

主題 5：認識程式設計的參數化思維

國家教育研究院
National Academy for Educational Research

雙語詞彙、學術名詞暨辭書資訊網

詞彙查詢 | 下載專區 | 詞彙建議 | 審譯會

釋義

翻譯

◀ 回簡目列表

✎ 修訂/勘誤建議

✎ 單筆輸出

參數化
parameterization

資訊與通信術語辭典

QR Code

名詞解釋:

使軟體通用化的一種技術。將各種可能影響軟體的因素（如作業方式、利率和稅率等環境變數）以參數表示，待軟體建置於一給定的環境時，再依該環境的實際需求填選參數，即可成為適合該環境的軟體。

建置專案的預設參數

即便是同一個專案，有時候我們也會希望可以傳遞一些不同的變數來當做此次的建置參數
這時就可修改 Spring Boot 原生的參數化設定值，來幫我們增加彈性

專案啟動時，指定套用哪一種配置文件

`application-local.yml` 代表在自己的電腦內模擬 IT 的測試環境，然而此時搭配【H2 記憶體型資料庫】

`application-dev.yml` 代表專案被啟動在 IT 的測試環境，或者在自己的電腦內模擬 IT 的測試環境，然而此時搭配公司資料庫

`application-uat.yml` 代表專案被啟動在 USER 的測試環境，或者在自己的電腦內模擬 USER 的測試環境，然而此時搭配公司資料庫

`application-prod.yml` 代表專案被啟動在 USER 的正式環境，然而此時嚴禁在自己的電腦內模擬 USER 的正式環境，然而此時搭配公司資料庫

```

1 # 方法 1
2 spring.profiles.active: uat
3
4 # 方法 2 (建議)
5 spring:
6   profiles:
7     active: uat

```

application.yml

檔案位置 `D:\workspace-IntelliJ\mytest-backend\mytest-common\src\main\resources\config\application.yml`

- 後端 API 包版資訊

Project Structure:

- mytest-backend
 - .idea
 - mytest-api
 - mytest-batch
 - mytest-common
 - src
 - main
 - java
 - resources
 - config
 - application.yml
 - application-local.yml
 - jasperreports
 - data-intra.sql
 - data-primary.sql
 - logback-spring.xml
 - schema-intra.sql
 - schema-primary.sql

pom.xml (mytest-backend) Content:

```

1 info: Tsai, 2022/10/7 上午 11:53 · Initial Commit
2 build:
3   artifact: '@project.artifactId@'
4   description: '@project.description@'
5   name: '@project.name@'
6   version: '@project.version@'
7 logging:
8   level:
9   org:
10    springframework:
11      web: INFO
12    zalando:
13      logbook: TRACE
14 server:
15   session-timeout: 120
16 spring:
17   servlet:
18     multipart:
19       max-file-size: 100MB
20       max-request-size: 100MB
21 profiles:
22   active: local

```

打包成 WAR 檔後，才會被 pom.xml 內容取代

- 上傳容量限制
- 指定套用哪一種配置文件
- 是否顯示資料庫 CRUD 的 SQL

```
application.yml x
16 spring:
17   servlet:
18     multipart:
19       max-file-size: 100MB 修改單一檔案最大上傳容量限制
20       max-request-size: 100MB 修改總上傳檔案，連同文字訊息，最多上傳容量限制
21   profiles:
22     active: local
23   session:
24     store-type: none
25   jpa:
26     # must set database to 'default' in order to auto select different dialect for each of the data source
27     database: default
28     hibernate:
29       naming:
30         physical-strategy: org.hibernate.boot.model.naming.PhysicalNamingStrategyStandardImpl
31     properties:
32       hibernate:
33         cache:
34           use_query_cache: false
35           use_second_level_cache: false
36     show-sql: true
```

