



**UNIVERSIDADE DE ITAÚNA  
PRÓ-REITORIA DE ENSINO  
COORDENAÇÃO DE CIÊNCIA DA COMPUTAÇÃO  
BACHARELADO EM CIÊNCIA DA COMPUTAÇÃO**

**FERNANDO GERALDO NOGUEIRA**

**GERADOR DE HTML EM JAVA**

**ITAÚNA  
2020**

# Sumário

<b>1</b>	<b>Introdução</b>	<b>3</b>
1.1	Gerador de HTML . . . . .	3
1.1.1	Modo de uso . . . . .	4
<b>2</b>	<b>Implementação</b>	<b>6</b>
2.1	<i>Composite</i> . . . . .	7
2.2	<i>Factory Method</i> . . . . .	7
2.3	<i>Flyheight</i> . . . . .	8
<b>3</b>	<b>Testes</b>	<b>9</b>
<b>4</b>	<b>Conclusão</b>	<b>13</b>
	<b>Referências Bibliográficas</b>	<b>14</b>

# 1 *Introdução*

O *HTML (HyperText Markup Language)* é o bloco de construção mais básico da Web. Ele define o significado e a estrutura do conteúdo da web (DEVDOCS, 2020).

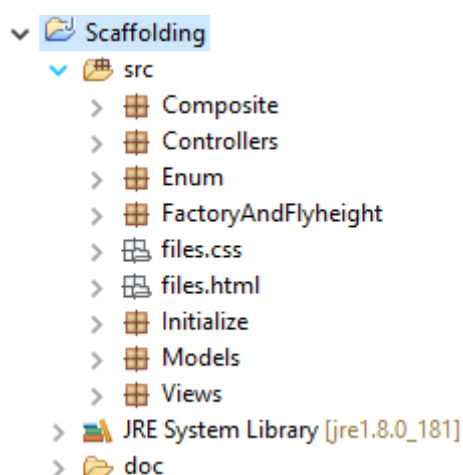
A escrita de uma página *html*, pode não ser tão trivial para a maioria das pessoas. Nesse sentido, cria-se uma necessidade de construir uma ferramenta para gerar automaticamente as páginas *html*, de acordo com a necessidade do usuário.

## 1.1 Gerador de HTML

Foi criada uma ferramenta com o nome de *Scaffolding* para gerar páginas em *html* automaticamente. O projeto foi estruturado em 7 pacotes, como mostrado na Figura 1:

- O primeiro é a estrutura do *Composite*;
- O segundo são os controladores do sistema, chamados de *Controllers*;
- O terceiro são as classes *Enum*;
- O quarto representa a fábrica e o *Flyheight*;

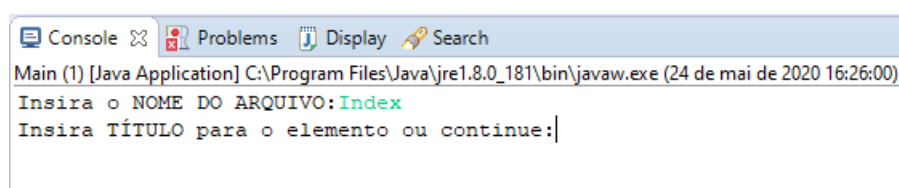
- O quinto é composto pelo método *Main*;
- O sexto representa os componentes do *html*;
- O sétimo controla o acesso de entrada e saída de informações.



