

SpringBoot 入門

下載官方專案包

spring boot initializr

1. 建議套件:
 1. Spring Boot DevTools
 2. Spring Web
2. DB 套件
 1. Spring Data JDBC
 2. ORM 套件:
 1. Hibernate (Spring Data JPA)
 2. MyBatis Framework
 3. DB driver
3. 其他:
 1. Spring Security
 2. Gateway
 3. Lombok

Dependencies

ADD DEPENDENCIES... CTRL + B

Spring Boot DevTools DEVELOPER TOOLS

Provides fast application restarts, LiveReload, and configurations for enhanced development experience.

Lombok DEVELOPER TOOLS

Java annotation library which helps to reduce boilerplate code.

Spring Web WEB

Build web, including RESTful, applications using Spring MVC. Uses Apache Tomcat as the default embedded container.

Spring Data JPA SQL

Persist data in SQL stores with Java Persistence API using Spring Data and Hibernate.

Spring Data JDBC SQL

Persist data in SQL stores with plain JDBC using Spring Data.

H2 Database SQL

Provides a fast in-memory database that supports JDBC API and R2DBC access, with a small (2mb) footprint. Supports embedded and server modes as well as a browser based console application.

MS SQL Server Driver SQL

A JDBC and R2DBC driver that provides access to Microsoft SQL Server and Azure SQL Database from any Java application.

Pom.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 https://maven.apache.org/xsd/maven-4.0.0.xsd">
    <modelVersion>4.0.0</modelVersion>
    <parent>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-parent</artifactId>
        <version>3.2.1</version>
        <relativePath/> <!-- lookup parent from repository -->
    </parent>
    <groupId>com.example</groupId>
    <artifactId>demo2</artifactId>
    <version>0.0.1-SNAPSHOT</version>
    <packaging>war</packaging>
    <name>demo2</name>
    <description>Demo project for Spring Boot</description>
    <properties>
        <java.version>17</java.version>
    </properties>
    <dependencies>
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-data-jpa</artifactId>
        </dependency>
        <dependency>
```

Spring Boot 專案資訊

本專案的專案描述

JAVA 版本

使用套件

Spring Boot 設定檔

application.properties



```
# 基本設定
server.port=8080
server.servlet.context-path=/demo2

# H2 DB 設定
spring.datasource.url=jdbc:h2:mem:todo1ist
spring.datasource.driverClassName=org.h2.Driver
spring.datasource.username=sa
spring.datasource.password=
spring.sql.init.platform=h2
spring.sql.init.data-locations=classpath:test/data.sql
spring.jpa.database=h2

# sql_server 設定
#spring.datasource.url=jdbc:sqlserver://;databaseName=demo2;trustServerCertificate=true
#spring.datasource.driver-class-name=com.microsoft.sqlserver.jdbc.SQLServerDriver
#spring.datasource.username=sa
#spring.datasource.password=az541462
#spring.jpa.database=sql_server

