



Instituto Politécnico de Castelo Branco
Escola Superior de Tecnologia

Thymeleaf

Unidade Curricular de
Aplicações Internet Distribuídas / Aplicações Distribuídas

Licenciatura em Engenharia Informática
Mestrado em Desenvolvimento de Software e Sistemas Interactivos
UTC Informática
Est-IPCB
Ano Letivo 2022/2023

Prof.º Doutor Alexandre Fonte
(adf@ipcb.pt)

Sumário

- Thymeleaf
 - Standard Expressions
 - Formulários
 - Expressões Condicionais
 - Ciclos

Declaração de Direitos de Autor

É Proibida a cópia, difusão, ou uso destes materiais não seja no âmbito exclusivo da Unidade Curricular de Aplicações Distribuídas do curso de Licenciatura em Engenharia Informática da Escola Superior de Tecnologia do Instituto Politécnico de Castelo Branco.

What is Thymeleaf?

- Thymeleaf is a view engine used in **Spring**
 - É uma biblioteca Java. Atualmente, opera melhor com html5 (in thymeleaf.org)
- It allows us to:
 - Use variables/collections in our views
 - Execute operations on our variables

"Thymeleaf is very, very extensible, and it allows you to define your own sets of template attributes (or even tags) with the names you want, evaluating the expressions you want in the syntax you want and applying the logic you want. It's more like a *template engine framework*." - thymeleaf.org

How to use Thymeleaf?

- Use Spring Initializr to import Thymeleaf, or use this dependency in your **pom.xml**:

```
<dependency>  
  <groupId>org.springframework.boot</groupId>  
  <artifactId>spring-boot-starter-thymeleaf</artifactId>  
</dependency>
```

- Define the Thymeleaf library in your html file:

```
<html xmlns="http://www.w3.org/1999/xhtml"  
      xmlns:th="http://www.thymeleaf.org">
```

How to use Thymeleaf?

```
<!DOCTYPE html SYSTEM "http://www.thymeleaf.org/dtd/xhtml1-strict-thymeleaf-4.dtd">
```

```
<html xmlns="http://www.w3.org/1999/xhtml"  
      xmlns:th="http://www.thymeleaf.org">
```

```
<head>  
  <title>Good Thymes Virtual Grocery</title>  
  <meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />  
  <link rel="stylesheet" type="text/css" media="all"  
        href="../../css/gtvg.css" th:href="@{/css/gtvg.css}" />  
</head>
```

```
<body>
```

```
<p th:text="#{home.welcome}">Welcome to our grocery store!</p>
```

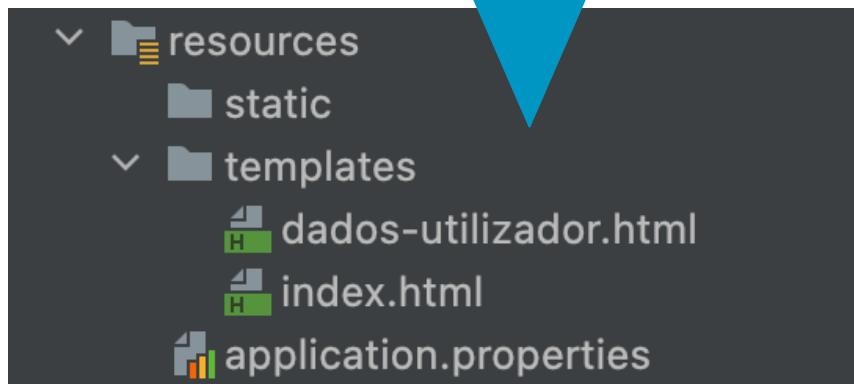
```
</body>
```

```
</html>
```

How to use Thymeleaf?

- Templates Dinâmicos

-In the Spring Boot Web MVC application, Thymeleaf templates are placed into a `/src/main/resources/templates` folder.



dados-utilizador.html

```
<html xmlns:th="https://thymeleaf.org">
<table>
<tr>
<td><h4>Nome Utilizador: </h4></td>
<td><h4 th:text="${utilizador.nome}"></h4></td>
</tr>
<tr>
<td><h4>Email ID: </h4></td>
<td><h4 th:text="${utilizador.email}"></h4></td>
</tr>
</table>
</html>
```

How to use Thymeleaf?

