

CH18 練習

鄭安翔

ansel_cheng@hotmail.com

練習1 使用Derby及JDBC

1. 下載安裝 MySQL資料庫
2. 建立EmployeeDB資料庫
 - 執行 employeeTable.sql
 - 檢驗資料
3. 開啟 SimpleJDBCExample 專案
 - 指定JDBC Driver路徑
4. 測試、執行
5. 修改主類別SimpleJDBCExample
 - 加入一筆客戶資料
6. 測試、執行

測試、執行

SimpleJDBCTest.java ×

```
10 public class SimpleJDBCTest {
11
12     public static void main(String[] args) {
13         String url = "jdbc:mysql://localhost:3306/EmployeeDB";
14         String username = "root";
15         String password = "abc123";
16         String query = "SELECT * FROM Employee";
17         try (Connection con = DriverManager.getConnection(url, username, password);
18             Statement stmt = con.createStatement();
19             ResultSet rs = stmt.executeQuery(query)){
20             while (rs.next()) {
21                 int empID = rs.getInt("ID");
22                 String first = rs.getString("FirstName");
23                 String last = rs.getString("LastName");
24                 Date birthDate = rs.getDate("BirthDate");
25                 float salary = rs.getFloat("Salary");
26                 System.out.println("Employee ID: " + empID + "\n"
27                                     + "Employee Name: " + first + " " + last + "\n"
28                                     + "Birth Date: " + birthDate + "\n"
29                                     + "Salary: " + salary + "\n");
30             } // end of while
31         } catch (SQLException ex) {
32             while (ex != null) {
33                 System.out.println("SQLState: " + ex.getSQLState());
34                 System.out.println("Error Code:" + ex.getErrorCode());
35                 System.out.println("Message: " + ex.getMessage());
36                 Throwable t = ex.getCause();
37                 while (t != null) {
38                     System.out.println("Cause:" + t);
39                     t = t.getCause();
40                 }
41                 ex = ex.getNextException();
42             }
43         }
44     }
45 }
```

Console ×

<terminated> SimpleJDBCTest [Java Application] C:\

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

加入一筆客戶資料

SimpleJDBCTest.java ×

```
10 public class SimpleJDBCTest {
11
12     public static void main(String[] args) {
13         String url = "jdbc:mysql://localhost:3306/EmployeeDB";
14         String username = "root";
15         String password = "abc123";
16         String query = "SELECT * FROM Employee";
17         try (Connection con = DriverManager.getConnection(url, username, password);
18             Statement stmt = con.createStatement();
19             ResultSet rs = stmt.executeQuery(query)) {
20             while (rs.next()) {
21                 int empID = rs.getInt("ID");
22                 String first = rs.getString("FirstName");
23                 String last = rs.getString("LastName");
24                 Date birthDate = rs.getDate("BirthDate");
25                 float salary = rs.getFloat("Salary");
26                 System.out.println("Employee ID: " + empID + "\n"
27                                     + "Employee Name: " + first + " " + last + "\n"
28                                     + "Birth Date: " + birthDate + "\n"
29                                     + "Salary: " + salary + "\n");
30             }
31             // Add a record
32             String update = "INSERT INTO Employee VALUES (11, 'Sean', 'Cheng', '1974-03-21', 75000)";
33             if (stmt.executeUpdate(update) == 1)
34                 System.out.println("新增11號員工成功!");
35             else
36                 System.out.println("新增11號員工失敗!");
37
38         } catch (SQLException ex) {
39             while (ex != null) {
40                 System.out.println("SQLState: " + ex.getSQLState());
41                 System.out.println("Error Code: " + ex.getErrorCode());
42                 System.out.println("Message: " + ex.getMessage());
43                 Throwable t = ex.getCause();
44                 while (t != null) {
45                     System.out.println("Cause: " + t);
46                     t = t.getCause();
47                 }
48                 ex = ex.getNextException();
49             }
50         }
51     }
52 }
53
```

