

Titulo

Jose Juan Montiel

Version 1.0, 2016-03-22

Table of Contents

| | |
|--|----|
| Primer fichero | 1 |
| Pongamos CSS, JS e IMG | 1 |
| Y ahora copy/paste | 1 |
| Slide Title will not be displayed | 1 |
| Empezemos con Thymeleaf | 1 |
| Para que coja los cambios en caliente (empecemos con el HTML) | 2 |
| Hasta ahora solo es html | 2 |
| Si los transformamos en templates | 2 |
| Hay que tener en cuenta que el html debe ser valido | 2 |
| ¿Ya no carga los css? | 3 |
| ¿Y como referencio a los css desde el servidor? | 3 |
| Punto de no retorno | 3 |
| Slide Title will not be displayed | 4 |
| Ahora que son templates, porque no usar toda la potencia | 4 |
| Layout | 4 |
| Uso del layout | 5 |
| Includes | 5 |
| Slide Title will not be displayed | 6 |
| Porque no ir mas alla, groovy, clases... .. | 6 |
| Ahora podemos montar formularios | 6 |
| Controlador | 6 |
| Controlador | 7 |
| Validacion de datos en servidor | 8 |
| Servicios rest - JSON | 8 |
| Flujos de control en thymeleaf | 9 |
| Ejemplos complejos html vs tipo dato | 9 |
| Ajax con thymeleaf, fragmentos | 9 |
| Thymeleaf pro tips | 9 |
| SASS / LESS - Gradle | 9 |
| Thymeleaf 3 | 9 |
| JBake | 10 |
| Grooscript | 10 |
| Para instalar (aunque no es necesario) gradle, springboot, jbake | 10 |
| Como usar Git Presenter | 11 |
| Como generar la presentacion con asciidoctor | 11 |

Primer fichero

Bueno, los frontender se dedican al HTML, basicamente esto es lo que haceis no? :)

Primera version de un html

```
<!doctype html>
<html>
<head></head>
<body></body>
</html>
```

Pongamos CSS, JS e IMG

Bueno, si, usais CSS, JS e IMAGENES.

Segunda version del html usando css, js e imagenes.

```
<!doctype html>
<html>
  <head>
    <link type="text/css" rel="stylesheet" href="css/main.css" media="all">
    <script type="text/javascript" src="js/main.js"></script>
  </head>
  <body>
    
    <button onclick="ejemplo()">LOGICA</button>
  </body>
</html>
```

Y ahora copy/paste

Una vez que tenemos la estructura nos ponemos a hacer paginas como churros, ¿no?

Slide Title will not be displayed

By Neil T

Empezemos con Thymeleaf

- Thymeleaf es un sistema de templating Java, que se suele utilizar dentro del ecosistema Spring.

- Vamos a crear un proyecto spring-boot usando esta web: <http://start.spring.io/>
- Con este lo arrancamos: `~/workspace/thymeleaf_talk$./gradlew bootRun --continuous`

Para que coja los cambios en caliente (empecemos con el HTML)

Añadimos al build.gradle

```
bootRun {  
    addResources = true  
}
```

Hasta ahora solo es html

Notar que hasta ahora estabamos mostrando estaticos, pero thymeleaf usa atributos distintos de los estandar con lo que podriamos seguir maquetando en "plano", notar los imports.

Si los transformamos en templates

Añadimos un controlador (para gobernarlos a todos) y ponemos los tags de thymeleaf (dejando los estandar) Hacer uso de un css distinto, y de un js distinto..,

<http://localhost:8080/generic/page1>

Hay que tener en cuenta que el html debe ser valido

<http://localhost:8080/generic/page1>

Error tipico que lanza cuando no puede procesar el HTML bien,

```
org.xml.sax.SAXParseException: El tipo de elemento "link" debe finalizar por la etiqueta  
final coincidente "</link>".
```

En thymeleaf 2.1 LEGACYHTML5 puede ayudar. <https://github.com/thymeleaf/thymeleaf/issues/391>

¿Ya no carga los css?

<file:///WORKSPACE/src/main/resources/templates/index.html>

Y para que siga viendose como html, sin necesidad de arrancar, debemos referenciar al los recursos en su nueva ruta tras mover los html.

```
<link type="text/css" rel="stylesheet" href="css/main.css" media="all" />
```

pasa a

```
<link type="text/css" rel="stylesheet" href="../../static/css/main.css" media="all" />
```

¿Y como referencio a los css desde el servidor?