# JPA 設定
spring.jpa.show-sql=true
spring.jpa.properties.hibernate.format_sql=true
#spring.jpa.database=sql_server
spring.jpa.generate-ddl=true
spring.jpa.hibernate.ddl-auto=update
```

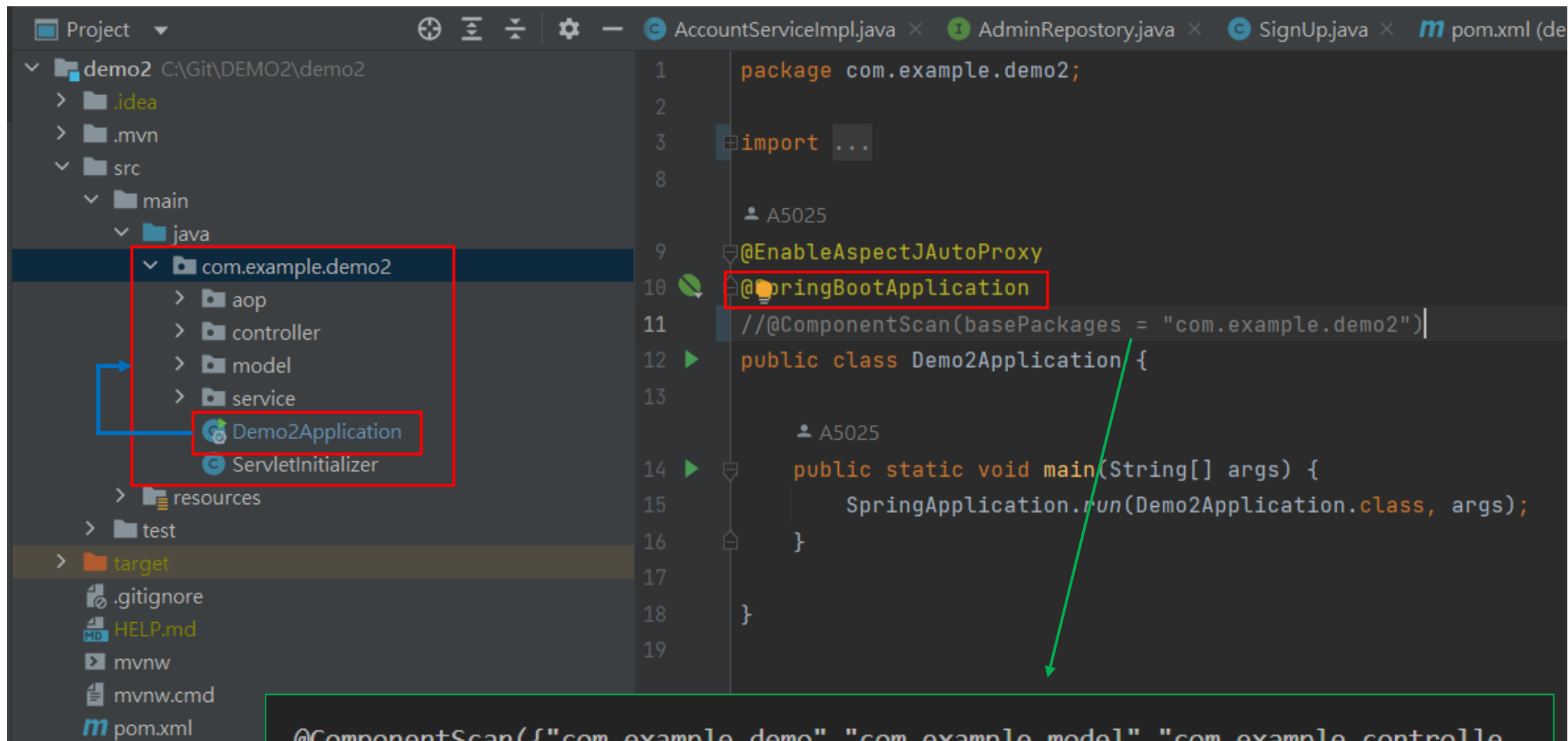
application.yml



