thymeleaf

JAVA · XML · XHTML · HTML5

Natural Templating in Spring MVC with Thymeleaf

Spring I/O 2012

José Miguel Samper · Daniel Fernández

Who are these guys?

José Miguel Samper

- Thymeleaf active contributor (since the beginning!)
- OSS author: osSeo, porQual, YAV Tags

Daniel Fernández

- Thymeleaf author & project lead
- Also jasypt, op4j, javatuples, javaRuntype, javagalician

AGENDA

1. Introducing Thymeleaf
2. Natural templating
3. Let's write templates!
4. Present + future

Introducing Thymeleaf

1. Introducing Thymeleaf

2. Natural templating

3. Let's write templates!

4. Present + future

The Project

- It's a **Java Template Engine**
- Can be used as view layer in Spring MVC
- First stable release: July 2011
- Currently: 2.0.x
- **Elegant**, configurable, extensible
- 21st-century feature set
- **FUN TO USE!**

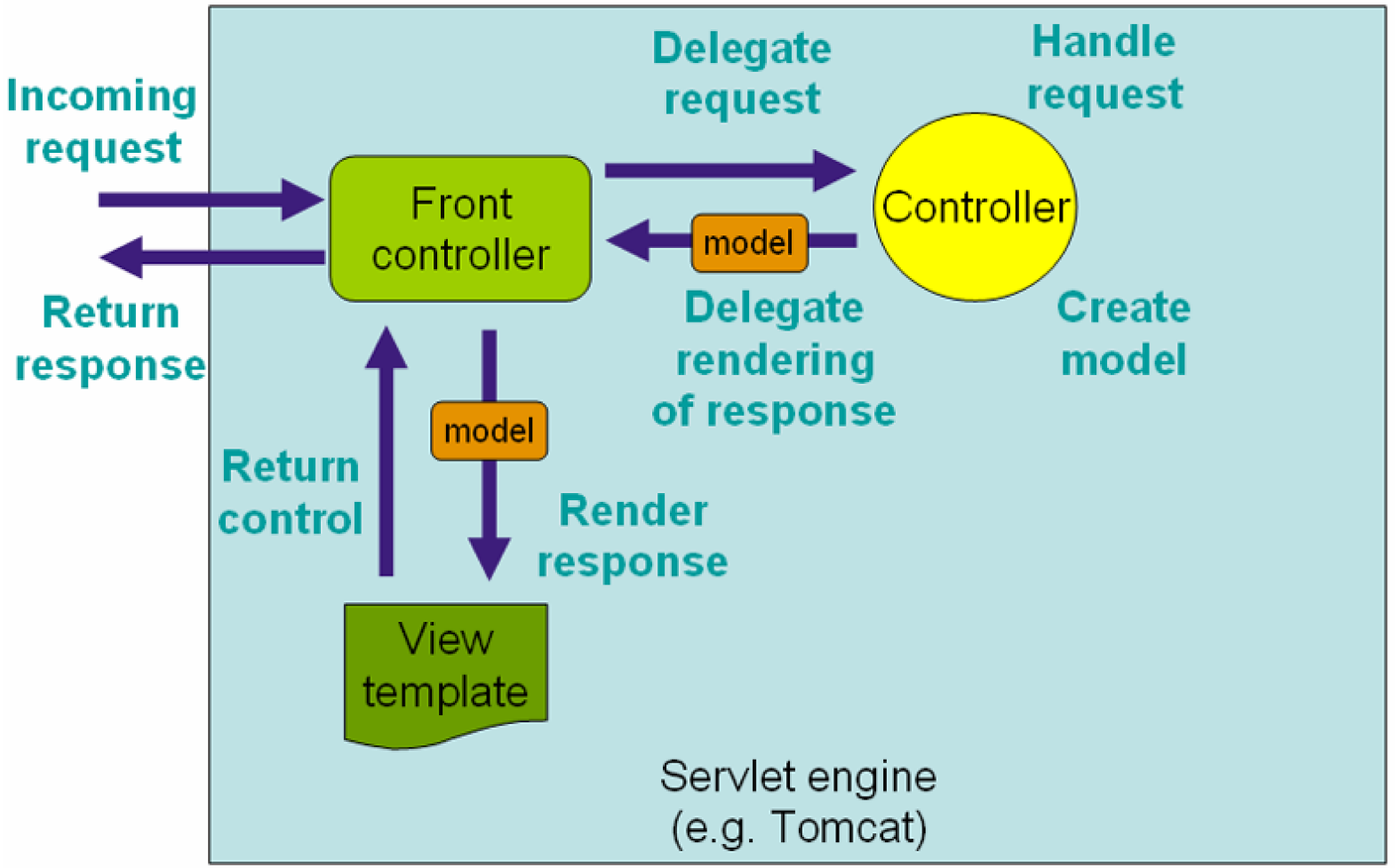
A template engine? What's that? (I)

