Spring MVC 與 統一例外處理

段維瀚 老師

Session

- 例外處理方式一
  - **@ExceptionHandler**
- 例外處理方式二
  - 實作 **HandlerExceptionResolver** 介面
- 例外處理方式三
  - **@ControllerAdvice** 與 **@ExceptionHandler**

@ExceptionHandler

- 可標示特定例外，被標示的方法將於例外發生後執行
- 支援多組例外
  - 語法：
    - `@ExceptionHandler(SQLException.class)`
    - `@ExceptionHandler({SQLException.class, Exception.class})`

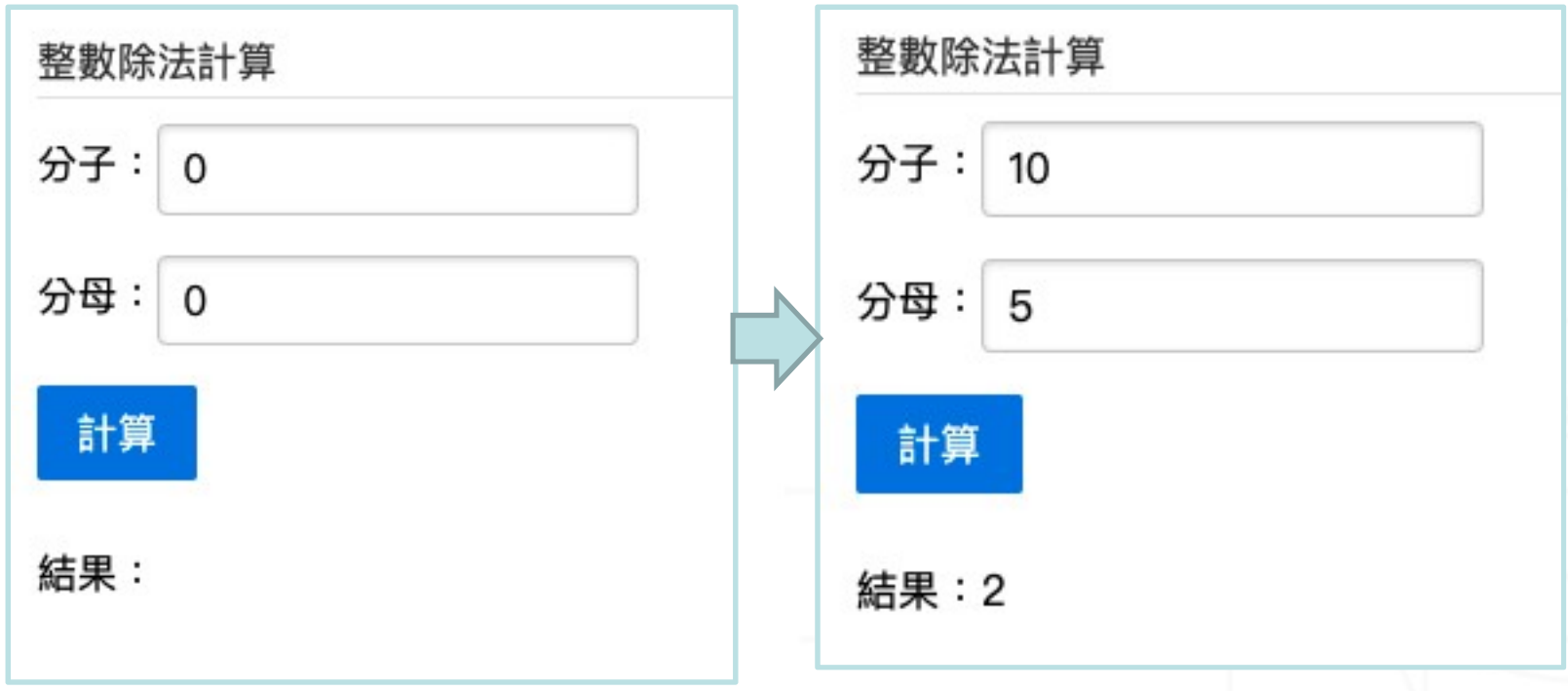
@ExceptionHandler

- 在 **controller** 層面上處理例外
  - 捕獲使用者輸入資料格式不正確的例外
    - 一個 Controller 若要配置多個 @ExceptionHandler 則例外類型不能重複，否則運行時會拋出異常。

```
// 捕獲使用者輸入資料格式不正確的例外
@ExceptionHandler({BindException.class})
public String fix(Exception ex, Model model, HttpServletRequest request){
    String referer = request.getHeader("Referer");
    model.addAttribute("referer", referer);
    model.addAttribute("ex", ex);
    return "session17/error";
}
```

