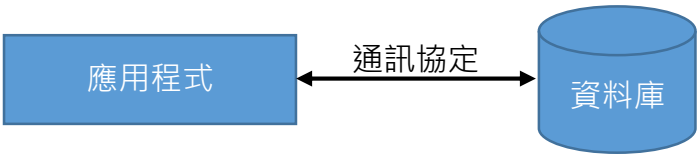
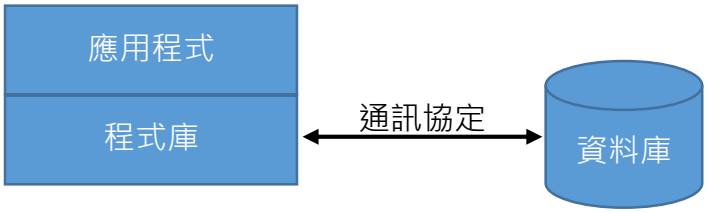
JDBC 入門

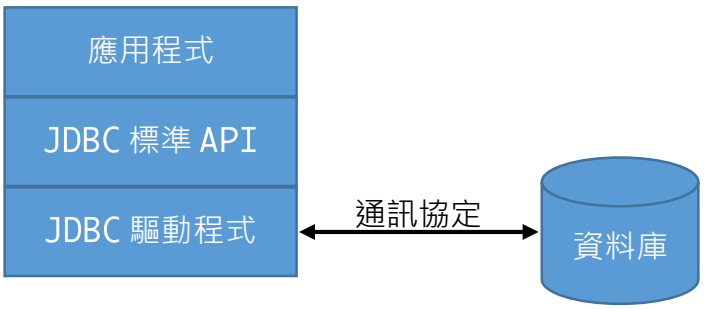
簡介 JDBC



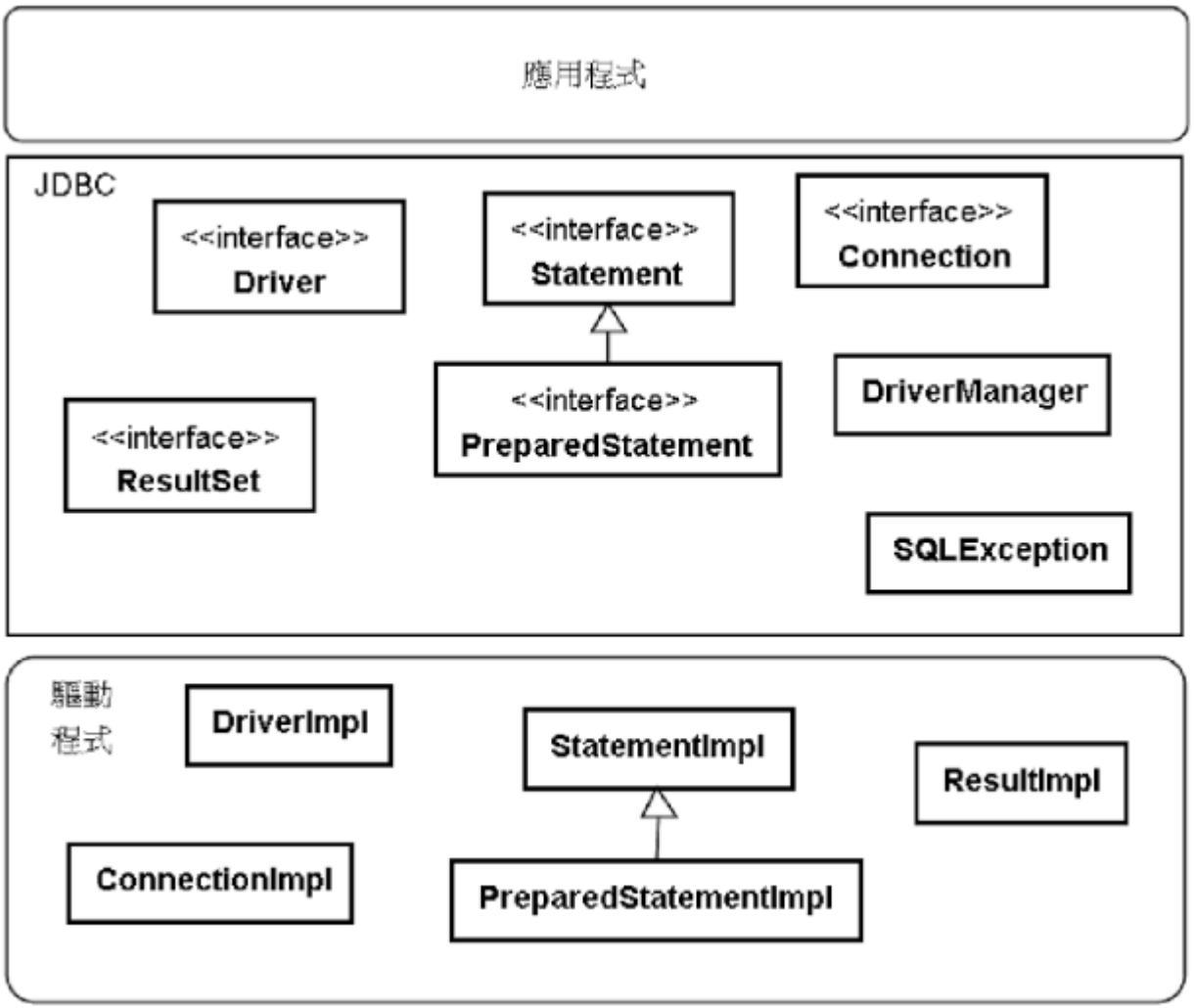
使用程式庫簡化程式撰寫

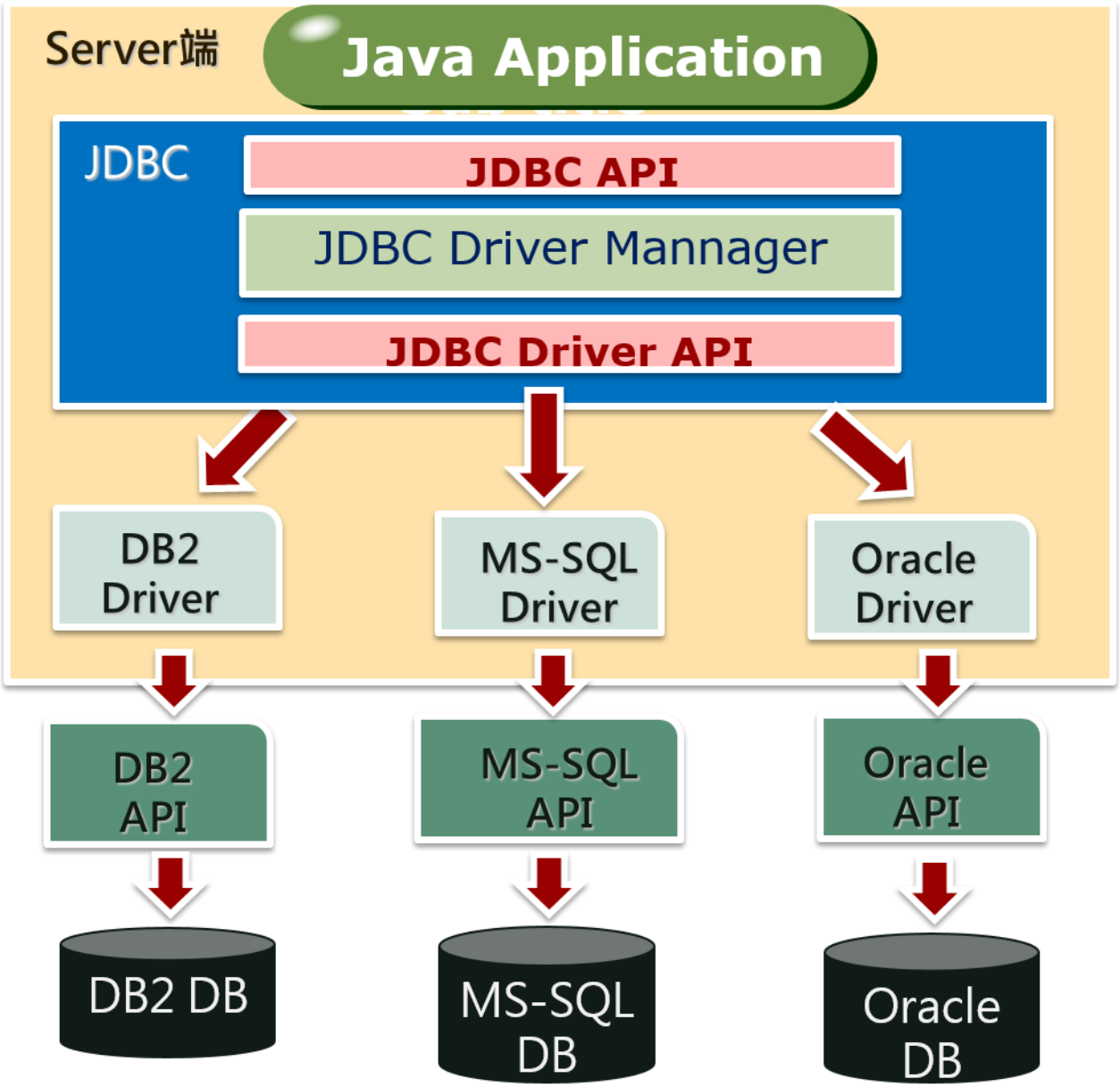


各資料庫的通訊協定不同，API 也不同，如果直接使用程式庫的話，要針對不同資料庫寫不同的程式



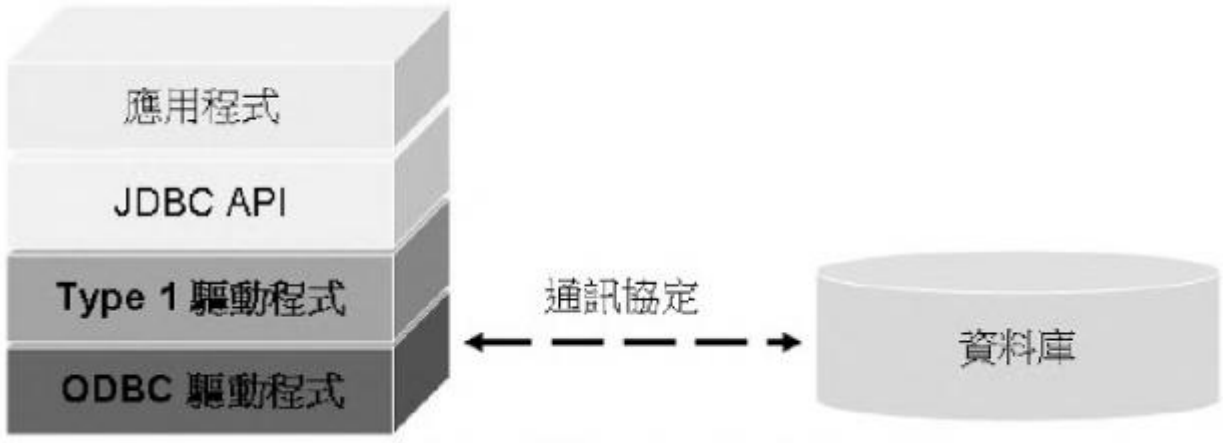
- JDBC 全名為 Java DataBase Connectivity，是 Java 連線資料庫的標準規範
- 定義一組標準類別與介面
- 分成兩個部分
  - JDBC 應用程式開發者介面(Application Developer Interface)
    - 應用程式使用(java.sql, javax.sql)
  - JDBC 驅動程式開發者介面(Driver Developer Interface)
    - 資料庫廠商會實作標準 API 中的介面，稱為 JDBC 驅動程式(Driver)


應用程式
JDBC
<<interface>>
Driver
<<interface>>
Statement
<<interface>>
Connection
<<interface>>
PreparedStatement
DriverManager
<<interface>>
ResultSet
SQLException
驅動
程式
DriverImpl
StatementImpl
ResultImpl
ConnectionImpl
PreparedStatementImpl



- 使用時需要在 classpath 設定 JDBC 驅動程式(jar 檔)
- 理想：一個程式操作所有資料庫
- 現實：轉移資料庫時，需要修改特定資料庫的 SQL 語法、資料型態或內建函式

