Spring JdbcTemplate

段維瀚 老師

Session

- 關於 Spring JdbcTemplate
- 配置 pom.xml 與專案資料夾
- SQLite 配置與測試
- JdbcTemplate CRUD 演練
- SimpleFlatMapper 微型資料對應框架

Spring JdbcTemplate

- 資料庫程式處理Connection的取得、Statement的建立、例外的處理、Statement的關閉、Connection的關閉等，對於一個基本的JDBC存取，這些流程是大同小異的，每一次您都必須作這樣的流程著實令人厭煩。
  好在 Spring 提供了 org.springframework.jdbc.core.JdbcTemplate 類別，它被設計為執行緒安全（Thread-safe），當中所提供的一些操作方法簡化了傳統 Java JDBC API 的使用。
  - ```
    @Autowired
    private JdbcTemplate jdbcTemplate;
    ```
    ```

配置 pom.xml

```xml
<!-- Spring JDBC -->
<dependency>
    <groupId>org.springframework</groupId>
    <artifactId>spring-jdbc</artifactId>
    <version>${spring.version}</version>
</dependency>

<dependency>
    <groupId>org.springframework</groupId>
    <artifactId>spring-orm</artifactId>
    <version>${spring.version}</version>
</dependency>

<dependency>
    <groupId>org.springframework</groupId>
    <artifactId>spring-tx</artifactId>
    <version>${spring.version}</version>
</dependency>
```

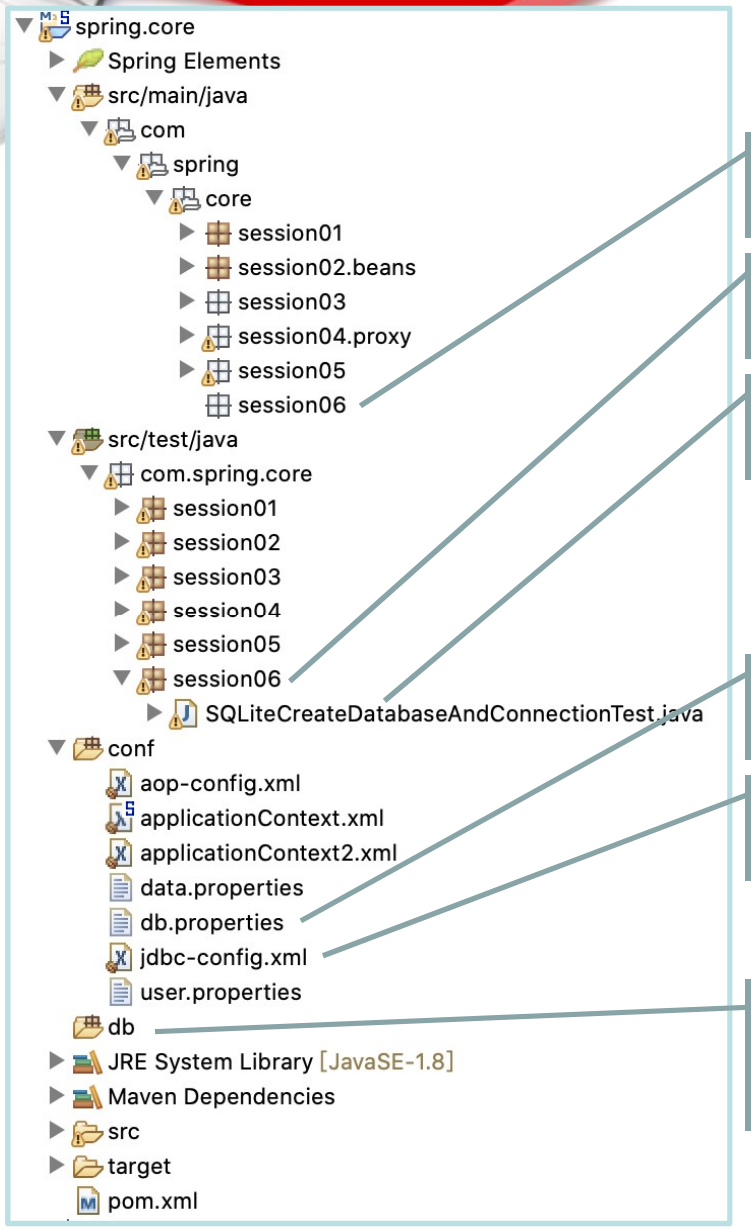
請自行決定資料庫要使用 MySQL 或 SQLite

