

chool

000111 10010101000000000000001 1010101

000111 100101010001 1

# spring *MVC* - 基礎篇

段維瀚 老師



01



# Session

- SpringMVC 部署
- @Controller
- 各種常用 @ 設定
- 參數資料封裝
- 對原生 API 的支援





# Spring MVC

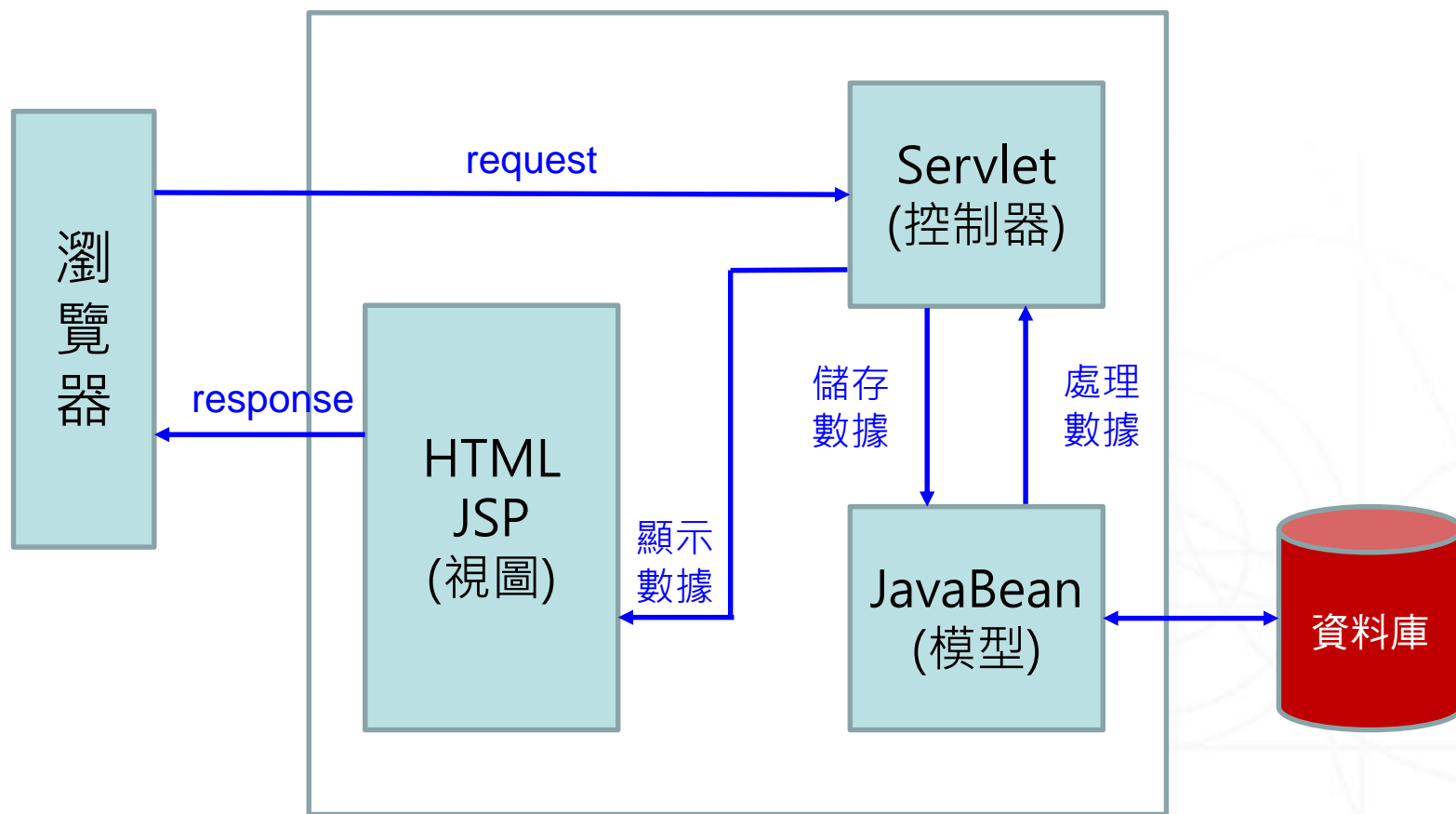


