SpringBoot with Thymeleaf

|  |  |
| --- | --- |
| 簡介 | SpringBoot使用Thymeleaf作為網頁呈現以及thymeleaf常用語法 |
| 作者 | 顏俊吉 |

1. 前言

Spring Boot推薦使用Thymeleaf作為模板引擎，因為Thymeleaf提供了完美的Spring MVC的支持。

Thymeleaf模版介紹：Thymeleaf 是一個java庫，它是一個XML/HTML5模板引擎，能夠應用於轉換於模板檔案，以顯示應用程式產生的資料和文本，它適合於基於XHTML/HTML5的web服務應用程式。

1. 環境

IDE： Eclipse IDE for Enterprise Java Developers. Version: 2019-12

Tool：sts-4.6.0.RELEASE

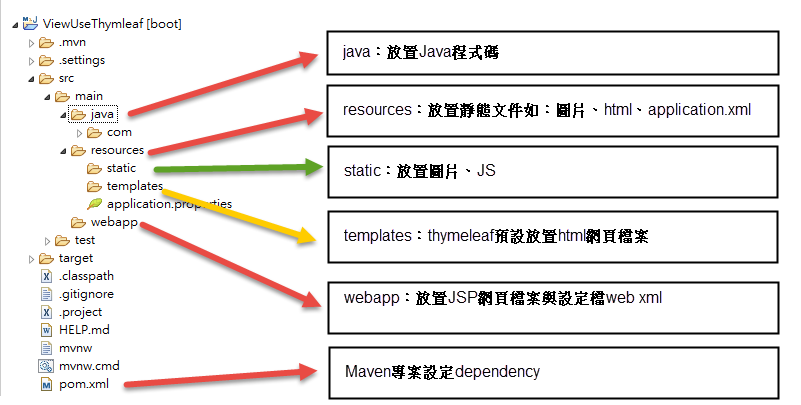
JavaJDK：1.8

Maven：4.0

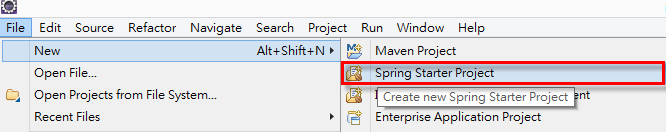
Framework：Springboot 2.2.5.RELEASE

Server：Tomcat 8.5

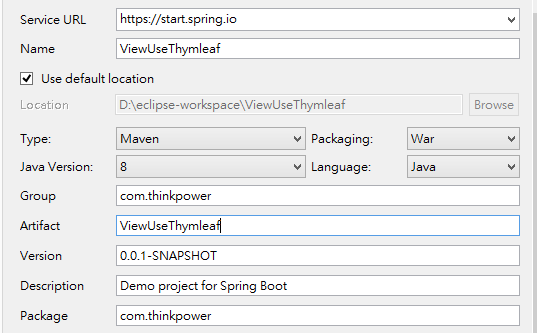
1. 建立專案
   1. 專案結構



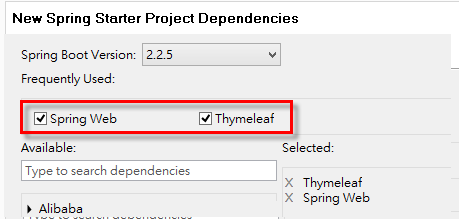
* 1. 建立Sprint Boot專案
     1. File->New->Spring Start Project。



* + 1. 設定專案資訊：Name=ViewUseThymeleaf、type=Maven、Java Version、Package=com.thinkpower。



* + 1. 設定Dependency：勾選Spring Web與Thymeleaf。



* 1. pom.xml
     1. Springboot專案建立時勾選Spring web，專案會自動加入以下的dependency。

|  |
| --- |
| <dependency>  <groupId>org.springframework.boot</groupId>  <artifactId>spring-boot-starter-web</artifactId>  </dependency>  <dependency>  <groupId>org.springframework.boot</groupId>  <artifactId>spring-boot-starter-tomcat</artifactId>  <scope>provided</scope>  </dependency>  <dependency>  <groupId>org.springframework.boot</groupId>  <artifactId>spring-boot-starter-test</artifactId>  <scope>test</scope>  <exclusions>  <exclusion>  <groupId>org.junit.vintage</groupId>  <artifactId>junit-vintage-engine</artifactId>  </exclusion>  </exclusions>  </dependency> |

