultSet)

# Đóng kết nối

---

❑ Đóng đối tượng *Connection*, thì đối tượng *Statement* và *ResultSet* sẽ tự động đóng.

❑ Ví dụ:

*con.close();*



# Đóng kết nối

- ❑ Khuyến cáo nên đóng connection sau khi hoàn tất

```
Connection connection =  
DriverManager.getConnection(...);  
/* use the database */  
...  
/* done using database */  
public void close( ) {  
    if ( connection == null ) return;  
    try {  
        connection.close();  
    }  
    catch ( SQLException sqle ) { /* ignore it */ }  
    finally { connection = null; }  
}
```

# Làm việc với ResultSet & Cursor

---

- ❑ **ResultSet** chứa các "rows" thỏa hay trả về từ câu query.  
Là đối tượng dạng bảng trong RAM, có gắn với **Cursor**.
- ❑ **Cursor** được sinh ra khi một **ResultSet** được sinh ra.
- ❑ **Cursor** cho phép chúng ta có thể xử lý một **ResultSet** từ trên xuống dưới hoặc từ dưới lên (theo cả 2 hướng) hoặc đi đến 1 dòng chỉ định nào đó.

# Làm việc với ResultSet & Cursor (tt)

- **ResultSet** hỗ trợ các phương thức để lấy dữ liệu từ cột:
  - "get" by column number -- starts at 1 (not 0)!
  - "get" by column name -- field names in table/query.

```
String query = "SELECT * FROM Country WHERE ...";  
ResultSet rs = statement.executeQuery( query );
```

```
// go to first row of results
```

```
rs.first( );
```

```
// display the values
```

```
System.out.println( rs.getString( 1 ) );
```

```
System.out.println( rs.getInt( "population" ) );
```

get by column number

get by name

# Làm việc với ResultSet & Cursor (tt)

- ResultSet hỗ trợ các phương thức để lấy từng dòng và cột trong kết quả trả về

## ResultSet

**next() : boolean**

**previous() : boolean**

**first() : boolean**

**last() : boolean**

**absolute( k )**

**getInt( name: String )**

**getInt( index: int )**

...

go to next row of results. "false" if no more.

go to previous row. "false" if 1st result.

go to first row of results.

go to last row of results.

go to k-th row of results.

get int value of field "name"

get int value of k-th column in a record

# Làm việc với ResultSet & Cursor (tt)

Các phương thức get(...) trả về dữ liệu cột:

getLong(3): get by column index (most efficient)

getLong("population"): get by field name (safest)

|                  |             |                                     |
|------------------|-------------|-------------------------------------|
| getInt( )        | getLong( )  | - get Integer field value           |
| getFloat( )      | getDouble() | - get floating pt. value            |
| getString( )     |             | - get Char or Varchar field value   |
| getDate( )       |             | - get Date or Timestamp field value |
| getBoolean( )    |             | - get a Bit field value             |
| getBytes( )      |             | - get Binary data                   |
| getBigDecimal( ) |             | - get Decimal field as BigDecimal   |
| getBlob( )       |             | - get Binary Large Object           |
| getObject( )     |             | - get any field value               |

# Ví dụ: đăng nhập hệ thống

```
public boolean login(String username, String password) throws SQLException {  
    Connection con = null;  
    boolean isExistedUser = false;  
    try {  
        Class.forName("com.mysql.jdbc.Driver");  
        con = DriverManager.getConnection("jdbc:mysql://localhost:3306/dbtest","root","root");  
        Statement stmt = con.createStatement();  
        ResultSet rs = stmt.executeQuery("SELECT * FROM USERACCOUNT WHERE username = '"  
            + username + "'" + " AND password = '" + password + "'");  
        if (rs != null && rs.next())  
            isExistedUser = true;  
        con.close();  
    }  
    catch (ClassNotFoundException ex) {  
        ex.printStackTrace();  
    }  
    finally {  
        if (con != null) {  
            con.close();  
        }  
    }  
    return isExistedUser;  
}
```

*Vấn đề SQL Injection*

# Đối tượng PreparedStatement

---

- ❑ *PreparedStatement* kế thừa từ *Statement*.
- ❑ Thường được dùng với các câu lệnh SQL có tham số → Thay vì thực thi đối tượng *Statement* nhiều lần, có thể dùng *PreparedStatement*.
- ❑ Đối tượng *PreparedStatement* chứa 1 câu lệnh SQL đã được biên dịch trước khi gửi đến DBMS, khác với *Statement* → Tránh được các lỗi *SQL injection*.