Type 1: JDBC-ODBC Bridge Driver



- 將 JDBC 呼叫轉為 ODBC 驅動程式的呼叫，再由 ODBC 驅動程式操作資料庫
  - ODBC(Open DataBase Connectivity)是由 Microsoft 主導的資料庫連接標準

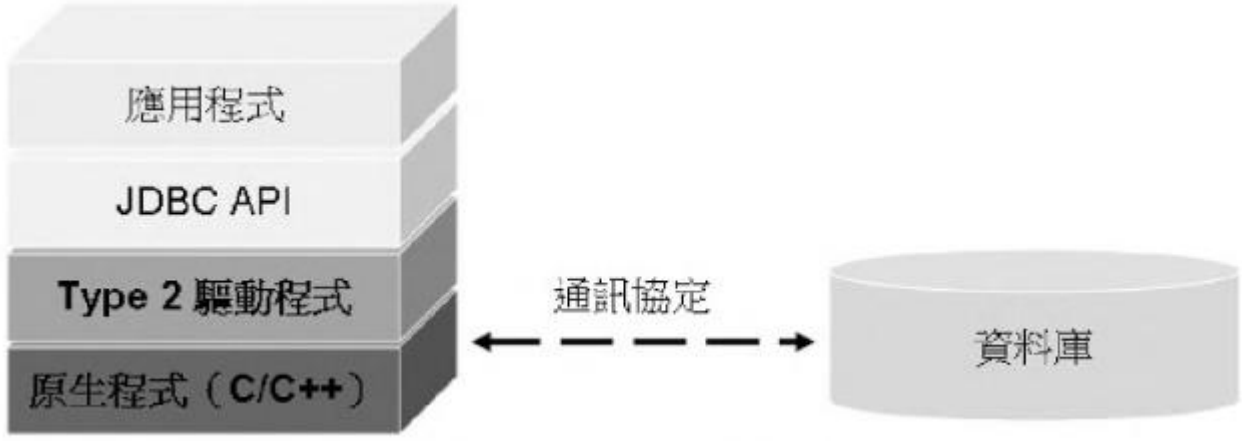
優點

- 容易實作驅動程式(Oracle JDK 有 sun.jdbc.odbc 開頭的套件)

缺點

- JDBC 與 ODBC 不是一對一，所以部分功能無法直接轉換，部份功能受限，即便可以多層轉換達成，存取速度也會因此受限
- ODBC 需在使用平台事先設定，彈性不足
- ODBC 驅動程式有跨平台限制