測試、執行

Result Grid					
Filter Rows:					
Edit:					
ID	FIRSTNAME	LASTNAME	BIRTHDATE	SALARY	
11	Sean	Cheng	1974-03-21	75000	
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	Molnair	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		

```
Console X
<terminated> SimpleJDBCTest [Java Application]

Employee ID: 200
Employee Name: Patricia Arnant
Birth Date: 1970-10-31
Salary: 79345.0

Employee ID: 201
Employee Name: Thomas Fitzpatrick
Birth Date: 1961-09-22
Salary: 75123.5

Employee ID: 202
Employee Name: Thomas Heimer
Birth Date: 1967-07-22
Salary: 79123.5

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

新增11號員工成功!
```

```
Console X
<terminated> SimpleJDBCTest [Java Application] C:\Program Files\Java\jdk-17.0.4\bin

Employee ID: 101
Employee Name: Abhijit Gopali
Birth Date: 1956-06-01
Salary: 89345.0

Employee ID: 11
Employee Name: Sean Cheng
Birth Date: 1974-03-21
Salary: 75000.0

SQLState: 23000
Error Code:1062
Message: Duplicate entry '11' for key 'employee.PRIMARY'
```

練習2 實作DAO設計模式

1. 開啟 EmployeeDAO 專案, 檢視下列類別:
 - ❑ com.example.test.EmployeeTestInteractive.java
 - ❑ com.example.model.Employee.java
 - ❑ com.example.dao.EmployeeDAOFactory.java
 - ❑ com.example.dao.EmployeeDAO介面
 - ❑ com.example.dao.EmployeeDAOFileImpl.java
 - ❑ com.example.dao.DAOException.java
2. 新增com.example.dao.EmployeeDAOJDBCImpl.java
 - ❑ 實作com.example.dao.EmployeeDAO介面
 - ❑ 新增私有屬性private Connection con;
 - ❑ 新增無參數建構子
 - 建立Connection物件con
 - url = "jdbc:mysql://localhost:3306/EmployeeDB";
 - username = "root";
 - password = "abc123";

練習2 實作DAO設計模式

- ❑ 實作 add(emp : Employee) 方法
 - INSERT INTO EMPLOYEE VALUES (*id, firstName, lastName, birthDate, salary*)
- ❑ 實作 update(emp : Employee) 方法
 - UPDATE EMPLOYEE SET FIRSTNAME=*firstName*, LASTNAME=*lastName*, BIRTHDATE=*birthDate*, SALARY=*salary* WHERE ID=*id*;
- ❑ 實作 delete(id : int) 方法
 - DELETE FROM EMPLOYEE WHERE ID=*id*
- ❑ 實作 findById(id : int) : Employee 方法
 - SELECT * FROM EMPLOYEE WHERE ID=*id*
 - 取得ResultSet後,取得欄位資訊,建立對應Employee物件傳回
- ❑ 實作 getAllEmployees(): Employee[] 方法
 - SELECT * FROM EMPLOYEE
 - 取得ResultSet後,走訪每一筆資料
 - 取得資料欄位資訊,建立對應Employee物件,加入ArrayList<Employee>
 - 走訪完成後所得之ArrayList<Employee>, 轉成Employee[]後傳回
- ❑ 實作 close()方法
 - con.close();
- ❑ 以上方法均會丟出DAOException

練習2 實作DAO設計模式

3. 修改EmployeeDAOFactory

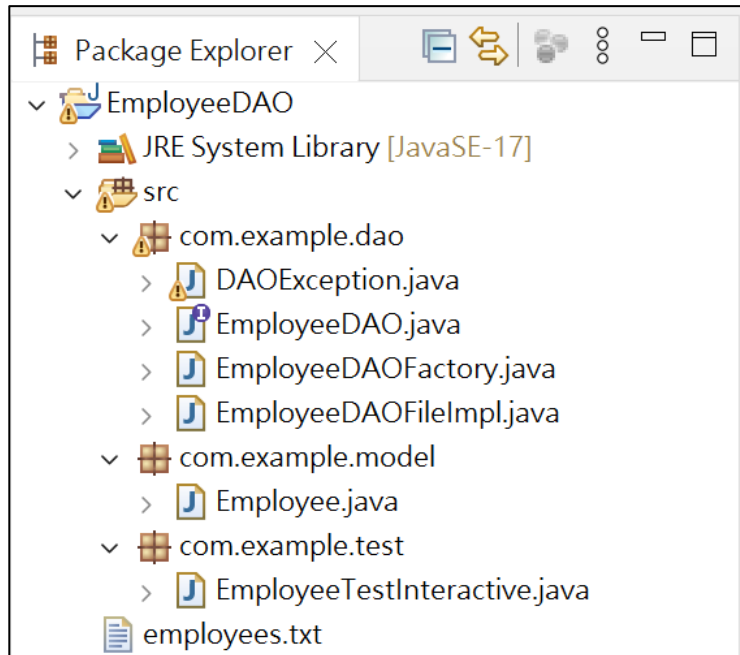
- 修改 createEmployeeDAO() : EmployeeDAO 方法
 - 傳回EmployeeDAOJDBCImpl物件