<http://localhost:8080/generic/page1>

```
<link type="text/css" rel="stylesheet" href="../../static/css/main.css" media="all" />

<li><a href="page1.html">Page 1</a></li>
```

pasa a

```
<link type="text/css" rel="stylesheet" href="../../static/css/main.css" th:href=
"@{/css/main.css}" media="all" />

<li><a href="page1.html" th:href="page1">Page 1</a></li>
```

Punto de no retorno

A partir de este momento, hay opciones para poder seguir viendo el "html plano" sin necesidad de levantar servidor:

- Thymoljs - <http://www.thymoljs.org/>
- Thymeleaf3 - Decoupled logic - <http://www.thymeleaf.org/doc/articles/thymeleaf3migration.html#decoupled-template-logic>

Podrias saltar a meter logica a las templates, bindings, rest... pero tu que eres maquetaador, te gusta estructurar tu HTML en componentes y no tener que repetir menus, headers, footes y componentes por todas partes.

Slide Title will not be displayed

Ahora que son templates, porque no usar toda la potencia

Layouts, includes... <http://www.thymeleaf.org/doc/articles/layouts.html>

Layout

```
<!DOCTYPE html>
<html>
  <head>
    <!--/* Each token will be replaced by their respective titles in the resulting page.
    */-->
    <title layout:title-pattern="$DECORATOR_TITLE - $CONTENT_TITLE">Gochez - </title>

    <link type="text/css" rel="stylesheet" href="../../static/css/main.css" th:href=
"@{/css/main.css}" media="all" />
    <script type="text/javascript" src="../../static/js/main.js" th:src="@{/js/main.js}"
></script>
  </head>
  <body>
    <!--/* Standard layout can be mixed with Layout Dialect */-->
    <div th:replace="fragments/header :: header">
      ...
    </div>
    <div class="container">
      <div layout:fragment="content">

      </div>
      <div th:replace="fragments/footer :: footer">&copy; 2016 The Gochez Templates</div>
    </div>
  </body>
</html>
```

Uso del layout

```
<!DOCTYPE html>
<html layout:decorator="layouts/main">
  <head>
    <title>Index</title>
    <link type="text/css" rel="stylesheet" href="../static/css/main.css" th:href=
"@{/css/index.css}" media="all" />
  </head>
  <body class="colortecni-index">
    <!-- /* Content of this page will be decorated by the elements of layout.html
(task/layout) */ -->
    <div layout:fragment="content">
      
      <button onclick="ejemplo()">LOGICA</button>
    </div>
  </body>
</html>
```

- Ojo al import de css que esta dentro de la pagina, que se añade al del layout...
- Tambien al decorator con el title...

Includes

Si nos fijamos en el layout, habia includes que se usaban para añadir partes comunes, en todas las templates que usen ese layout, y asi se pueden reusar en otras, componentes, vamos...

```
<!DOCTYPE html>
<html>
  <head>
  </head>
  <body>
    <div th:fragment="header">
      <ul>
        <li><a href="page1.html" th:href="page1">Page 1</a></li>
        ...
        <li><a href="page6.html" th:href="page6">Page 6</a></li>
      </ul>
    </div>
  </body>
</html>
```

```
<!DOCTYPE html>
<html>
  <head>
  </head>
  <body>
    <div th:fragment="footer">&copy; 2016 The Gochez Templates</div>
  </body>
</html>
```

Slide Title will not be displayed

Porque no ir mas alla, groovy, clases...

- Con este comando aceleramos las build de gradle: touch ~/.gradle/gradle.properties && echo "org.gradle.daemon=true" >> ~/.gradle/gradle.properties
- Y lo arrancamos con: ~/workspace/thymeleaf_talk\$./gradlew build --continuous
- Y en otro terminal: ./gradle bootRun

Asi de esta manera, sin un ide, podemos aprovechar la ventaja de las devtools de spring-boot <https://spring.io/blog/2015/06/17/devtools-in-spring-boot-1-3>

Ahora podemos montar formularios

Binding, y la potencia de groovy

<https://spring.io/guides/gs/handling-form-submission/>

Controlador


```

@Controller
public class GreetingController {

    @RequestMapping(value="/greeting", method=RequestMethod.GET)
    public String greetingForm(Model model) {
        model.addAttribute("greeting", new Greeting())
        return "greeting"
    }

    @RequestMapping(value="/greeting", method=RequestMethod.POST)
    public String greetingSubmit(@ModelAttribute Greeting greeting, Model model) {
        model.addAttribute("greeting", greeting)
        return "results"
    }

}