- Spring MVC

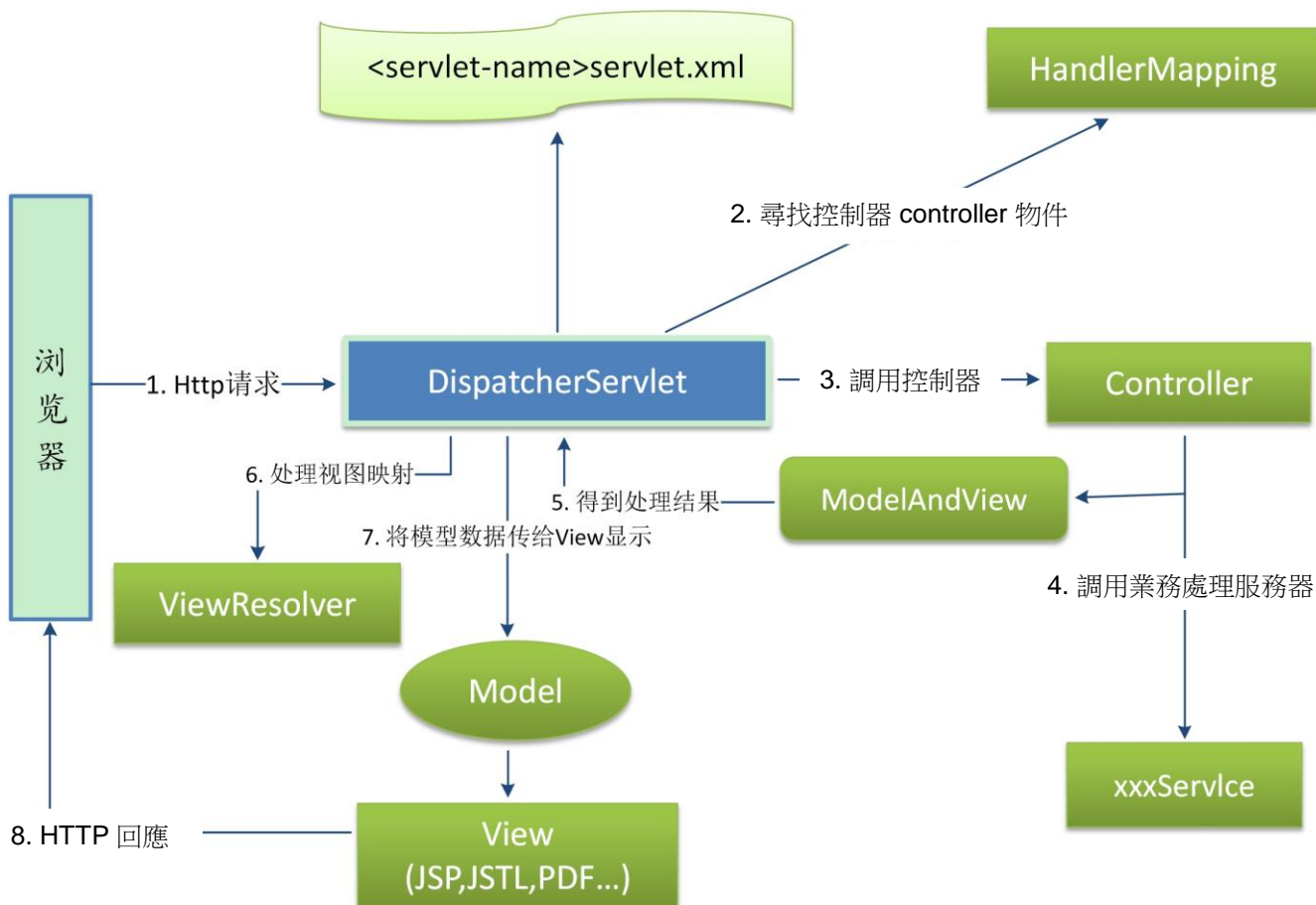
- 在Web MVC架構中，使用者並不直接連接至所需的資源，而必須先連接至前端控制器（Front controller），由前端控制器判斷使用者的請求要分派（Dispatch）給哪一個控制物件（Controller）來處理請求，藉此執行到控制使用者可請求的資源之目的。