```xml
<!-- MySQL -->
<dependency>
    <groupId>mysql</groupId>
    <artifactId>mysql-connector-java</artifactId>
    <version>8.0.19</version>
</dependency>

<!-- SQLite -->
<dependency>
    <groupId>org.xerial</groupId>
    <artifactId>sqlite-jdbc</artifactId>
    <version>3.36.0.3</version>
</dependency>

<!- c3p0 connection pool -->
<dependency>
    <groupId>com.mchange</groupId>
    <artifactId>c3p0</artifactId>
    <version>0.9.5.5</version>
</dependency>
```

專案資料夾



建立package：session06

建立package：session06

建立測試檔案

建立 db.properties檔案

建立 jdbc-config.xml

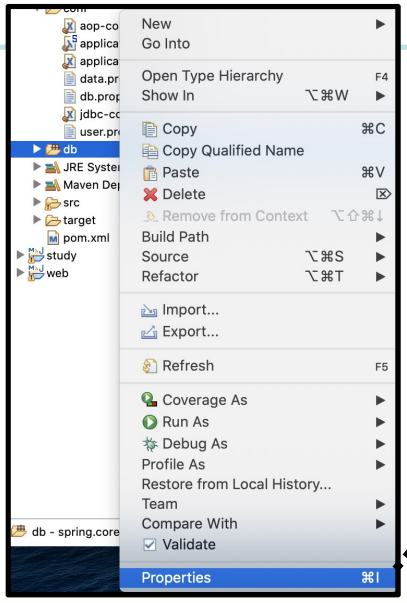
建立資源資料夾（Source Folder）
名稱：db

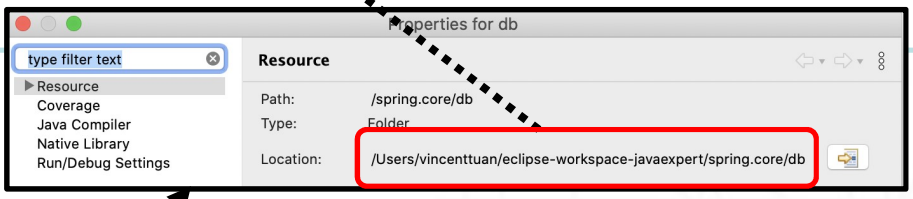
SQLite 配置

- db.properties

/mydb.db
資料庫名稱

```
jdbc.driver=org.sqlite.JDBC
jdbc.url=jdbc:sqlite:/Users/.../spring.core/db/mydb.db
jdbc.username=
jdbc.password=
```





SQLite 配置

- jdbc-config.xml

```
<!-- component-scan -->
<context:component-scan base-package="com.spring.core.session06" />

<!-- load db.properties -->
<context:property-placeholder location="db.properties" />

<!-- c3p0 pool -->
<bean id="dataSource" class="com.mchange.v2.c3p0.ComboPooledDataSource">
    <property name="driverClass" value="${jdbc.driver}" />
    <property name="jdbcUrl" value="${jdbc.url}" />
    <property name="user" value="${jdbc.username}" />
    <property name="password" value="${jdbc.password}" />
    <property name="maxConnectionAge" value="300" /> <!-- -1 表示沒有限制 -->
    <property name="maxIdleTimeExcessConnections" value="50" /> <!-- -1 表示沒有限制 -->
    <property name="debugUnreturnedConnectionStackTraces" value="true" />
</bean>

<!-- jdbc template -->
<bean id="jdbcTemplate" class="org.springframework.jdbc.core.JdbcTemplate">
    <property name="dataSource" ref="dataSource" />
</bean>