- 本機預設用 Spring Boot 內嵌式 Tomcat 來啟動專案
所以此參數會影響 Tomcat 的預設值

- 在 SIT 或 UAT 則是用 JBoss 內 Undertow 來啟動專案
然而此參數異動並不會影響 Undertow 的預設值

- Open API (Swagger) 文件自動生成
 - 預設不啟用 Swagger 來當作線上版 API 規格說明文件（文件位置 <http://<hostname or ip>:<port>/swagger-ui.html>）

```
application.yml x
51 swagger:
52   # For security reason, it's better to explicitly enable on each non-production profile
53   enabled: false
54   forCodeGeneration: true
55   hide-error-endpoint: true
56   api-info:
57     version: latest
58   ui:
59     deep-linking: true
60     display-operation-id: false
61     default-models-expand-depth: 1
62     default-model-expand-depth: 1
63     default-model-rendering: EXAMPLE
64     display-request-duration: true
65     doc-expansion: NONE
66     filter: false
67     operations-sorter: ALPHA
68     show-extensions: false
69     tags-sorter: ALPHA
70     supported-submit-methods: ["get", "put", "post", "delete", "options", "head", "patch", "trace"]
```

- HTTP 請求與回應日誌
 - 使用第三方套件 Logbook 攔截 Request、Response，後續會在由 RequestLoggingHandler 將收集到的資訊紀錄在資料表 MYTEST_LOG_TRACE

```
application.yml x
71 logbook:
72   include:
73     - /api/**
74   filter:
75     enabled: true
76     form-request-mode: body
77     strategy: default
78   format:
79     style: json
```

application-local.yml

檔案位置 *D:\workspace-IntelliJ\mytest-backend\mytest-common\src\main\resources\config\application-local.yml*

- 在自己的電腦內模擬測試環境的 PORT

```
application-local.yml x
1 server:
2   port: 8181  僅在本機端才會發揮作用
```

- 資料庫連線資訊（預設連線 2 個資料庫）

```
application-local.yml x
3  spring:
4    datasource:
5      driver-class-name: org.h2.Driver
6      password: password
7      jdbc-url: jdbc:h2:mem:primarydb;MODE=Oracle
8      username: sa
9    platform: primary
10  intra-datasource:
11    driver-class-name: org.h2.Driver
12    password: password
13    jdbc-url: jdbc:h2:mem:intradb;MODE=MSSQLServer
14    username: sa
15    platform: intra
16  h2: 僅在 H2 資料庫才會發揮作用
17  console:
18    enabled: true 允許直接透過瀏覽器對資料庫資料進行CRUD操作
19    settings.web-allow-others: true 允許其他人透過瀏覽器進行CRUD操作
```

- JSON Web Token (JWT) 身份驗證
 - JWT 內容的驗證金鑰
 - JWT 過期時間
 - JWT 簽發者

```
application-local.yml x
19  jwt:
20    secret: "5c5a1817b6caa4d5247571f8d120c7c89a8358de54c4039541ff9d8a88b80362"
21    expiration: 86400
22    refresh-expiration: 86400
23    issuer: dtw.fbl
```

- Crowd SSO 身份驗證

```
application-local.yml ×
24  crowd:
25      hostname: ssouat.dtw.intranet
26      port: 8095
27      scheme: http
28      appname: YOUR_CROWD_APP_NAME
29      password: YOUR_CROWD_APP_PASSWORD
30      is-mock: false
31  cookie-sso-enabled: false
```

- 當spring.profiles.active 為【 local 】，啟用 Swagger 來當作線上版 API 規格說明文件（文件位置 <http://<hostname or ip>:<port>/swagger-ui.html>）

```
application-local.yml ×
34  swagger:
35      enabled: true
```