Type 2: Native API Driver



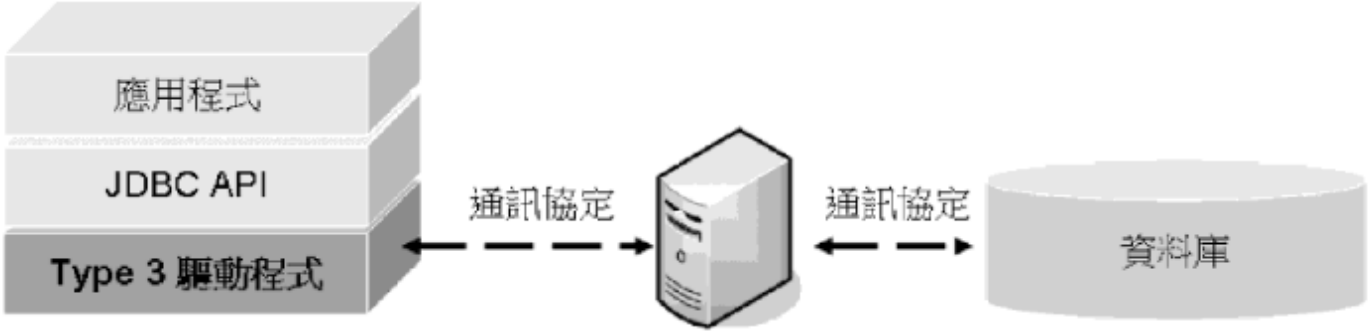
優點

- 速度快（仍比 Type 3, Type 4 慢）：因為直接呼叫資料庫原生 API，所以在資料庫回應資料後，建構相關 JDBC API 實作物件時最快

缺點

- 沒有跨平台：驅動程式與平台相依

Type 3: JDBC-Net Driver



- 將 JDBC 轉為特定的網路協定(Protocol)呼叫，與資料庫特定的中介伺服器或元件進行協定操作，再由中介伺服器或元件操作資料庫

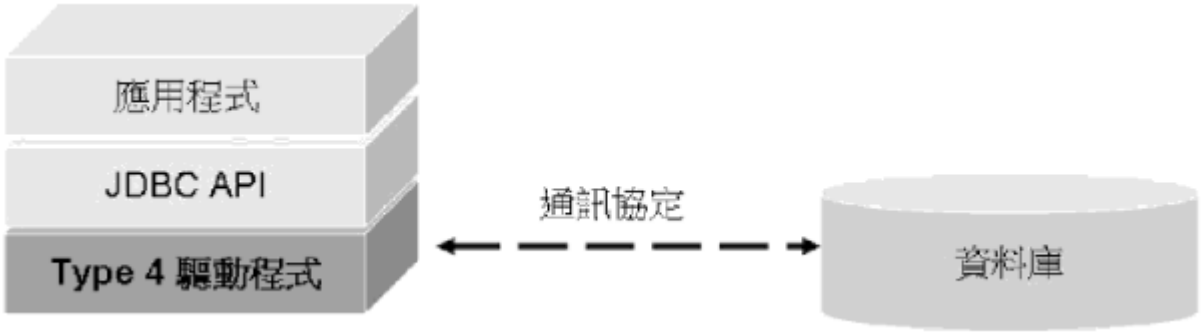
優點

- 架構有彈性、可用純 Java 技術實現：因為只是將 JDBC 呼叫對應至網路協定
- 只要更換中介伺服器或元件即可更換資料庫

缺點

- 速度比 Type 4 慢

Type 4: Native Protocol Driver



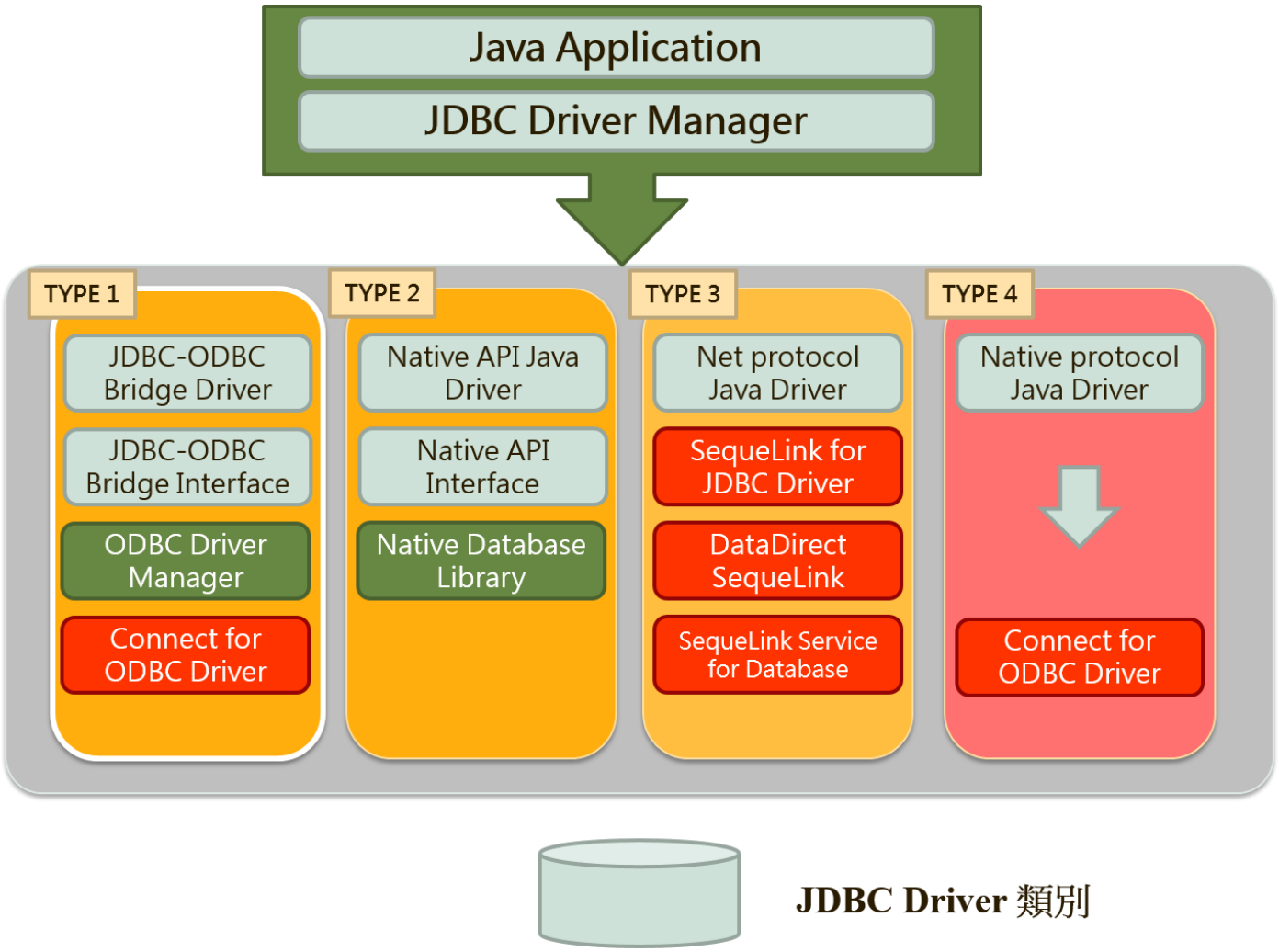
- 驅動程式實作由廠商提供
- 驅動程式實作會將 JDBC 呼叫轉換為特定的網路協定，以與資料庫溝通。可用純 Java 實現