4. 刪除 EmployeeDAOFileImpl 類別

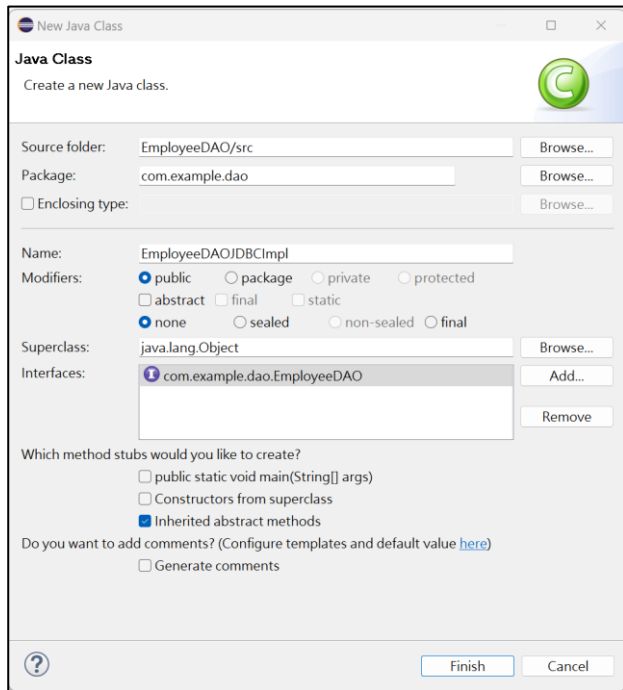
5. 新增 Java DB Driver 類別函式庫

6. 測試、執行

檢視 EmployeeDAO 專案



新增 com.example.dao. EmployeeDAOJDBCImpl.java



```
EmployeeDAOJDBCImpl.java ×
1 package com.example.dao;
2
3 import com.example.model.Employee;
4
5 public class EmployeeDAOJDBCImpl implements EmployeeDAO {
6
7     @Override
8     public void close() throws Exception {
9         // TODO Auto-generated method stub
10    }
11
12
13     @Override
14     public void add(Employee emp) throws DAOException {
15         // TODO Auto-generated method stub
16    }
17
18
19     @Override
20     public void update(Employee emp) throws DAOException {
21         // TODO Auto-generated method stub
22    }
23
24
25     @Override
26     public void delete(int id) throws DAOException {
27         // TODO Auto-generated method stub
28    }
29
30
31     @Override
32     public Employee findById(int id) throws DAOException {
33         // TODO Auto-generated method stub
34         return null;
35    }
36
37     @Override
38     public Employee[] getAllEmployees() throws DAOException {
39         // TODO Auto-generated method stub
40         return null;
41    }
42
43 }
44
```

新增屬性conn及建構子

```
EmployeeDAOJDBCImpl.java ×
1 package com.example.dao;
2
3 import com.example.model.Employee;
4 import java.sql.*;
5
6 public class EmployeeDAOJDBCImpl implements EmployeeDAO {
7
8     private Connection conn;
9
10    EmployeeDAOJDBCImpl() {
11        String url = "jdbc:mysql://localhost:3306/EmployeeDB";
12        String username = "root";
13        String password = "abc123";
14        try {
15            conn = DriverManager.getConnection(url, username, password);
16        } catch (SQLException ex) {
17            System.out.println("資料庫連線建立失敗:"+ex);
18            System.exit(0);
19        }
20    }
21
22
23    public void close() throws Exception {}
24
25
26    public void add(Employee emp) throws DAOException {}
27
28
29    public void update(Employee emp) throws DAOException {}
30
31
32    public void delete(int id) throws DAOException {}
33
34
35    public Employee findById(int id) throws DAOException {}
36
37
38    public Employee[] getAllEmployees() throws DAOException {}
39
40
41    }
42
43 }
```

