Java常用類別應用
應用程式與資料庫的整合

鄭安翔

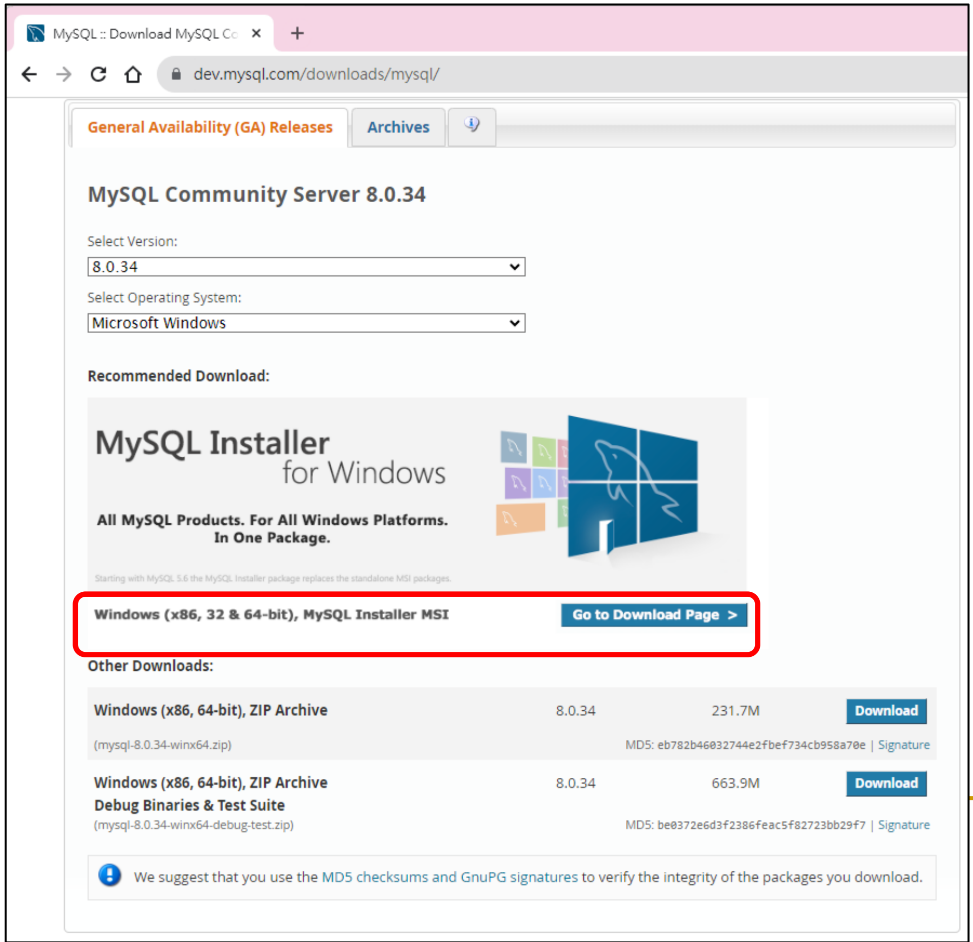
ansel_cheng@hotmail.com

課程大綱

1) **課前準備**
   - **MySQL資料庫安裝設定**
   - **JDBC 簡介**
2) 使用JDBC API 開發資料庫應用程式
3) 預編程式
4) Transaction 交易模式
5) DAO pattern

下載MySQL資料庫伺服器軟體

- http://dev.mysql.com/downloads/mysql/
  - MySQL Installer安裝程式



MySQL :: Download MySQL Co

dev.mysql.com/downloads/mysql/

General Availability (GA) Releases | Archives

MySQL Community Server 8.0.34

Select Version: 8.0.34

Select Operating System: Microsoft Windows

Recommended Download:

MySQL Installer for Windows

All MySQL Products. For All Windows Platforms. In One Package.

Starting with MySQL 5.6 the MySQL Installer package replaces the standalone MSI packages.

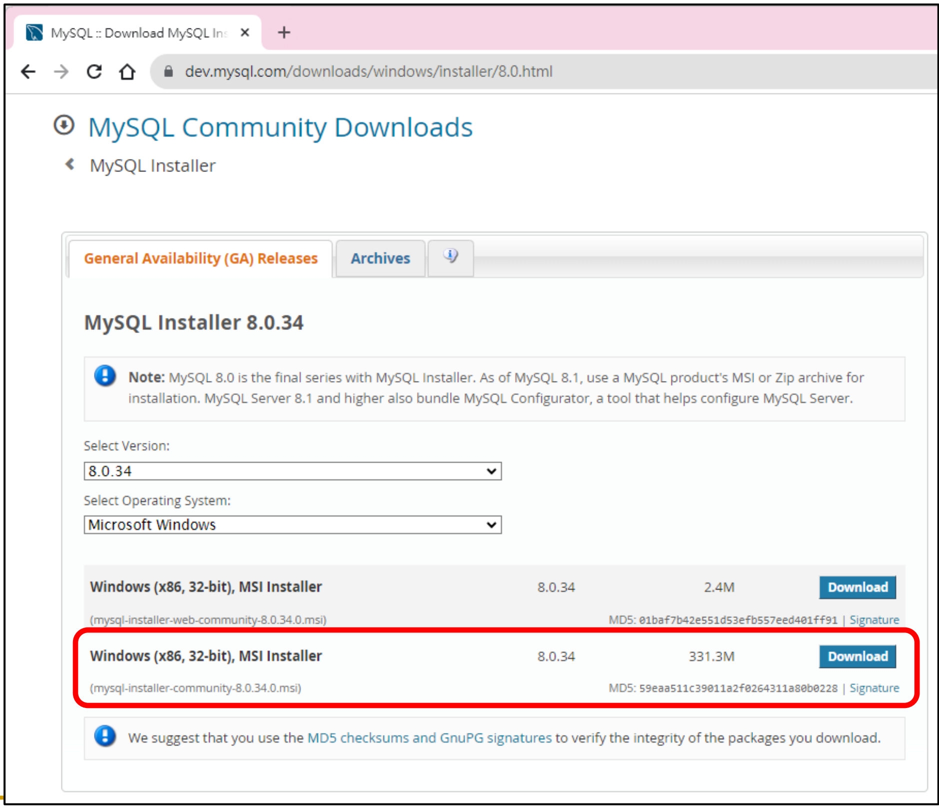
Windows (x86, 32 & 64-bit), MySQL Installer MSI — Go to Download Page >

Other Downloads:

Windows (x86, 64-bit), ZIP Archive — 8.0.34 — 231.7M — Download
(mysql-8.0.34-winx64.zip) — MD5: eb782b46032744e2fbef734cb958a70e | Signature

Windows (x86, 64-bit), ZIP Archive Debug Binaries & Test Suite — 8.0.34 — 663.9M — Download
(mysql-8.0.34-winx64-debug-test.zip) — MD5: be0372e6d3f2386feac5f82723bb29f7 | Signature

We suggest that you use the MD5 checksums and GnuPG signatures to verify the integrity of the packages you download.



MySQL :: Download MySQL Ins

dev.mysql.com/downloads/windows/installer/8.0.html

MySQL Community Downloads

MySQL Installer

General Availability (GA) Releases | Archives

MySQL Installer 8.0.34

Note: MySQL 8.0 is the final series with MySQL Installer. As of MySQL 8.1, use a MySQL product's MSI or Zip archive for installation. MySQL Server 8.1 and higher also bundle MySQL Configurator, a tool that helps configure MySQL Server.

Select Version: 8.0.34

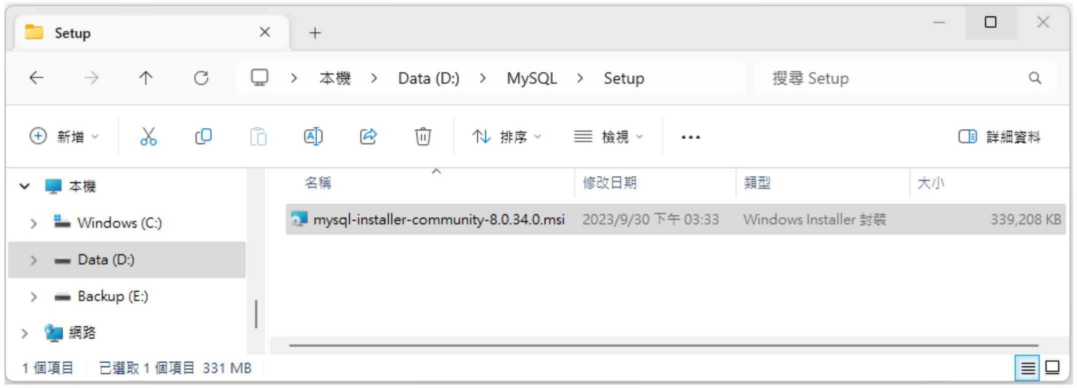
Select Operating System: Microsoft Windows

Windows (x86, 32-bit), MSI Installer — 8.0.34 — 2.4M — Download
(mysql-installer-web-community-8.0.34.0.msi) — MD5: 01baf7b42e551d53efb557eed401ff91 | Signature

Windows (x86, 32-bit), MSI Installer — 8.0.34 — 331.3M — Download
(mysql-installer-community-8.0.34.0.msi) — MD5: 59eaa511c39011a2f0264311a80b0228 | Signature

We suggest that you use the MD5 checksums and GnuPG signatures to verify the integrity of the packages you download.

安裝時設定Root帳號密碼

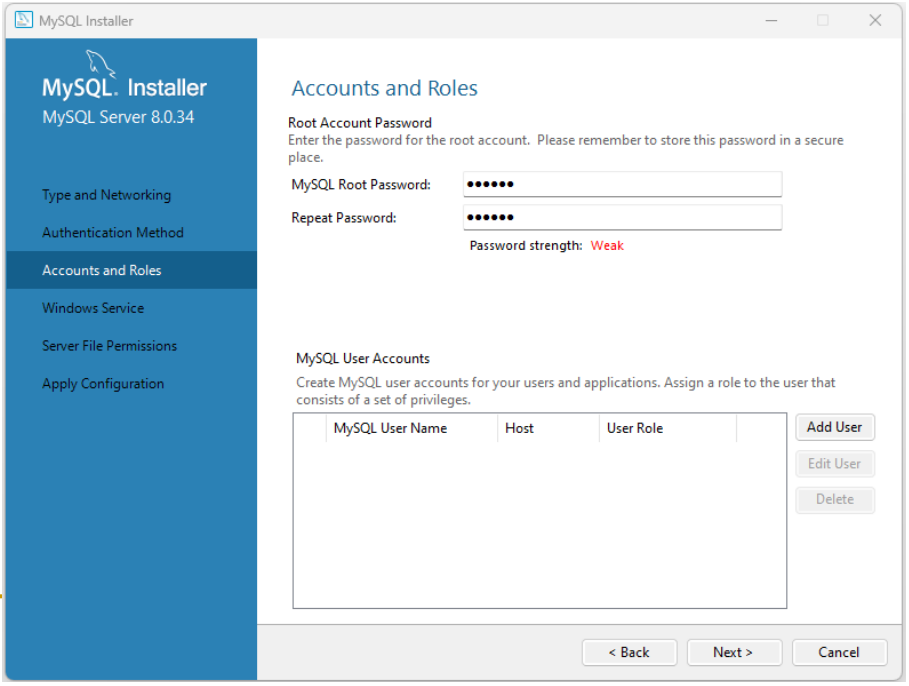

Setup

本機 › Data (D:) › MySQL › Setup

名稱	修改日期	類型	大小
mysql-installer-community-8.0.34.0.msi	2023/9/30 下午 03:33	Windows Installer 封裝	339,208 KB