* + 1. Thymeleaf：專案建立時勾選thymeleaf的dependency，專案會自動加入dependency，如果專案建立時沒有勾選，可至pom.xml中加入以下代碼：

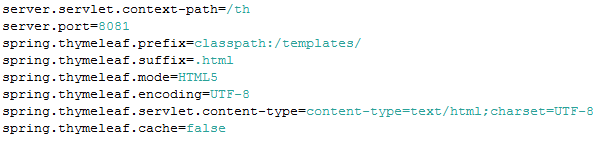
|  |
| --- |
| <dependency>  <groupId>org.springframework.boot</groupId>  <artifactId>spring-boot-starter-thymeleaf</artifactId>  </dependency> |

* 1. 設定application.properties

application.properties位於「src/main/resources/」，加入以下設定到application.properties中。

一張含有 螢幕擷取畫面 的圖片

自動產生的描述



Server setting：設定網頁連線時使用的路徑及port

|  |
| --- |
| server.servlet.context-path:/th  server.servlet.port=8081 |

Thymeleaf：設定thymeleaf相關參數

|  |
| --- |
| spring.thymeleaf.prefix=classpath:/templates/  spring.thymeleaf.suffix=.html  spring.thymeleaf.mode=HTML  spring.thymeleaf.encoding=UTF-8  spring.thymeleaf.servlet.content-type=text/html;charset=UTF-8  spring.thymeleaf.cache=false |

* 1. @SpringBootApplication

專案建立時會自動產生有使用@SpringBootApplication的Class，如未產生則手動建立Class並使用@SpringBootApplication進行註釋，在IDE中，運行此Class以啟動整個Web。

|  |
| --- |
| package com.thinkpower;  import org.springframework.boot.SpringApplication;  import org.springframework.boot.autoconfigure.SpringBootApplication;  @SpringBootApplication  public class SpringBootApplication {  public static void main(String[] args) {  SpringApplication.run(SpringBoot22Application.class, args);  }  } |

* 1. 建立Controller

於「src/main/java/con/thinkpower/controller」建立Class為thController.java，並加入@Controller，web運作時會依所輸入的網址路徑分配執行再將處理結果發送給前端畫面，程式碼如下：