- **Not** a Web Framework
- Usually a part of them
  - Many web frameworks have their own
  - Takes care of the view layer
  - Template + Data = Document

```
${user.name} → John Apricot
```

Template engines in Spring MVC (I)

- Where in Spring MVC architecture?



Template engines in Spring MVC (II)

- Abstraction:
  - `ViewResolver`, `View`
- Default: **JSP** + **JSTL** + **Spring taglibs**
- Other integrations:
  - Apache Velocity
  - FreeMarker
  - Apache Tiles, XSTL, JasperReports, ...

How does it look like?

```
<table>
  <thead>
    <tr>
      <th th:text="#{msgs.headers.name}">Name</th>
      <th th:text="#{msgs.headers.price}">Price</th>
    </tr>
  </thead>
  <tbody>
    <tr th:each="prod : ${allProducts}">
      <td th:text="${prod.name}">Oranges</td>
      <td th:text="${#numbers.formatDecimal(prod.price,1,2)}">0.99</td>
    </tr>
  </tbody>
</table>
```

The (main) features

- Java, DOM-based
- Online (Web) or Offline (email, XML data...)
- Produces **XML**, **XHTML** or **HTML5**
- Expression eval, i18n, URL rewriting...
- **Full Spring MVC integration**
  - Spring EL, form binding, i18n...
- Configurable and extensible
- Static prototyping abilities

Some features are special (I)

- **DOM-based**: Especially made for the web
  - Web UIs are represented as DOM @ browsers
  - DOM allows powerful processing of documents
  - Thymeleaf's DOM means processing power
  - Better than sequential text processing

Some features are special (II)

- **Configurability** & **Extensibility**
  - Dialects
    - From "*create your own processor libraries*"...
    - ... to "*create your own template engine*"
  - Resolvers ("finders"): templates, messages...
  - Cache strategies
  - Even "Template Modes"
    - Decide what you want to call "*a template*"
    - If it's DOM-able, it's processable.

Some features are special (III)

- **Static prototypes**
  - Static prototyping is not your enemy anymore
  - UI usually starts with static prototypes
  - Prototype-to-working-UI usually hard path
  - A new approach: **NATURAL TEMPLATING**!

Natural Templating

1. Introducing Thymeleaf
2. **Natural templating**
3. Let's write templates!
4. Present + future

Natural what?

- From Wikipedia: `Template Engine (web)`

"Natural Templates = the template can be a document as valid as the final result, the engine syntax doesn't break the document's structure"

How do we evaluate it?

- *“valid document, don't break structure”*
- Templates should be statically displayable
- Static = Open in browser, no web server
- Templates should work as **prototypes**

How can that be done?

- Take profit of browsers' display behaviour
- Use custom attribs, browsers ignore them

```
<div exec="doit()">...</div>
```

- No expressions inside tag bodies

```
<div exec="substitute_body('hello!')">
   Some nice prototyping text...
</div>
```

Can JSP do it?

```
<table>
    <caption>Product list</caption>
    <thead>
        <tr>
            <th>Description</th>
            <th>Price</th>
            <th>Available from</th>
        </tr>
    </thead>
    <tbody>
        <%
        List<Product> products = ProductDAO.getAllProducts();
        for (Product product : products) {
            %>
            <tr>
                <td><%= product.getDescription() %></td>
                <td><%= df.format(product.getPrice()) %></td>
                <td><%= sdf.format(product.getAvailableFrom()) %></td>
            </tr>
            <%
        }
        %>
    </tbody>
</table>
```

Can JSP do it?