# Ví dụ: PreparedStatement

```
public void updateCoffeeSales(HashMap<String, Integer> salesForWeek)
    throws SQLException {

    PreparedStatement updateSales = null;
    PreparedStatement updateTotal = null;

    String updateString =
        "update " + dbName + ".COFFEES " +
        "set SALES = ? where COF_NAME = ?";

    String updateStatement =
        "update " + dbName + ".COFFEES " +
        "set TOTAL = TOTAL + ? " +
        "where COF_NAME = ?";
```



# Ví dụ: PreparedStatement

---

```
try {
    con.setAutoCommit(false);
    updateSales = con.prepareStatement(updateString);
    updateTotal = con.prepareStatement(updateStatement);

    for (Map.Entry<String, Integer> e : salesForWeek.entrySet()) {
        updateSales.setInt(1, e.getValue().intValue());
        updateSales.setString(2, e.getKey());
        updateSales.executeUpdate();
        updateTotal.setInt(1, e.getValue().intValue());
        updateTotal.setString(2, e.getKey());
        updateTotal.executeUpdate();
        con.commit();
    }
} catch (SQLException e ) {
```

# Ví dụ: PreparedStatement

```
JDBCTutorialUtilities.printStackTrace(e);
if (con != null) {
    try {
        System.err.print("Transaction is being rolled back");
        con.rollback();
    } catch (SQLException excep) {
        JDBCTutorialUtilities.printStackTrace(excep);
    }
}
} finally {
    if (updateSales != null) {
        updateSales.close();
    }
    if (updateTotal != null) {
        updateTotal.close();
    }
    con.setAutoCommit(true);
}
}
```

# Ví dụ: PreparedStatement

---

- ❑ Dùng Loop để *reset* giá trị các tham số → dùng lại đối tượng *PreparedStatement*.
- ❑ *con.setAutoCommit(false)*: không câu lệnh SQL nào được commit cho đến khi commit được gọi → đảm bảo tính toàn vẹn dữ liệu.
- ❑ *Rollback* khi có ngoại lệ *SQLException*.
- ❑ Giải phóng tài nguyên trong khối *finally*.

# Làm việc với Stored Procedure

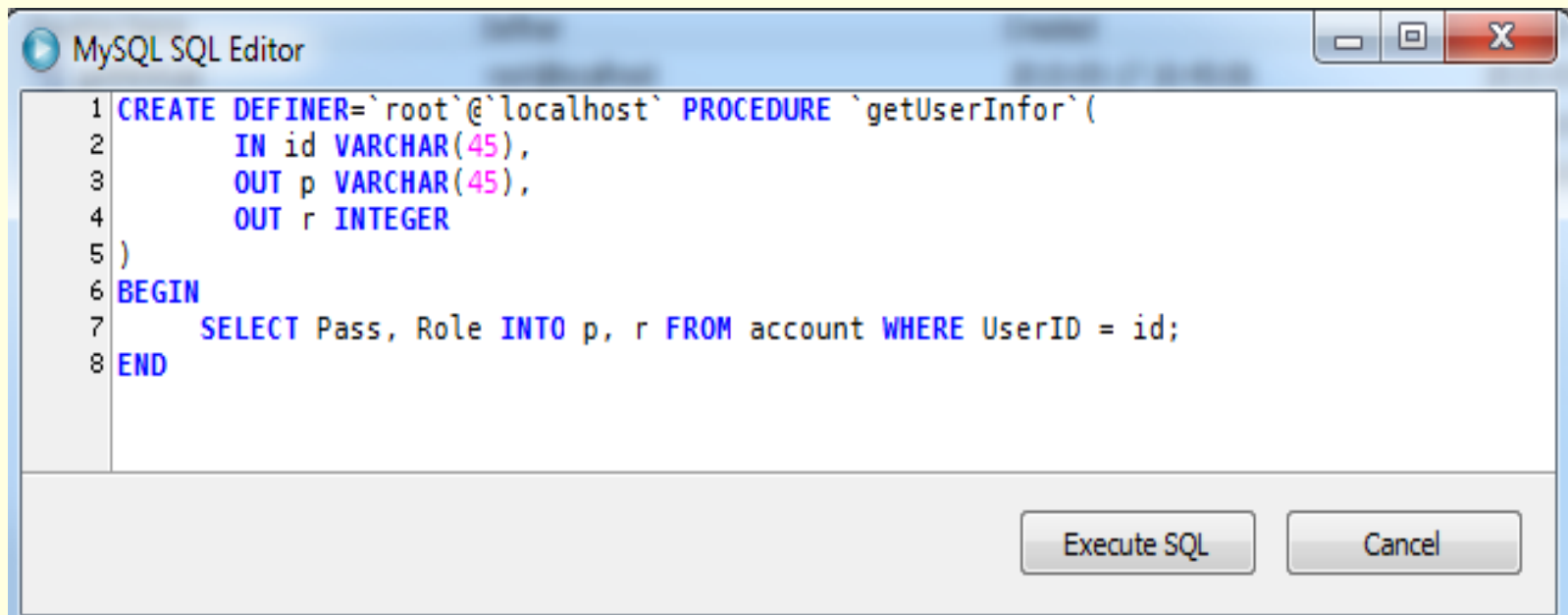
---

## ❑ **Stored Procedure:**

- ❑ Là nhóm các câu lệnh SQL có thể được gọi bằng 1 tên.
- ❑ Đoạn mã, chương trình con thực hiện 1 công việc nào đó được gọi như một phương thức hàm.
- ❑ Trước đây các stored procedure được viết theo ngôn ngữ riêng của các DBMS, các DBMS thế hệ mới cho phép viết các store procedure dùng Java và JDBC API.