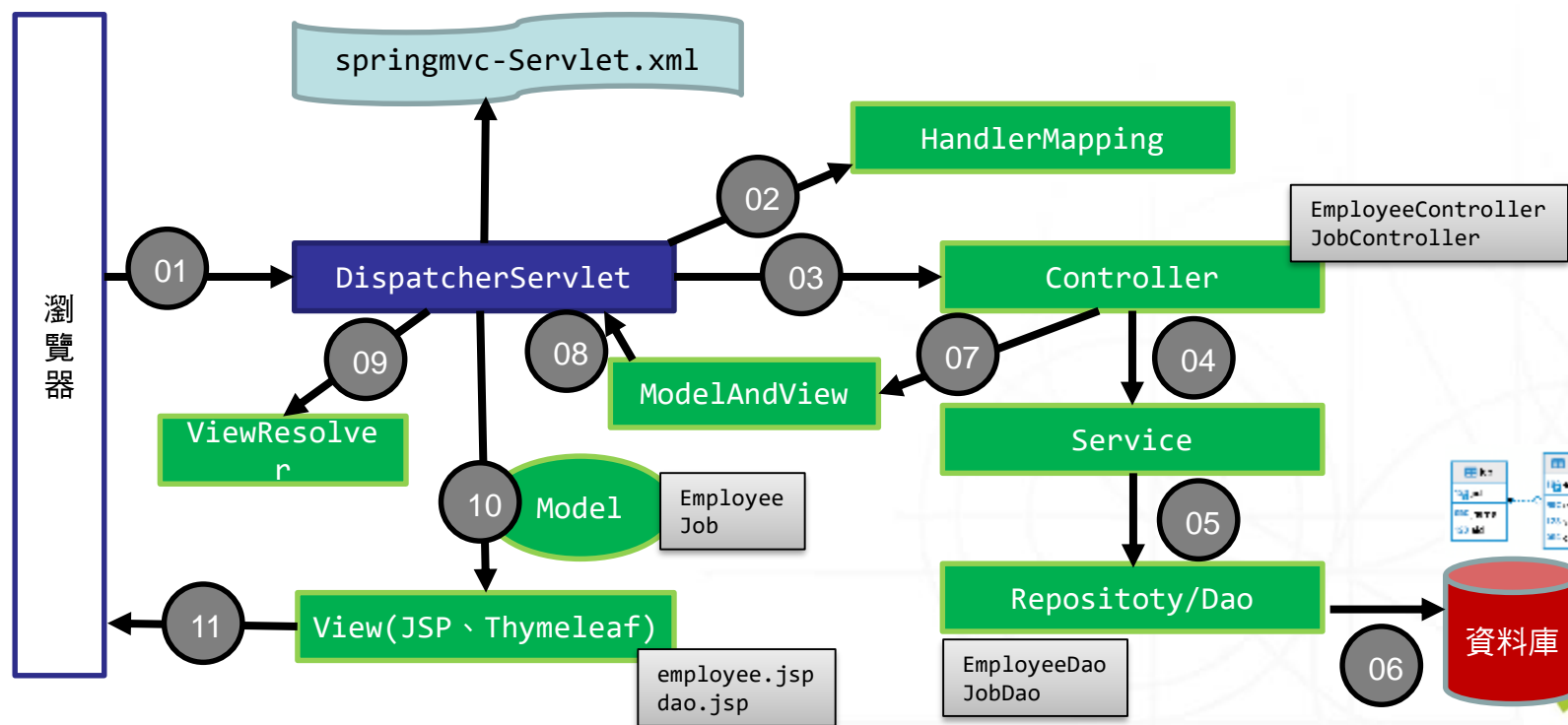
# JSP 中的 MVC 模式



# Spring MVC



# 系統流程圖





# SpringMVC 官方文档

- <https://docs.spring.io/spring/docs/current/spring-framework-reference/web.html>