NO :-(

Can JSP+JSTL do it?

```
<table>
    <caption>Product list</caption>
    <thead>
        <tr>
            <th>Description</th>
            <th>Price</th>
            <th>Available from</th>
        </tr>
    </thead>
    <tbody>
        <c:forEach var="product" items="${products}">
            <tr>
                <td><c:out value="${product.description}" /></td>
                <td><fmt:formatNumber value="${product.price}" pattern="#.00" /></td>
                <td><fmt:formatDate value="${product.availableFrom}" type="date" /></td>
            </tr>
        </c:forEach>
    </tbody>
</table>
```

Can JSP+JSTL do it?

NO :-|

Can Velocity do it?

```
<table>
    <caption>Product list</caption>
    <thead>
        <tr>
            <th>Description</th>
            <th>Price</th>
            <th>Available from</th>
        </tr>
    </thead>
    <tbody>
        #foreach($product in $products)
            <tr>
                <td>$product.description</td>
                <td>$number.format($product.price)</td>
                <td>$date.format($product.availableFrom)</td>
            </tr>
        #end
    </tbody>
</table>
```

Can Velocity do it?

NO :o(

Can FreeMarker do it?

```
<table>
    <caption>Product list</caption>
    <thead>
        <tr>
            <th>Description</th>
            <th>Price</th>
            <th>Available from</th>
        </tr>
    </thead>
    <tbody>
        <#list products as product>
            <tr>
                <td>${product.description}</td>
                <td>${product.price?string("#.00")}</td>
                <td>${product.availableFrom?date}</td>
            </tr>
        </#list>
    </tbody>
</table>
```

Can FreeMarker do it?

NO :-[

And... can Thymeleaf do it?

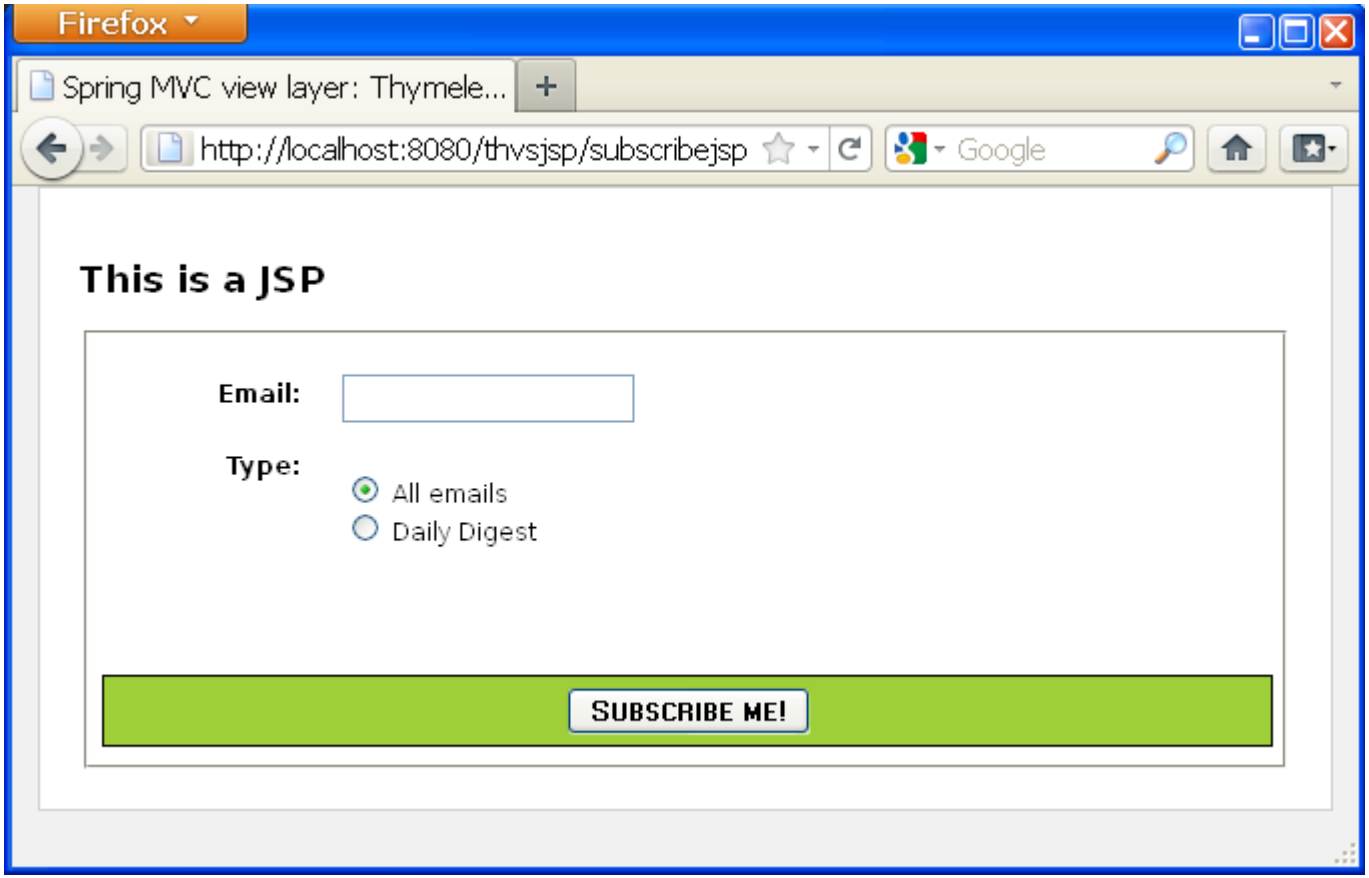
```
<table>
    <caption>Product list</caption>
    <thead>
        <tr>
            <th>Description</th>
            <th>Price</th>
            <th>Available from</th>
        </tr>
    </thead>
    <tbody>
        <tr th:each="product : ${products}">
            <td th:text="${product.description}">Chair</td>
            <td th:text="${#numbers.formatDecimal(product.price, 0, 2)}">30.00</td>
            <td th:text="${#dates.format(product.availableFrom, 'dd/MM/yyyy')}">
                27/12/2012
            </td>
        </tr>
    </tbody>
</table>
```