1 個項目　已選取 1 個項目 331 MB

密碼: abc123


MySQL Installer

MySQL. Installer
MySQL Server 8.0.34

Accounts and Roles

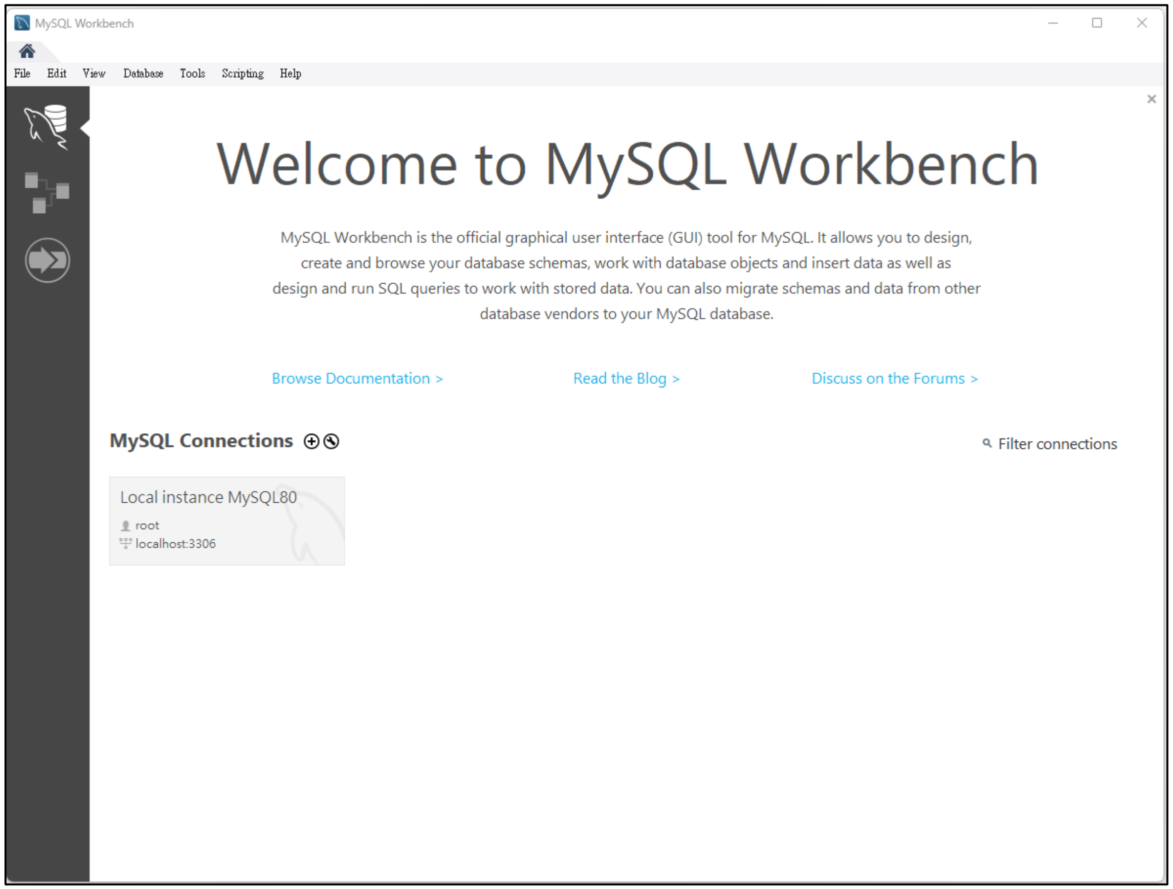
Root Account Password
Enter the password for the root account. Please remember to store this password in a secure place.

MySQL Root Password: ••••••
Repeat Password: ••••••
Password strength: Weak

MySQL User Accounts
Create MySQL user accounts for your users and applications. Assign a role to the user that consists of a set of privileges.

MySQL User Name	Host	User Role

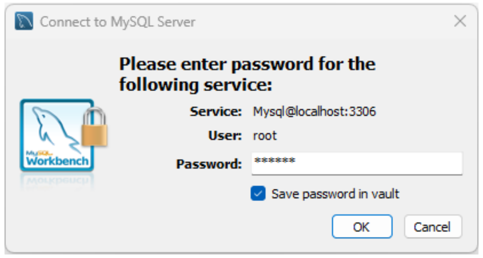
使用MySQL Workbench 設定資料庫



Welcome to MySQL Workbench

MySQL Workbench is the official graphical user interface (GUI) tool for MySQL. It allows you to design, create and browse your database schemas, work with database objects and insert data as well as design and run SQL queries to work with stored data. You can also migrate schemas and data from other database vendors to your MySQL database.

Local instance MySQL80
root
localhost:3306



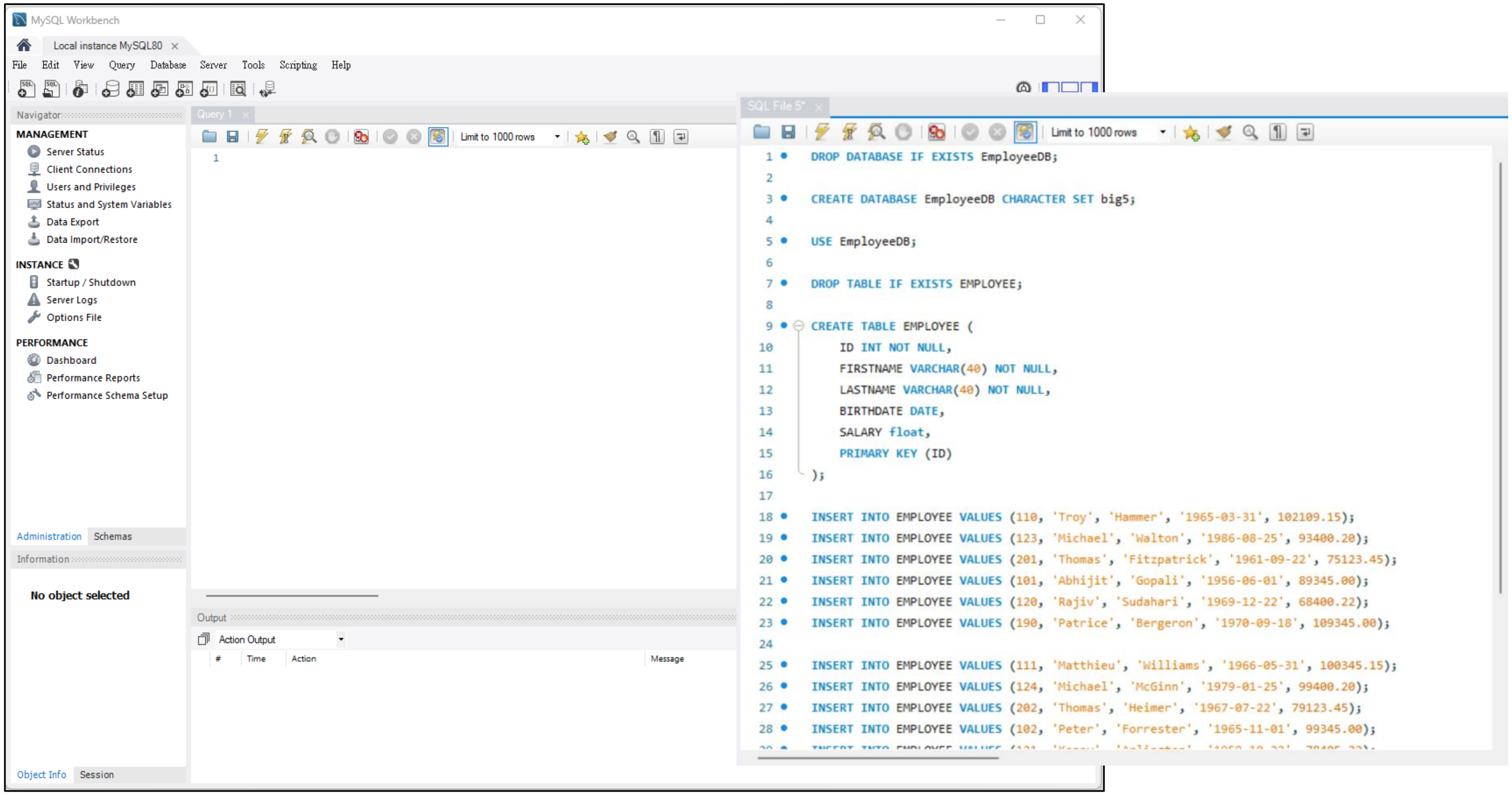
Please enter password for the following service:

