spring

MVC – 表單數據驗證

段維瀚 老師

Session

- 表單數據驗證
- JSR 303 優雅的數據驗證
- 驗證文字訊息配置(properties)
- 自訂驗證-實作 Validator
- 表單數據驗證 Lab 練習
  - 範例：股票資訊輸入驗證

表單數據驗證

- 表單數據的資料是否通過驗證檢測 ?

JSR 303 優雅的數據驗證

- JSR 303 不需要編寫驗證器直接透過@進行配置設定
- 或加載 Hibernate Validator 外掛
  - pom.xml

```
<!-- Hibernate Validator -->
<dependency>
    <groupId>org.hibernate</groupId>
    <artifactId>hibernate-validator</artifactId>
    <version>5.4.1.Final</version>
</dependency>
```

參考文章：https://codingnote.cc/zh-hk/p/265841/

注意事項

- 在使用 **javax.validation.constraints.***
  與使用 **org.hibernate.validator.constraints.***
  - 若同時使用要注意是否會有設定上的衝突？
  - 例如：
    - **import javax.validation.constraints.Size;**
      與
      import org.hibernate.validator.constraints.Length;

springmvc-servlet.xml

```
<!-- JSR 303 Validator 驗證: Hibernate Validator 實作驗證配置-->
<bean id="validator"
        class="org.springframework.validation.beanvalidation.LocalValidatorFactoryBean">
    <property name="providerClass" value="org.hibernate.validator.HibernateValidator" />
    <property name="validationMessageSource" ref="messageSource" />
</bean>

<!--  解決中文 與 JSR 303 Validator 設定 -->
<mvc:annotation-driven validator="validator">
    <mvc:message-converters
        register-defaults="true">
        <bean
            class="org.springframework.http.converter.StringHttpMessageConverter">
            <property name="supportedMediaTypes">
                <list>
                    <value>text/html;charset=UTF-8</value>
                </list>
            </property>
        </bean>
    </mvc:message-converters>
</mvc:annotation-driven>
```

Person.java

```
public class Person {
  @NotNull(message = "姓名不可以是空值")
  @Size(min = 2, max = 50, message = "姓名字數範圍必須介於 2~50 個字之間")
  private String name; // 姓名

  @NotNull(message = "年齡不可以是空值")
  @Range(min = 18, max = 99, message = "年齡範圍必須介於 18~99 歲之間")
  private Integer age; // 年齡

  @NotNull(message = "會員的設定不可以是空值")
  private Boolean member; // 是否是會員

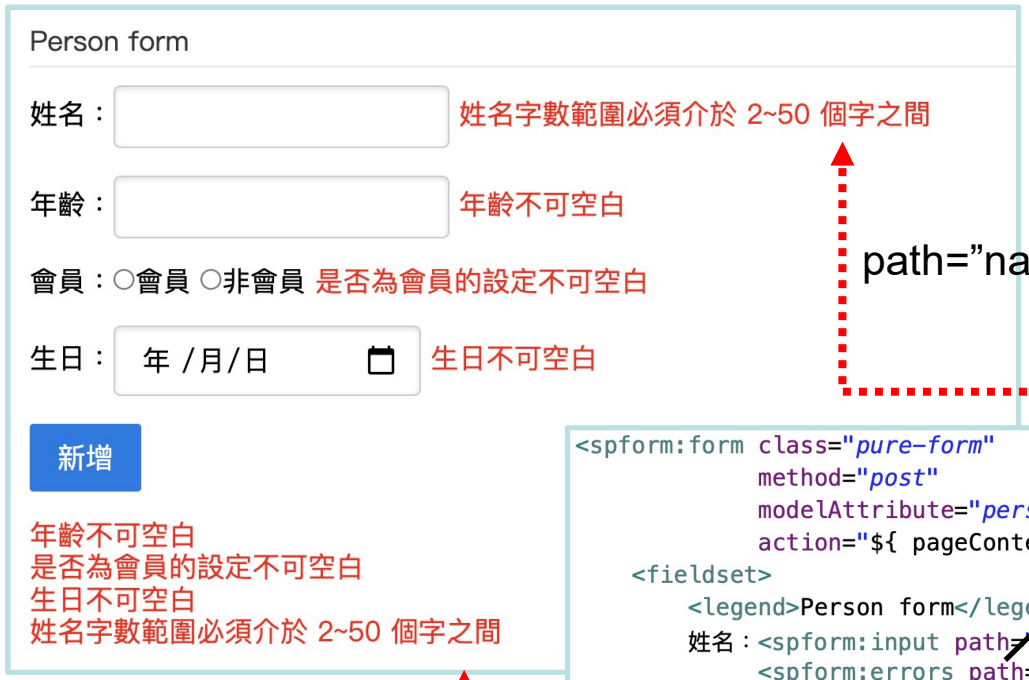
 @NotNull(message = "生日不可以是空值")
  @Past(message = "生日不可以大於現在的日期")
  @JsonFormat(pattern="yyyy-MM-dd",timezone="GMT+8") // 返回日期類型
  @DateTimeFormat(pattern="yyyy-MM-dd") //接收日期類型
  private Date birth; // 生日

    // getter、setter、toString

}
```