# Làm việc với Stored Procedure (tt)

- ❑ Tạo Store Procedure trong MySQL.



The screenshot shows a window titled "MySQL SQL Editor" with a standard Windows interface (minimize, maximize, close buttons). The editor contains the following SQL code:

```
1 CREATE DEFINER='root'@'localhost' PROCEDURE `getUserInfor`(  
2     IN id VARCHAR(45),  
3     OUT p VARCHAR(45),  
4     OUT r INTEGER  
5 )  
6 BEGIN  
7     SELECT Pass, Role INTO p, r FROM account WHERE UserID = id;  
8 END
```

At the bottom right of the window, there are two buttons: "Execute SQL" and "Cancel".

# Làm việc với Stored Procedure (tt)

- ❑ Dùng đối tượng JDBC *CallableStatements* để gọi thực thi một stored procedure trong java.

```
private void callStoredProc() throws SQLException {
    Connection dbConnection = getDBConnection();
    CallableStatement callableStatement = null;
    String getUserInformation = "{call getUserInfor(?,?,?)}";
    try {
        callableStatement = dbConnection.prepareCall(getUserInformation);
        callableStatement.setString(1, "admin");
        callableStatement.registerOutParameter(2, java.sql.Types.VARCHAR);
        callableStatement.registerOutParameter(3, java.sql.Types.INTEGER);
        callableStatement.executeUpdate();
        String pass = callableStatement.getString(2);
        int role = callableStatement.getInt(3);
        // ....
    } catch (SQLException e) {
        System.out.println(e.getMessage());
    } finally {
        Close connection
    }
}
```

# Làm việc với Transaction

---

- ❑ Là cơ chế đảm bảo tính toàn vẹn của dữ liệu trong database.
- ❑ Một **Transaction** mặc định bắt đầu khi có một câu lệnh DML (Data Manipulation Language) như SELECT, INSERT, UPDATE, DELETE hoặc DDL (Data Definition Language) như CREATE TABLE, CREATE INDEX, ...
- ❑ Nếu **AutoCommit = TRUE/ON** thì Transaction sẽ tự động *commit* sau mỗi câu lệnh INSERT, UPDATE, DELETE. → như vậy cần thiết lập **FALSE/OFF** khi cần kiểm soát nhiều Transaction.

# Làm việc với Transaction

- ❑ Ví dụ chuyển tiền từ người này sang người khác, không quản lý Transaction

```
$balance = "GET BALANCE FROM your ACCOUNT";  
if ($balance < $amount_being_paid) {  
    charge_huge_overdraft_fees();  
}  
$balance = $balance - $amount_being_paid;  
UPDATE your ACCOUNT SET BALANCE = $balance;
```

```
$balance = "GET BALANCE FROM receiver ACCOUNT"  
charge_insane_transaction_fee();  
$balance = $balance + $amount_being_paid  
UPDATE receiver ACCOUNT SET BALANCE = $balance
```



# Làm việc với Transaction

---

- ❑ **Transaction** kết thúc với **commit** hoặc **rollback**.
- ❑ **Commit**: cho kết quả thay đổi từ các câu DQL trong giao tác.
- ❑ **Rollback**: phục hồi tất cả các thay đổi gây ra do các câu SQL trong giao tác.