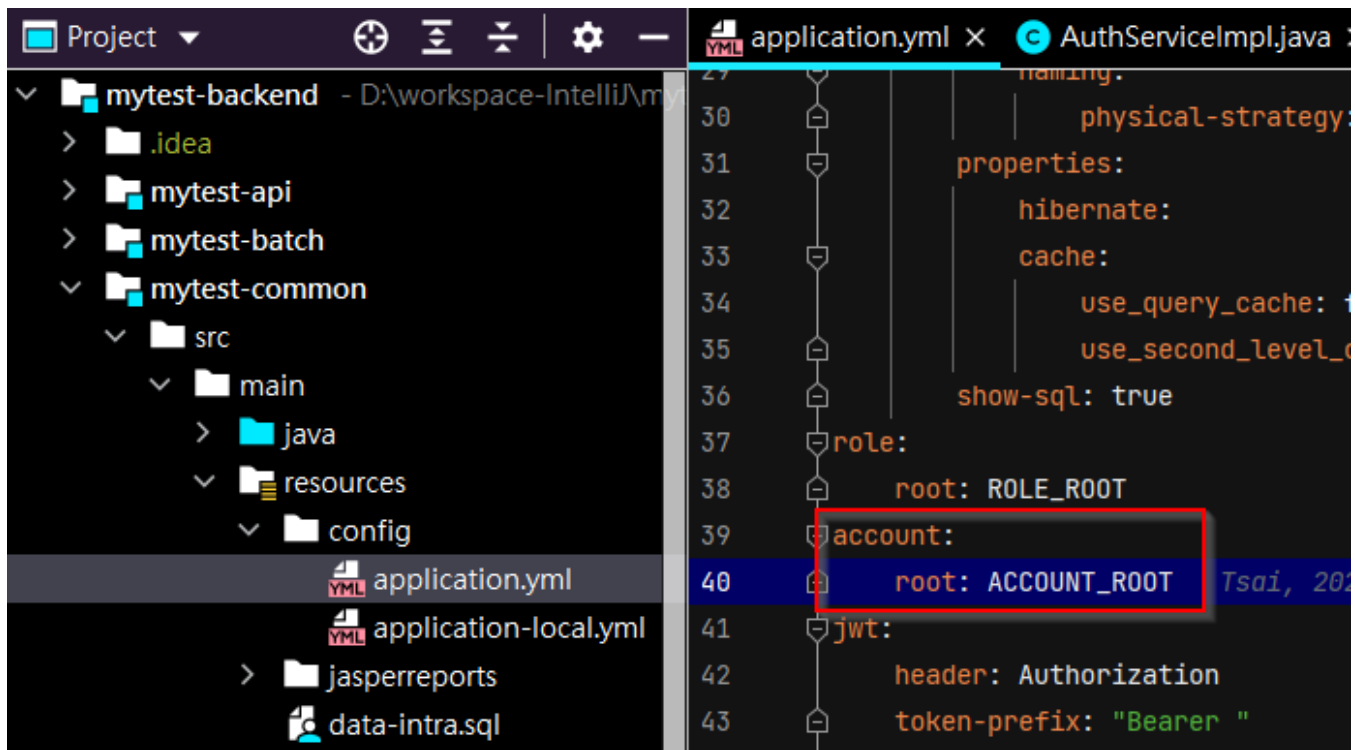
專案被啟動在 USER 的正式環境
建議改成 false

讀取來自配置文件內的參數設定值

方法一：@Value

舉例說明

- 配置文件



- 讀取並使用

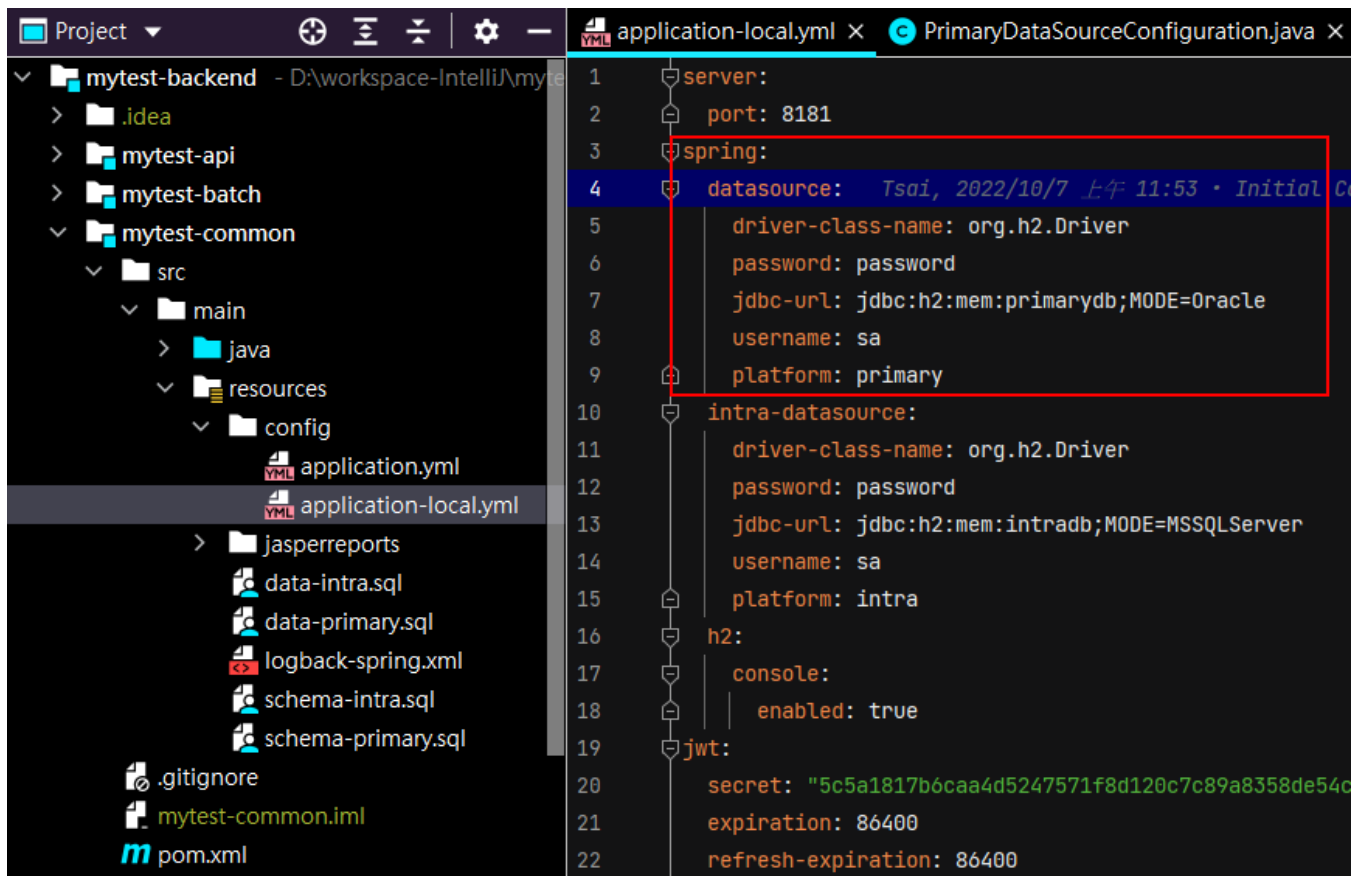
```
application.yml × AuthServiceImpl.java ×
27
28     @Autowired
29     private AccountService accountService;
30
31     @Value("${account.root}") Tsai, 2022/10/7 上午 11:53 • Initial Commit
32     private String rootAccount;
33
34     @Override
35     public Account getCurrent() {...}
36
37
38
39
40
41
42
43
44
45
46
47
48     @Override
49     public TokenPair generateJwtToken() {...}
50
51
52
53
54
55     @Override
56     public TokenPair refreshJwtToken(TokenPair oldPair) {...}
57
58
59
60
61
62
63
64     @Override
65     public TokenPair generateRootJwtToken() {
66         Account account = accountService.findById(rootAccount);
67         TokenPair retVal = jwtHelper.generateToken(account, shouldExpire: true);
68
69         return retVal;
70     }
```