# Spring pom.xml 配置

```
<properties>
  <endorsed.dir>${project.build.directory}/endorsed</endorsed.dir>
  <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
  <spring.version>4.3.29.RELEASE</spring.version>
</properties>
```

```
<build>
  <plugins>
    <plugin>
      <groupId>org.apache.maven.plugins</groupId>
      <artifactId>maven-compiler-plugin</artifactId>
      <version>3.1</version>
      <configuration>
        <source>1.8</source>
        <target>1.8</target>
        ...
      </configuration>
    ...
  ...
</build>
```





# Spring pom.xml 配置

```
<dependencies>
  <dependency>
    <groupId>javax</groupId>
    <artifactId>javaee-web-api</artifactId>
    <version>8.0.1</version>
    <scope>provided</scope>
  </dependency>
```

更多  
配置

<https://github.com/vincenttuan/SpringMVCExpert2022/blob/main/pom.xml>

```
</ dependencies >
```



# 配置檔



Web Pages



META-INF



WEB-INF



jsp



springmvc-servlet.xml



web.xml





# SpringMVC 部署

- /WEB-INF/springmvc-servlet.xml

```
<?xml version="1.0" encoding="UTF-8"?>  
<beans ...>  
    <mvc:annotation-driven />  
  
</beans>
```

<https://github.com/vincenttuan/SpringMVCExpert2022/blob/main/src/main/webapp/WEB-INF/springmvc-servlet.xml>





# SpringMVC 部署

- /WEB-INF/web.xml

```
<web-app ... >
...
<!-- 部署 DispatcherServlet -->
<servlet>
  <servlet-name>springmvc</servlet-name>
  <servlet-class>org.springframework.web.servlet.DispatcherServlet</servlet-class>
  <load-on-startup>1</load-on-startup>
</servlet>

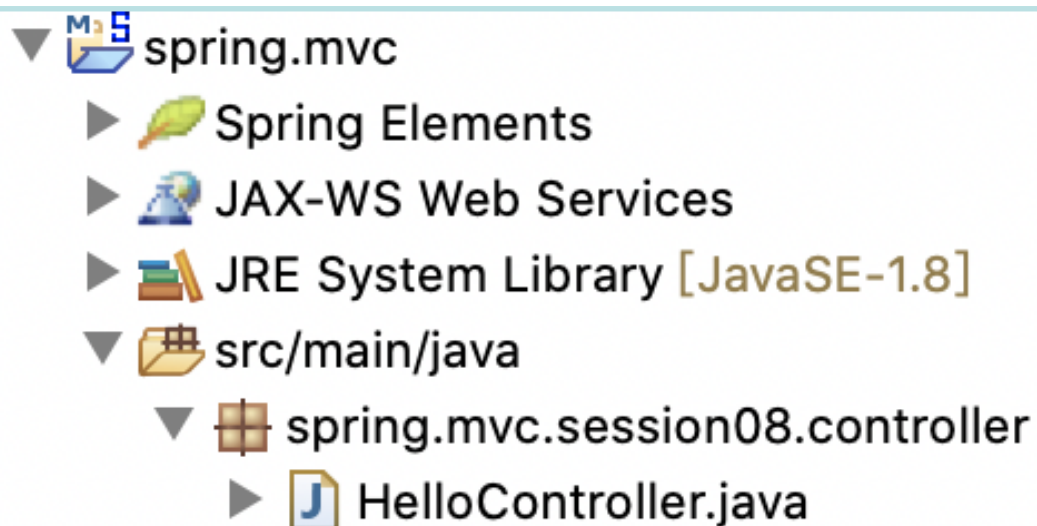
<servlet-mapping>
  <servlet-name>springmvc</servlet-name>
  <url-pattern>/mvc/*</url-pattern>
</servlet-mapping>
...
</web-app>
```

<https://github.com/vincenttuan/SpringMVCExpert2022/blob/main/src/main/webapp/WEB-INF/web.xml>