```
1 # 基本設定
2 server:
3   servlet:
4     context-path: /demo2
5     port: '8080'
6
7 spring:
8   jpa:
9     generate-ddl: 'true'
10    database: h2
11    show-sql: 'true'
12    hibernate:
13      ddl-auto: update
14    properties:
15      hibernate:
16        format_sql: 'true'
17  sql:
18    init:
19      data-locations: classpath:test/data.sql
20      platform: h2
21  datasource:
22    driverClassName: org.h2.Driver
23    password: ''
24    username: sa
25    url: jdbc:h2:mem:todo1ist
26
27 #swagger-ui 設定
28 springdoc:
29   swagger-ui:
```

1. 會先讀 .yml 後再讀 .properties
2. .properties 會覆蓋 .yml

Spring Boot 專案結構(建議的)



The screenshot displays an IDE with a project named 'demo2' located at 'C:\Git\DEMO2\demo2'. The project structure on the left shows a 'src/main/java' directory containing a 'com.example.demo2' package. This package has sub-packages 'aop', 'controller', 'model', and 'service', and two classes: 'Demo2Application' and 'ServletInitializer'. A red box highlights the 'com.example.demo2' package and its contents, with a blue arrow pointing to the 'Demo2Application' class. The main editor shows the code for 'Demo2Application.java', which includes the package declaration 'package com.example.demo2;', imports for '@EnableAspectJAutoProxy' and '@SpringBootApplication', and a comment for '@ComponentScan(basePackages = "com.example.demo2")'. The class is 'public class Demo2Application' with a 'main' method that calls 'SpringApplication.run(Demo2Application.class, args);'. A green arrow points from the '@SpringBootApplication' annotation to the '@ComponentScan' comment. A green box at the bottom contains the full '@ComponentScan' annotation: '@ComponentScan({"com.example.demo", "com.example.model", "com.example.controller", "com.example.service", "com.example.repository"})' followed by '@ComponentScan [複雜版]'.

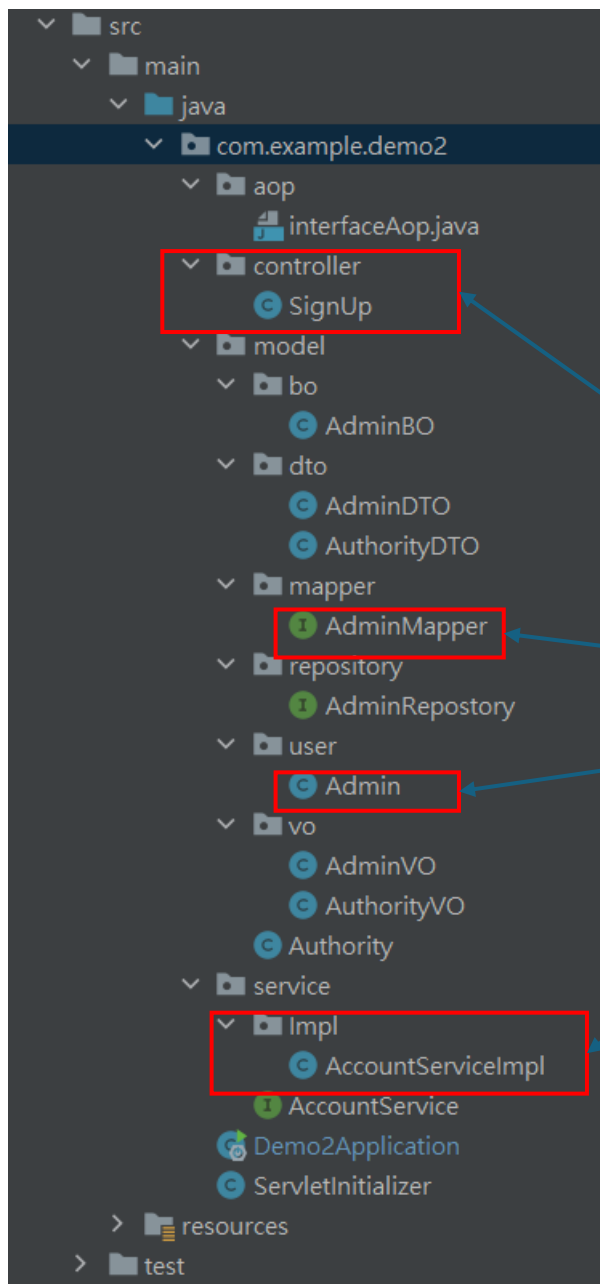
```
package com.example.demo2;

import ...