Demo

<spform:errors path="欄位名" cssClass="css設定" />



path="name" 特定欄位錯誤

要用 path

path="*" 所有錯誤

```
<spform:form class="pure-form"
             method="post"
             modelAttribute="person"
             action="${ pageContext.request.contextPath }/mvc/session13/person/">
    <fieldset>
        <legend>Person form</legend>
        姓名：<spform:input path="name" />
             <spform:errors path="name" cssClass="error" /><p />
        年齡：<spform:input path="age" />
             <spform:errors path="age" cssClass="error" /><p />
        會員：<spform:radiobutton path="member" value="true" />會員
             <spform:radiobutton path="member" value="false" />非會員
             <spform:errors path="member" cssClass="error" /><p />
        生日：<spform:input path="birth" type="date" />
             <spform:errors path="birth" cssClass="error" /><p />
        <button type="submit" class="pure-button pure-button-primary">新增</button><p />
        <spform:errors path="*" cssClass="error" />
    </fieldset>
</spform:form>
```

將錯誤文字編寫在 properties

/WEB-INF/resource/errorMessage.properties


```
person.name.empty=姓名不可空白
person.name.range=姓名字數範圍必須介於 {min}~{max} 個字之間
person.age.empty=年齡不可空白
person.age.range=年齡範圍必須介於 {min}~{max} 歲之間
person.member.empty=是否為會員的設定不可空白
person.birth.empty=生日不可空白
person.birth.past=生日不可以大於現在的日期
```

Person.java

/WEB-INF/resource/errorMessage.properties

```
person.name.empty=姓名不可空白
person.name.range=姓名字數範圍必須介於 {min}~{max} 個字之間
person.age.empty=年齡不可空白
person.age.range=年齡範圍必須介於 {min}~{max} 歲之間
person.member.empty=是否為會員的設定不可空白
person.birth.empty=生日不可空白
person.birth.past=生日不可以大於現在的日期
```

```
public class Person {
  @NotNull(message = "{person.name.empty}")
  @Size(min = 2, max = 50, message = "{person.name.range}")
  private String name; // 姓名
  @NotNull(message = "{person.age.empty}")
  @Range(min = 18, max = 99, message = "{person.age.range}")
  private Integer age; // 年齡
  @NotNull(message = "{person.member.empty}")
  private Boolean member; // 是否是會員
  @NotNull(message = "{person.birth.empty}")
  @Past(message = "{person.birth.past}")
  @JsonFormat(pattern="yyyy-MM-dd",timezone="GMT+8") // 返回日期類型
  @DateTimeFormat(pattern="yyyy-MM-dd") //接收日期類型
  private Date birth; // 生日
 ... 略
}
```

JSR 303

- **springmvc-servlet.xml**

```
<!-- 錯誤訊息設定 properites -->
<bean id="messageSource"
          class="org.springframework.context.support.ReloadableResourceBundleMessageSource">
    <property name="basename" value="/WEB-INF/resource/errorMessage" />
    <property name="useCodeAsDefaultMessage" value="false"/>
    <property name="defaultEncoding" value="UTF-8"/>
</bean>

<!-- JSR 303 Validator 驗證: Hibernate Validator 實作驗證配置-->
<bean id="validator"
           class="org.springframework.validation.beanvalidation.LocalValidatorFactoryBean">
    <property name="providerClass" value="org.hibernate.validator.HibernateValidator" />
    <property name="validationMessageSource" ref="messageSource" />
</bean>
```