實作 add(emp : Employee) 方法

```
33- @Override
34 public void add(Employee emp) throws DAOException {
35     //INSERT INTO EMPLOYEE VALUES (id, firstName, lastName, birthDate, salary)
36     String sql = "INSERT INTO EMPLOYEE VALUES (?, ?, ?, ?, ?)";
37     try(PreparedStatement pstmt = conn.prepareStatement(sql)){
38         pstmt.setInt(1, emp.getId());
39         pstmt.setString(2, emp.getFirstName());
40         pstmt.setString(3, emp.getLastName());
41         pstmt.setDate(4, new java.sql.Date(emp.getBirthDate().getTime()));
42         pstmt.setFloat(5, emp.getSalary());
43         if(pstmt.executeUpdate() != 1)
44             throw new DAOException("新增員工失敗!");
45     } catch(SQLException ex) {
46         throw new DAOException("資料庫新增發生錯誤:", ex);
47     }
48 }
49
```

實作 update(emp : Employee)方法

```
50 @Override
51 public void update(Employee emp) throws DAOException {
52     //UPDATE EMPLOYEE SET FIRSTNAME=firstName, LASTNAME=lastName, BIRTHDATE=birthDate, SALARY=salary WHERE ID=id;
53     String sql = "UPDATE EMPLOYEE SET FIRSTNAME=?, LASTNAME=?, BIRTHDATE=?, SALARY=? WHERE ID=?";
54     try(PreparedStatement pstmt = conn.prepareStatement(sql)){
55         pstmt.setInt(5, emp.getId());
56         pstmt.setString(1, emp.getFirstName());
57         pstmt.setString(2, emp.getLastName());
58         pstmt.setDate(3, new java.sql.Date(emp.getBirthDate().getTime()));
59         pstmt.setFloat(4, emp.getSalary());
60         if(pstmt.executeUpdate() != 1)
61             throw new DAOException("更新員工失敗!");
62     } catch(SQLException ex) {
63         throw new DAOException("資料庫更新發生錯誤:", ex);
64     }
65 }
66
```

實作 delete(id : int) 方法

```
67  @Override
68  public void delete(int id) throws DAOException {
69      //DELETE FROM EMPLOYEE WHERE ID=id
70      String sql = "DELETE FROM EMPLOYEE WHERE ID=?";
71      try(PreparedStatement pstmt = conn.prepareStatement(sql)) {
72          pstmt.setInt(1, id);
73          if(pstmt.executeUpdate() != 1)
74              throw new DAOException("刪除員工失敗!");
75      } catch(SQLException ex) {
76          throw new DAOException("資料庫刪除發生錯誤:", ex);
77      }
78  }
79
```

實作 findById(id:int):Employee 方法

```
81 ① @Override
82  public Employee findById(int id) throws DAOException {
83      //SELECT * FROM EMPLOYEE WHERE ID=id
84      String query = "SELECT * FROM EMPLOYEE WHERE ID=?";
85      Employee emp = null;
86      try(PreparedStatement pstmt = conn.prepareStatement(query)){
87          pstmt.setInt(1, id);
88          ResultSet rs = pstmt.executeQuery();
89          if(rs.next())
90              emp = new Employee(rs.getInt("ID"),
91                                rs.getString("FIRSTNAME"), rs.getString("LASTNAME"),
92                                rs.getDate("BIRTHDATE"), rs.getFloat("SALARY"));
93          return emp;
94      } catch(SQLException ex) {
95          throw new DAOException("資料庫查詢發生錯誤:", ex);
96      }
97  }
98
```

實作getAllEmployees():Employee[]