- Classe Controlador

-The controller class has a single method that returns a name of a View. Because the name of the View is “**index**”, you will need to create a Thymeleaf template that is called **index.html**.

```
@Controller
public class HomeController {

    @GetMapping("/")
    public String homePage()
    {
        return "index";
    }
}
```


Thymeleaf Tags and Attributes

- All Thymeleaf tags and attributes begin with **th:** by default
- Example of Thymeleaf attribute:

```
<p th:text="example"></p>
```

- Example of Thymeleaf tag(element processor):

```
<th:block>  
  ...  
</th:block>
```

Variáveis e Links Thymeleaf

- Variable Expressions are executed on the context variables

```
${...}
```

Permitem mostrar os atributos nas Vistas

Field of a object:

```
*{...}
```

- Examples:

```
${dog.name}
```

```
${title}
```

- Link Expressions are used to build URLs

Fragments

```
@{...}
```

```
~{...}
```

- Examples:

```
<a th:href="@{/register}">Register</a>
```

```
<a th:href="@{/details/{id}(id=${game.id})}">Details</a>
```

Thymeleaf Selection Expressions

- Selection Expressions are executed on the previously selected object
- Example:


```
*{...}
```

```
<div th:object="${book}">
  ...
  <span
th:text="*{title}">...</span>
  //equivalent to ${book.title}
  ...
</div>
```

Forms in Thymeleaf

- In Thymeleaf you can create almost normal HTML forms:

```
<form th:action="@{/salva}" th:method="post" th:object=${utilizador}>
  <input type="number" th:field=*{id}/>
  <input type="text" th:field=*{name}/>
  <input type="email" th:field=*{email}/>
  <input type="submit"/>
</form>
```



Nome do
Atributo Modelo

- The th:field attribute creates different attributes based on the input type.
- Por exemplo:

```
<input type="number" th:field=*{id}/>
```

- É *renderizado* como:

```
<input type="number" id="id" name="id" th:value=*{id}/>
```

The input field names must be the same as the object field names

Forms in Thymeleaf (2) - Controlador que aceitará o objeto utilizador

- In Thymeleaf you can create almost normal HTML forms:

```
<form th:action="@{/salva}" th:method="post" th:object=${utilizador}>
  <input type="number" name="id" th:field=*{id}/>
  <input type="text" name="name" th:field=*{name}/>
  <input type="email" name="email" th:field=*{email}/>
  <input type="submit"/>
</form>
```

- You can have a controller that will accept an object of given type:

```
@PostMapping("/salva")
public ModelAndView regista(@ModelAttribute Utilizador
utilizador) { ... }
```

The input field names must be the same as the object field names

Mostrar Model Attributes

- (1) Simple Attributes - th:text="{attributename}"

In the Controller class:

```
model.addAttribute("servertime", dateFormat.format(new Date()));
```

Can be accessed as:

```
Current time is <span h:text:"${servertime}">
```

- (2) Collection Attributes

In the Controller class:

```
List<Student> studentsList = new ArrayList<>();  
studentsList.add(student1);  
studentsList.add(student2);  
model.addAttribute("studentsList", studentsList);
```

Can be accessed as:

```
<tbody>  
  <tr th:each="student:${studentList}">  
    <td th:text=${student.id}>  
    <td th:text=${student.name}>  
  </tr>  
</tbody>
```

O Model será automaticamente passado à vista como uma variável de contexto e os atributos pode ser acedidos do Thymeleaf usando `${studentsList}`

Conditional Statements in Thymeleaf

- If statements can be created in the following way:

```
<div th:if="${...}">
  <p>The statement is true</p>
</div>
```

- You can create inverted if statements using **th:unless**:

```
<div th:unless="${...}">
  <p>The statement is false</p>
</div>
```

- Exemplo

```
<div th:if="${student.gender}=='M'" th:text="Male"/>

<div th:unless="${student.gender}=='M'"
th:text="Female"/>
```

Conditional Statements in Thymeleaf (2)

- You can also create switch in which the default case is "*":

```
<div th:switch="${writer.role}">
  <p th:case="'ADMIN'">The user is admin</p>
  <p th:case="'MOD'">The user is moderator</p>
  <p th:case="*">The user has no privileges</p>
</div>
```

- Or use ternary operator:

```
<p th:text="${row.even} ? 'even' : 'odd'">
  ...
</p>
```


Loops in Thymeleaf

- Creating a for loop:

```
<div th:each="element :  
    ${#numbers.sequence(start, end, step)}">  
    <p th:text="${element}"></p>  
</div>
```

- Example:

```
<div th:each="element : ${#numbers.sequence(1, 5)}">  
    <p th:text="${element}"></p>  
</div>  
// 1 2 3 4 5
```

Loops in Thymeleaf (2)

- Creating a for-each loop:

```
<div th:each="book : ${books}">  
  <p th:text="*{author}"></p>  
</div>
```

- Getting the iterator:

```
<div th:each="u, iter : ${users}" th:object="${u}">  
  <p th:text="|ID: *{id}, Username: *{name}|"></p>  
  <p th:text="|${iter.index} of ${iter.size}|"></p>  
</div>
```

A minha primeira Aplicação Spring Boot MVC Web com Templates Thymeleaf

- **Atividade Prática n.º2:** Criar uma aplicação multinível chamada **DemoMVCWeb-Thymeleaf**