# @Controller

- 配置

- 建立 controller package



# 取得字串資料

```
@Controller
@RequestMapping(value = "/hello")
public class HelloController {
    // Base 路徑 : http://localhost:8080/spring.mvc/mvc
    /*
    * 1. 取得字串資料
    * 子路徑 : /hello/welcome
    * 完整路徑 = Base 路徑 + 子路徑
    */
    @RequestMapping(value = "/welcome")
    @ResponseBody
    public String welcome() {
        return "Welcome String MVC !";
    }
}
```



# 帶入參數

```
@RequestParam(value = "name",  
                defaultValue = "unknown",  
                required = false)
```

```
/*  
 * 2. 帶入參數 @RequestParam  
 * 子路徑：/hello/sayhi?name=Vincent&age=18  
 */  
@RequestMapping(value = "/sayhi")  
@ResponseBody  
public String sayhi(@RequestParam(value = "name") String name,  
                    @RequestParam(value = "age") Integer age) {  
    return String.format("Hi %s %d", name, age);  
}
```





# 帶入參數並計算 Lab 練習

```
/*  
* 3. 帶入參數並計算 (Lab 練習)  
* 子路徑：/hello/bmi?h=170.0&w=60.0  
*/  
// 請設計方法 api，結果會得到 bmi = 20.76
```





# 路徑參數 @PathVariable

```
/*
 * 4. 路徑參數並計算 @PathVariable
 * 子路徑：/hello/exam/75 => 結果 75 pass
 * 子路徑：/hello/exam/45 => 結果 45 fail
 */
@RequestMapping(value = "/exam/{score}")
@ResponseBody
public String examScore(@PathVariable("score") Integer score) {
    return String.format("%d %s", score, (score>=60)?"pass":"fail");
}
```



# @RequestParam + @PathVariable

```
/*
 * 5. @RequestParam + @PathVariable (Lab 練習)
 * 子路徑: /calc/add?x=30&y=20 -> Result: 50
 * 子路徑: /calc/sub?x=30&y=20 -> Result: 10
 * 子路徑: /calc/sub?y=20 -> Result: -20
 * 子路徑: /calc/add -> Result: 0
 */
// 請設計方法 api
@RequestMapping(value = "/calc/{exp}")
@ResponseBody
public String calcExp(@PathVariable("exp") String exp,
    @RequestParam(value = "x", required=false, defaultValue = "0") Integer x,
    @RequestParam(value = "y", required=false, defaultValue = "0") Integer y) {
    int result = 0;
    switch (exp) {
        case "add":
            result = x + y;
            break;
        case "sub":
            result = x - y;
            break;
        default:
            return "exp value error";
    }
    return String.format("Result: %d", result);
}
```

## \* 任意多字 ? 任意一字

```
/*  
* 6. @PathVariable (萬用字元： * 任意多字、? 任意一字)  
* 子路徑：/any/aabbcc/java8  
* 子路徑：/any/abcdefghijkl/java7  
* 子路徑：/any/a/java6  
*/  
@RequestMapping(value = "/any/*/java?")  
@ResponseBody  
public String any() {  
    return "Any";  
}
```



# 多筆參數

```
/*
 * 7. 多筆參數資料
 * 子路徑：/age?age=18&age=19&age=20
 * 並計算總和與平均
 */
@RequestMapping("/age")
@ResponseBody
public String age(@RequestParam("age") List<Integer> ageList) {
    // int 統計物件
    IntSummaryStatistics stat = ageList.stream()
                                        .mapToInt(Integer::intValue)
                                        .summaryStatistics();

    long sum = stat.getSum(); // 總和
    double avg = stat.getAverage(); // 平均
    return String.format("%s sum: %d avg: %.1f", ageList, sum, avg);
}
```



# 多筆參數 Lab 練習

```
/*  
* 8. 得到多筆 score 資料 (Lab 練習)  
* 網址輸入：/max?score=80&score=100&score=50  
* 結果得到：max score = 100  
* 網址輸入：/min?score=80&score=100&score=50  
* 結果得到：min score = 80  
*/
```