thController.java

|  |
| --- |
| @Controller  **public** **class** thController {  @Value("WellCome to my Thymeleaf Example !")  **private** String welcomeMessage;  @Value("<b>WellCome to my Thymeleaf Example !<b>")  **private** String welcomeMessageWithHtml;  **private** List<String> appliancesIndex = Arrays.*asList*("C","B","A","E","D");  **private** List<String> appliancesArr = Arrays.*asList*("冷氣","冰箱","電風扇","氣炸鍋","吸塵器");  **private** List<String> serarchArr = Arrays.*asList*("A","E");  **private** List<Integer> numList = Arrays.*asList*(238,900,602,242,101);  **private** List<Integer> intList = Arrays.*asList*(1,2,3,4,5,6,7,8,9,10);  **private** Integer[] intArr= {20,21,12,81,11,92};    @GetMapping("/")  **public** String toIndex(Model model) {  System.***out***.println("to home ");  model.addAttribute("welcomeMessage", welcomeMessage);  **return** "home";  }    @GetMapping("/thymeleaf")  **public** String toExample(Model model) {  **int** numA = 12;  **int** numB = 4;  String[] stringArr = {"B","C","D","F","E","A"};  String[] stringArr2 = {"A","F"};  model.addAttribute("welcomeMessageWithHtml", welcomeMessageWithHtml);  model.addAttribute("welcomeMessage", welcomeMessage);  model.addAttribute("numA", numA);  model.addAttribute("numB", numB);  model.addAttribute("stringArr", stringArr);  model.addAttribute("stringArr2", stringArr2);  model.addAttribute("numList", numList);  model.addAttribute("intArr", intArr);  model.addAttribute("intList", intList);  model.addAttribute("appliancesIndex", appliancesIndex);  model.addAttribute("appliancesArr", appliancesArr);  model.addAttribute("serarchArr", serarchArr);    VisitInfo visitor = **new** VisitInfo("Tony Stark","159.11.133.25","2021/02/28");  model.addAttribute("visitor", visitor);    List<VisitInfo> visitorList= **new** ArrayList<VisitInfo>();  visitorList.add(**new** VisitInfo("Ken","172.18.12.250","2019/08/15"));  visitorList.add(**new** VisitInfo("Joe","192.168.12.20","2019/12/22"));  visitorList.add(**new** VisitInfo("Tin","201.108.11.143","2019/10/11"));  visitorList.add(**new** VisitInfo("Jessica","211.10.111.14","2020/03/01"));  model.addAttribute("visitorList", visitorList);    List<Person> personList =**new** ArrayList<Person>();  personList.add(**new** Person("A123456789","Jimmy",30));  personList.add(**new** Person("B123456789","Jerry",31));  personList.add(**new** Person("C123456789","Denny",20));  personList.add(**new** Person("D223456789","Dora",21));  personList.add(**new** Person("E223456789","Emily",55));  personList.add(**new** Person("F123456789","Ken",78));  personList.add(**new** Person("G223456789","Tina",22));  personList.add(**new** Person("H223456789","Lucy",25));  model.addAttribute("personifno", personList);    HashMap<String , String> appliancesMap = **new** HashMap<String,String>();  appliancesMap.put("C", "冷氣");  appliancesMap.put("B", "冰箱");  appliancesMap.put("A", "電風扇");  appliancesMap.put("E", "氣炸鍋");  appliancesMap.put("D", "吸塵器");  model.addAttribute("appliancesMap", appliancesMap);  model.addAttribute("appliancesSelected", "E");    Set<String> nameSet = **new** HashSet<String>();  nameSet.add("Tina");  nameSet.add("Jimmy");  nameSet.add("Hank");  nameSet.add(**null**);  nameSet.add("Denny");  model.addAttribute("nameSet", nameSet);    String[] searchNames = {"Tina","Denny"};  model.addAttribute("searchNames", searchNames);    model.addAttribute("company", "thinkpower");  model.addAttribute("isHidden" ,**true**);  model.addAttribute("today" ,**new** Date());    model.addAttribute("flagTrue", "TRUE");  model.addAttribute("flagFalse", **false**);    model.addAttribute("oddeven", "odd");  model.addAttribute("cssbgpink", "cssbgpink");  **return** "thymeleafExample";  }  } |

Person.java

|  |
| --- |
| **public** **class** Person {  **public** Person() {};  **public** Person(String idno , String name , **int** age) {  **this**.idno = idno;  **this**.name = name ;  **this**.age = age;  }  **private** String idno;  **private** String name;  **private** **int** age;  **public** **void** setIdno(String idno) { **this**.idno = idno; }  **public** String getIdno() {**return** idno; }  **public** **void** setName(String name) { **this**.name = name; }  **public** String getName() {**return** name; }  **public** **void** setAge(**int** age) {**this**.age = age; }  **public** **int** getAge() { **return** age; }  } |

VisitInfo.java

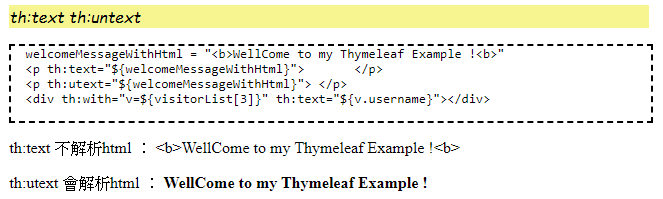
|  |
| --- |
| **public** **class** VisitInfo {  **public** String username;  **public** String visitip;  **public** String visittime;  **public** VisitInfo(String username , String visitip , String visittime) {  **this**.username = username;  **this**.visitip =visitip;  **this**.visittime =visittime;  }  **public** String getUserName() {**return** username;}  **public** String getVisitIP() {**return** visitip;}  **public** String getVisitTime() {**return** visittime;}  } |