優點

- 跨平台
- 速度不錯

缺點

- 驅動程式由各資料庫廠商提供



連接資料庫

註冊 Driver 實作物件

- 管理 Driver 實作物件的類別是 java.sql.DriverManager
- 註冊(實際上很少這樣寫)

```
DriverManager.registerDriver(new 資料庫驅動程式類別)
```

- 載入實作類別的.class 檔即可完成註冊

```
Class.forName("com.ibm.db2.jcc.DB2Driver");
```

載入.class 檔的方式

1. 使用 Class.forName()
2. 自行建立 Driver 介面實作類別的實例

```
Java.sql.Driver driver = new com.ibm.db2.jcc.DB2Driver();
```

3. 啟動 JVM 時指定 jdbc.drivers 屬性

```
Java uDjdbc.drivers=com.ibm.db2.jcc.DB2Driver;ooo.XXXDriver YourProgram
```

4. 啟動 JAR 中/META-INF/java.sql.Driver 檔案(JDK6↑, JDBC4.0)
    - 在驅動程式實作的 jar 檔/META-INF/services 資料夾中放一個 java.sql.Driver 檔，裡面撰寫 Driver 介面的實作類別名稱全名，DriverManager 會自動讀取這個檔案並找到指定類別進行註冊

取得 Connection 實作物件

- 是資料庫連線的代表物件

```
Connection conn = DriverManager.getConnection(jdbcUrl, username,
password);
```

- JDBC URL:定義了連接資料庫的協定(jdbc)、子協定（各資料庫皆不同）、資料來源識別（資料庫的位址、埠號、名稱）

```
jdbc:子協定:資料來源識別
```

- 使用者名稱、密碼

關閉 Connection 實作物件

- isClosed(): 測試與資料庫的連結是否關閉
- close(): 關閉與資料庫的連接（釋放連接時的相關資源，例如：連線相關物件、授權資源等）

[練習] com.cathay.practice.Q1

使用 Statement、ResultSet

- 執行 SQL 的話，要取得 java.sql.Statement 實作物件

```
Statement stmt = conn.createStatement();
```