Service: Mysql@localhost:3306
User: root
Password: ******

Save password in vault

建立EmployeeDB Schema及內容

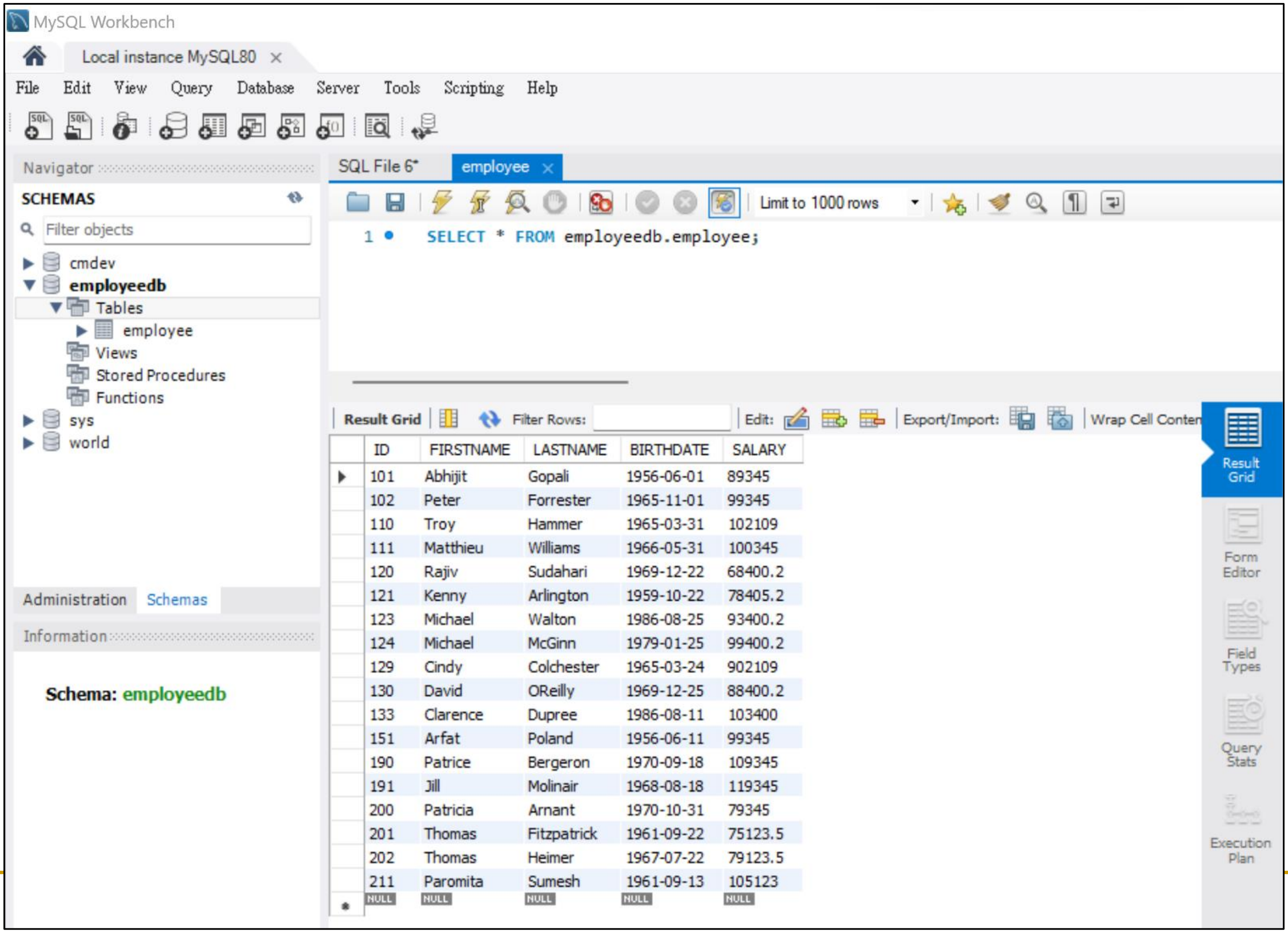
- 複製employeeTable.sql
- 執行sql



No object selected

```sql
DROP DATABASE IF EXISTS EmployeeDB;

CREATE DATABASE EmployeeDB CHARACTER SET big5;

USE EmployeeDB;

DROP TABLE IF EXISTS EMPLOYEE;

CREATE TABLE EMPLOYEE (
    ID INT NOT NULL,
    FIRSTNAME VARCHAR(40) NOT NULL,
    LASTNAME VARCHAR(40) NOT NULL,
    BIRTHDATE DATE,
    SALARY float,
    PRIMARY KEY (ID)
);

INSERT INTO EMPLOYEE VALUES (110, 'Troy', 'Hammer', '1965-03-31', 102109.15);
INSERT INTO EMPLOYEE VALUES (123, 'Michael', 'Walton', '1986-08-25', 93400.20);
INSERT INTO EMPLOYEE VALUES (201, 'Thomas', 'Fitzpatrick', '1961-09-22', 75123.45);
INSERT INTO EMPLOYEE VALUES (101, 'Abhijit', 'Gopali', '1956-06-01', 89345.00);
INSERT INTO EMPLOYEE VALUES (120, 'Rajiv', 'Sudahari', '1969-12-22', 68400.22);
INSERT INTO EMPLOYEE VALUES (190, 'Patrice', 'Bergeron', '1970-09-18', 109345.00);

INSERT INTO EMPLOYEE VALUES (111, 'Matthieu', 'Williams', '1966-05-31', 100345.15);
INSERT INTO EMPLOYEE VALUES (124, 'Michael', 'McGinn', '1979-01-25', 99400.20);
INSERT INTO EMPLOYEE VALUES (202, 'Thomas', 'Heimer', '1967-07-22', 79123.45);
INSERT INTO EMPLOYEE VALUES (102, 'Peter', 'Forrester', '1965-11-01', 99345.00);
```

檢視EmployeeDB



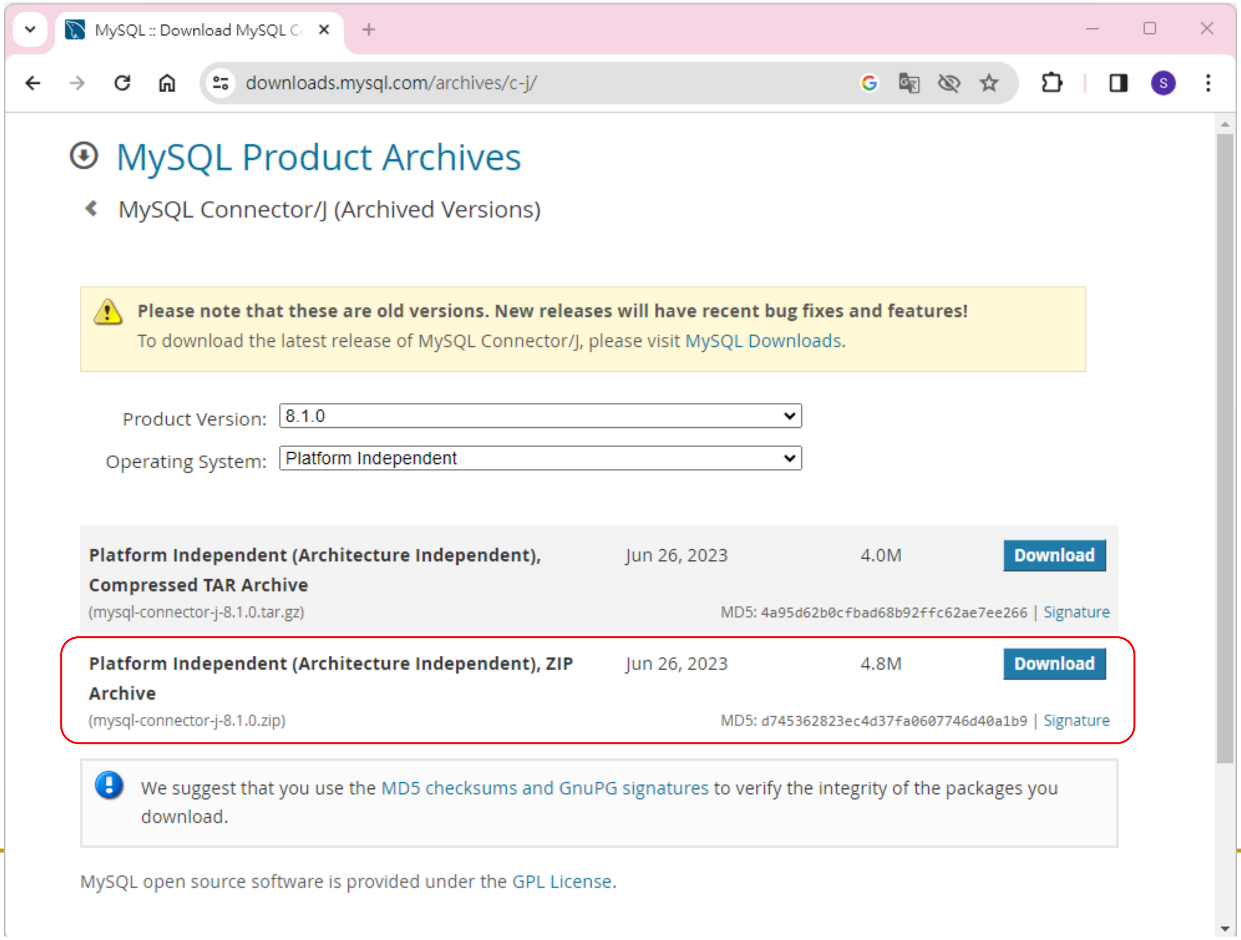
Schema: employeedb

```sql
SELECT * FROM employeedb.employee;
```

ID	FIRSTNAME	LASTNAME	BIRTHDATE	SALARY
101	Abhijit	Gopali	1956-06-01	89345
102	Peter	Forrester	1965-11-01	99345
110	Troy	Hammer	1965-03-31	102109
111	Matthieu	Williams	1966-05-31	100345
120	Rajiv	Sudahari	1969-12-22	68400.2
121	Kenny	Arlington	1959-10-22	78405.2
123	Michael	Walton	1986-08-25	93400.2
124	Michael	McGinn	1979-01-25	99400.2
129	Cindy	Colchester	1965-03-24	902109
130	David	OReilly	1969-12-25	88400.2
133	Clarence	Dupree	1986-08-11	103400
151	Arfat	Poland	1956-06-11	99345
190	Patrice	Bergeron	1970-09-18	109345
191	Jill	Molinair	1968-08-18	119345
200	Patricia	Arnant	1970-10-31	79345
201	Thomas	Fitzpatrick	1961-09-22	75123.5
202	Thomas	Heimer	1967-07-22	79123.5
211	Paromita	Sumesh	1961-09-13	105123
NULL	NULL	NULL	NULL	NULL

下載MySQL JDBC Driver

- https://dev.mysql.com/downloads/connector/j/



MySQL Product Archives

MySQL Connector/J (Archived Versions)

Please note that these are old versions. New releases will have recent bug fixes and features!
To download the latest release of MySQL Connector/J, please visit MySQL Downloads.

Product Version: 8.1.0

Operating System: Platform Independent

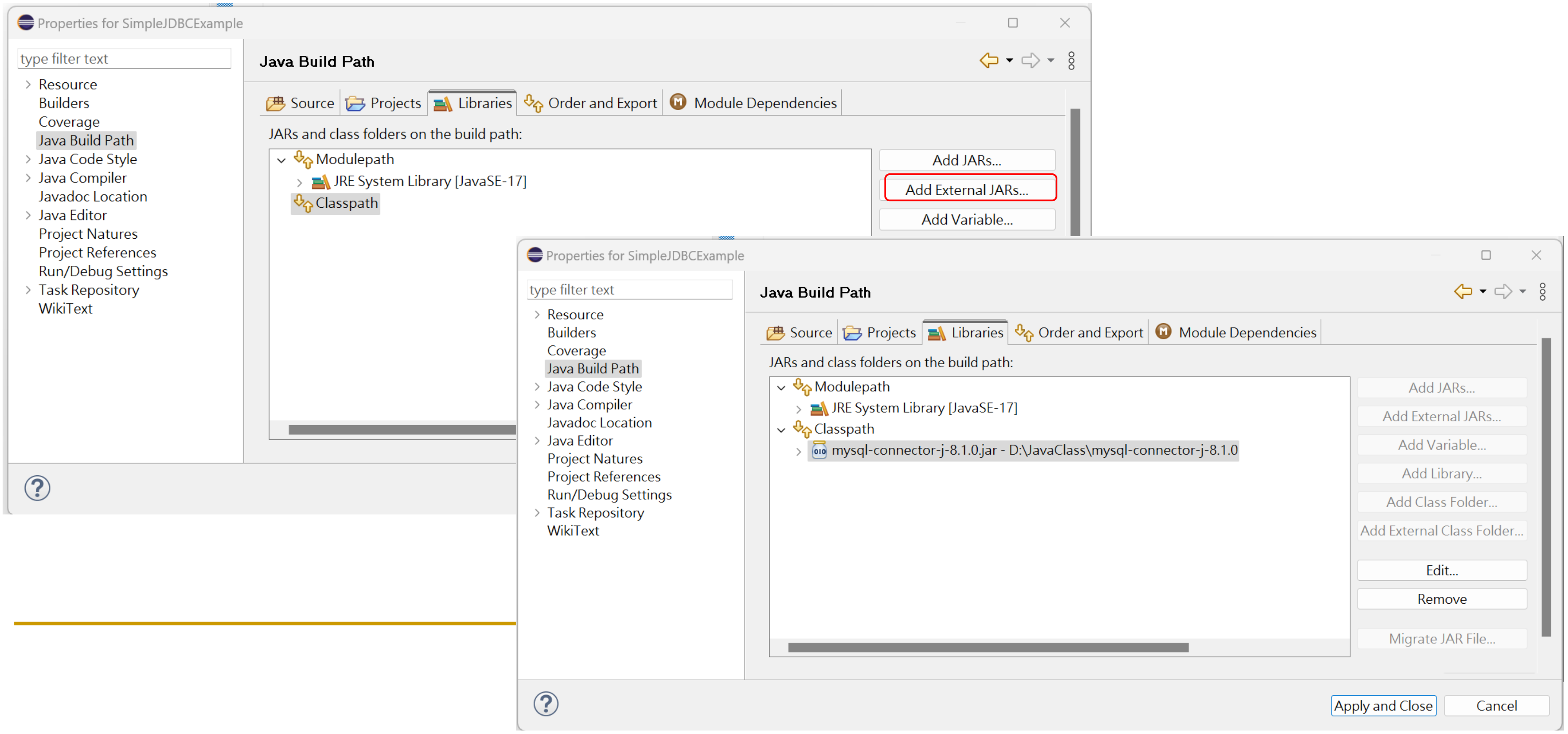
Platform Independent (Architecture Independent), Compressed TAR Archive (mysql-connector-j-8.1.0.tar.gz)	Jun 26, 2023	4.0M	Download MD5: 4a95d62b0cfbad68b92ffc62ae7ee266 | Signature
Platform Independent (Architecture Independent), ZIP Archive (mysql-connector-j-8.1.0.zip)	Jun 26, 2023	4.8M	Download MD5: d745362823ec4d37fa0607746d40a1b9 | Signature

We suggest that you use the MD5 checksums and GnuPG signatures to verify the integrity of the packages you download.