**Figura 1:** Estrutura do Projeto *Scaffolding*  
Fonte: Criado pelo autor.

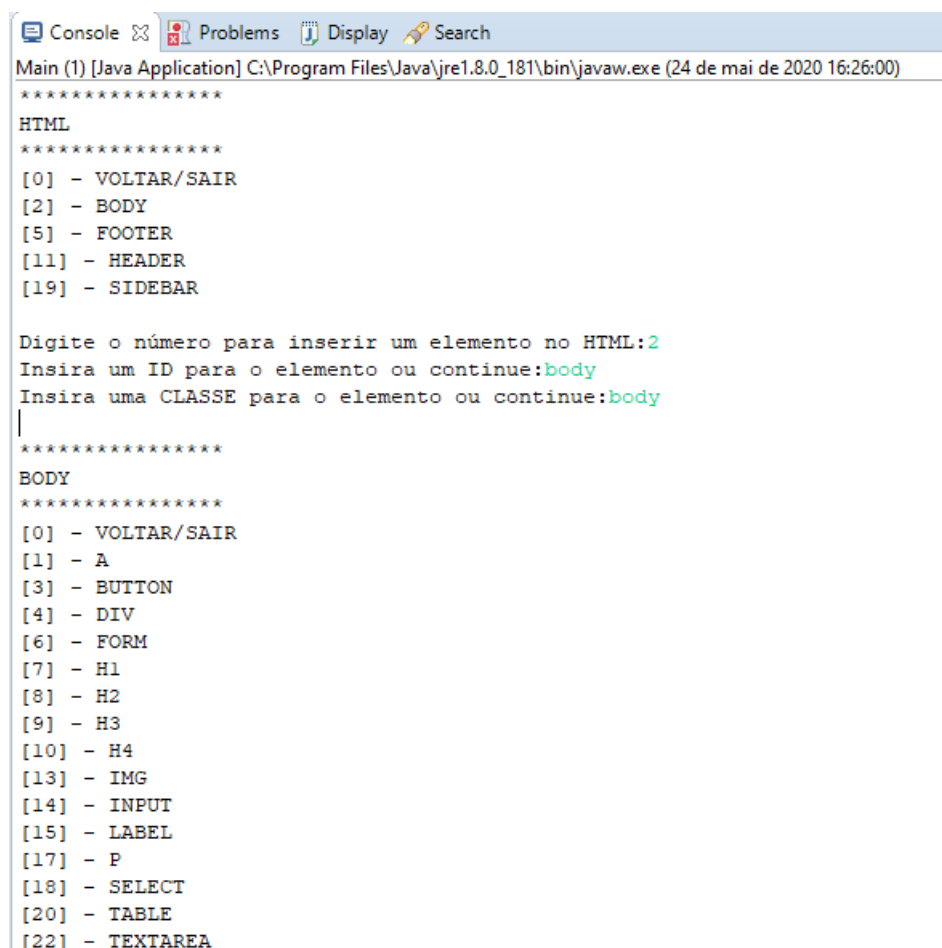
### 1.1.1 Modo de uso

Como mostrado na Figura 2, inicialmente a ferramenta pede o usuário para inserir o nome do arquivo que será gerado.



**Figura 2:** Início da ferramenta.  
Fonte: Criado pelo autor.

Depois de inserido o nome do arquivo, o programa vai retornar quais as *tags* podem ser utilizadas dentro da *tag* atual, por exemplo, na Figura 3, a *tag* *body* foi criada solicitando para o usuário as opções de *tags* que a *body* pode compor.



```
Console Problems Display Search
Main (1) [Java Application] C:\Program Files\Java\jre1.8.0_181\bin\javaw.exe (24 de mai de 2020 16:26:00)
*****
HTML
*****
[0] - VOLTAR/SAIR
[2] - BODY
[5] - FOOTER
[11] - HEADER
[19] - SIDEBAR

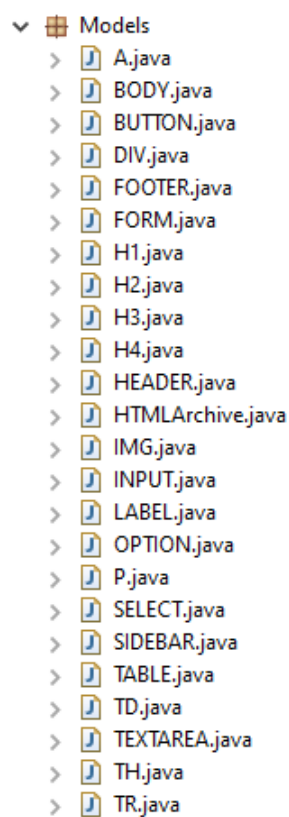
Digite o número para inserir um elemento no HTML:2
Insira um ID para o elemento ou continue:body
Insira uma CLASSE para o elemento ou continue:body
|
*****
BODY
*****
[0] - VOLTAR/SAIR
[1] - A
[3] - BUTTON
[4] - DIV
[6] - FORM
[7] - H1
[8] - H2
[9] - H3
[10] - H4
[13] - IMG
[14] - INPUT
[15] - LABEL
[17] - P
[18] - SELECT
[20] - TABLE
[22] - TEXTAREA
```

**Figura 3:** A *tag* atual, informando quais os componentes ela pode compor  
Fonte: Criado pelo autor.

Posteriormente o usuário pode continuar inserindo suas *tags* ou voltar, quando o usuário seleciona a opção de *VOLTAR/SAIR*, ele automaticamente será redirecionado para a *tag* anterior que ele esta, fechando consequentemente a *tag* que estava.