方法二：@ConfigurationProperties

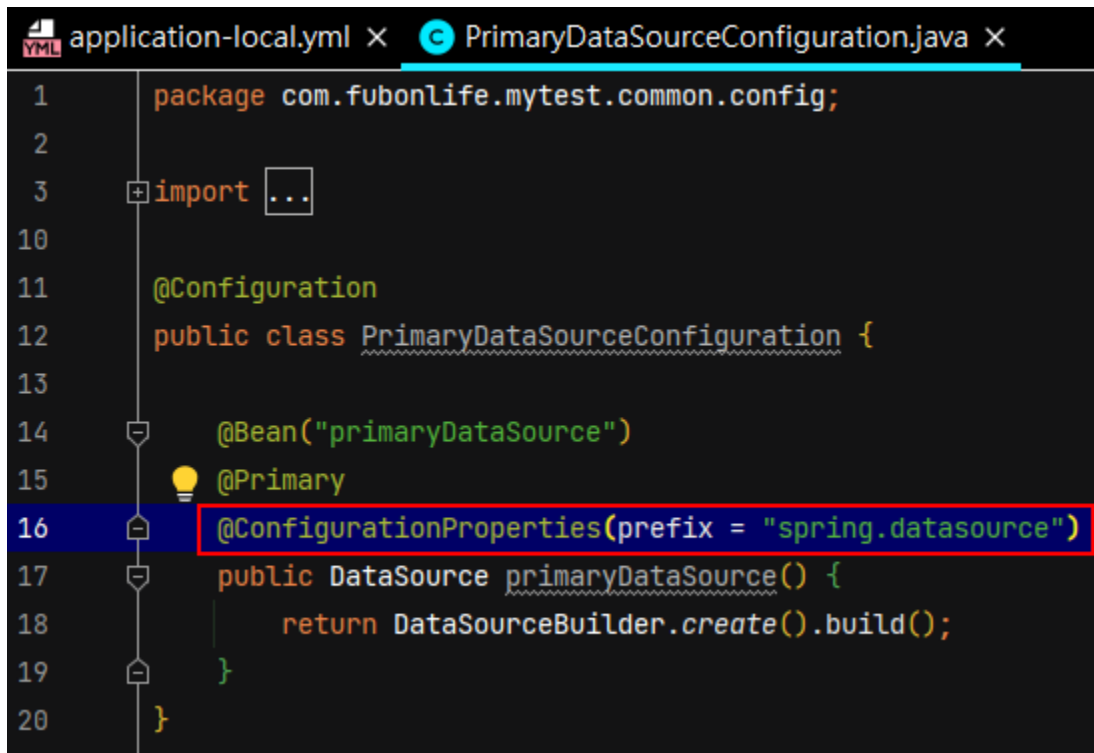
Configuration檔案建議放在子專案 xxx-common\src\main\java\com\fubonlife\mytest\common\config\000Configuration.java