And... can Thymeleaf do it?

YES! ;-)

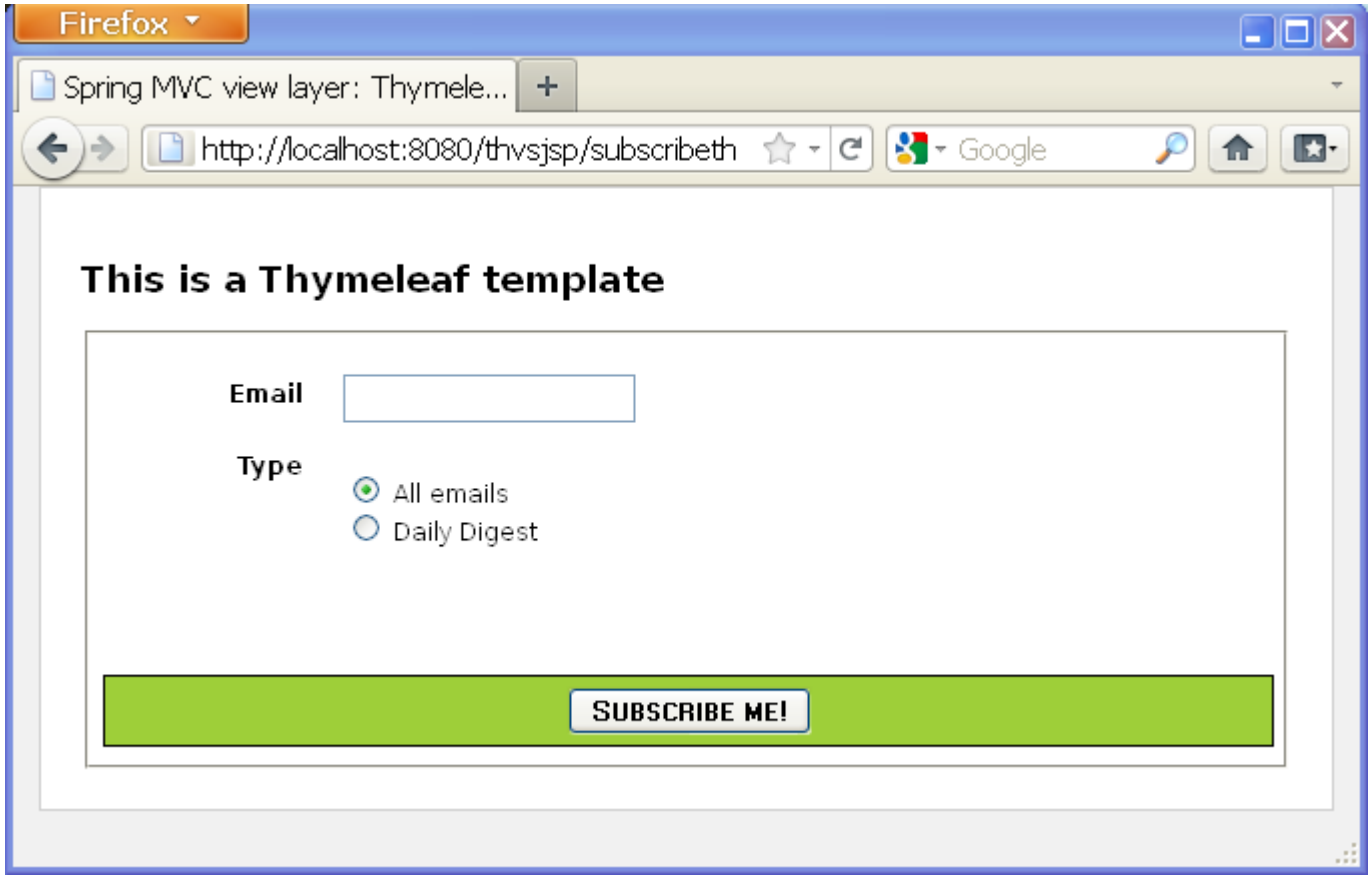
Just how bad is not having this? (I)