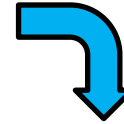
```
99  @Override
100  public Employee[] getAllEmployees() throws DAOException {
101      //SELECT * FROM EMPLOYEE
102      String query = "SELECT * FROM EMPLOYEE";
103      ArrayList<Employee> emps = new ArrayList<>();
104      try(Statement stmt = conn.createStatement()){
105          ResultSet rs = stmt.executeQuery(query);
106          while(rs.next())
107              emps.add(new Employee(rs.getInt("ID"),
108                                   rs.getString("FIRSTNAME"), rs.getString("LASTNAME"),
109                                   rs.getDate("BIRTHDATE"), rs.getFloat("SALARY")));
110          return emps.toArray(new Employee[0]);
111      } catch(SQLException ex) {
112          throw new DAOException("資料庫查詢發生錯誤:", ex);
113      }
114  }
115
```


實作 close() 方法

```
23  @Override
24  public void close() throws Exception {
25      try {
26          if (conn!=null) {
27              conn.close();
28          }
29      } catch(SQLException ex) {
30          System.out.println("資料庫連線關閉失敗："+ex);
31      }
32  }
33
```

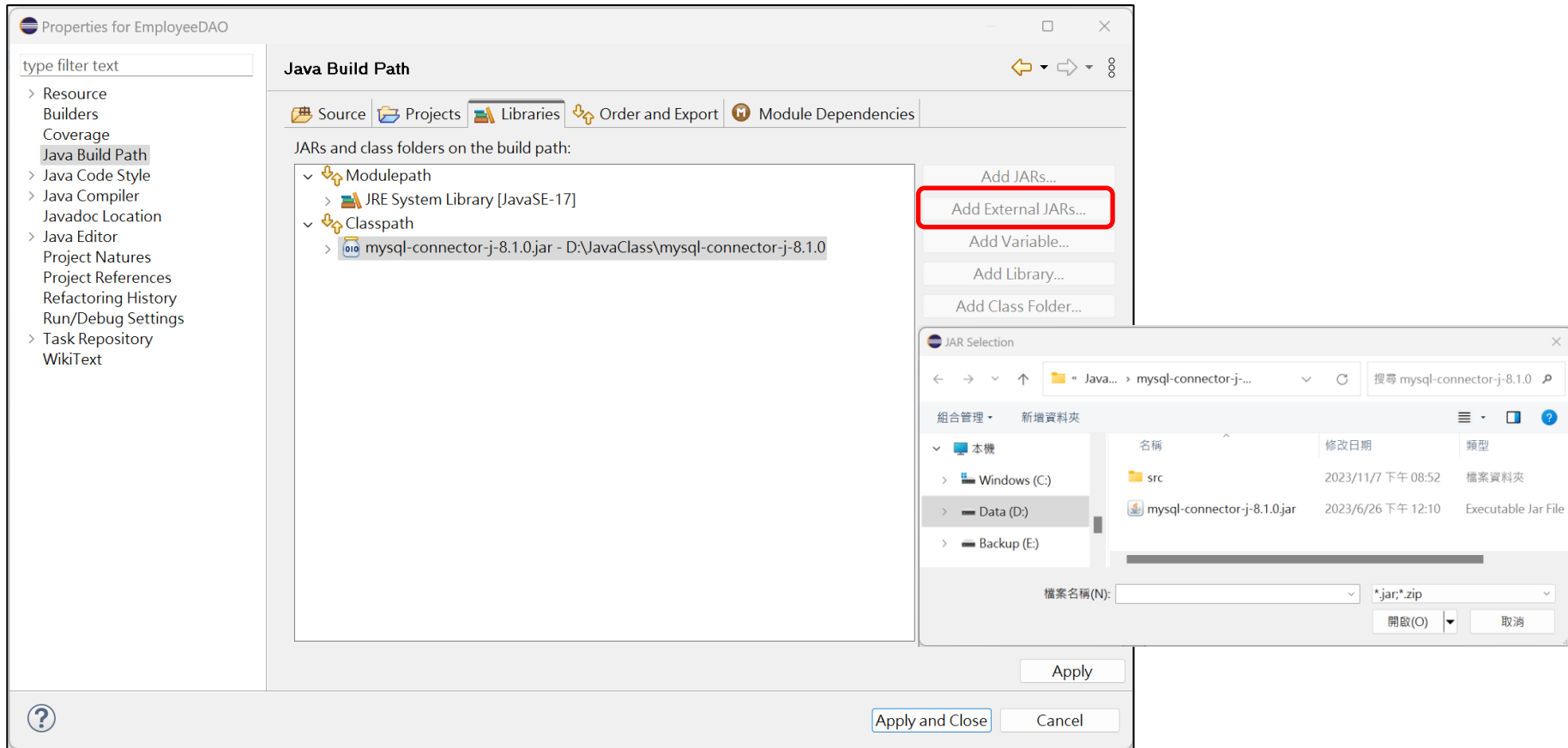
修改EmployeeDAOFactory

```
EmployeeDAOFactory.java ×
1 package com.example.dao;
2
3 public class EmployeeDAOFactory {
4
5     public EmployeeDAO createEmployeeDAO() {
6         return new EmployeeDAOFileImpl("employees.txt");
7     }
8
9 }
10
```