<!-- named parameter jdbc template -->
<bean id="namedParameterJdbcTemplate" class="org.springframework.jdbc.core.namedparam.NamedParameterJdbcTemplate">
    <constructor-arg ref="dataSource"/>
</bean>
```

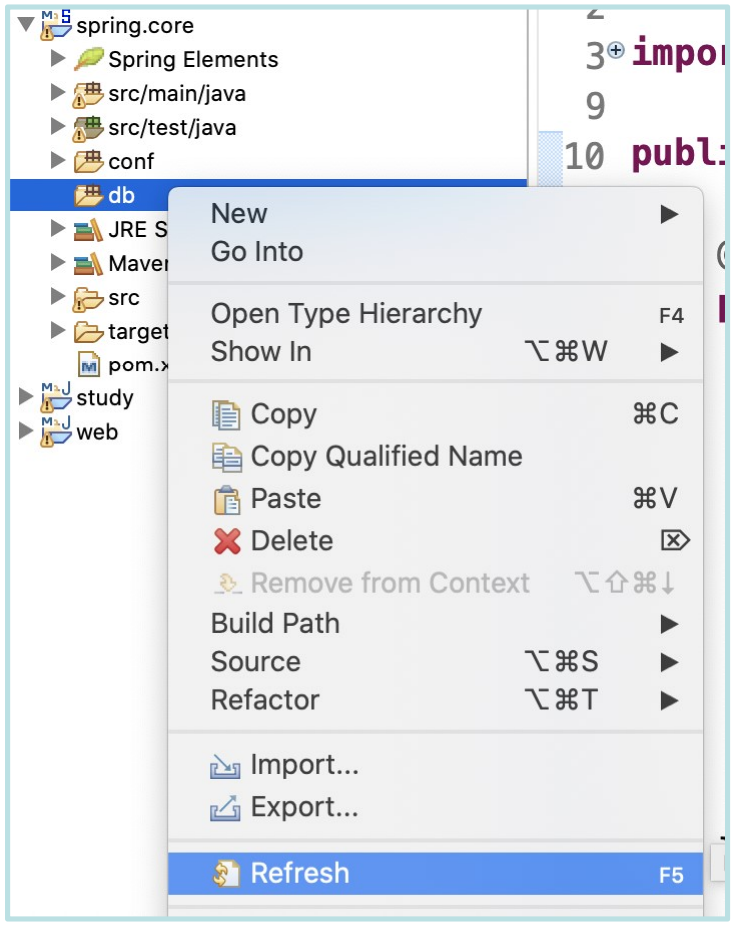
SQLite 配置測試

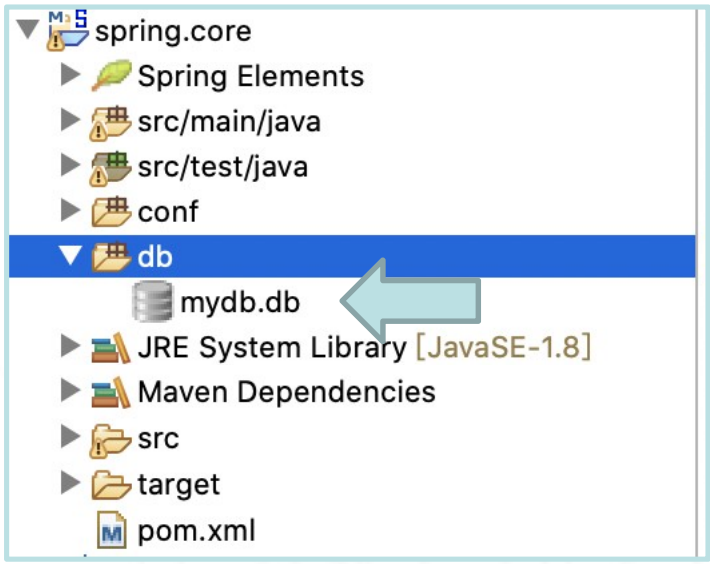
SQLiteCreateDatabaseAndConnectionTest.java