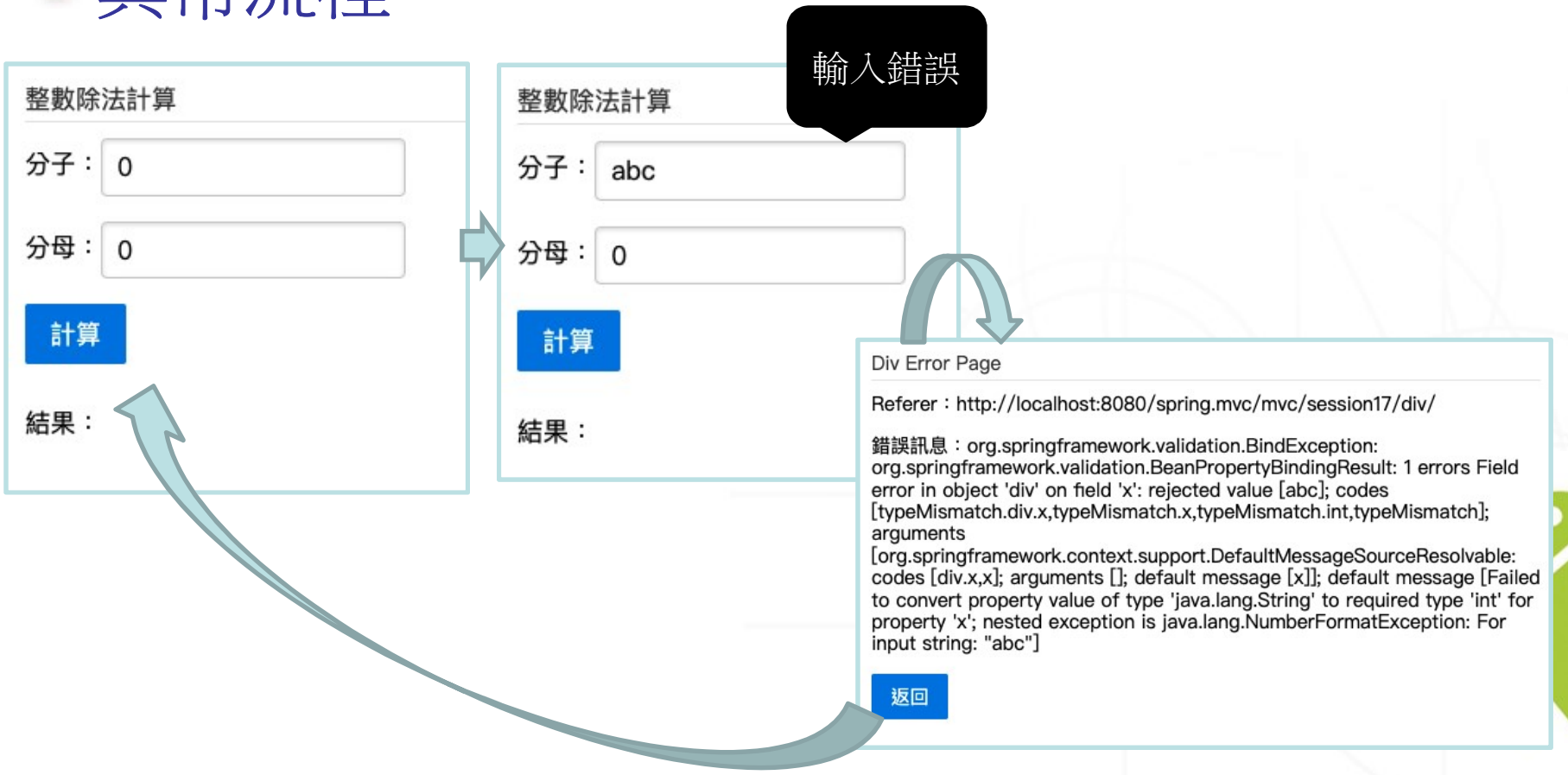
範例演示：

- 正常流程



範例演示：

- 異常流程


整數除法計算
分子：0
分母：0
計算
結果：
輸入錯誤
整數除法計算
分子：abc
分母：0
計算
結果：
Div Error Page
Referer：http://localhost:8080/spring.mvc/mvc/session17/div/
錯誤訊息：org.springframework.validation.BindException: org.springframework.validation.BeanPropertyBindingResult: 1 errors Field error in object 'div' on field 'x': rejected value [abc]; codes [typeMismatch.div.x,typeMismatch.x,typeMismatch.int,typeMismatch]; arguments [org.springframework.context.support.DefaultMessageSourceResolvable: codes [div.x,x]; arguments []; default message [x]]; default message [Failed to convert property value of type 'java.lang.String' to required type 'int' for property 'x'; nested exception is java.lang.NumberFormatException: For input string: "abc"]
返回

實作 HandlerExceptionResolver 介面

- 目的：
  - 可以自訂全局異常

```
// 自訂全局例外異常處理
@Component
public class MyHandlerExceptionResolver implements HandlerExceptionResolver {
    @Override
    public ModelAndView resolveException(HttpServletRequest request, HttpServletResponse response,
            Object handler, Exception ex) {
        System.out.println("MyHandlerExceptionResolver: 發生全局例外異常");
        String referer = request.getHeader("Referer");
        ModelAndView mv = new ModelAndView();