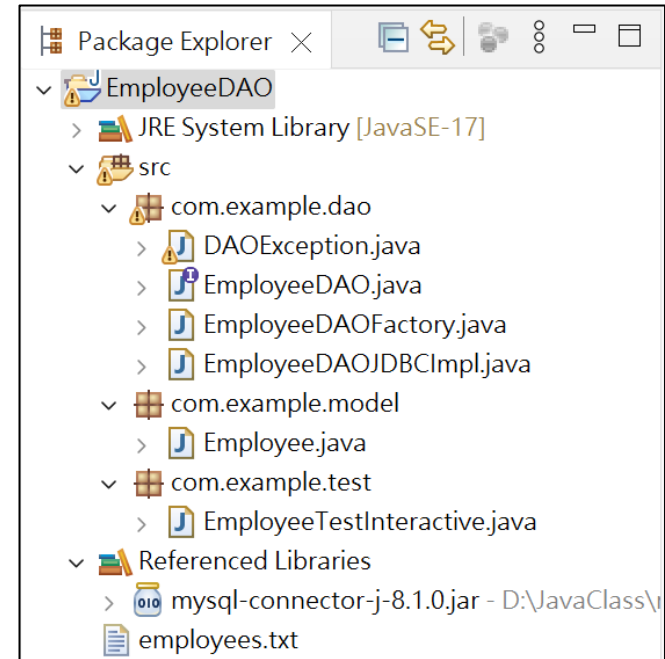
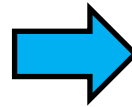
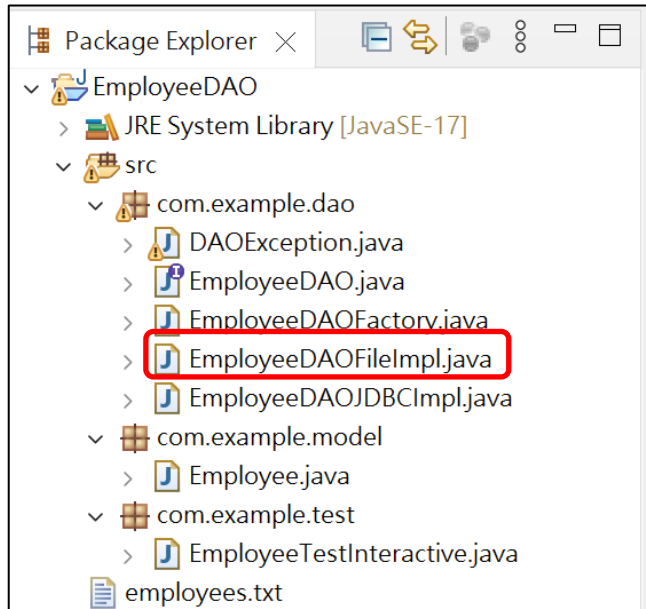


```
*EmployeeDAOFactory.java ×
1 package com.example.dao;
2
3 public class EmployeeDAOFactory {
4
5     public EmployeeDAO createEmployeeDAO() {
6         return new EmployeeDAOJDBCImpl();
7     }
8
9 }
10
```

新增 Java DB Driver 類別函式庫



删除 EmployeeDAOMemoryImpl



測試、執行

```
Console ×
EmployeeTestInteractive [Java Application] C:\Program Files\Java\jdk-17.0.4\bin\javaw

[C]reate | [R]ead | [U]pdate | [D]elete | [L]ist | [Q]uit:
C
Enter int value for employee id:
12
Enter value for employee first name :
David
Enter value for employee last name :
Wang
Enter value for employee birth date (MMM d, yyyy) :
Dec 25, 1980
Enter float value for employee salary :
80000
Successfully added Employee Record: 12

Created Employee ID: 12
Employee Name: David Wang
Birth Date: 12月 25, 1980
Salary: $80,000.00

[C]reate | [R]ead | [U]pdate | [D]elete | [L]ist | [Q]uit:
```