# XÂY DỰNG ỨNG DỤNG KẾT NỐI CSDL THEO KIẾN MVC

# Kiến trúc MVC là gì?

## Tại sao MVC?

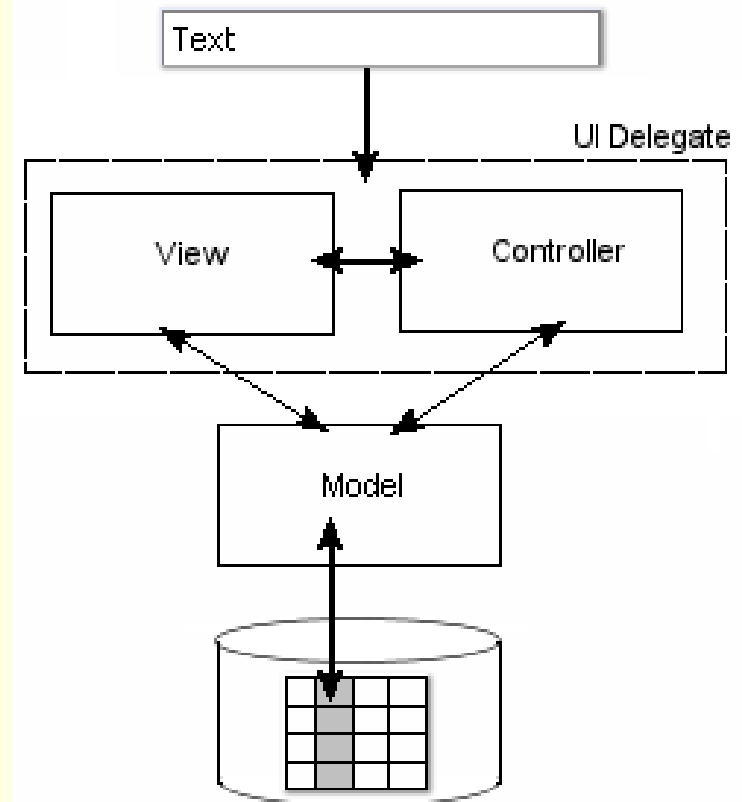
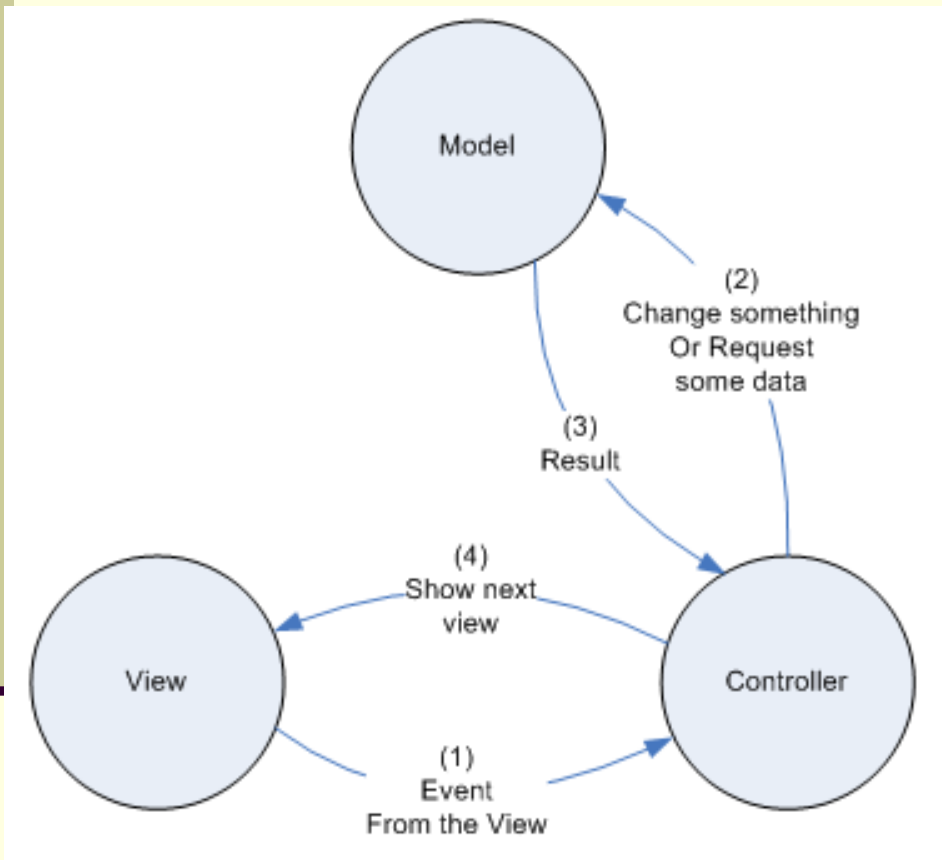
---

- MVC: Model-View-Controller
- Trygve Reenskaug, a Smalltalk developer ở the Xerox Palo Alto Research Center, 1979.
- Tách biệt Data Access và Business Logic với cách mà data được hiển thị cho user.
  - **Model**: mô hình biểu diễn data và các rules chi phối việc truy cập và cập nhật data. Model có thể xem như quá trình xử lý thế giới thực.
  - **View**: chỉ định data nên được hiển thị như thế nào.
  - Controller: chuyển những tương tác của user với Views thành những hành động mà Model sẽ thực hiện.

(Ref: <http://www.oracle.com/technetwork/articles/javase/index-142890.html>)

# Kiến trúc MVC là gì?

## Tại sao MVC?



Nguồn: <http://stackoverflow.com/questions/5966905/which-mvc-diagram-is-correct-web-app>,  
<http://www.oracle.com/technetwork/articles/nimphius-mills-swing-jsf-092891.html>