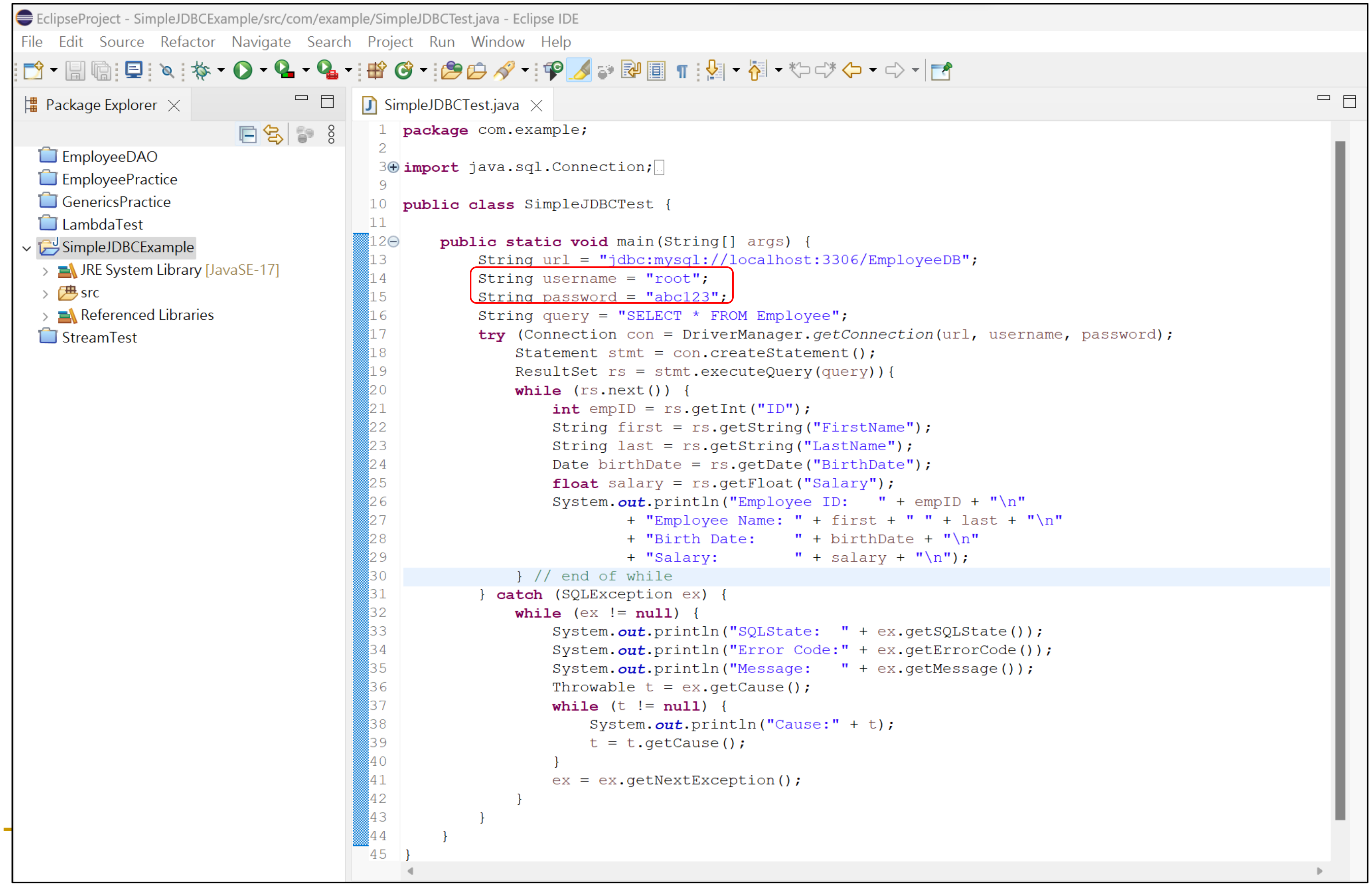
MySQL open source software is provided under the GPL License.

設定MySQL JDBC Driver路徑

- 解壓縮JDBC Driver
- 專案Classpath加入mysql-connector-j-8.x.x

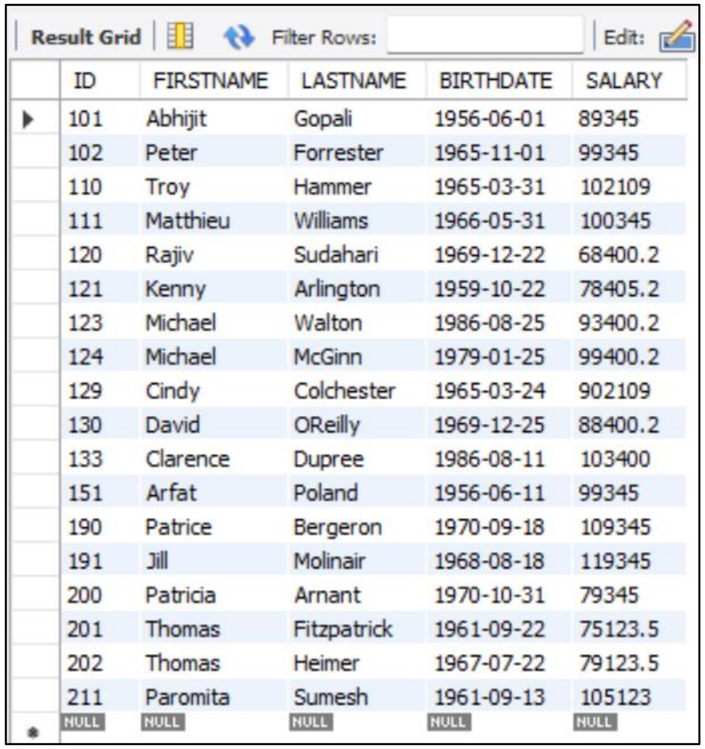


確認username/password


EclipseProject - SimpleJDBCExample/src/com/example/SimpleJDBCTest.java - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help
Package Explorer
EmployeeDAO
EmployeePractice
GenericsPractice
LambdaTest
SimpleJDBCExample
JRE System Library [JavaSE-17]
src
Referenced Libraries
StreamTest
SimpleJDBCTest.java

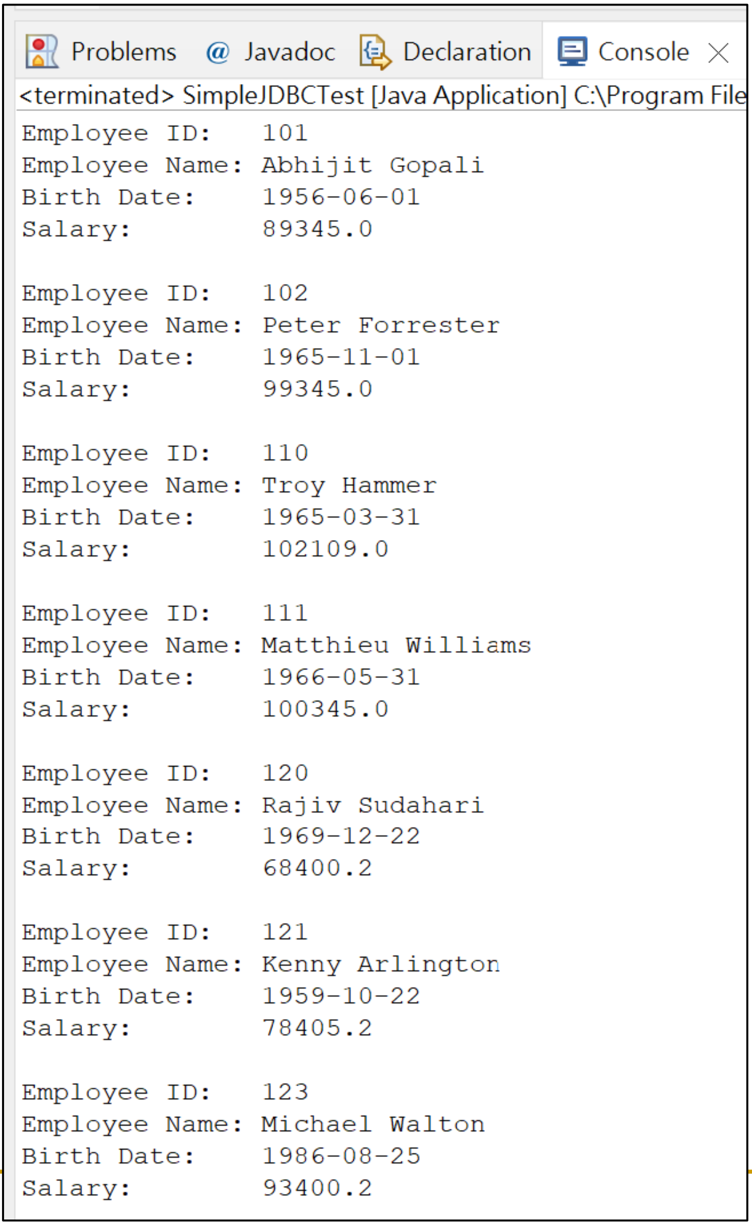
```java
package com.example;

import java.sql.Connection;

public class SimpleJDBCTest {

    public static void main(String[] args) {
        String url = "jdbc:mysql://localhost:3306/EmployeeDB";
        String username = "root";
        String password = "abc123";
        String query = "SELECT * FROM Employee";
        try (Connection con = DriverManager.getConnection(url, username, password);
            Statement stmt = con.createStatement();
            ResultSet rs = stmt.executeQuery(query)){
            while (rs.next()) {
                int empID = rs.getInt("ID");
                String first = rs.getString("FirstName");
                String last = rs.getString("LastName");
                Date birthDate = rs.getDate("BirthDate");
                float salary = rs.getFloat("Salary");
                System.out.println("Employee ID:   " + empID + "\n"
                        + "Employee Name: " + first + " " + last + "\n"
                        + "Birth Date:    " + birthDate + "\n"
                        + "Salary:        " + salary + "\n");
            } // end of while
        } catch (SQLException ex) {
            while (ex != null) {
                System.out.println("SQLState:  " + ex.getSQLState());
                System.out.println("Error Code:" + ex.getErrorCode());
                System.out.println("Message:   " + ex.getMessage());
                Throwable t = ex.getCause();
                while (t != null) {
                    System.out.println("Cause:" + t);
                    t = t.getCause();
                }
                ex = ex.getNextException();
            }
        }
    }
}
```

測試



Result Grid | Filter Rows: | Edit:

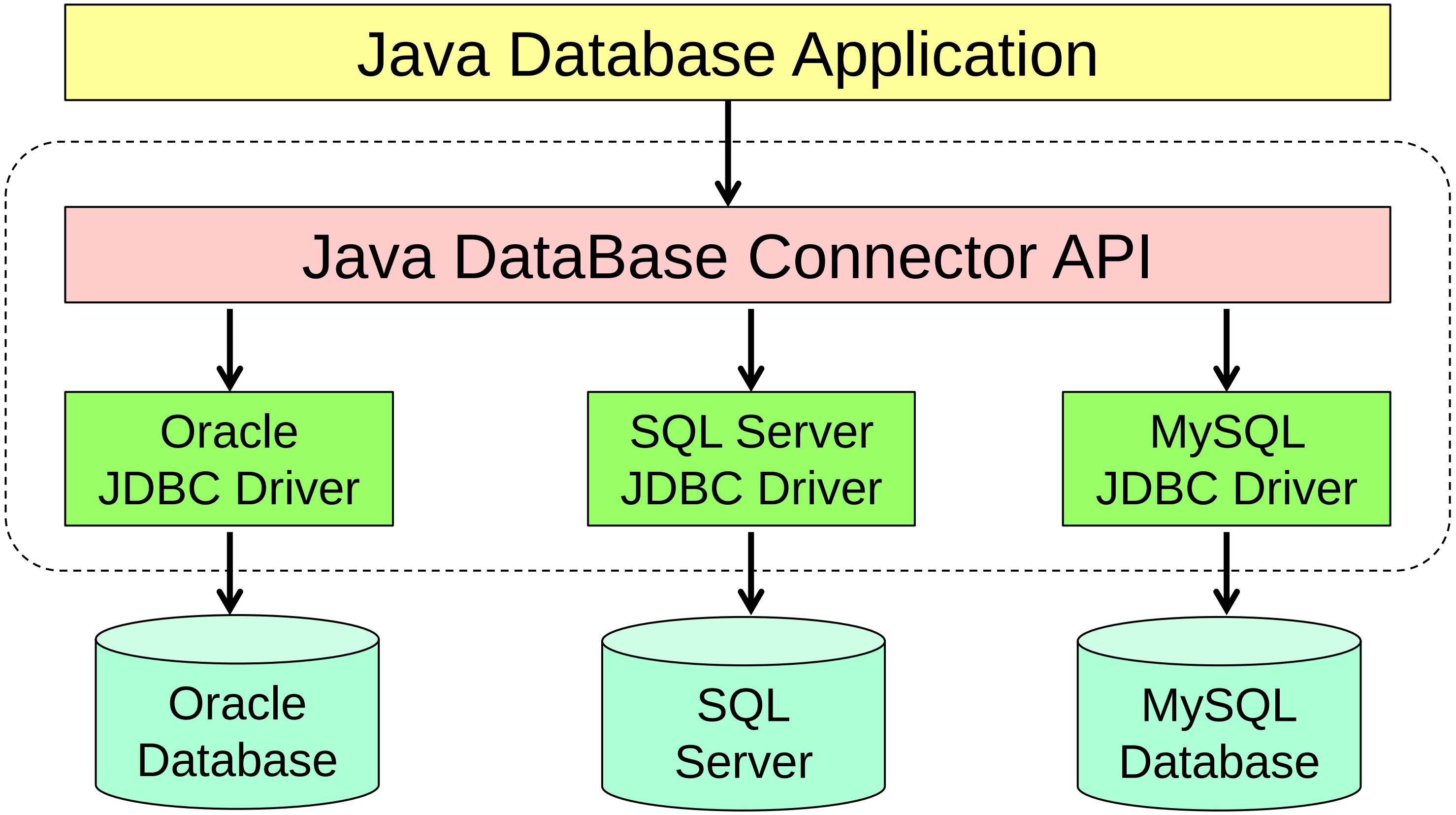
	ID	FIRSTNAME	LASTNAME	BIRTHDATE	SALARY
▶	101	Abhijit	Gopali	1956-06-01	89345
	102	Peter	Forrester	1965-11-01	99345
	110	Troy	Hammer	1965-03-31	102109
	111	Matthieu	Williams	1966-05-31	100345
	120	Rajiv	Sudahari	1969-12-22	68400.2
	121	Kenny	Arlington	1959-10-22	78405.2
	123	Michael	Walton	1986-08-25	93400.2
	124	Michael	McGinn	1979-01-25	99400.2
	129	Cindy	Colchester	1965-03-24	902109
	130	David	OReilly	1969-12-25	88400.2
	133	Clarence	Dupree	1986-08-11	103400
	151	Arfat	Poland	1956-06-11	99345
	190	Patrice	Bergeron	1970-09-18	109345
	191	Jill	Molinair	1968-08-18	119345
	200	Patricia	Arnant	1970-10-31	79345
	201	Thomas	Fitzpatrick	1961-09-22	75123.5
	202	Thomas	Heimer	1967-07-22	79123.5
	211	Paromita	Sumesh	1961-09-13	105123
*	NULL	NULL	NULL	NULL	NULL



Problems @ Javadoc Declaration Console

```
<terminated> SimpleJDBCTest [Java Application] C:\Program File
Employee ID:   101
Employee Name: Abhijit Gopali
Birth Date:    1956-06-01
Salary:        89345.0

Employee ID:   102
Employee Name: Peter Forrester
Birth Date:    1965-11-01
Salary:        99345.0

Employee ID:   110
Employee Name: Troy Hammer
Birth Date:    1965-03-31
Salary:        102109.0

Employee ID:   111
Employee Name: Matthieu Williams
Birth Date:    1966-05-31
Salary:        100345.0

Employee ID:   120
Employee Name: Rajiv Sudahari
Birth Date:    1969-12-22
Salary:        68400.2

Employee ID:   121
Employee Name: Kenny Arlington
Birth Date:    1959-10-22
Salary:        78405.2

Employee ID:   123
Employee Name: Michael Walton
Birth Date:    1986-08-25
Salary:        93400.2
```

JDBC 簡介



Write Once Match to All
Database Management System

JDBC驅動程式

- 廠商在實作JDBC驅動程式時，依方式可將驅動程式分作四種類型
  - Type 1：JDBC-ODBC Bridge Driver
    - 利用Bridge的方式將JDBC的呼叫方式轉換為ODBC的呼叫方式
  - Type 2：Native API Driver
    - 驅動程式將Java應用程式中JDBC呼叫轉為原生程式碼的呼叫(ex:C、C++)來與資料庫作溝通
  - Type 3：JDBC-Middleware Driver
    - 驅動程式以特定中介伺服器(Network Server)的網路協定，來完成資料庫存取動作
  - Type 4：Native Protocol Driver
    - 直接以資料庫的通訊協定與資料庫作溝通，而不透過橋接或中介伺服器來存取資料庫

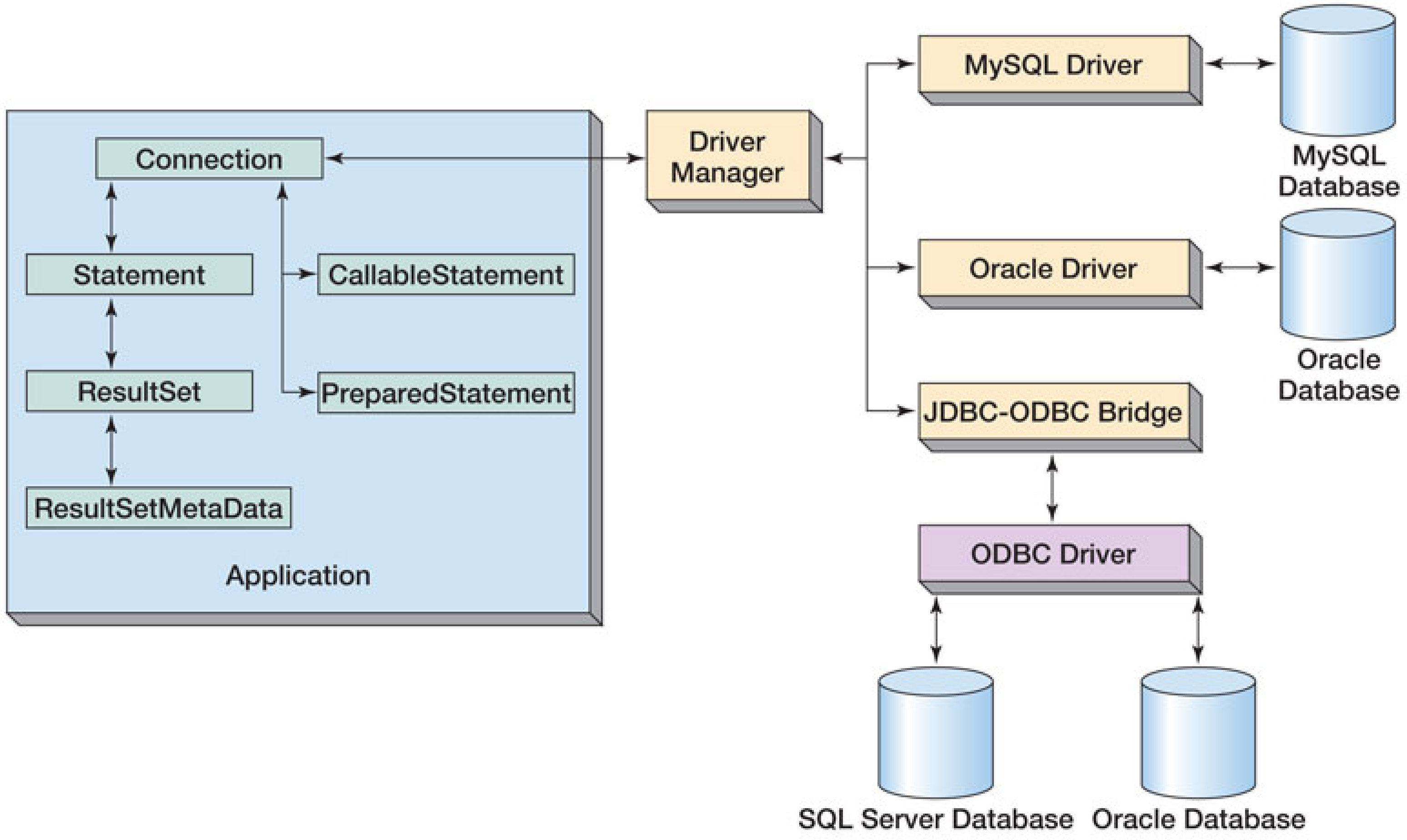
JDBC 演進

- JDBC 1.0
  - 提供對資料庫基本的存取功能
  - 需要自行撰寫連接資料庫及關閉連線的程式碼
  - 選擇和啟動JDBC驅動程式也需要手動控制
- JDBC 2.0
  - 引入data source 概念,由data source 取得資料庫連線
  - 資料庫連接池 Connection Pool
- JDBC 3.0
  - 使用JNDI 來獲得data source , 將業務邏輯與資料庫層分離
  - 資料庫連接池成為應用伺服器或Servlet 容器標準功能
- JDBC 4.0
  - 不必再使用Class.forName註冊JDBC驅動程式. DriverManager透過在classpath中搜尋並載入JDBC驅動程式
  - 支援XML資料類型