舉例說明－以【主要資料庫】連線設定值為例

- 配置文件

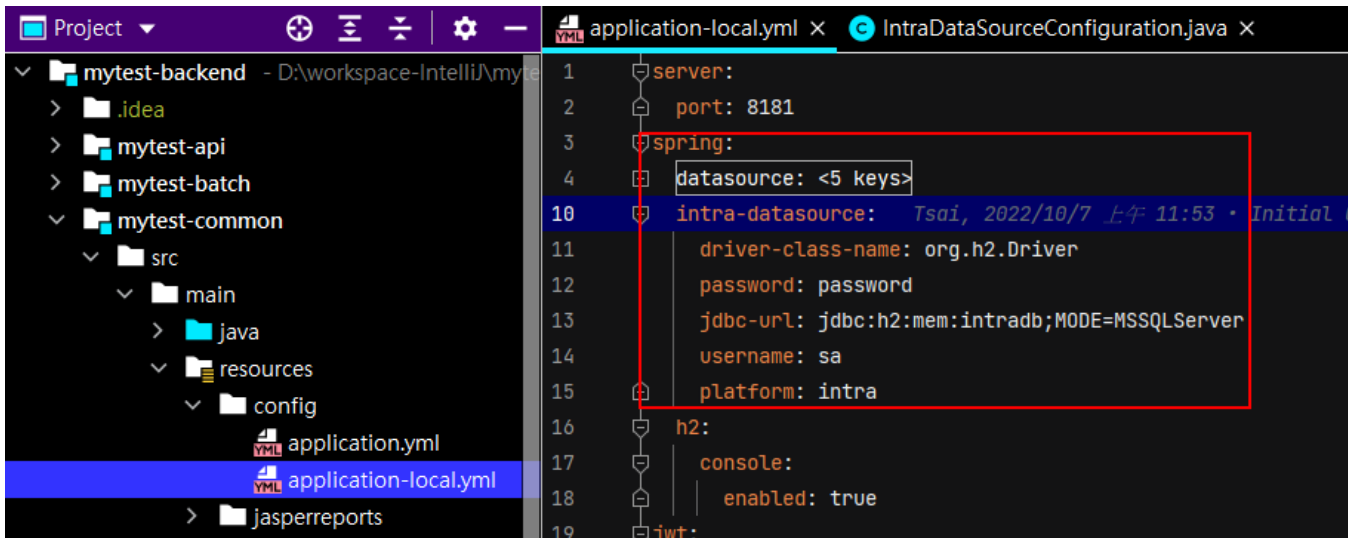


- 讀取並使用

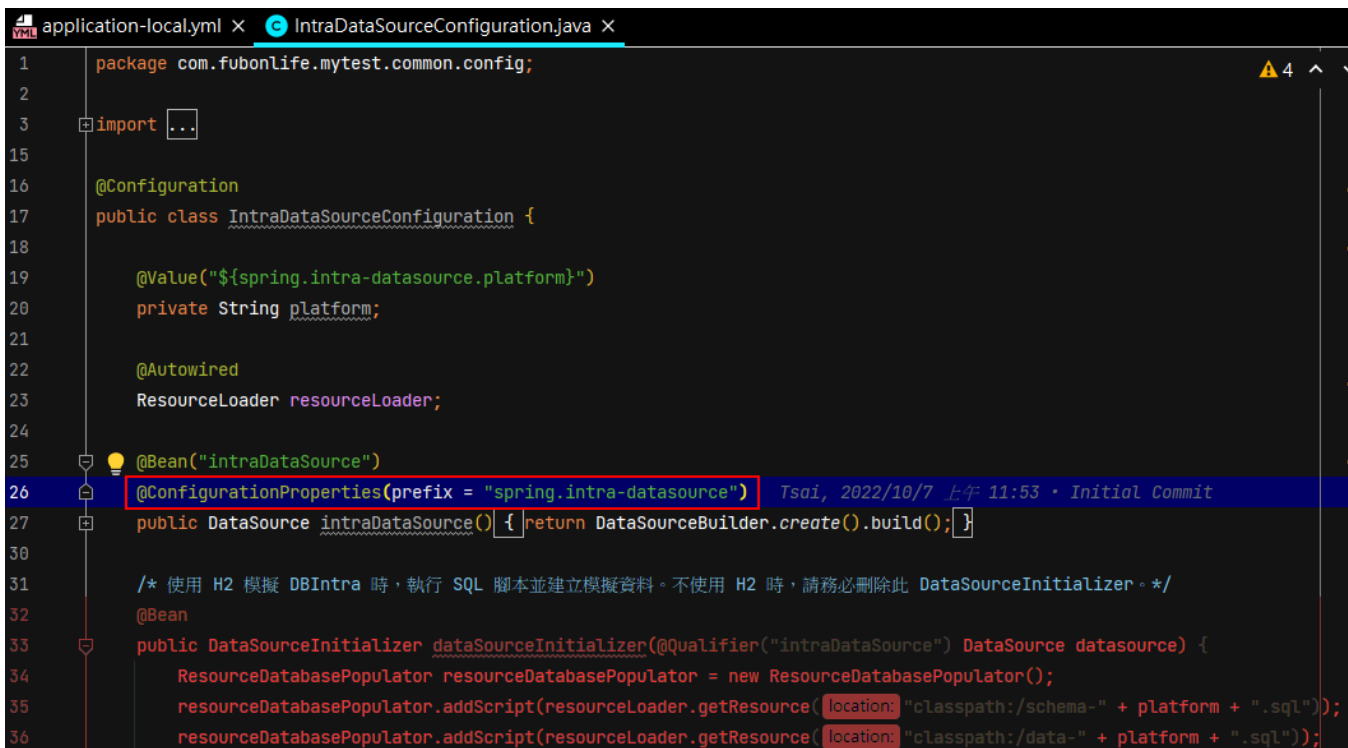


舉例說明—以【次要資料庫】連線設定值為例

- 配置文件



- 讀取並使用



使用 H2 模擬 DBIntra 時，執行 SQL 腳本並建立模擬資料。
不使用 H2 時，請務必刪除此 DataSourceInitializer !!

舉例說明－客製化一個 DataSourceConnectionConfiguration 並註冊成 Bean

- 配置文件

配置文件

```
spring.datasource.url=jdbc:mysql://127.0.0.1:7108/test?
useUnicode=false&autoReconnect=true&characterEncoding=utf-8
spring.datasource.username=sa
spring.datasource.password=b1384
spring.datasource.driver-class-name=com.microsoft.sqlserver.jdbc.SQLServerDriver
```

- 讀取

讀取

```
@Configuration
@ConfigurationProperties(prefix = "spring.datasource")
public class DatasourceConnectionConfiguration {

    private String url;
    private String username;
    private String password;
    private String driverClassName; // driver-class-name,

    public String getUrl() {
        return url;
    }

    public void setUrl(String url) {
        this.url = url;
    }

    public String getUsername() {
        return username;
    }

    public void setUsername(String username) {
        this.username = username;
    }

    public String getPassword() {
        return password;
    }

    public void setPassword(String password) {
        this.password = password;
    }