Truy cập 10/05/2014

# Tương tác giữa các MVC component

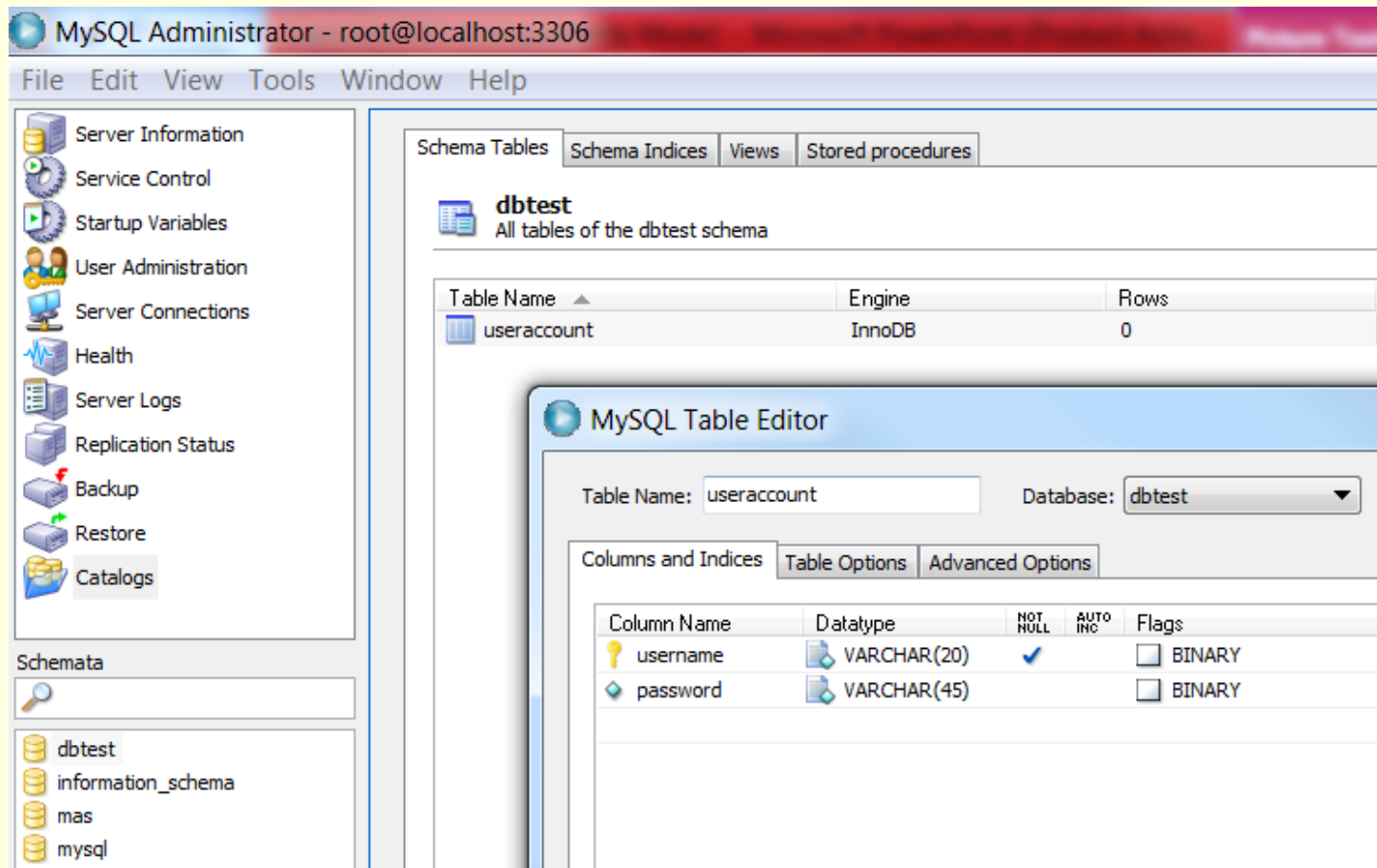
---

- Mọi user actions thực hiện trên Views, sẽ gọi các phương thức listener được đăng ký trong các lớp Controller. Khi User tương tác với GUI, những hành động sau sẽ xuất hiện:
  - 1) GUI nhận diện hành động: nhấn nút, kéo chuột, ... Dùng phương thức listener đã đăng ký.
  - 2) GUI sẽ gọi phương thức tương ứng trong Controller
  - 3) Controller truy xuất Model, cập nhật, xử lý tương ứng với thao tác người dùng.