`thvsjsp` (example app): JSP on web server



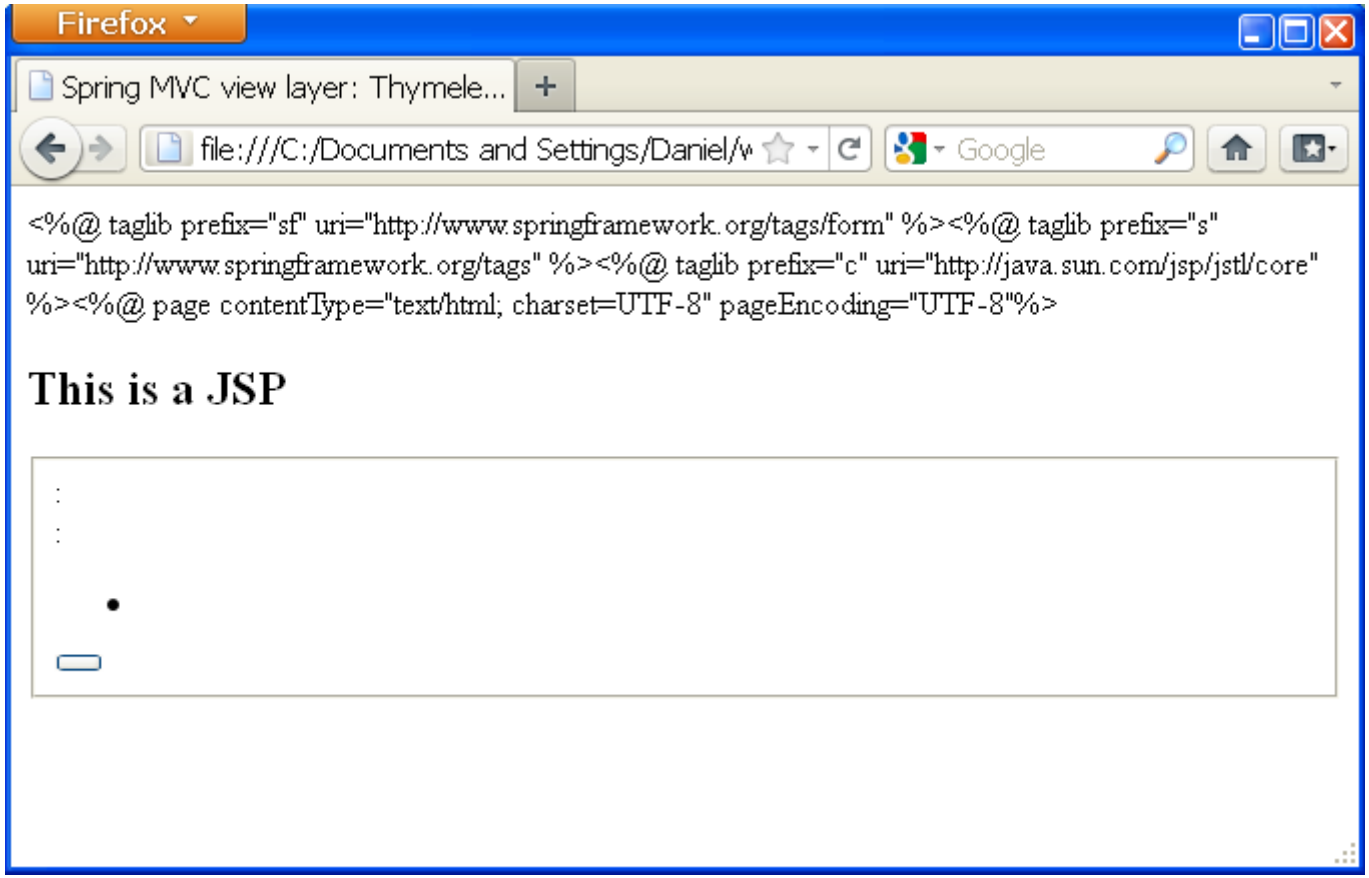
Just how bad is not having this? (II)