- 取得之後，可以使用 executeUpdate()、executeQuery()執行 SQL

- executeUpdate: CREATE TABLE, INSERT, UPDATE, DELETE, DROP TABLE, ALTER TABLE
  - 回傳更新筆數(int)
- executeQuery: SELECT
  - 回傳查詢結果(ResultSet)
- execute:
  - TRUE: 回傳 ResultSet，可用 getResultSet()取得
  - FALSE: 回傳更新筆數或沒有結果，可用 getUpdateCount()取得更新筆數

- java.sql.ResultSet
  - 表示查詢結果
  - 可以使用 next()移動到下一筆資料，回傳結果(true, false)則會知道有沒有資料
  - 取得資料要用 getXXX(XXX 表示要取得的資料型態)
- Statement 和 ResultSet 不使用時，也可以用 close()關閉，而且 Statement 關閉時會把關聯的 ResultSet 一起關閉；通常驅動程式實作時會在關閉 Connection 時，一併關閉 Statement

[例] com.cathay.lesson.basic.Statement1

使用 PreparedStatement、CallableStatement

PreparedStatement

- Statement 執行的 SQL 需要用串接的方式將值與 SQL 連接起來
  - 缺點
    - 容易造成 SQL Injection
    - 對 DB 來說相同 SQL 但是值不同就是不同 SQL，每次執行都要先經過解析
- PreparedStatement 可以建立預先編譯的 SQL，會變動的參數則用?取代，等到要執行的時候才將值設定到對應的位置
- 執行完成後，可以使用 clearParameters()清除設定的參數，讓 PreparedStatement 重複使用

[例] com.cathay.lesson.basic.PreparedStatement1

CallableStatement

- 如果要使用 JDBC 來呼叫 Stored Procedure 的話，則可使用 java.sql.CallableStatement
- 使用方式和 PreparedStatement 差不多，除了要用 prepareCall()產生 CallableStatement 實例，也可以使用 registerOutParameter 註冊輸出參數
- Java 型態和 SQL 型態不是一對一關係
- Date, Time, Timestamp 是用 java.sql 下的

JDBC 進階

使用 DataSource 取得連線

[例] com.cathay.lesson.advanced

- 應用程式更常透過 JNDI 從伺服器取得設定好的 DataSource，再從 DataSource 取得 Connection
  - JNDI: Java Naming and Directory Interface
    - Java EE 提供的介面，讓 Java 程式透過該介面找尋及取得所需的資源(e.g. 資料庫連線、EJB、JMS、java mail 等)

使用 ResultSet 捲動、更新資料

ResultSet 的類型

1. ResultSet.TYPE_FORWARD_ONLY(預設)
   - 資料游標只能前進
2. ResultSet.TYPE_SCROLL_INSENSITIVE
   - 資料游標可以前進後退
   - 不會反映資料庫的資料異動
3. ResultSet.TYPE_SCROLL_SENSITIVE
   - 資料游標可以前進後退
   - 會反映資料庫的資料異動

更新設定

1. ResultSet.CONCUR_READ_ONLY(預設)
   - 只能讀資料
2. ResultSet.CONCUR_UPDATABLE
   - 可以更新資料

- 可以在建立 Statement 或 PreparedStatement 時指定

要使用 ResultSet 修改資料有下列限制

- 必須選取單一表格
- 必須選取主鍵
- 必須選取所有 NOT NULL 的值

ResultSet API

API	說明
absolute	
afterLast	
beforeFirst	
first	
last	

relative	
previous	
next	移動資料游標到下一筆
isAfterLast	
isBeforeFirst	
isFirst	
isLast	
getXXX	取得資料
setXXX	設定資料
updateXXX	更新資料
updateRow	完成更新
cancelRowUpdates	取消更新(需在 updateRow 之前)
moveToInsertRow	在新增資料前(需在 updateXXX 之前)
insertRow	新增資料列(需在 updateXXX 之後)
deleteRow	刪除資料列(需在 updateXXX 之後)

簡介交易

交易的四個基本要求

1. 原子性(Atomicity)
2. 一致性(Consistency)
3. 隔離行為(Isolation behavior)
4. 持續性(Durability)

JDBC 的原子性實例

- setAutoCommit(false)
- commit(), rollback()
- setSavepoint(), releaseSavepoint()

[例] com.cathay.lesson.advanced.Transaction1

[例] com.cathay.lesson.advanced.Transaction2