Result Grid						Filter Rows:	Edit:
	ID	FIRSTNAME	LASTNAME	BIRTHDATE	SALARY		
▶	11	Sean	Cheng	1974-03-21	75000		
	12	David	Wang	1980-12-25	80000		
	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		

測試、執行

Console X

EmployeeTestInteractive [Java Application] C:\Program Files\Java\jdk-17.0.4\bin\javaw.exe (2024年3月31日 下午7:11)

[C]reate | [R]ead | [U]pdate | [D]elete | [L]ist | [Q]uit:

U

Enter int value for employee id:

12

Modify the fields of Employee record: 12. Press return to accept current value.

Enter value for employee first name [David] :

Enter value for employee last name [Wang] :




Enter value for employee birth date (MMM d, yyyy) [Dec 25, 1980] :

Enter float value for employee salary [\$80,000.00] :

90000

Successfully updated Employee Record: 12

[C]reate | [R]ead | [U]pdate | [D]elete | [L]ist | [Q]uit:

Result Grid   Filter Rows: <input type="text"/> Edit: 					
	ID	FIRSTNAME	LASTNAME	BIRTHDATE	SALARY
▶	11	Sean	Cheng	1974-03-21	75000
	12	David	Wang	1980-12-25	90000
	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	O'Reilly	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

測試、執行

```
Console X
EmployeeTestInteractive [Java Application] C:\Program Files\Java\jdk-17.0.4\bin\javaw.

[C]reate | [R]ead | [U]pdate | [D]elete | [L]ist | [Q]uit:
D
Enter int value for employee id:
12
Deleted Employee 12

[C]reate | [R]ead | [U]pdate | [D]elete | [L]ist | [Q]uit:
R
Enter int value for employee id:
12

Employee 12 not found

[C]reate | [R]ead | [U]pdate | [D]elete | [L]ist | [Q]uit:
```

Result Grid						Filter Rows:	Edit:
	ID	FIRSTNAME	LASTNAME	BIRTHDATE	SALARY		
▶	11	Sean	Cheng	1974-03-21	75000		
	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		

測試、執行

```
Console ×
EmployeeTestInteractive [Java Application] C:\Program Files\Java\jdk-17.0.4\bin\javaw.

[C]reate | [R]ead | [U]pdate | [D]elete | [L]ist | [Q]uit:
L
Employee ID: 11
Employee Name: Sean Cheng
Birth Date: 3月 21, 1974
Salary: $75,000.00

Employee ID: 101
Employee Name: Abhijit Gopali
Birth Date: 6月 1, 1956
Salary: $89,345.00

Employee ID: 102
Employee Name: Peter Forrester
Birth Date: 11月 1, 1965
Salary: $99,345.00

Employee ID: 110
Employee Name: Troy Hammer
Birth Date: 3月 31, 1965
Salary: $102,109.00

Employee ID: 111
Employee Name: Matthieu Williams
Birth Date: 5月 31, 1966
Salary: $100,345.00

Employee ID: 120
Employee Name: Rajiv Sudahari
Birth Date: 12月 22, 1969
Salary: $68,400.20
```

```
Console ×
<terminated> EmployeeTestInteractive [Java Application] C:\Program Files\Java\jdk-17.0.4\bin\javaw.

Employee ID: 200
Employee Name: Patricia Arnant
Birth Date: 10月 31, 1970
Salary: $79,345.00

Employee ID: 201
Employee Name: Thomas Fitzpatrick
Birth Date: 9月 22, 1961
Salary: $75,123.50

Employee ID: 202
Employee Name: Thomas Heimer
Birth Date: 7月 22, 1967
Salary: $79,123.50

Employee ID: 211
Employee Name: Paromita Sumesh
Birth Date: 9月 13, 1961
Salary: $105,123.00

[C]reate | [R]ead | [U]pdate | [D]elete | [L]ist | [Q]uit:
Q
```