讓 properties 支援 UTF-8

JSR 303 標註類型

- 空檢查
  - @Null、@NotNull、@NotBlank
    - @NotBlank(message="使用者名稱不得空白")
    - @NotNull(message="{user.name.required}")
      - @NotNull: CharSequence, Collection, Map 和 Array 不能是 null, 但可以是空集（size = 0）
      - @NotEmpty: CharSequence, Collection, Map 和 Array 不能是 null 並且相關物件的 size 大於 0
      - @NotBlank: String 不是 null 且 至少包含一个字符

JSR 303 標註類型

- boolean 檢查
  - @AssertTure、@AssertFalse
- 長度檢查
  - @Size(min=, max=)
    @Length(min=, max=)

JSR 303 標註類型

- 日期檢查(Date、Calendar)
  - @Past 驗證是否在當前時間之前(過去時間)
    @Future 驗證是否在當前時間之後(未來時間)
    @PastOrPresent、@FutureOrPresent （包含現在）
- 數值檢查
  - @Min、@Max、@Range、@Digits
- 其他驗證
  - @Email、@CreditCardNumber
  - @Pattern 驗證是否符合表達式的規則格式
    - ```
      @Pattern(regexp = "^[0-9]{1,3}\\.[0-9]{1,3}\\.[0-9]{1,3}\\.[0-9]{1,3}$")
      private String ipAddress;
      ```


      ```

JSR 303 補充

- jsr303，hibernate validator的注解使用教程和爬坑
  - https://blog.csdn.net/w8y56f/article/details/103705934

自訂驗證-實作 Validator

- org.springframework.validation.Validator
  - 實作方法
    - public boolean supports(Class<?> klass)
      - 設定要驗證的物件，返回值若為 false 則不驗證
    - public void validate(Object object, Errors errors)
      - 驗證 object，並將驗證錯誤的訊息傳入 errors
  - 驗證工具
    - ValidationUtils

Stock.java

```
public class Stock {
  private String symbol; // 股票代號：股票代號必須股票代號要存在且有進行交易
  private Double price; // 買進價格：買進價格必須是昨日收盤價的±10%之間
  private Integer amount; // 買進股數：買進股數必須是1000的倍數(1000股 = 1張)
 ... 略
}
```

Spring 驗證器實作

```java
@Component
public class StockValidator implements Validator {
    // 判斷當前的類是否是要驗證的類別
    @Override
    public boolean supports(Class<?> clazz) {
        // 因為 StockValidator 只會判定/驗證 Stock 類
        // 所以要先判斷傳進來的 clazz 是否是 Stock 類別
        return Stock.class.isAssignableFrom(clazz);
    }

    @Override
    public void validate(Object target, Errors errors) {
        Stock stock = (Stock)target;
        // 基礎驗證
        ValidationUtils.rejectIfEmpty(errors, "symbol", "stock.symbol.empty");
        ValidationUtils.rejectIfEmpty(errors, "price", "stock.price.empty");
        ValidationUtils.rejectIfEmpty(errors, "amount", "stock.amount.empty");
        // 進階驗證
        yahoofinance.Stock yStock = null;
        try {
            yStock = YahooFinance.get(stock.getSymbol());
            double previousCose = yStock.getQuote().getPreviousClose().doubleValue(); // 昨日收盤價
            // 買進價格必須是昨日收盤價的±10%之間
            if(stock.getPrice() < previousCose * 0.9 || stock.getPrice() > previousCose * 1.1) {
                errors.rejectValue("price", "stock.price.range");
            }
            // 買進股數必須大於或等於1000
            if(stock.getAmount() < 1000) {
                errors.rejectValue("amount", "stock.amount.notenough");
            }
            // 買進股數必須是1000的倍數(1000股 = 1張)
            if(stock.getAmount() % 1000 != 0) {
                errors.rejectValue("amount", "stock.amount.range");
            }

        } catch (Exception e) {
            e.printStackTrace();
            if(yStock == null) {
                errors.rejectValue("symbol", "stock.symbol.notfound");
            }
        }
    }
}
```