* 1. 建立html網頁，使用thymeleaf模版功能及檔案的放置
     1. 使用thymeleaf模版功能，於頁面中的html上需要宣告thymeleaf 的名稱空間，具體程式碼如下：

|  |
| --- |
| <html xmlns:th="http://www.thymeleaf.org"> |

* + 1. Thymeleaf頁面需放到預設路徑「src/main/resources/templates/ 」下或是application.properties有另外設定路徑「spring.thymeleaf.prefix=…」。
    2. 頁面所使用參照的檔案如.jpg、.png、.js等檔案應放置在「src/main/resources/static/」路徑的資料夾中。

1. Themyleaf模版功能實作
   1. Thymeleaf表示式
      1. 變數表示式：thymeleaf頁面使用 ${...}和\*{…}顯示變數內容，但是\*{...}一般和th:object進行一起使用來完成物件屬性的簡寫。
      2. 連結網址表示式：通過使用@{…}來拼接訪問路徑，通常與th:href、th:src、th:action一起使用。
      3. 片段表示式：~{…}模版Layout一起使用，可將已編輯好的layout直接插入頁面。
   2. 常用功能實作



一張含有 螢幕擷取畫面, 鳥 的圖片

自動產生的描述

一張含有 螢幕擷取畫面 的圖片

自動產生的描述

一張含有 螢幕擷取畫面, 鳥 的圖片

自動產生的描述

一張含有 螢幕擷取畫面 的圖片

自動產生的描述

一張含有 螢幕擷取畫面, 鳥 的圖片

自動產生的描述

一張含有 螢幕擷取畫面 的圖片

自動產生的描述

一張含有 螢幕擷取畫面 的圖片

自動產生的描述

一張含有 螢幕擷取畫面 的圖片

自動產生的描述

一張含有 螢幕擷取畫面, 鳥 的圖片

自動產生的描述

一張含有 螢幕擷取畫面 的圖片

自動產生的描述

一張含有 螢幕擷取畫面 的圖片

自動產生的描述

一張含有 螢幕擷取畫面 的圖片

自動產生的描述

一張含有 螢幕擷取畫面 的圖片

自動產生的描述

一張含有 螢幕擷取畫面 的圖片

自動產生的描述

一張含有 螢幕擷取畫面 的圖片

自動產生的描述

一張含有 螢幕擷取畫面 的圖片

自動產生的描述

一張含有 螢幕擷取畫面, 鳥 的圖片

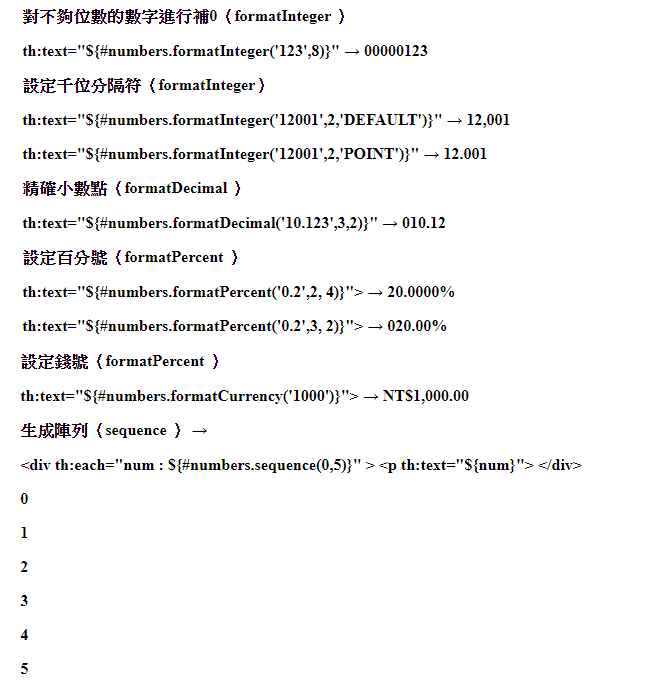
自動產生的描述

#dates：處理日期資料、生成，轉換，獲取日期的具體天數、年數。

一張含有 螢幕擷取畫面, 鳥 的圖片

自動產生的描述

#numbers：處理數字資料的轉換