`thvsjsp`: Thymeleaf on web server



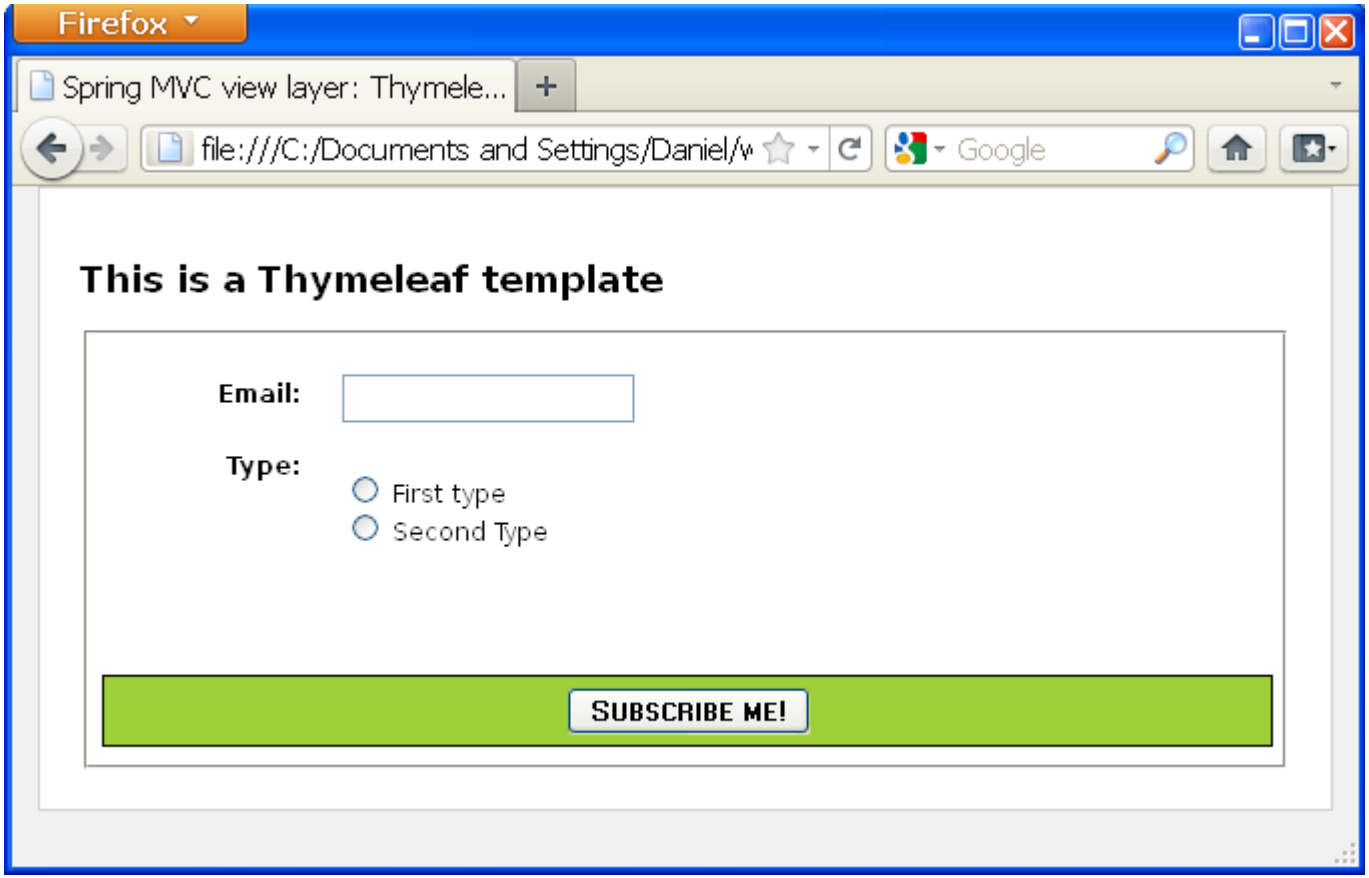
Just how bad is not having this? (III)

