8.3. Query with SQL File

如果今天 SQL 是以檔案的形式記錄，我們就必須從指令的路徑讀取相對 SQL 檔案再進行查詢。所以接下來會新增一個專門用來讀取 SQL 檔案的 Class `SqlUtils.java` 。

```
|--src/main/java
   |--com.example.demospringboot
      |--DemospringbootApplication.java
   |--com.example.demospringboot.configuration
      |--SwaggerConfig.java
      |--RestConfiguration.java
   |--com.example.demospringboot.controller
      |--TestController.java
      |--ProductController.java
      |--EmpController.java
      |--EmpBySqlFileController.java // 新增的檔案
   |--com.example.demospringboot.entity
      |--Car.java
      |--CarPK.java
   |--com.example.demospringboot.model
      |--Product.java
   |--com.example.demospringboot.repository
      |--CarRepository.java
   |--com.example.demospringboot.service
      |--CarService.java
      |--ProductService.java
   |--com.example.demospringboot.service.impl
      |--CarServiceImpl.java
      |--ProductServiceImpl.java
   |--com.example.demospringboot.service.sql
      |--SqlAction2.java // 匯入的檔案
      |--SqlAction.java // 匯入的檔案
      |--SqlUtils.java // 匯入的檔案
      |--GetSqlStrSrvUtil.java // 匯入的檔案
|--src/main/resources
   |--sql
      |--FIND_BY_PK_001.sql // 新增的檔案
```

由於本章匯入的檔案有引用到第三方套件，請在 `pom.xml` 加入 `commons-lang3`

```
<dependency>
    <groupId>org.apache.commons</groupId>
    <artifactId>commons-lang3</artifactId>
    <version>3.10</version>
</dependency>
```

引入完套件，先在 `src/main/resources` 這個路徑下新增一個 `sql` 資料夾 (folder)，再建立一個新檔案 (file) `FIND_BY_PK_001.sql` ，將 SQL 寫在這個檔案內。動態傳入的參數一樣使用 `:` (冒號) 加上參數名稱表示。如果有動態參數，需用中括號將該行括起。

```
select EMP_NAME
 from STUDENT.TB_EMP
 where EMP_ID = :EMP_ID
[ or EMP_ID like :LIKE_EMP_ID]
[ or EMP_NAME = :EMP_NAME]
```

由於是從指定資料夾讀入 SQL ，匯入三個 Class `SqlAction2.java` 、 `SqlUtils.java` 及 `GetSqlStrSrvUtil.java` ，專門用來處理 SQL 檔案的讀取以及設定查詢欄位等功能。

介紹三個檔案用途：

1. GetSqlStrSrvUtil.java
   這支檔案是使用於 `SqlUtils.java` 僅簡單介紹，主要用於讀取 resources 目錄的檔案 ，其中注意 ClassPathResource ， 這是 Spring Boot 提供給我們讀檔 Class

   ```
   ClassPathResource resource = new ClassPathResource("sql/" + sqlName);
   ```

2. SqlUtils.java
   在讀取完 SQL 語法後，由 `SqlUtils.java` 組成動態查詢 SQL，主要介紹兩個方法，我們可以發現兩個方法都有兩個相同的參數： `String sqlName` 、 `Map<String, Object> dataMap` ，前者是 SQL 檔的檔名，該參數傳入後，會利用第一個介紹的 `GetSqlStrSrvUtil.java` 抓取檔案內容，後者是指我們需要動態加入的參數，如有在 `Map<String, Object> dataMap)` 內找尋到與 SQL 語法對應的 Key ，會將中括號解開，詳細使用方式可以看最下面的例子。
   - 取得並組成動態查詢 SQL

   ```
   public String getDynamicQuerySQL(String sqlName, Map<String, Object> dataMap) throws IOExceptior
   ```

   - 取得並組成動態查詢 SQL + order by

   ```
   public String getDynamicQuerySQL(String sqlName, Map<String, Object> dataMap, String[] orderBy)
   ```

3. SqlAction2.java
   主要是查詢方法，所有跟 SQL 檔案的操作都會透過這支程式來做。要取得相對應的 SQL 檔只需要傳入相對應的檔案名稱就可以了，其他的解析交給 `SqlUtils.java` 。
   - 查詢

   ```
   public List<Map<String, Object>> queryForList(String sql, Map<String, ?> parameters) {}
   ```

   - 查詢並轉換成指定物件，注意如果 SQL 語法包含 join ，需要再自建物件，才能對應欄位

   ```
   public <T> List<T> queryForList(String sql, Map<String, ?> parameters, Class<T> clazz) {}
   ```

介紹完畢後，我們開始調整程式碼。

在 `EmpController.java` 中增加程式碼：

```
@RestController
public class EmpController {

    @Autowired
    private SqlAction2 sqlAction2;

    @Autowired
    private SqlUtils sqlUtils;

    @PostMapping(path = "/empNameBySql")
    public List<Map<String, Object>> queryEmpByDynamicSql() throws IOException {

        // 放進動態參數至 Map
        Map<String, Object> map = new HashMap<>();
        StringBuilder sb = new StringBuilder();
        map.put("EMP_ID", "00001");
        map.put("LIKE_EMP_ID", sb.append('%').append("2").append('%').toString());

        // 組成動態 SQL
        String sql = sqlUtils.getDynamicQuerySQL("FIND_BY_PK_001.sql", map);

        // 查詢動態 SQL
        return sqlAction2.queryForList(sql, map);
    }
}
```

方法內有一個 `Map<String, Object> map = new HashMap<>();` ，主要是用來裝動態參數的內容，回顧一下 SQL 語法，裡面有三個動態參數 `:EMP_ID` 、 `:LIKE_EMP_ID` 、 `:EMP_NAME` ，除了 `:EMP_ID` 外，其他都是視情況設定的條件，在程式碼內，我們放進兩組 key : value，預期除了 `:EMP_ID` 外， `:LIKE_EMP_ID` 也是查詢條件。

```
select EMP_NAME
 from STUDENT.TB_EMP
 where EMP_ID = :EMP_ID
[ or EMP_ID like :LIKE_EMP_ID]
[ or EMP_NAME =  :EMP_NAME]
```

另外在 Controller 中傳入參數的部份我們直接使用 `RequestBody` 作為傳入參數主體，如果需要定義傳入物件格式，可以再另外定義 `EmpRequest` 及 `EmpResponse` DTO 物件。(下面會提到如何定義 request 及 response 物件格式。)

最後在 Postman 輸入相對應的 url ，便可以拿到我們想要的資料。

```
[
   {"EMP_NAME":"Tom"},
   {"EMP_NAME":"88854"},
   {"EMP_NAME":"88863"},
   {"EMP_NAME":"將軍"},
   {"EMP_NAME":"1223331"}
]
```


稍微觀察console訊息

```
[SQL log]- : Query DB-> select EMP_NAME from STUDENT.TB_EMP where EMP_ID = 86594 or EMP_ID like %888%
```

，可以發現參數有動態組合上去， SQL 語法中的 `:EMP_NAME` ，因為查詢 Map 內沒有放入 Key（EMP_NAME），所以不會解開中括號。

參考

https://mvnrepository.com/artifact/org.apache.commons/commons-lang3/3.0