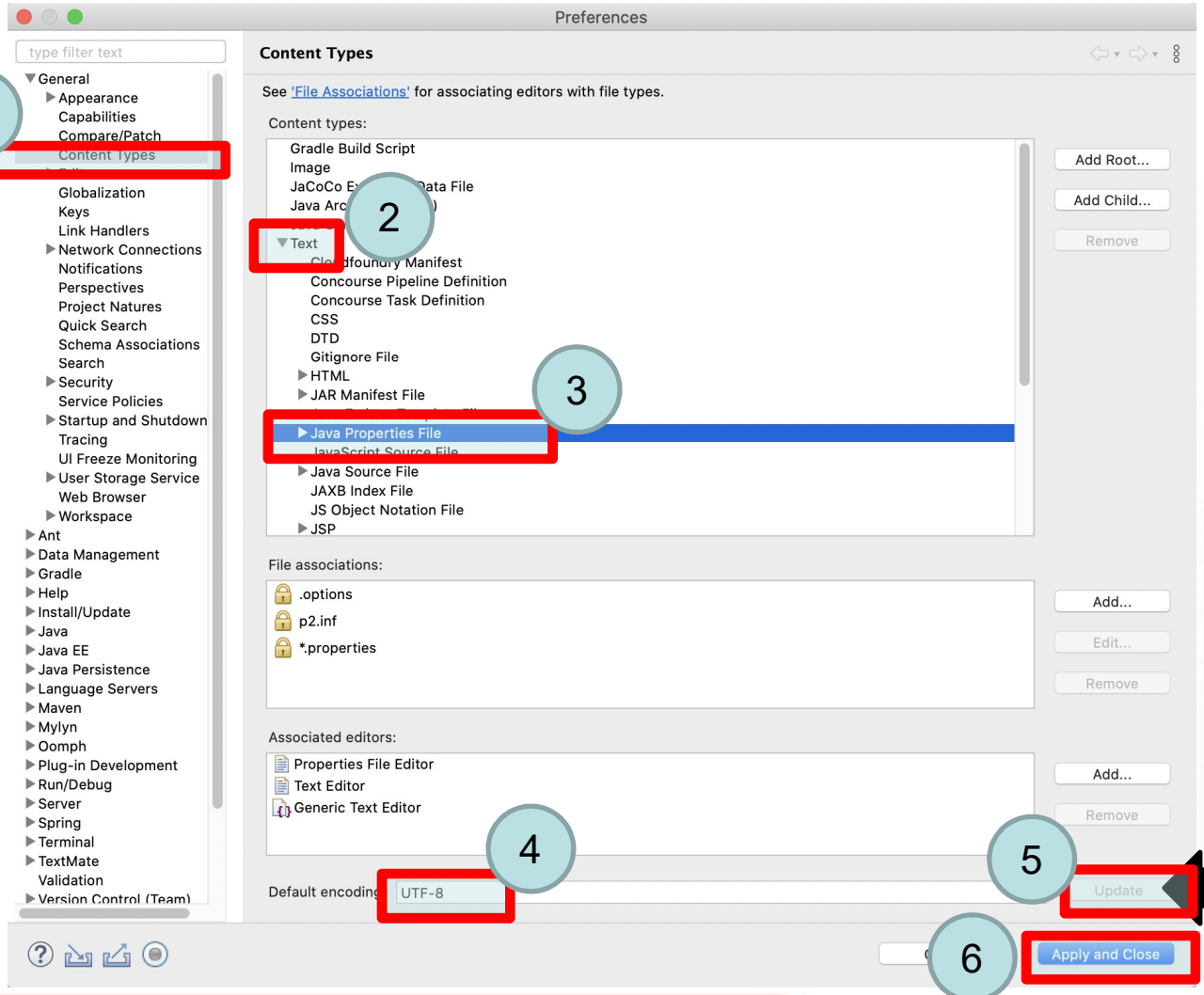
errorMessage.properties

```
stock.symbol.empty=股票代號不可空白
stock.symbol.notfound=股票代號不存在
stock.price.empty=買進價格不可空白
stock.price.range=買進價格必須是昨日收盤價的±10%之間
stock.amount.empty=買進數量不可空白
stock.amount.notenough=買進股數必須大於或等於1000
stock.amount.range=買進股數必須是1000的倍數(1000股 = 1張)
```

Spring 驗證器調用

```
@PostMapping("/")
public String add(Model model, @Valid Stock stock, BindingResult result) {
    // 自主驗證錯誤
    stockValidator.validate(stock, result);
    if(result.hasErrors()) { // 若有錯誤發生，會自動將錯誤訊息傳導到指定 view 中
        model.addAttribute("stocks", stocks); // 手動增加額外要給 view 呈現的資訊
        return "session14/stock";
    }
    stocks.add(stock);
    return "redirect:./";
}
```

讓 properties 顯示中文



注意要按