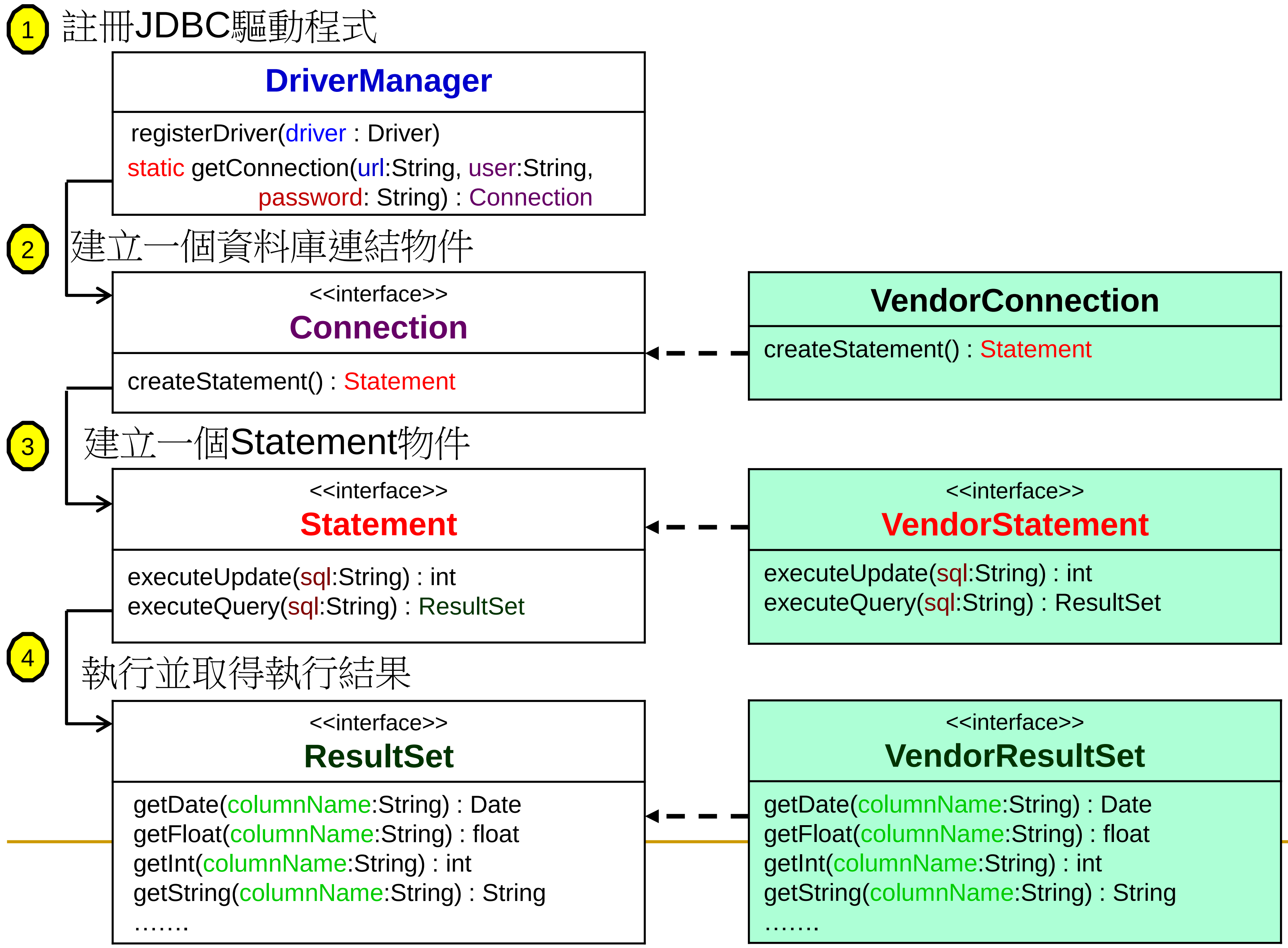
課程大綱

1) 課前準備
2) **使用JDBC API 開發資料庫應用程式**
   - **JDBC使用步驟**
   - **ResultSet**
3) 預編程式
4) Transaction 交易模式
5) DAO pattern

JDBC 元件


Connection
Statement
CallableStatement
ResultSet
PreparedStatement
ResultSetMetaData
Application
Driver Manager
MySQL Driver
MySQL Database
Oracle Driver
Oracle Database
JDBC-ODBC Bridge
ODBC Driver
SQL Server Database
Oracle Database

JDBC使用步驟


1 註冊JDBC驅動程式
DriverManager
registerDriver(driver : Driver)
static getConnection(url:String, user:String, password: String) : Connection
2 建立一個資料庫連結物件
<<interface>>
Connection
createStatement() : Statement
VendorConnection
createStatement() : Statement
3 建立一個Statement物件
<<interface>>
Statement
executeUpdate(sql:String) : int
executeQuery(sql:String) : ResultSet
<<interface>>
VendorStatement
executeUpdate(sql:String) : int
executeQuery(sql:String) : ResultSet
4 執行並取得執行結果
<<interface>>
ResultSet
getDate(columnName:String) : Date
getFloat(columnName:String) : float
getInt(columnName:String) : int
getString(columnName:String) : String
…….
<<interface>>
VendorResultSet
getDate(columnName:String) : Date
getFloat(columnName:String) : float
getInt(columnName:String) : int
getString(columnName:String) : String
…….

載入及註冊JDBC驅動程式

- 連接資料庫前準備工作
  - 必需要有廠商實作的JDBC驅動程式
  - CLASSPATH中設定指向驅動程式JAR檔案
- 註冊JDBC驅動程式(JDBC4.0之前才需要)

```
Driver driver = (Driver) Class.forName(jdbcDriverClass).newInstance();

DriverManager.registerDriver(driver);
```

java.sql.DriverManager 類別

- java.sql.DriverManager

常用方法	傳回值	說明
registerDriver(Driver driver)	static void	註冊指定JDBC driver 至 DriverManager.
deregisterDriver(Driver driver)	static void	將指定JDBC driver 由 DriverManager中清單移除
getConnection(String url)	Static Connection	以指定URL 字串建立資料庫連線
getConnection(String url, String user, String password)	Static Connection	以指定URL 字串,帳號及密碼,建立資料庫連線

JDBC使用步驟

- 建立一個資料庫連結物件
  - JDBC URL 語法

    <Protocol>:<Subprotocol>:<DataSourceName>
  - 建立一個Connection物件

```
String url = "jdbc:mysql://localhost:3306/EmployeeDB";
String user = "root";
String password = "abc123";
Connection con = DriverManager.getConnection(url, user, password);
```

JDBC使用步驟

- 建立一個SQL敘述物件
  - createStatement()

    Statement stmt = con.createStatement();

- 建立陳述字串

```
String sql1 = “SELECT * FROM EMPLOYEE”;
String sql2 = “INSERT INTO EMPLOYEE VALUES
                    (123, 'Sean', 'Cheng', '1974-03-21', 75000.00)”;
String sql3 = “UPDATE EMPLOYEE SET SALARY=85000
                    WHERE ID=123”;
String sql4 = “DELETE EMPLOYEE WHERE ID=123”;
```

JDBC使用步驟

- executeUpdate() : int
  - 用於更新(Insert/Update/Delete) 敘述
  - 傳回值為修改成功筆數

```
Statement stmt = con.createStatement();
String sql = “INSERT INTO EMPLOYEE VALUES
                  (123, 'Sean', 'Cheng', '1974-03-21', 75000.00)”;
int rs = stmt.executeUpdate(sql);
```

JDBC使用步驟

- 執行並取得執行結果
  - executeQuery() : ResultSet
    - 用於查詢 (Select) 敘述
    - 傳回值為 ResultSet 物件

```
Statement stmt = con.createStatement();
String sql = “SELECT * FROM EMPLOYEE”;
ResultSet rs = stmt.executeQuery(sql);
while(rs.next()){
    int id = rs.getInt(1);
    String first = rs.getString(2);
    String last = rs.getString(3);
    …….
}
```

java.sql.ResultSet

- java.sql.ResultSet物件
  - 代表查詢的結果
  - 具有指向當前資料行的游標
    - 游標一開始在查詢結果的第一行
    - 透過 next() 方法將游標移動到下一行查詢結果

rs.next()

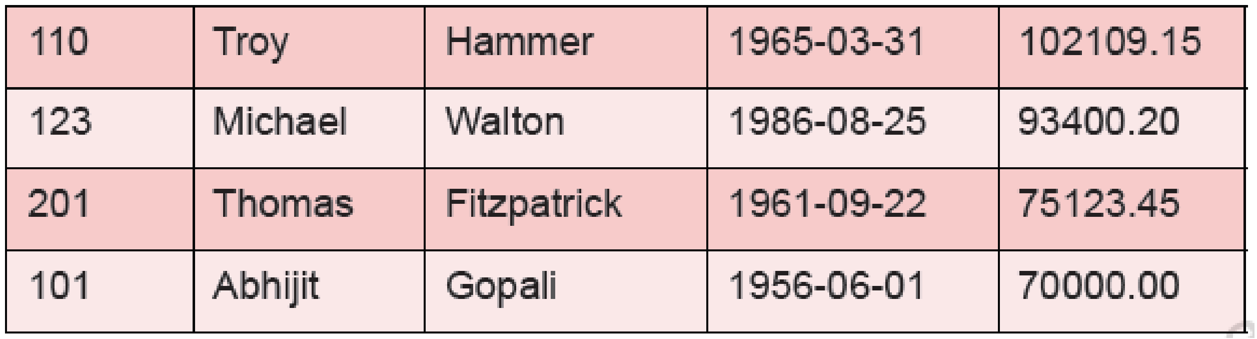
rs.next() ⟶

rs.next() ⟶

rs.next() ⟶

rs.next() ⟶

⟶ false



110	Troy	Hammer	1965-03-31	102109.15
123	Michael	Walton	1986-08-25	93400.20
201	Thomas	Fitzpatrick	1961-09-22	75123.45
101	Abhijit	Gopali	1956-06-01	70000.00

java.sql.ResultSet 常用方法	傳回值	說明
first()	boolean	將游標移動到 ResultSet 物件的第一行(筆)
next()	boolean	將游標由ResultSet從當前位置移動到下一行(筆)
last()	boolean	將游標移動到 ResultSet 物件的最後一行(筆)
close()	void	釋放此 ResultSet 物件的資料庫和 JDBC 資源
isFirst()	boolean	游標是否位於此 ResultSet 物件的第一行(筆)
isLast()	boolean	游標是否位於此 ResultSet 物件的最後一行(筆)
isClosed()	boolean	此 ResultSet 物件是否已關閉
getRow()	int	取得當前游標指向的資料行(筆)數
setFetchSize(int rows)	void	設定ResultSet一次取得資料的最多行(筆)數
getInt(int columnIndex)	int	取得ResultSet 當前行中第幾欄位的 int 值
getInt(String columnLabel)	int	取得ResultSet 當前行中指定欄位名稱的 int 值
getString(int columnIndex)	String	取得ResultSet 當前行中第幾欄位的字串值
getString(String columnLabel)	String	取得ResultSet 當前行中指定欄位名稱的字串值
getDouble(int columnIndex)	double	取得ResultSet 當前行中第幾欄位的double值
getDouble(String columnLabel)	double	取得ResultSet 當前行中指定欄位名稱的double值
getDate(int columnIndex)	java.sql.Date	取得ResultSet 當前行中第幾欄位的日期值
getDate(String columnLabel)	java.sql.Date	取得ResultSet 當前行中指定欄位名稱的日期值
getTimestamp(int columnIndex)	java.sql. Timestamp	取得ResultSet 當前行中第幾欄位的時間值
getTimestamp(String columnLabel)	java.sql. Timestamp	取得ResultSet 當前行中指定欄位名稱的時間值

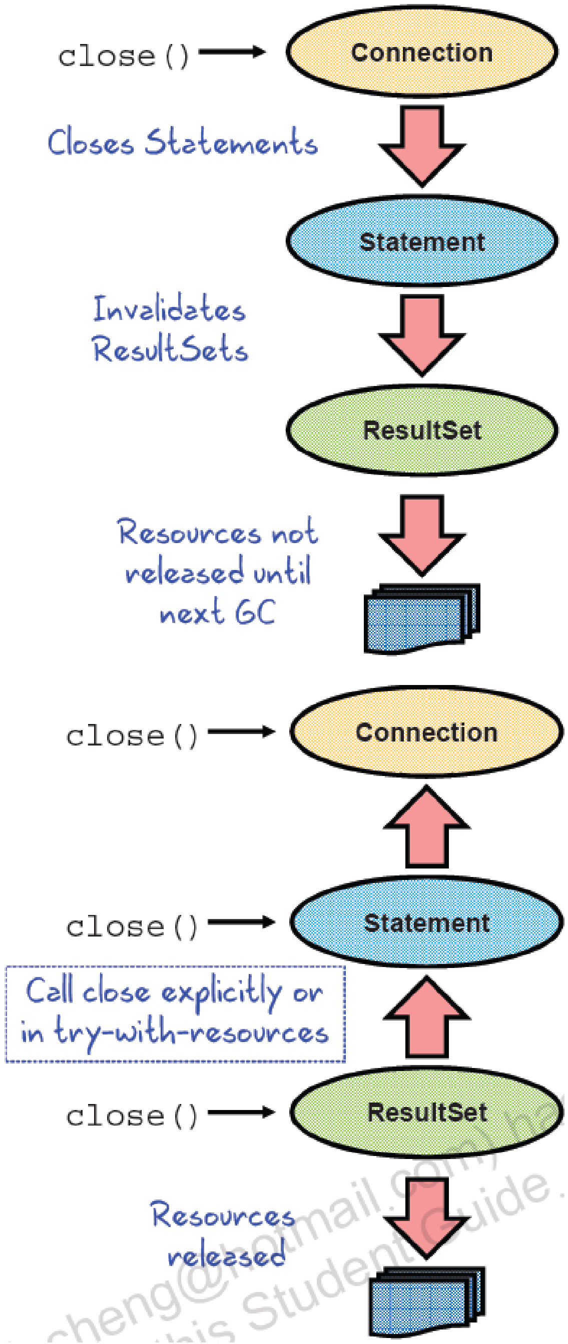
java.sql.SQLException

- java.sql.SQLException物件
  - 提供關於資料庫存取錯誤的例外
  - 包含下列資訊
    - message:描述錯誤的字串,
    - SQLState: XOPEN SQLstate 或 SQL:2003
    - vendorCode:資料庫廠商提供之異常/錯誤代碼
    - cause: 產生此SQLException 的例外(被包覆)