    public String getDriverClassName() {
        return driverClassName;
    }

    public void setDriverClassName(String driverClassName) {
        this.driverClassName = driverClassName;
    }
}
```

- 透過 Spring 容器取得指定的 Configuration Bean 後，就能用了

使用

```
@RestController
@RequestMapping(value = "/api/b1384")
public class B1384Controller {

    @Autowired
    DataSourceConnectionConfiguration dataSourceConnectionConfiguration;

    @PostMapping("/datasource-config/list")
    public Map<String, Object> listDataSourceConfigInB1384(){

        Map<String, Object> map = new HashMap<>();
        map.put("url", dataSourceConnectionConfiguration.getUrl());
        map.put("userName", dataSourceConnectionConfiguration.getUsername());
        map.put("password", dataSourceConnectionConfiguration.getPassword());
        map.put("className", dataSourceConnectionConfiguration.getDriverClassName());

        return map;
    }
}
```

方法三：@ConditionalOnProperty

舉例說明

- 配置文件

配置文件

```
demo.enabled=false
```

- 讀取並使用

讀取並使用

```
@Configuration
@ConditionalOnProperty(
    prefix = "demo",          //
    name = {"enabled"},       //
    havingValue = "true",     // havingValue TestConfigurationBean
    matchIfMissing = true     // TestConfigurationBean
)
public class TestConfiguration {

    // ....
}
```