A minha primeira Aplicação Spring Boot MVC Web com Templates Thymeleaf

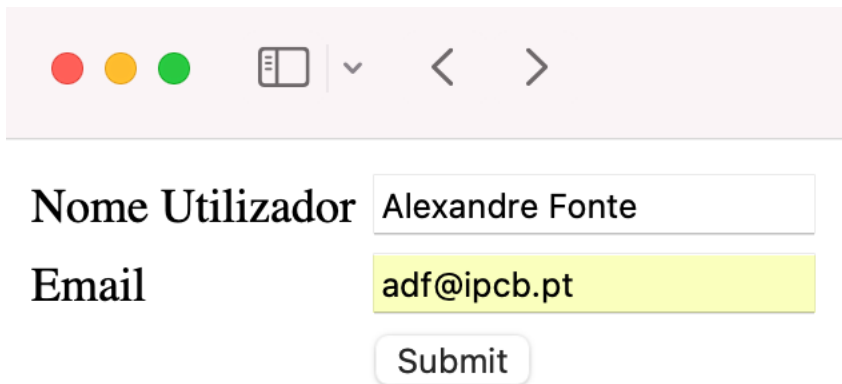
- **Passo 1: Criar uma Aplicação DemoMVC-Thymeleaf**
Utilize o Spring Initializr para adicionar ao ficheiro pom.xml os Starter Spring Web, e Thymeleaf.
- **Passo 2: Criar o modelo Utilizador e um Controlador UtilizadorControlador**
 - A classe utilizador mantém os dados de um utilizador
 - O controlador atualiza o modelo Utilizador e atualiza as vistas (páginas Web)

```
public class Utilizador {  
    private String nome;  
    private String email;  
    private String password;  
    //getters e setters  
}
```

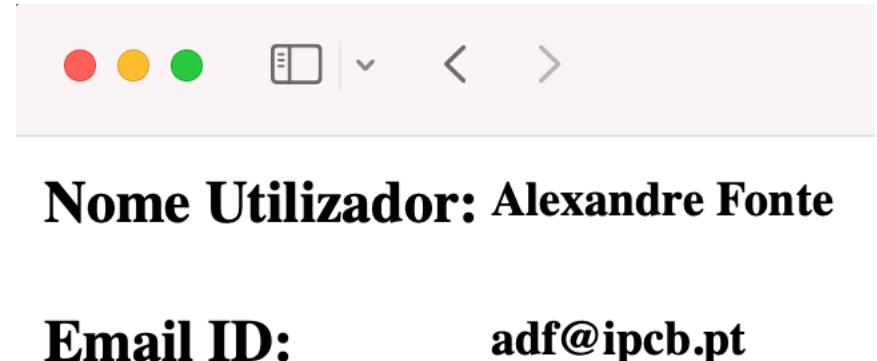
```
@Controller  
public class ControladorUtilizador  
{  
    // Métodos que executam as ações  
}
```

A minha primeira Aplicação Spring Boot MVC Web com Templates Thymeleaf

- **Passo 3:** Implementar os seguintes métodos ação
 - String index();
 - Retorna a Vista index.html
 - ModelAndView registaUtilizador (Utilizador utilizador);
 - Retorna o Modelo Utilizador e a Vista dados-utilizador.html
- **Passo 4:** Criar duas templates Thymeleaf html
 - index.html
 - dados-utilizador.html



A screenshot of a web browser window displaying a registration form. The browser's address bar is empty. The form has two input fields: "Nome Utilizador" with the value "Alexandre Fonte" and "Email" with the value "adf@ipcb.pt". Below the email field is a "Submit" button.



A screenshot of a web browser window displaying the result of the registration. The browser's address bar is empty. The page shows the text "Nome Utilizador: Alexandre Fonte" and "Email ID: adf@ipcb.pt".

A minha primeira Aplicação Spring Boot MVC Web com Templates Thymeleaf

Exemplo: index.html

```
<html xmlns:th="https://thymeleaf.org">
<head>
<title>Página Gestão de Utilizadores</title>
</head>
<body>
<form th:action="@{/registar}" th:method="post">
<table>
<tr>
<td><label for="nome-utilizador">Nome Utilizador</label></td>
<td><input type="text" name="nome"></input></td>
</tr>
<tr>
<td><label for="email">Email</label></td>
<td><input type="text" name="email"></input></td>
</tr>
<tr>
<td></td>
<td><input type="submit" value="Submit"></input></td>
</tr>
</table>
</form>
</body>
</html>
```

A minha primeira Aplicação Spring Boot MVC Web com Templates Thymeleaf

Exemplo: dados-utilizador.html

```
<html xmlns:th="https://thymeleaf.org">
<table>
<tr>
<td><h4>Nome Utilizador: </h4></td>
<td><h5 th:text="${utilizador.nome}"></h5></td>
</tr>
<tr>
<td><h4>Email ID: </h4></td>
<td><h5 th:text="${utilizador.email}"></h5></td>
</tr>
</table>
</html>
```

Questões