# Ứng dụng MVC như thế nào? (1)

## ■ Xây dựng ứng dụng minh họa

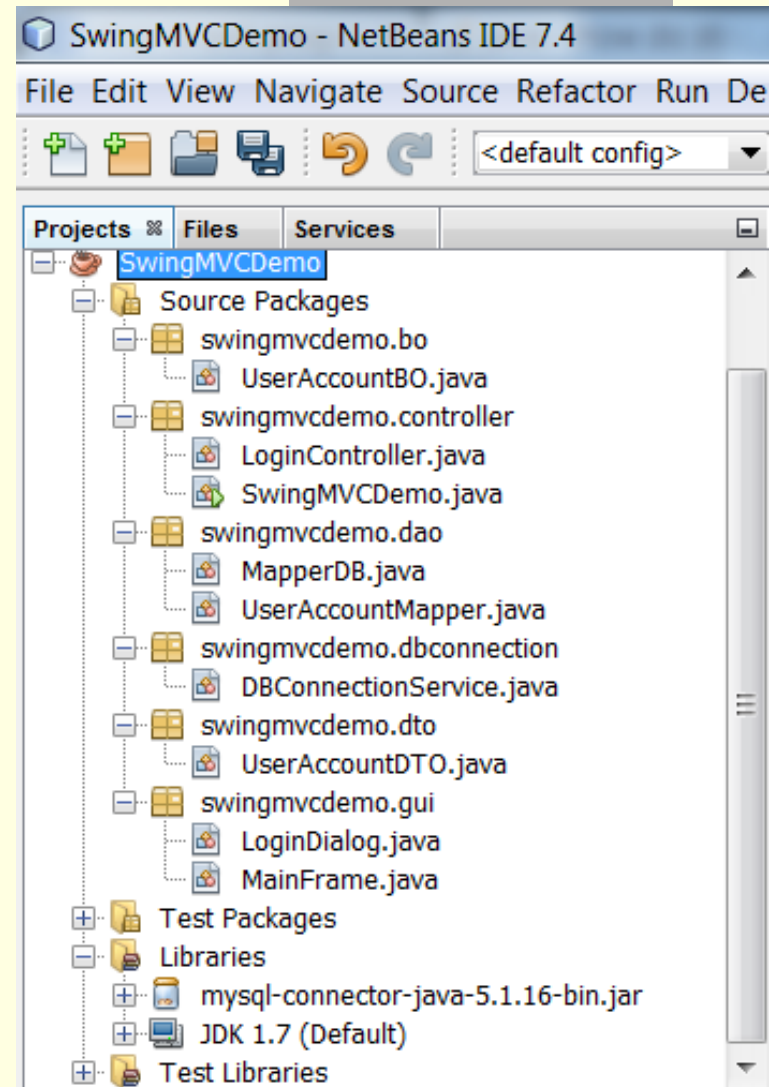
1) Tạo CSDL: chỉ 1 table UserAccount để minh họa



# Ứng dụng MVC như thế nào? (2)

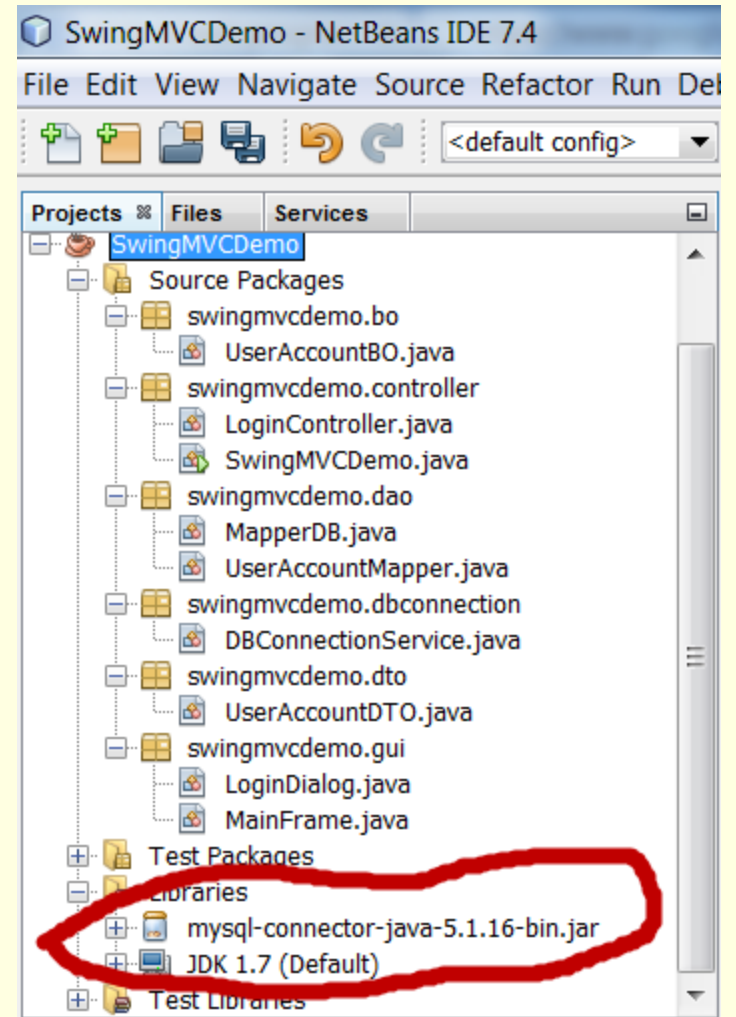
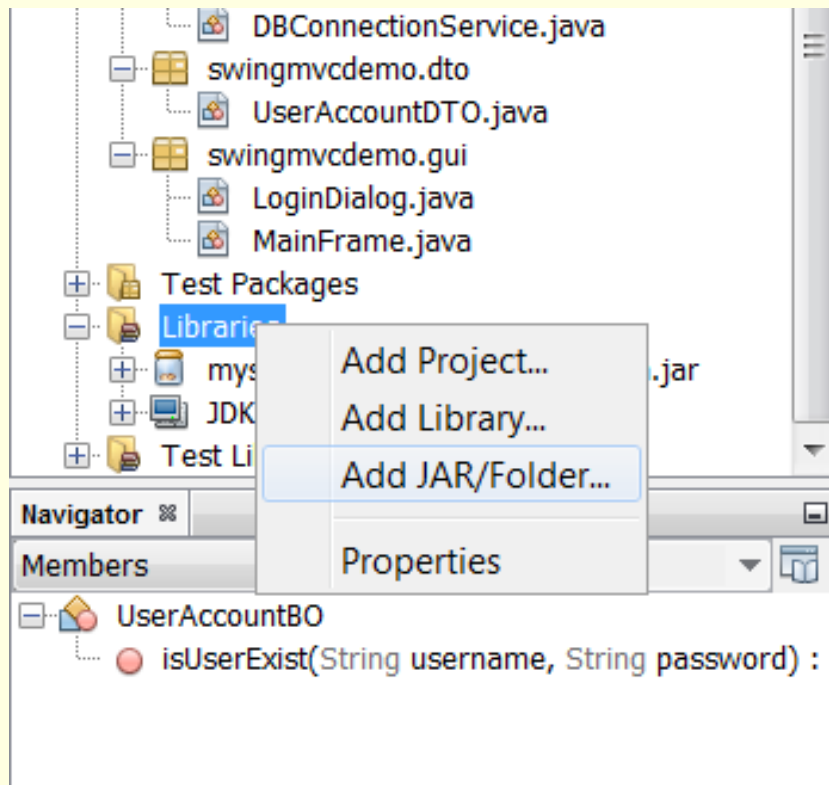
## 2. Tạo các packages tương ứng