常用方法	傳回值	說明
getMessage()	String	取得描述錯誤的字串
getSQLState()	String	取得SQLException 物件的SQLState
getErrorCode()	int	取得資料庫廠商提供之異常/錯誤代碼
iterator()	Iterator <Throwable>	SQLExceptions 上進行迭代的迭代器,以取得被包覆的例外物件

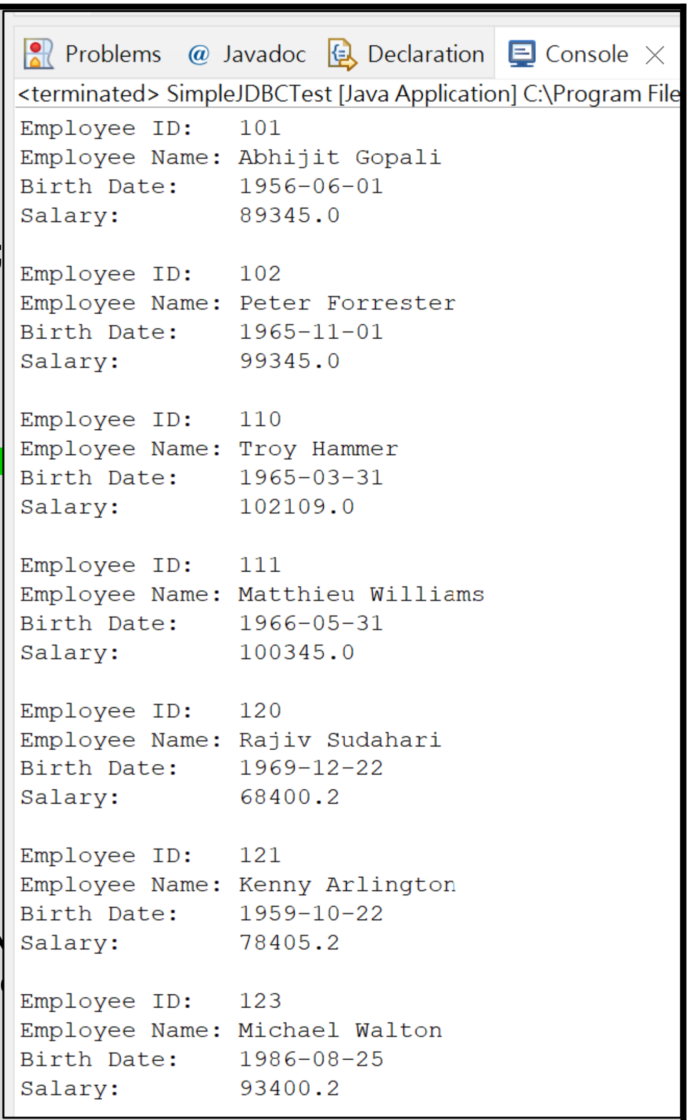
JDBC資源釋放

- JDBC資源釋放
  - 只關閉Connection物件
    - 自動關閉Statement
    - 自動作廢ResultSet
    - ResultSet資源需等垃圾收集才釋放
  - 自行關閉所有資源
    - ResultSet -> Statement -> Connection
    - ResultSet資源於關閉ResultSet時釋放
  - 使用try with resources
    - JDBC 4.1 資源均實作AutoCloseable
    - 多個可關閉資源用’；’區隔
    - 依資源建立相反順序關閉



簡易 JDBC 範例

```java
import java.sql.*;
import java.util.Date;
public class SimpleJDBCTest {
    public static void main(String[] args) {
        String url = "jdbc:mysql://localhost:3306/EmployeeDB";
        String username = "root";
        String password = "abc123";
        String query = "SELECT * FROM Employee";
        try (Connection con = DriverManager.getConnection(url
               Statement stmt = con.createStatement();
               ResultSet rs = stmt.executeQuery(query)){
            while (rs.next()) {
                int empID = rs.getInt("ID");
                String first = rs.getString("FirstName");
                String last = rs.getString("LastName");
                Date birthDate = rs.getDate("BirthDate");
                float salary = rs.getFloat("Salary");
                System.out.printf("Employee ID: %d%nEmployee N
                System.out.printf("Birth Date: %tF%nSalary: %f%n
           } // end of while
        } catch (SQLException ex) {    ……..   }
    }
}
```



```
<terminated> SimpleJDBCTest [Java Application] C:\Program File
Employee ID:   101
Employee Name: Abhijit Gopali
Birth Date:    1956-06-01
Salary:        89345.0

Employee ID:   102
Employee Name: Peter Forrester
Birth Date:    1965-11-01
Salary:        99345.0

Employee ID:   110
Employee Name: Troy Hammer
Birth Date:    1965-03-31
Salary:        102109.0

Employee ID:   111
Employee Name: Matthieu Williams
Birth Date:    1966-05-31
Salary:        100345.0

Employee ID:   120
Employee Name: Rajiv Sudahari
Birth Date:    1969-12-22
Salary:        68400.2

Employee ID:   121
Employee Name: Kenny Arlington
Birth Date:    1959-10-22
Salary:        78405.2

Employee ID:   123
Employee Name: Michael Walton
Birth Date:    1986-08-25
Salary:        93400.2
```

簡易 JDBC 範例

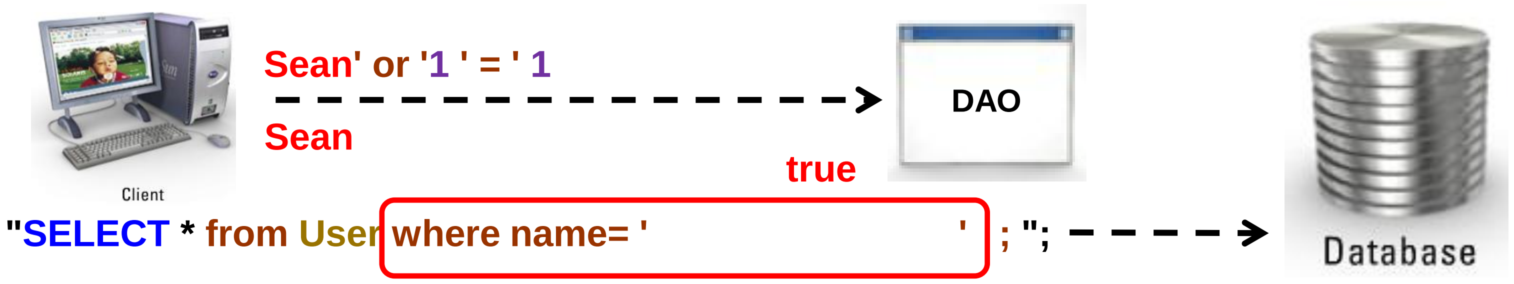
```
        try (Connection con = DriverManager.getConnection(url, username, password);
                Statement stmt = con.createStatement();
                ResultSet rs = stmt.executeQuery(query)){
                …….
        } catch (SQLException ex) {
            while (ex != null) {
                System.out.println("SQLState:  " + ex.getSQLState());
                System.out.println("Error Code:" + ex.getErrorCode());
                System.out.println("Message:   " + ex.getMessage());
                Throwable t = ex.getCause();
                while (t != null) {
                    System.out.println("Cause:" + t);
                    t = t.getCause();
                }
                ex = ex.getNextException();
            }
        }
    }
}
```

課程大綱

1) 課前準備
2) 使用JDBC API 開發資料庫應用程式
3) **預編程式**
4) Transaction 交易模式
5) DAO pattern

Prepared Statement

- 預編程式
  - 將SQL敘述事先編譯
    - Statement 物件則是執行前才編譯
  - 提高執行效率
  - 避免 SQL Injection 隱碼攻擊
    - 應用程式中使用字串串接方式組合SQL指令
    - 輸入的資料中夾帶被資料庫認為是正常的SQL指令而執行



Prepared Statement

- 使用步驟
  - SQL敘述中未知值用問號表示
    - 字串, 日期不用加 ''
  - Connection 物件之 prepareStatment(String sql)取得 java.sql.PreparedStatement物件
  - 執行前用setXXX()來設定

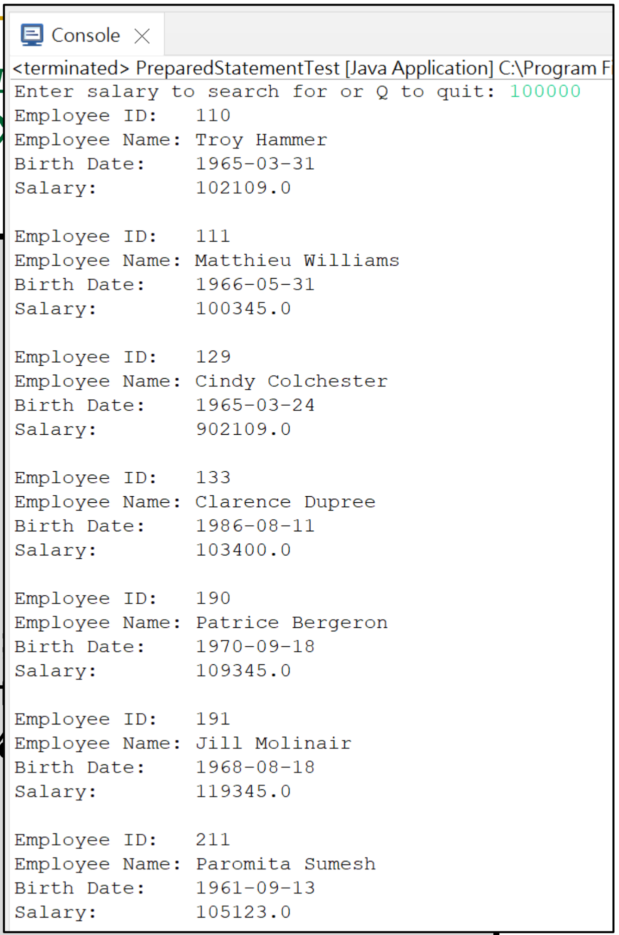
```
String sql = “SELECT * FROM Employee WHERE Salary > ?”;
PreparedStatement pstmt = con.prepareStatement(sql);
double value = 100_000.00;
pstmt.setDouble(1, value);
ResultSet rs = pstmt.executeQuery();
```

Prepared Statement範例

```
import java.sql.*;
import java.util.Date;
public class PreparedStatementTest {
    public static void main(String[] args) {
        String url = "jdbc:mysql://localhost:3306/EmployeeDB";
        String username = "root";        String password = "abc123";
        String input = "";
        double searchValue;
        BufferedReader in = new BufferedReader(new InputStreamReader(System.in));
        try (Connection con = DriverManager.getConnection(url, username, password);
                PreparedStatement pStmt = con.prepareStatement
                                                        ("SELECT * FROM Employee Where salary > ?“){
            while (true) {
                System.out.print("Enter salary to search for or Q to quit: ");
                input = in.readLine();
                if (input.equals("q") || input.equals("Q")) {
                    break;
                }
                searchValue = Double.valueOf(input);
                pStmt.setDouble(1, searchValue);
```