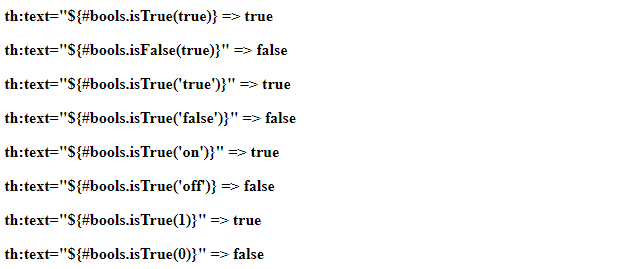


#strings：處理String的相關操作

一張含有 文字 的圖片

自動產生的描述

#bools：處理bool的相關操作



#arrays：處理arrays的相關操作

一張含有 桌, 鳥 的圖片

自動產生的描述

#lists：處理lists的相關操作

一張含有 桌 的圖片

自動產生的描述

#sets：處理sets的相關操作

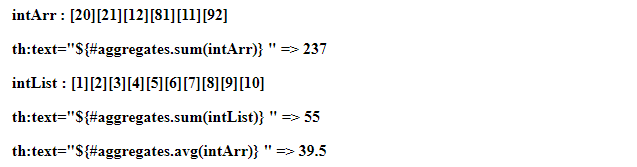
一張含有 鳥 的圖片

自動產生的描述

#maps：處理maps的相關操作



#aggregates：處理#aggregates的相關操作



Layout

SpringBoot2.0 使用模版Layout功能需要先加入 thymeleaf的 dependency

|  |
| --- |
| <dependency>  <groupId>nz.net.ultraq.thymeleaf</groupId>  <artifactId>thymeleaf-layout-dialect</artifactId>  </dependency> |

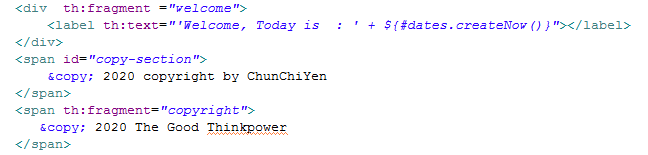
th:fragment ：

我們可以通過 th:fragment 來定義引用片段，然後可以在其他頁面進行引用

一張含有 螢幕擷取畫面, 鳥 的圖片

自動產生的描述

extra.html內容



於thymeleafExample.html使用「th:insert」、「th:include」、「th:replace」執行加入模版功能。

一張含有 螢幕擷取畫面, 鳥 的圖片

自動產生的描述

一張含有 螢幕擷取畫面 的圖片

自動產生的描述

一張含有 螢幕擷取畫面, 鳥 的圖片

自動產生的描述

此外， ~{extra::copyright}" 也可簡化為 "extra::copyright"。

1. 小結

相較於傳統開發前端技術，Thymeleaf幾乎都可以做到，且Thymeleaf不管在有網路或無網路環境下皆可執行，即可讓使用者查看靜態頁面效果，也可以讓程式設計師在伺服器查看帶數據的動態頁面效果。

有使用springBoot的話也沒試過Thymeleaf的話，不妨試試看springBoot+Thymeleaf

1. 參考網站與分享

https://www.thymeleaf.org/doc/articles/layouts.html

https://www.jishuwen.com/d/pmaz/zh-tw

專案檔案放於GitHub：<https://github.com/chunchiYen/ViewUseThymeleaf>