- GUI
- Controller
- BO (Business Object)
- DAO (Data Access Object)
- DTO (Data Transfer Object)



# Ứng dụng MVC như thế nào? (3)

## 3. Đăng ký thư viện JDBC Driver tương ứng với DBMS





# Ứng dụng MVC như thế nào? (4)

## 4. Xây dựng Kết nối CSDL “DBConnectionService”

```
package swingmvcdemo.dbconnection;
+ import ...6 lines

public class DBConnectionService {
    public static String dbURL;
    public static String dbUserName;
    public static String dbPassword;
    protected static void loadJDBCdriver() throws Exception {
        try {
            Class.forName("com.mysql.jdbc.Driver");
        } catch (java.lang.ClassNotFoundException e) {
            throw new Exception("SQL JDBC Driver not found ...");
        }
    }

    public static Connection getConnection() throws Exception {
        Connection connect = null;
        dbURL = "jdbc:mysql://localhost:3306/DBTest?useUnicode=true&characterEncoding=UTF-8";
        dbUserName = "root";
        dbPassword = "root";
        if (connect == null) {
            loadJDBCdriver();
            try {
                connect = DriverManager.getConnection(dbURL, dbUserName, dbPassword);
            } catch (java.sql.SQLException e) {
                throw new Exception("Can not access to Database Server ..." + dbURL + e.getMessage());
            }
        }
        return connect;
    }
}
```

# Ứng dụng MVC như thế nào? (5)

## 5. Xây dựng Lớp MapperDB

```
1 package swingmvcdemo.dao;
2 import java.sql.Connection;
3 import swingmvcdemo.dbconnection.DBConnectionService;
4
5 public class MapperDB {
6     private Connection connection;
7     public MapperDB() throws Exception {
8         try {
9             connection = DBConnectionService.getConnection();
10        }
11        catch (Exception e) {
12            System.out.println("Failed in constructor method in MapperDB:" + e);
13        }
14    }
15
16    public void closeConnection() throws Exception {
17        try {
18            getConnection().close();
19        }
20        catch (Exception e) {
21            System.out.println("Failed in closeConnection method in MapperDB:" + e);
22        }
23    }
24
25    public Connection getConnection() {
26        return connection;
27    }
28 }
```

# Ứng dụng MVC như thế nào? (6)

## 6. Xây dựng Lớp UserAccountMapper

```
1  package swingmvcdemo.dao;
2  import ...7 lines
3
4
5
6
7
8
9
10 public class UserAccountMapper extends MapperDB {
11     public UserAccountMapper() throws Exception {
12         super();
13     }
14
15     public boolean isUserExist(String username, String password) throws Exception {
16         boolean isExist = false;
17         try {
18             StringBuffer sql = new StringBuffer();
19             sql.append(" SELECT * FROM dbtest.useraccount userTable");
20             sql.append(" WHERE userTable.username = ? AND userTable.password = ?");
21             PreparedStatement stmt = getConnection().prepareStatement(sql.toString());
22             stmt.setString(1, username);
23             stmt.setString(2, password);
24             ResultSet rs = stmt.executeQuery();
25             if ((rs != null) && (rs.next())) {
26                 isExist = true;
27             }
28             stmt.close();
29         } catch (Exception e) {
30             e.printStackTrace();
31             throw e;
32         }
33         return isExist;
34     }
35 }
```