# Map 參數

```
/*
 * Map 參數
 * 子路徑：/person?name=John&score=100&age=18&pass=true
 * 子路徑：/person?name=Mary&score=90&age=20&level=2
 * 常與於 form 表單傳來的參數
 * */
@RequestMapping("/person")
@ResponseBody
public String getPerson(@RequestParam Map<String, String> person) {
    return person.toString();
}
```

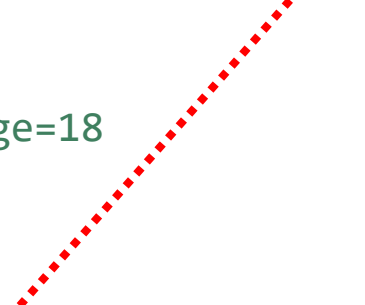




# *pojo(entity) 參數自動匹配*

```
public class User {  
    private String name;  
    private Integer age;  
    // getter、setter、toString()  
}
```

```
/*  
 * pojo(entity) 參數匹配  
 * 子路徑： /user?name=John&age=18  
 * */  
@RequestMapping("/user")  
@ResponseBody  
public String getUser(User user) { // 參數會自動匹配  
    return user.toString();  
}
```





# SpringMVC 部署

- /WEB-INF/jsp/hello.jsp

```
<%@page import="java.util.Date"%>
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
    <title>Hello SpringMVC</title>
  </head>
  <body>
    <h1>成功訪問 SpringMVC</h1>
    <h1><%=new Date() %></h1>
  </body>
</html>
```







# 傳統配置 Controller

```
public class TestController implements Controller {  
  
    @Override  
    public ModelAndView handleRequest(HttpServletRequest req,  
                                     HttpServletResponse resp) throws Exception {  
  
        ModelAndView mv = new ModelAndView("/WEB-INF/jsp/hello.jsp");  
        return mv;  
  
    }  
}
```

在 springmvc-servlet.xml 加入

```
<bean name="/test" class="com.web.ssh.HelloController" />
```



# 新版標註 @Controller 配置

- 配置

- springmvc-servlet.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<beans ...>
```

掃描

```
<context:component-scan base-package="com.web.ssh.controller"/>
```

```
<bean class="org.springframework.web.servlet.view.InternalResourceViewResolver"
      id="internalResourceViewResolver">
  <property name="prefix" value="/WEB-INF/jsp/" />
  <property name="suffix" value=".jsp" />
</bean>
```

```
</beans>
```





# @Controller

**@Controller**

**@RequestMapping("/demo")**

```
public class DemoController {  
    @RequestMapping("/hello1")  
    public ModelAndView hello1() {  
        ModelAndView mv = new ModelAndView("hello")  
        return mv;  
    }  
}
```

**@Controller**

**@RequestMapping("/demo")**

```
public class DemoController {  
    @RequestMapping("/hello2")  
    public String hello2() {  
        return "hello"; // 指向 /WEB-INF/jsp/hello.jsp  
    }  
}
```

**@Controller**

**@RequestMapping("/demo")**

```
public class DemoController {  
    @RequestMapping("/hello3")  
    @ResponseBody  
    public String hello3() {  
        return "hello"; // 直接印出字串  
    }  
}
```





# 各種常用 @ 設定

- @RequestMapping

- value

- /get

- /get/{id}

- 可透過 @PathVariable 取得 id 值

- method

- {RequestMethod.GET, RequestMethod.POST}





# 各種常用 @ 設定

- @RequestParam
  - value 、 defaultValue 、 required 、 params
- @CookieValue
  - ex: @CookieValue("JSESSIONID")
- @RequestHeader
  - ex: @RequestHeader(value="User-Agent")





# 各種常用 @ 設定

- @CookieValue
  - ex:@CookieValue("JSESSIONID")
- @RequestHeader
  - ex:@RequestHeader(value="User-Agent")





# 對原生 API 的支援

- `HttpServletRequest`
- `HttpServletResponse`
- `HttpSession`
- `Reader`、`Writer`
- `InputStream`、`OutputStream`