```

Controlador

Formulario

```

<div layout:fragment="content">
    <div>Page1</div>
    <h1>Form</h1>
    <form action="#" th:action="@{/greeting}" th:object="${greeting}" method="post">
        <p>Id: <input type="text" th:field="*{id}" /></p>
        <p>Message: <input type="text" th:field="*{content}" /></p>
        <p><input type="submit" value="Submit" /> <input type="reset" value="Reset"
    /></p>
    </form>
</div>

```

Resultado

```

<div layout:fragment="content">
    <div>Page2</div>
    <p>Id: <span th:text="${greeting.id}" /></p>
    <p>Message: <span th:text="${greeting.content}" /></p>
</div>

```

Validacion de datos en servidor

Anotaciones para validar

```
public class Greeting {  
    @Min(1L)  
    long id  
    @NotBlank  
    String content  
}
```

Mas info en:

- <https://spring.io/guides/gs/validating-form-input/>
- <http://hibernate.org/validator/>

Servicios rest - JSON

¿Quieres exponer un servicio rest para consumirlo desde tu maqueta HTML con el framework js de moda?

Controlador rest

```
private static final String template = "Hello, %s!"  
private final AtomicLong counter = new AtomicLong()  
  
@RequestMapping("/rest/greeting")  
public Greeting greeting(@RequestParam(value="name", defaultValue="World") String name) {  
    return new Greeting(  
        id: counter.incrementAndGet(),  
        content: String.format(template, name)  
    )  
}
```

<http://localhost:8080/rest/greeting>

Mas info en:

- <https://spring.io/guides/gs/rest-service/>
- <https://github.com/olivergierke/spring-restbucks>
- <https://github.com/ilopmar/contest>

Flujos de control en thymeleaf

Ejemplos complejos html vs tipo dato

<https://github.com/thymeleaf/thymeleafexamples-petclinic>

Ajax con thymeleaf, fragmentos

<https://github.com/sunnydyal/spring-webflow-thymeleaf-ajax-fragment-sample/tree/master/spring-webflow-thymeleaf-ajax-fragment-sample>

<http://www.thymeleaf.org/doc/tutorials/2.1/thymeleafspring.html#ajax-fragments>

<https://unpocodejava.wordpress.com/2014/04/10/ajax-y-thymeleaf-fragments/> Correo: Investigación Ajax-Thymeleaf

Thymeleaf pro tips

Projection & selection on collection

<https://doanduyhai.wordpress.com/2012/04/14/spring-mvc-part-iv-thymeleaf-advanced-usage/>

```
<tr th:each="artist,rowStat : ${listArtits.[alive == true]}">
<tr th:each="artist,rowStat : ${listArtits.![firstname+ ' '+lastname]}">
<tr th:each="artist,rowStat : ${listArtits.[alive == true].![firstname+ ' '+lastname]}">
```

SASS / LESS - Gradle

<https://github.com/robletcher/gradle-compass-bootstrap-with-gulp-2/>

<http://broonix-rants.ghost.io/spring-boot-building->

Thymeleaf 3

<http://www.thymeleaf.org/doc/articles/thymeleaf3migration.html>

Decoupled logic: <http://www.thymeleaf.org/doc/articles/thymeleaf3migration.html#decoupled-template-logic>

home.html Template sin logica extra

```
<!DOCTYPE html>
<html>
  <body>
    <table id="usersTable">
      <tr>
        <td class="username">Jeremy Grapefruit</td>
        <td class="usertype">Normal User</td>
      </tr>
      <tr>
        <td class="username">Alice Watermelon</td>
        <td class="usertype">Administrator</td>
      </tr>
    </table>
  </body>
</html>
```

home.th.html Logica para la template

```
<?xml version="1.0"?>
<thlogic>
  <attr sel="#usersTable" th:remove="all-but-first">
    <attr sel="/tr[0]" th:each="user : ${users}">
      <attr sel="td.username" th:text="${user.name}" />
      <attr sel="td.usertype" th:text="#{|user.type.${user.type}||}" />
    </attr>
  </attr>
</thlogic>
```

JBake

http://jbake.org/docs/2.4.0/#project_structure

Grooscript

<http://grooscript.org/doc.html>

Para instalar (aunque no es necesario) gradle, springboot, jbake

<http://sdkman.io/install.html>

Como usar Git Presenter

Se han seguido los pasos de [esta documentacion](#).

Como generar la presentacion con asciidoctor

Se han seguido los pasos de [esta documentacion](#).

Para generar (ejecutar dentro de docs):

- HTML
 - `asciidoctor -T asciidoctor-deck.js/templates/haml manual.adoc`
- PDF
 - `asciidoctor -r asciidoctor-pdf -b pdf manual.adoc`

Usndo gradle (en la raiz)

- `gradle asciidoctor`