`thvsjsp`: JSP statically displayed



Just how bad is not having this? (IV)

`thvsjsp`: Thymeleaf statically displayed



Let's write templates!

1. Introducing Thymeleaf
2. Natural templating
3. **Let's write templates!**
4. Present + future

Writing texts

- `th:text` HTML-escaped text (default)
- `th:utext` unescaped text

```
<h2 th:text="#{product.info}">Product information</h2>
<dl>
    <dt th:text="#{product.name}">Product name</dt>
    <dd th:utext="${product.description}">
        Wooden <b>red</b> chair
    </dd>

    <dt th:text="#{product.price}">Product price</dt>
    <dd th:text="${product.price}">350 €</dd>
</dl>
```

Formatting

#dates, #calendars, #numbers, #strings...

```
<dl>
    <dt>Product name</dt>
    <dd th:text="${#strings.capitalize(product.name)}">350</dd>

    <dt>Product price</dt>
    <dd th:text="${#numbers.formatDecimal(product.price, 1, 2)}">350</dd>

    <dt>Product available from</dt>
    <dd th:text="${#dates.format(product.availableFrom, 'dd-MM-yyyy')}">
        28-Jun-2012
    </dd>
</dl>
```

URLs

- `@{...}` syntax

```
<a th:href="@{/product.action(id=${product.id})}">View product</a>
```

...produces...

```
<a href="/myAppContext/product.action?id=254">View product</a>
```

- Automatic URL-rewriting is performed

Iteration

- th:each

```
<tr th:each="product : ${productList}">
    <td th:text="${product.description}">Red chair</td>
    <td th:text="${product.price}">350 €</td>
</tr>
```

...produces...

```
<tr>
    <td>White table</td>
    <td>200 €</td>
</tr>
<tr>
    <td>Reb table</td>
    <td>200 €</td>
</tr>
<tr>
    <td>Blue table</td>
    <td>200 €</td>
</tr>
```

Iteration status

- th:each

```
<tr th:each="product : ${productList}">
    <td th:text="${productStat.count}">1</td>
    <td th:text="${product.description}">Red chair</td>
    <td th:text="${product.price}">350 €</td>
</tr>
```

