# Spring.thymeleaf

InformAtelier

Thymeleaf's main goal is to bring elegant natural templates to your development workflow — HTML that can be correctly displayed in browsers and also work as static prototypes, allowing for stronger collaboration in development teams.

# Natural Template

```
<table>
  <thead>
    <tr>
      <th th:text="#{msgs.headers.name}">Name</th>
      <th th:text="#{msgs.headers.price}">Price</th>
    </tr>
```

InformAtelier

# Form

```
<form action="#" th:action="@{/seedstartermng}" th:object="${seedStarter}"
            method="post">
<label th:for="${#ids.next('covered')}"
          th:text="#{seedstarter.covered}">Covered</label>
 <input type="checkbox" th:field="*{covered}" />
<select th:field="*{type}">
 <option th:each="type : ${allTypes}"
       th:value="${type}"
       th:text="#{${'seedstarter.type.' + type}}">Wireframe</option>
</select>
</form>
```

InformAtelier

# Validation - Model

```
import javax.validation.constraints.Min;
import javax.validation.constraints.NotNull;
import javax.validation.constraints.Size;

public class PersonForm {

    @NotNull
    @Size(min=2, max=30)
    private String name;
```

# Validation - Controller

```
@PostMapping("/")
    public String checkPersonInfo(@Valid PersonForm personForm,
BindingResult bindingResult) {


        if (bindingResult.hasErrors()) {
            return "form";
```

# Validation - View

```
<form action="#" th:action="@{/}" th:object="${personForm}" method="post">
<table>
    <tr>
      <td>Name:</td>
       <td><input type="text" th:field="*{name}" /></td>
       <td th:if="${#fields.hasErrors('name')}" th:errors="*{name}">Name Error</td>
     </tr>
```

# ConvertionService

```
<bean id="conversionService"
      class="org.springframework.format.support.FormattingConversionServiceFactoryBean">
   <property name="formatters">
    <set>
     <bean class="thymeleafexamples.stsm.web.conversion.VarietyFormatter" />
     <bean class="thymeleafexamples.stsm.web.conversion.DateFormatter" />
    </set>
   </property>
 </bean>
```

# Formatter

```
public class DateFormatter implements Formatter<Date> {

  @Autowired
  private MessageSource messageSource;

  public DateFormatter() {
    super();
  }

  public Date parse(final String text, final Locale locale) throws ParseException {
    final SimpleDateFormat dateFormat = createDateFormat(locale);
    return dateFormat.parse(text);
  }
```

# Formatter

```
    public String print(final Date object, final Locale locale) {
        final SimpleDateFormat dateFormat = createDateFormat(locale);
        return dateFormat.format(object);
    }

    private SimpleDateFormat createDateFormat(final Locale locale) {
        final String format = this.messageSource.getMessage("date.format", null, locale);
        final SimpleDateFormat dateFormat = new SimpleDateFormat(format);
        dateFormat.setLenient(false);
        return dateFormat;
    }

}
```

# Riferimenti

- https://www.thymeleaf.org/doc/tutorials/2.1/thymeleafspring.html
- https://spring.io/guides/gs/serving-web-content/
- https://spring.io/guides/gs/validating-form-input/