Titulo

Jose Juan Montiel

Version 1.0, 2016-03-22

Table of Contents

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>
    <img src="img/logo_0.png">
    <button onclick="ejemplo()">LOGICA</button>
</body>
</html>
```

Y ahora copy/paste

Una vez que tenemos la estructura nos ponemos a hacer paginas como churros.

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/

Y lo arrancamos con: ~/workspace/thymeleaf_talk$ ./gradlew bootRun

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

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

¿Ya no carga los css?

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

Y para que sigua 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:

- Thymol
- Thymeleaf3 - Decoupled logic

Podrias saltar a meter logica a las templates, bindings, rest... pero tu que eres maquetador, 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">
        <img src="../static/img/logo_0.png" th:src="@{/img/logo_0.png}"/>
        <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...

Includes

```
<!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, 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

Flujose de control en 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]}">
```

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

Servicios rest - JSON

https://github.com/olivergierke/spring-restbucks https://github.com/ilopmar/contest

SASS / LESS - Gradle

https://github.com/robfletcher/gradle-compass http://broonix-rants.ghost.io/spring-boot-building-

bootstrap-with-gulp-2/

Grooscript

http://grooscript.org/doc.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