...produces...

```
<tr>
    <td>1</td>
    <td>White table</td>
    <td>200 €</td>
</tr>
<tr>
    <td>2</td>
    <td>Reb table</td>
    <td>200 €</td>
</tr>
<tr>
    <td>3</td>
    <td>Blue table</td>
    <td>200 €</td>
</tr>
```

Conditionals (I)

- th:if

```
<span th:if="${product.price lt 100}">Special offer!</span>
```

- th:unless

```
<span th:unless="${product.notInStock}">Currently in stock!</span>
```

Conditionals (II)

- `th:switch` / `th:case`

```
<div th:switch="${user.role}">
    <p th:case="'admin'">User is an administrator</p>
    <p th:case="#{roles.manager}">User is a manager</p>
    <p th:case="*">User is some other thing</p>
</div>
```

Forms and bean-binding (I)

- th:object, th:field

```
<form th:action="@{/saveCustomer}" th:object="${customer}" method="post">
    <input type="hidden" th:field="*{id}" />

    <label for="firstName">First name:</label>
    <input type="text" th:field="*{firstName}" />

    <label for="lastName">Last name:</label>
    <input type="text" th:field="*{lastName}" />

    <label for="countryId">Country:</label>
    <select th:field="*{countryId}">
        <option th:each="country : ${allCountries}"
                th:value="${country.id}" th:text="${country.name}">Ireland</option>
    </select>

    <label for="balance">Balance (euro):</label>
    <input type="text" th:field="*{balance}" />

    <input type="submit" />
</form>
```

Forms and bean-binding (II)

- **Forms integrate fully with Spring**
  - `th:field` acts exactly as Spring taglib tags
    - Slightly different behaviour depending on host tag
  - *PropertyEditors* work OK
  - *Spring EL* expressions in th:field work OK
  - *Validations* work OK (`th:errors`)

Page composition

- Declare fragment with `th:fragment`

```
<h2>Product information</h2>
<div th:fragment="banner">
    <img th:src="@{/logo.png}" alt="Thymeleaf logo" />
    <span>The Thymeleaf tutorial</span>
</div>
...
```

- Reuse `th:include`

```
<h2>Product list</h2>
<div th:include="index.html::banner">Banner</div>
...
```

- ...and reuse again...

```
<h2>Product edition</h2>
<div th:include="index.html::banner">Banner</div>
...
```

Inlining

- Text, Javascript and Dart inlining
- th:inline="text"|"javascript"|"dart"

```
<textarea name="body" th:inline="text">
Dear [[${customerName}]],

it is our sincere pleasure to congratulate you:
    Happy birthday [[${customerName}]]!!!

See you soon, [[${customerName}]].

Regards,
    The Thymeleaf team
</textarea>
```

Present + Future

1. Introducing Thymeleaf
2. Natural templating
3. Let's write templates!
4. **Present + future**

Does anybody use this thing?

- OSS = # of users difficult to know
- Jul 2011 - Jan 2012: **2,528 downloads**
  - Strongly increasing rate
  - Top country @SF.net: China (37%)
  - With Spring integrations @maven: 89%

Thymeleaf in production

- **sahibinden.com**
  - Online classifieds & ecommerce, Turkey
  - [alexa.com] rank 625th worldwide, 9th Turkey
  - Search engine frontend
    - View layer: Thymeleaf 2.0
    - Size: 20 servers
    - 1 Billion hits/month ~385/sec
  - Helped in boosting Thymeleaf performance

The Future

- Detached template modes
  - One file for HTML, another for instructions
- More performance fine-tuning
- More docs, tutorials, example apps...
- Maven archetype(s)
- ...

Where to go, what to see...

- Documentation, articles, code examples

  `http://www.thymeleaf.org/documentation.html`

- User forum

  `http://forum.thymeleaf.org/`

- Twitter

  `@thymeleaf`

Don't forget!

- **Workshop** this afternoon (3:00PM)

Any questions?

Thank you!

José Miguel Samper

jmiguel.samper@atotarreu.com

Daniel Fernández

dfernandez@users.sourceforge.net

This talk at Spring I/O 2012's Lanyrd:

http://lanyrd.com/smtwy