# Ứng dụng MVC như thế nào? (7)

## 7. Xây dựng Lớp UserAccountBO

```
1  package swingmvcdemo.bo;
2  import swingmvcdemo.dao.UserAccountMapper;
3  public class UserAccountBO {
4      public boolean isUserExist(String username, String password) throws Exception {
5          UserAccountMapper mapper = null;
6          try {
7              mapper = new UserAccountMapper();
8              return mapper.isUserExist(username, password);
9          } catch (Exception e) {
10             throw e;
11          } finally {
12              mapper.closeConnection();
13          }
14      }
15  }
16
```

# Ứng dụng MVC như thế nào? (8)

## 8. Xây dựng Lớp UserAccountController

```
1  package swingmvcdemo.controller;
2  [+ import ...4 lines
6
7  public class UserAccountController {
8  [- public void loginAction(UserAccountDTO userAccountDTO) {
9      try {
10         UserAccountBO bo = new UserAccountBO();
11         boolean loginStatus = bo.isUserExist(userAccountDTO.getUsername(),
12         userAccountDTO.getPassword());
13         if (loginStatus) {
14             MainFrame mainFrame = new MainFrame();
15         }
16         else {
17             System.exit(0);
18         }
19     } catch (Exception ex) {
20         ex.printStackTrace();
21     }
22 }
23 }
24
```

# Ứng dụng MVC như thế nào? (9)

## 9. Xây dựng lớp UserAccountDTO

```
1  package swingmvcdemo.dto;
2
3  public class UserAccountDTO {
4      private String username;
5      private String password;
6
7      + public String getUsername() {...3 lines }
10
11      + public void setUsername(String username) {...3 lines }
14
15      + public String getPassword() {...3 lines }
18
19      + public void setPassword(String password) {...3 lines }
22  }
23
```

# Ứng dụng MVC như thế nào? (10)

## 10. Xây dựng lớp LoginDialog

```
1  package swingmvcdemo.gui;
2  import ...12 lines
14
15  public class LoginDialog {
16      Khai bao cac thanh phan cua lop
17  public LoginDialog() {
18      //<editor-fold defaultstate="collapsed" desc="Tao cac doi tuong trong cua so Login">
19      userNameLabel = new JLabel("User name: ", JLabel.RIGHT);
20      userNameField = new JTextField("");
21      passwordLabel = new JLabel("Password: ", JLabel.RIGHT);
22      passwordField = new JPasswordField("");
23      connectionPanel = new JPanel(false);
24      connectionPanel.setLayout(new BoxLayout(connectionPanel, BoxLayout.X_AXIS));
25      namePanel = new JPanel(false);
26      namePanel.setLayout(new GridLayout(0, 1));
27      namePanel.add(userNameLabel);
28      namePanel.add(passwordLabel);
29      fieldPanel = new JPanel(false);
30      fieldPanel.setLayout(new GridLayout(0, 1));
31      fieldPanel.add(userNameField);
32      fieldPanel.add(passwordField);
33      connectionPanel.add(namePanel);
34      connectionPanel.add(fieldPanel);
35      //</editor-fold>
```

# Ứng dụng MVC như thế nào? (11)

## 11. Xây dựng lớp LoginDialog

```
1  package swingmvcdemo.gui;
2  import ...12 lines
14
15  public class LoginDialog {
16      Khai bao cac thanh phan cua lop
27      public LoginDialog() {
28          Tao cac doi tuong trong cua so Login
46          if (JOptionPane.showOptionDialog(null, connectionPanel,
47              ConnectTitle,
48              JOptionPane.OK_CANCEL_OPTION,
49              JOptionPane.INFORMATION_MESSAGE,
50              null, ConnectOptionNames,
51              ConnectOptionNames[0]) == 1) {
52              System.exit(0);
53          }
54
55          UserAccountDTO userAccountDTO = new UserAccountDTO();
56          userAccountDTO.setUsername(userNameField.getText());
57          userAccountDTO.setPassword(passwordField.getText());
58          UserAccountController controller = new UserAccountController();
59          controller.loginAction(userAccountDTO);
60      }
61  }
```



# Ứng dụng MVC như thế nào? (12)

## 12. Xây dựng lớp Main

```
1  package swingmvcdemo.controller;
2  import ...2 lines
3
4
5  public class SwingMVCDemo {
6
7      public static void main(String[] args) {
8          LoginDialog loginDialog = new LoginDialog();
9      }
10 }
```

# Tài liệu tham khảo

---

1. <http://docs.oracle.com/javase/tutorial/jdbc/index.html>
2. [http://www.ntu.edu.sg/home/ehchua/programming/java/JDBC\\_Basic.html](http://www.ntu.edu.sg/home/ehchua/programming/java/JDBC_Basic.html)