        mv.setViewName("error/global_error");
        mv.addObject("referer", referer);
        mv.addObject("ex", ex);
        return mv;
    }
}
```

範例演示：

console

整數除法計算

分子： 100

分母： 0

計算

結果：

分母 = 0

MyHandlerExceptionResolver: 發生全局例外異常

Global Error Page

Referer：http://localhost:8080/spring.mvc/mvc/session17/div/

Error message：java.lang.ArithmeticException: / by zero

返回

@ControllerAdvice

- @ControllerAdvice顧名思義就是
  - Controller + AOP
  - Controller 的增強版
  - 可以調撥/處理全局控制
- 在Spring裡，我們可以使用@ControllerAdvice來聲明一些全局性的東西

@ControllerAdvice

- 配合 **@ControllerAdvice** 的標注
  - **@ExceptionHandler**
    - 用於捕獲全局異常
  - **@ModelAttribute**
    - 會在執行目標**Controller**方法之前執行，可以配置一些全局參數。
  - **@InitBinder**
    - 初始化繫結器，用於資料繫結、設定資料轉換器等
    - 用於請求中註冊自定義參數的解析，從而達到自定義請求參數格式的目的

@ControllerAdvice

- 加註 @ExceptionHandler

```
@ControllerAdvice
public class GlobalController {

    @ExceptionHandler({RuntimeException.class, SQLException.class})
    public String fix(Exception ex, Model model, HttpServletRequest request){
        System.out.println("全局例外處理");
        String referer = request.getHeader("Referer");
        model.addAttribute("referer", referer);
        model.addAttribute("ex", ex);
        return "error/global_error";
    }
}
```