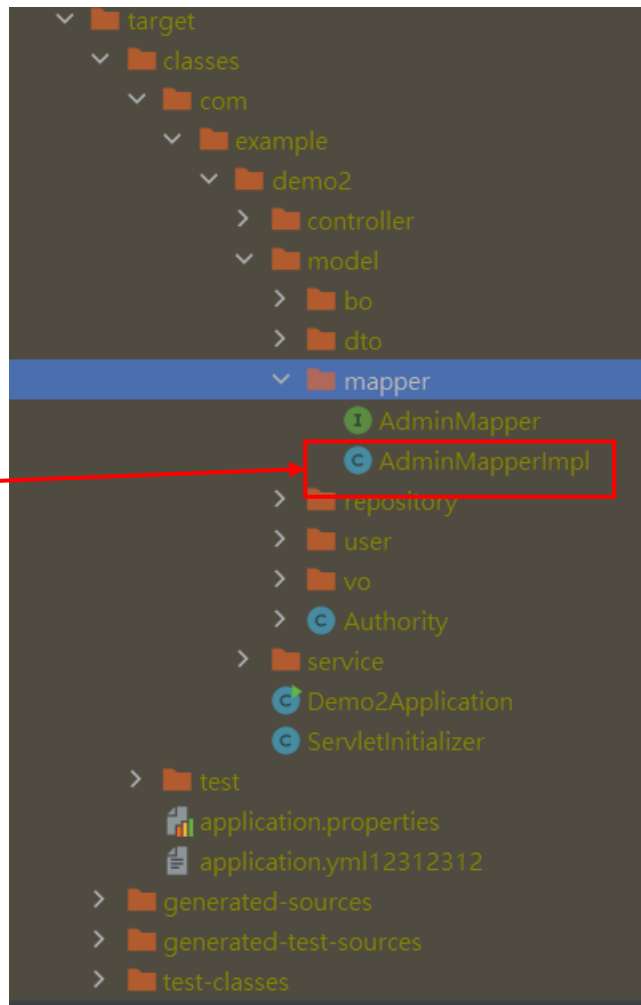
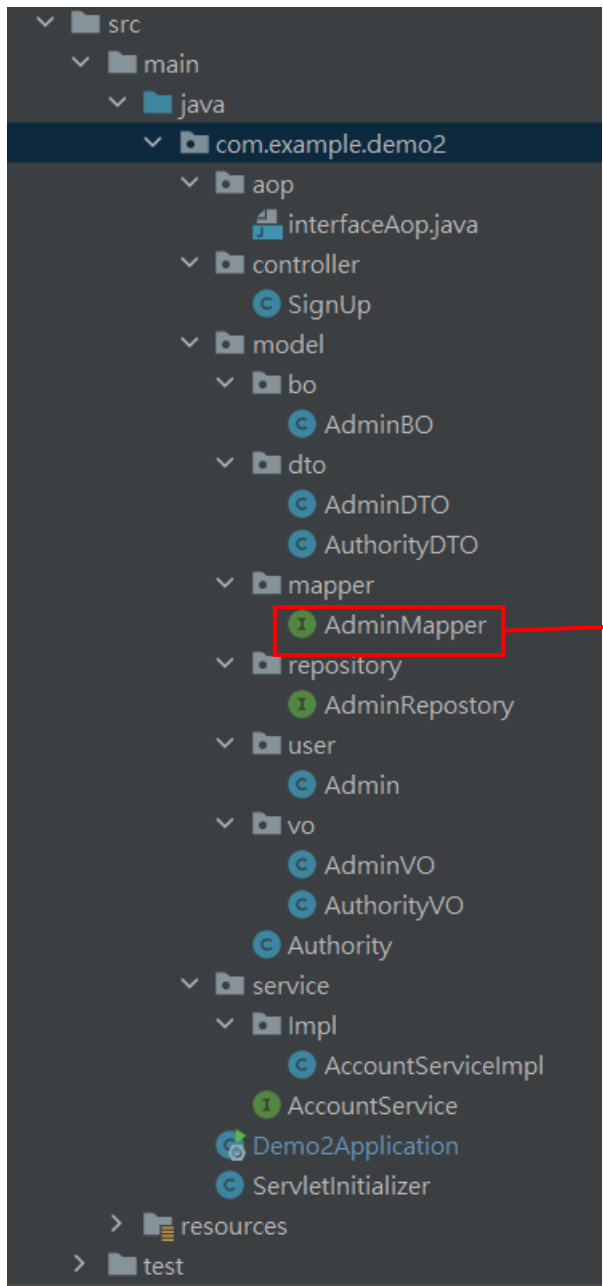
@EnableAspectJAutoProxy
@SpringBootApplication
//@ComponentScan(basePackages = "com.example.demo2")
public class Demo2Application {