Prepared Statement範

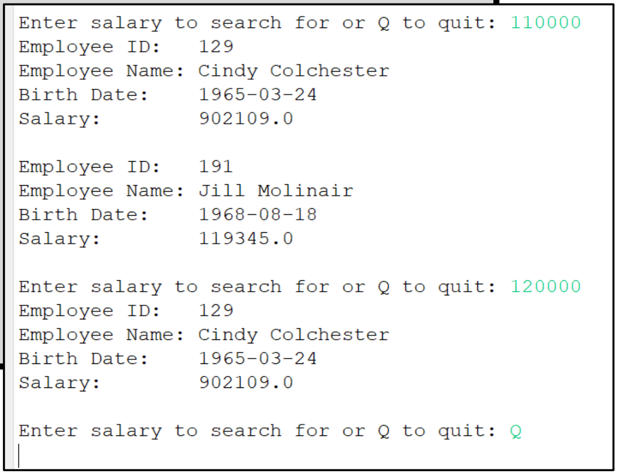
```java
                ResultSet rs = pStmt.executeQuery();
                while (rs.next()) {
                    int empID = rs.getInt("ID");
                    String first = rs.getString("FIRSTNAME");
                    String last = rs.getString("LASTNAME");
                    Date birth_date = rs.getDate("BIRTHDATE");
                    float salary = rs.getFloat("SALARY");
                    System.out.printf("Employee ID: %d%n", empID)
                    System.out.printf("Employee Name: %s %s%n",
                    System.out.printf("Birth Date: %tF%nSalary: %f%
                }// end of while
            }// end of while
        } catch (NumberFormatException n) {
            System.out.println("Please enter a valid number.");
        } catch (IOException | SQLException e) {
            System.out.println("SQLException: " + e);
        } // end of try-with-resources
    }
}
```



Console

<terminated> PreparedStatementTest [Java Application] C:\Program F

```
Enter salary to search for or Q to quit: 100000
Employee ID:   110
Employee Name: Troy Hammer
Birth Date:    1965-03-31
Salary:        102109.0

Employee ID:   111
Employee Name: Matthieu Williams
Birth Date:    1966-05-31
Salary:        100345.0

Employee ID:   129
Employee Name: Cindy Colchester
Birth Date:    1965-03-24
Salary:        902109.0

Employee ID:   133
Employee Name: Clarence Dupree
Birth Date:    1986-08-11
Salary:        103400.0

Employee ID:   190
Employee Name: Patrice Bergeron
Birth Date:    1970-09-18
Salary:        109345.0

Employee ID:   191
Employee Name: Jill Molinair
Birth Date:    1968-08-18
Salary:        119345.0

Employee ID:   211
Employee Name: Paromita Sumesh
Birth Date:    1961-09-13
Salary:        105123.0
```



```
Enter salary to search for or Q to quit: 110000
Employee ID:   129
Employee Name: Cindy Colchester
Birth Date:    1965-03-24
Salary:        902109.0

Employee ID:   191
Employee Name: Jill Molinair
Birth Date:    1968-08-18
Salary:        119345.0

Enter salary to search for or Q to quit: 120000
Employee ID:   129
Employee Name: Cindy Colchester
Birth Date:    1965-03-24
Salary:        902109.0

Enter salary to search for or Q to quit: Q
```

課程大綱

1) 課前準備
2) 使用JDBC API 開發資料庫應用程式
3) 預編程式
4) **Transaction 交易模式**
5) DAO pattern

Transaction 交易模式

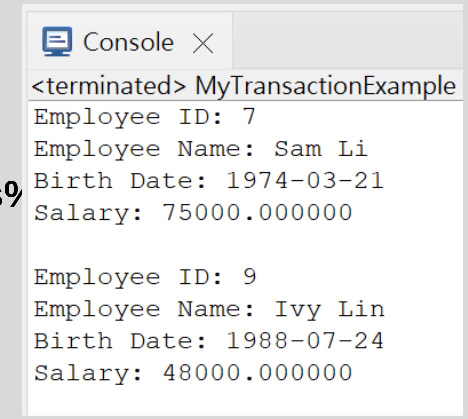
- Transaction 交易模式
  - 一系列不可拆分完整的資料庫操作
  - Atomicity 原子性
    - 交易中的所有操作，要麼全部完成，要麼全部不完成
    - 執行過程中發生錯誤，會被回復(Rollback)到交易開始前狀態
  - Consistency 一致性
    - 在交易開始之前和交易結束以後，資料庫的完整性沒有被破壞
  - Isolation 隔離性
    - 指多個交易並行執行時資料的相互關係
    - 分為Read uncommitted、read committed、repeatable read和Serializable等四個等級
  - Durablity 持久性
    - 交易完成後，該變更會永久改變資料的狀態

交易模式相關方法

- java.sql.Connection 物件提供交易模式相關方法

Connection 方法	傳回值	說明
setAutoCommit(boolean autoCommit)	void	設定此Connection 的自動認可模式. true:每個敘述執行完後自動確認, false: 手動呼叫commit()才確認
setSavepoint(String name)	Savepoint	設定一個交易的回復點，並為其指定名稱
releaseSavepoint(Savepoint savepoint)	void	刪除指定名稱的回復點
rollback()		取消在當前交易中進行的所有更改，並釋放此 Connection 物件持有的所有資料庫鎖
rollback(Savepoint savepoint)	void	取消在當前交易中進行的所有更改至指定回復點，並釋放此 Connection 物件持有的所有資料庫鎖
commit()	void	確認自上一次確認/回復後進行的所有變更，並釋放此 Connection 物件持有的所有資料庫鎖

```java
import java.sql.*;     import java.util.Date;
public class MyTransactionExample {
    public static void main(String[] args) {
        String url = "jdbc:mysql://localhost:3306/EmployeeDB";
        String username = "root";        String password = "abc123";
        try (Connection con = DriverManager.getConnection(url, username, password);
               Statement stmt = con.createStatement() ){
            con.setAutoCommit(false);
            stmt.executeUpdate("INSERT INTO EMPLOYEE VALUES (7, ‘Sam', ‘Li', '1974-03-21', 75000)");
            Savepoint sp = con.setSavepoint();
            stmt.executeUpdate("INSERT INTO EMPLOYEE VALUES (8, ‘Sue', ‘Hu', '1975-11-26', 50000)");
            con.rollback(sp);
            stmt.executeUpdate("INSERT INTO EMPLOYEE VALUES (9, ‘Ivy', ‘Lin', '1988-07-24', 48000)");
            con.commit();
            ResultSet rs = stmt.executeQuery("SELECT * FROM Employee WHERE id < 10");
            while (rs.next()) {
                int empID = rs.getInt("ID");
                String first = rs.getString("FirstName");
                String last = rs.getString("LastName");
                System.out.printf("Employee ID: %d%nEmployee Name: %s %s%
            }
            con.commit();       rs.close();
        } catch (SQLException ex) {    ……..   }
    }
}
```



Console

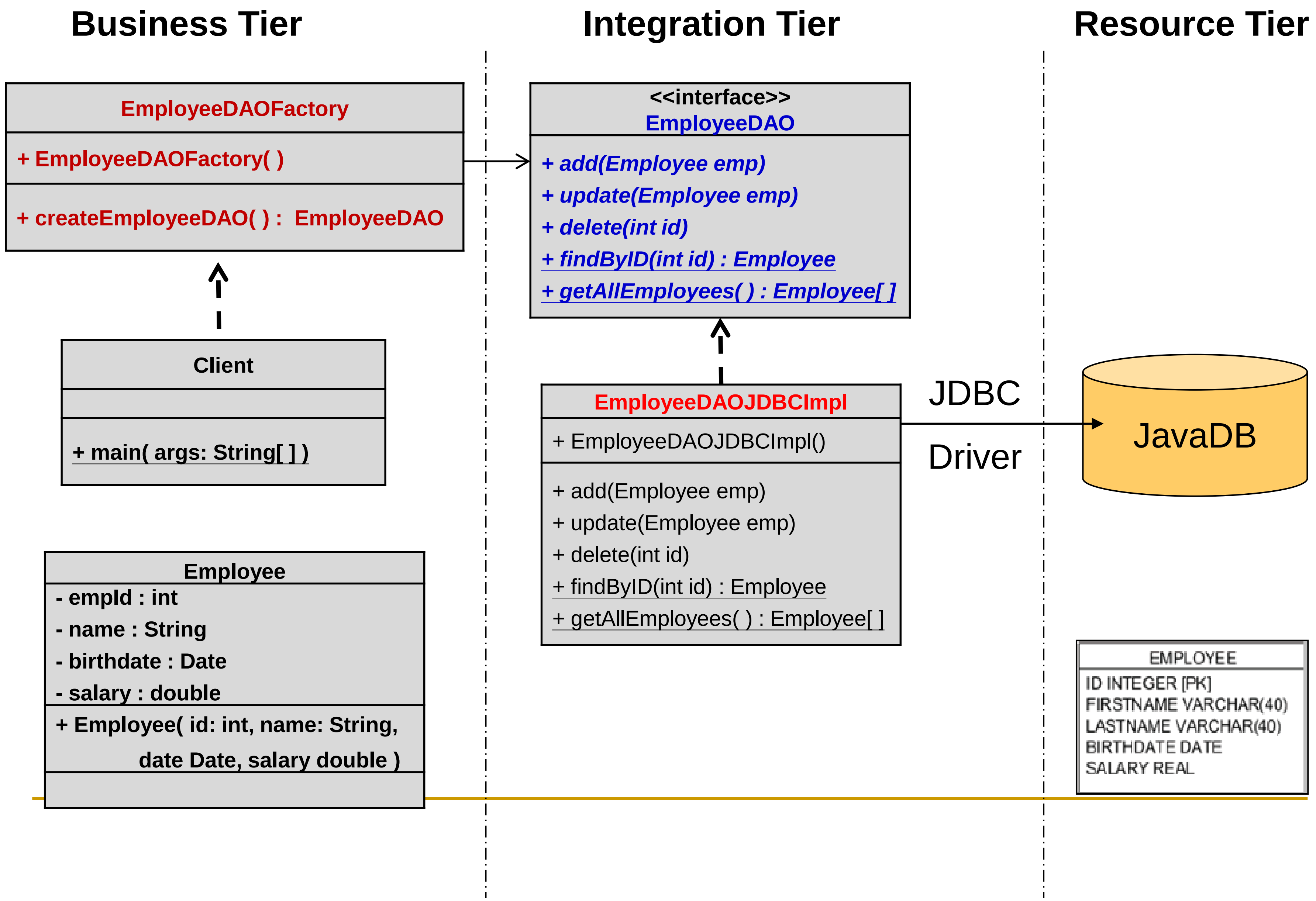
<terminated> MyTransactionExample

```
Employee ID: 7
Employee Name: Sam Li
Birth Date: 1974-03-21
Salary: 75000.000000

Employee ID: 9
Employee Name: Ivy Lin
Birth Date: 1988-07-24
Salary: 48000.000000
```

課程大綱

1) 課前準備
2) 使用JDBC API 開發資料庫應用程式
3) 預編程式
4) Transaction 交易模式
5) **DAO pattern**

DAO Design Pattern


Business Tier
Integration Tier
Resource Tier
EmployeeDAOFactory
+ EmployeeDAOFactory( )
+ createEmployeeDAO( ) : EmployeeDAO
<<interface>>
EmployeeDAO
+ add(Employee emp)
+ update(Employee emp)
+ delete(int id)
+ findByID(int id) : Employee
+ getAllEmployees( ) : Employee[ ]
Client
+ main(args: String[ ])
EmployeeDAOJDBCImpl
+ EmployeeDAOJDBCImpl()
+ add(Employee emp)
+ update(Employee emp)
+ delete(int id)
+ findByID(int id) : Employee
+ getAllEmployees( ) : Employee[ ]
JDBC
Driver
JavaDB
Employee
- empId : int
- name : String
- birthdate : Date
- salary : double
+ Employee(id: int, name: String, date Date, salary double)
EMPLOYEE
ID INTEGER [PK]
FIRSTNAME VARCHAR(40)
LASTNAME VARCHAR(40)
BIRTHDATE DATE
SALARY REAL