全局例外衝突

- 在全局例外攔截相同例外類型時
  - 若同時使用「自訂全局例外異常」與 **@ControllerAdvice**，則 **@ControllerAdvice** 會覆蓋「自訂全局例外異常」

作用

```
@ControllerAdvice
public class GlobalController {
```

不作用

```
// 自訂全局例外異常處理
@Component
public class MyHandlerExceptionResolver implements HandlerExceptionResolver {
```

@ControllerAdvice

- 加註 @ModelAttribute

```java
@ModelAttribute(name = "globalMapData")
public Map<String,Object> mydata() {
    Map<String, Object> map = new LinkedHashMap<>();
    map.put("language", "Java");
    map.put("version", 11);
    map.put("framework", "Spring");
    return map;
}
```

```java
@Controller
@RequestMapping("/session17/div")
public class DivController {

    @GetMapping("/")
    public String index(@ModelAttribute Div div, Model model) {
        // 可以看到在 GlobalController 裡定義的 @ModelAttribute 全域資料
        System.out.println(model.asMap());
        return "session17/div";
    }
}
```

@ControllerAdvice

- 加註 **@InitBinder**
  - 有 **Book.java、Author.java**

```
public class Book {
    private String name;
    private Long price;
    ... getter/setter
}
```

```
public class Author {
    private String name;
    private Integer age;
    ... getter/setter
}
```

因為都有相同的屬性name
這會導致同時配置時產生錯誤

問題演示：

```
<form class="pure-form"
            method="post"
            action="${ pageContext.request.contextPath }/mvc/session17/book_author/">
    <fieldset>
        <legend>Book And Author</legend>
        書名：<input id="name" name="name" /><p />
        價格：<input id="price" name="price" /><p />
        作者：<input id="name" name="name" /><p />
        年齡：<input id="age" name="age" /><p />
        <button type="submit" class="pure-button pure-button-primary">新增</button><p />
    </fieldset>
</form>
```

```
@Controller
@RequestMapping("/session17/book_author")
public class BookAndAuthorController {

    @GetMapping("/")
    public String index() {
        return "session17/book_author";
    }

    @RequestMapping("/")
    @ResponseBody
    public String add(Book book, Author author) {
        return book + " " + author;
    }

}
```

Book And Author

書名： Java

價格： 500

作者： Tom

年齡： 30

新增

name 資料被合併 ??

```
Book [name=Java,Tom, price=500]

Author [name=Java,Tom, age=30]
```

@ControllerAdvice

- 加註 **@InitBinder**
  - 利用 **a, b** 前綴區分 **Author**、**Book**

```
@InitBinder("b")
public void b(WebDataBinder binder) {
    binder.setFieldDefaultPrefix("b.");
}

@InitBinder("a")
public void a(WebDataBinder binder) {
    binder.setFieldDefaultPrefix("a.");
}
```

要加「.」

結果演示：

加入前綴 a.、b.

```
<form class="pure-form"
            method="post"
            action="${ pageContext.request.contextPath }/mvc/session17/book_author/">
    <fieldset>
        <legend>Book And Author</legend>
        書名：<input id="b.name" name="b.name" /><p />
        價格：<input id="b.price" name="b.price" /><p />
        作者：<input id="a.name" name="a.name" /><p />
        年齡：<input id="a.age" name="a.age" /><p />
        <button type="submit" class="pure-button pure-button-primary">新增</button><p />
    </fieldset>
</form>
```

```
@Controller
@RequestMapping("/session17/book_author")
public class BookAndAuthorController {

    @GetMapping("/")
    public String index() {
        return "session17/book_author";
    }

    @RequestMapping("/")
    @ResponseBody
    public String add(@ModelAttribute("b") Book book,
                      @ModelAttribute("a") Author author) {
        return book + " " + author;
    }

}
```

加入 a、b 設定

Book And Author

書名： Java

價格： 500

作者： Tom

年齡： 30

新增

name 資料被正確配置

```
Book [name=Java, price=500]

Author [name=Tom, age=30]
```