```
public class SQLiteCreateDatabaseAndConnectionTest {

    @Test
    public void test() {
        ApplicationContext ctx = new ClassPathXmlApplicationContext("jdbc-config.xml");
        ComboPooledDataSource cp = ctx.getBean("dataSource", ComboPooledDataSource.class);
        try {
            System.out.println(cp.getConnection()); // 資料庫連線
        } catch (SQLException e) {
            System.out.println("資料庫建立連線失敗");
            e.printStackTrace();
            return;
        }
        System.out.println("資料庫建立連線成功");
    }
}
```

SQLite 配置測試


spring.core
Spring Elements
src/main/java
src/test/java
conf
db
JRE S
Maven
src
target
pom.x
study
web
New
Go Into
Open Type Hierarchy F4
Show In ⌥⌘W
Copy ⌘C
Copy Qualified Name
Paste ⌘V
Delete
Remove from Context ⌥⇧⌘↓
Build Path
Source ⌥⌘S
Refactor ⌥⌘T
Import...
Export...
Refresh F5


spring.core
Spring Elements
src/main/java
src/test/java
conf
db
mydb.db
JRE System Library [JavaSE-1.8]
Maven Dependencies
src
target
pom.xml

Lab for SQLite

```
-- 建立 mydb 資料庫
-- 執行 SQLiteCreateDatabaseAndConnectionTest.java

-- 建立 Emp 資料表
create table emp (
    eid integer PRIMARY KEY, -- 主鍵 (自行產生序號: 1, 2, 3, ...)
    ename text, -- 員工姓名
    age integer, -- 員工年齡
    createtime datetime default current_timestamp -- 建檔時間
);

-- 建立 Emp 範例資料
insert into emp(ename, age) values('john', 28);
insert into emp(ename, age) values('mary', 30);
insert into emp(ename, age) values('bobo', 29);

-- 查詢 Emp 資料
select eid, ename, age, createtime from emp;
```

Lab for MySQL

```
-- 建立 mydb 資料庫
create database mydb character set utf8mb4 collate utf8mb4_general_ci;

-- 建立 Emp 資料表
create table emp(
          eid int not null auto_increment, -- 主鍵 (自行產生序號: 1, 2, 3, ...)
          ename varchar(50) not null unique, -- 員工姓名
          age int, -- 員工年齡
          createtime timestamp default current_timestamp, -- 建檔時間
          primary key(eid)
);

-- 建立 Emp 範例資料
insert into emp(ename, age) values('john', 28);
insert into emp(ename, age) values('mary', 30);
insert into emp(ename, age) values('bobo', 29);

-- 查詢 Emp 資料
select eid, ename, age, createtime from emp;
```

JdbcTemplate CRUD 演練

- CreateEmp
- ReadEmp
- UpdateEmp
- DeleteEmp

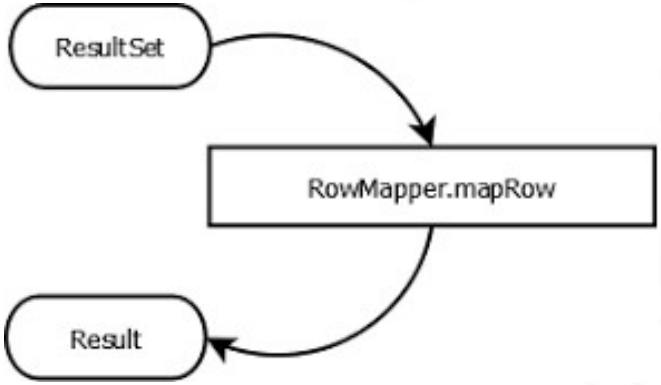
建立 Entity/Bean

```
public class Emp {
    private Integer eid;
    private String ename;
    private Integer age;
    private Date createtime;

    // getter / setter
}
```

RowMapper

- ORM Mapper
  - Spring 提供 RowMapper 可將查詢結果封裝於Domain物件中。




```
List<Customer> list = jdbcTemplate.query(sql, (ResultSet rs, int i) -> {
            Customer customer = new Customer();
            // block of code
            return customer;
});
```

SimpleFlatMapper

- 簡單數據庫映射
  - https://simpleflatmapper.org/

```
<dependency>
    <groupId>org.simpleflatmapper</groupId>
    <artifactId>sfm-springjdbc</artifactId>
    <version>3.11.7</version>
</dependency>
```