    public static void main(String[] args) {
        SpringApplication.run(Demo2Application.class, args);
    }
}
```

@ComponentScan({"com.example.demo", "com.example.model", "com.example.controller", "com.example.service", "com.example.repository"}) @ComponentScan [複雜版]

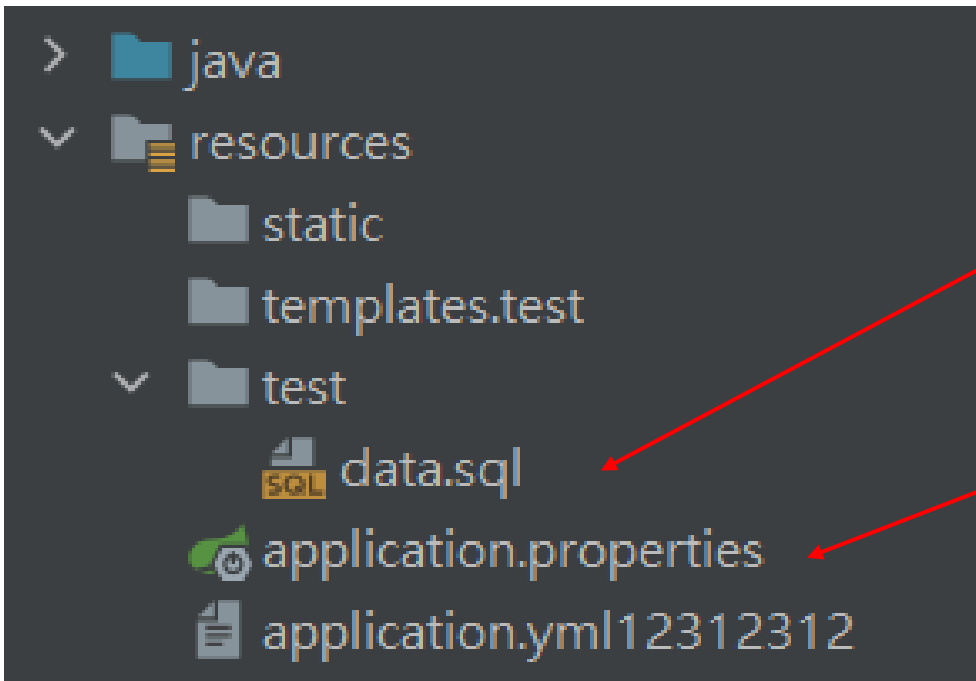


一定要扫描的元素



一定要掃描的元件
但要 Build 後才會產生

靜態文件位置



DB 初始資料

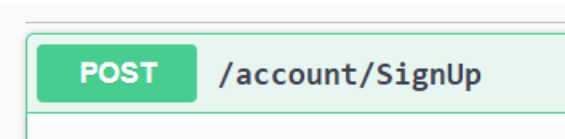
設定檔或其他靜態文件

1. 會被一同打包入專案內(jar 、 war)
2. 可以 ResourceUtils 取出
3. 不可直接運行(.bat)

```
Properties properties = new Properties();
try {
    File file = ResourceUtils.getFile("classpath:application.properties");
    InputStream in = new FileInputStream(file);
    properties.load(in);
} catch (IOException e) {
    LOGGER.error(e.getMessage());
}
```


Controller

RESTful API 路徑



```
@RestController
@RequestMapping("account")
public class SignUp {

    @Autowired
    AccountService accountService;

    @Autowired
    AdminMapper adminMapper;

    // A5025 *
    @PostMapping(value = "SignUp")
    public AdminVO signAccount(
        // @Validated
        @RequestBody AdminDTO adminDTO) {
        Admin admin = adminMapper.toPO(adminDTO);
        accountService.saveData(admin);
        return adminMapper.toVO(admin);
    }
}
```

```
@Data
public class AdminDTO {
    @NotBlank(message = "姓氏不得空白")
    private String adminLastName;
    @NotBlank(message = "名字不得空白")
    private String adminFirstName;
    @Email(message = "Email 格式異常")
    private String adminEmail;
    @NotBlank
    @Pattern(regexp = "^09[0-9]{8}$", message = "手機號碼異常")
    private String adminMobilePhone;
    @NotBlank
    private String adminAccount;
    @NotBlank
    private String adminPwd;
    private AuthorityDTO authorityDTO;
}
```

檢核請求參數

作業

spring boot initializr

基礎

- 建立 1 個spring boot 專案
- 先不寫 application.properties
- 寫一個 Controller 並設定他的 URL

進階一 application.properties

- 指定 port
- 指定專案RESTful 的路徑

進階二

- 利用 SpringDoc 建立測試用的 API 文件(Swagger-ui) [有雷注意]

進階三

- 建立 Service 並用 @Autowired 的方式讓 Controller 找到 Service