## 2 Implementação

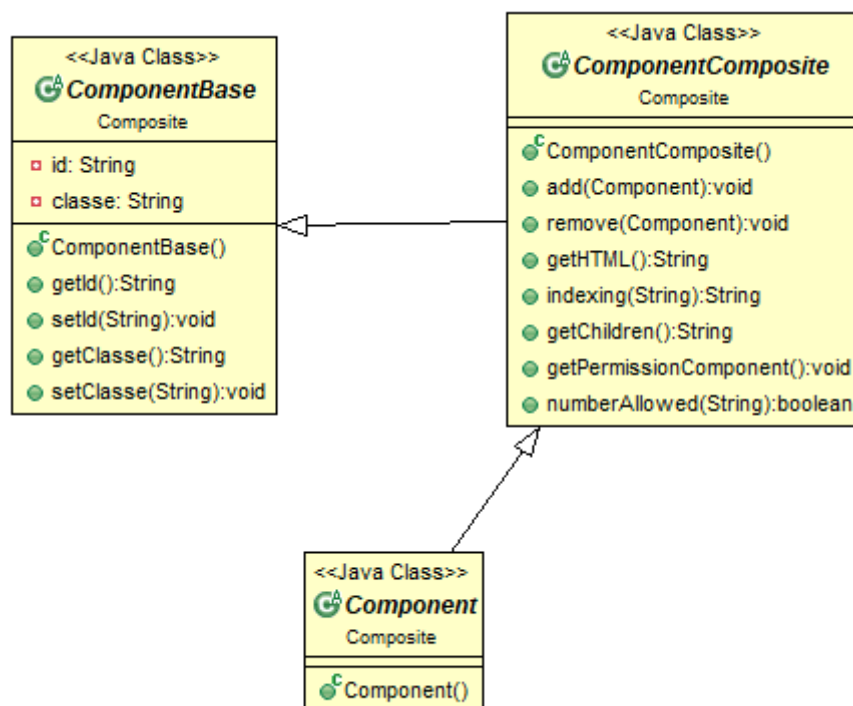
Para a implementação da ferramenta, foram criados modelos (objetos) para cada *tag* de *html*, como mostrado na Figura 4. A estrutura do projeto foi realizado com o auxílio de alguns padrões de projeto, sendo eles, *Composite*, *Factory Method* e o *Flyheight*.



**Figura 4:** Modelos da ferramenta.  
Fonte: Criado pelo autor.

## 2.1 Composite

O *Composite*, é um padrão estrutural, responsável por compor componentes entre objetos de uma mesma classe abstrata, criando então uma árvore e níveis hierárquicos, possuindo também a característica de controlar responsabilidade permitidas entre os objetos. Ele foi utilizado para compor objetos e controlar as características do objeto, a Figura 5 mostra a estrutura do pacote *Composite*.



**Figura 5:** Diagrama do pacote *Composite*.  
Fonte: Criado pelo autor.

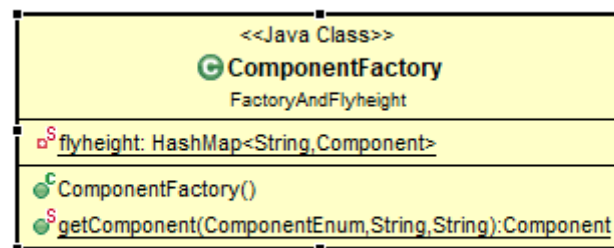
## 2.2 Factory Method

O *Factory Method*, é um padrão de criação, responsável pela gerência da inicialização dos objetos, é basicamente uma fábrica do sistema. Ele foi

utilizado para ser o responsável pela criação dos objetos, a Figura 6 mostra o diagrama da classe.

## 2.3 Flyheight

O *Flyheight*, Figura 6, é responsável pela economia de memória. Em um sistema de criação de elementos *html*, é bem comum ocorrer repetições de *tags*, nesse sentido, deve-se criar um controle para restringir a criação objetos já existentes. Esse padrão foi utilizado junto com o *Factory Method*, sendo os responsáveis por fornecer os objetos para o sistema.



**Figura 6:** Diagrama do pacote *FactoryAndFlyheight*.  
Fonte: Criado pelo autor.



### 3 Testes

Foi realizado um teste de criação de um arquivo de cadastro de usuário, possuindo um *form*, como mostrado o resultado na Figura 7 e seu código fonte na Figura 8. Foi criado também um segundo arquivo com a geração de uma tabela, Figura 9 e o código gerado pelo terminal na Figura 10.



**Figura 7:** Arquivo de um formulário gerado pela ferramenta.  
Fonte: Criado pelo autor.

```
<html lang='pt-br' id='html1'><head>
<meta charset='utf-8'>
<link href='../css/style.css' rel='stylesheet'>
<title>Cadastro</title></head>
<header id='header' class='header'>Cadastro</header>

<footer class='footer' id='footer' class='footer'>By:Fernando</footer>

<div class='sidenav' id='menu' class='menu'><a href='#'>MENU</a>
<a id='11' class='11' href='index.html'>Home</a>

<a id='12' class='12' href='cadastro.html'>Cadastro</a>
</div>

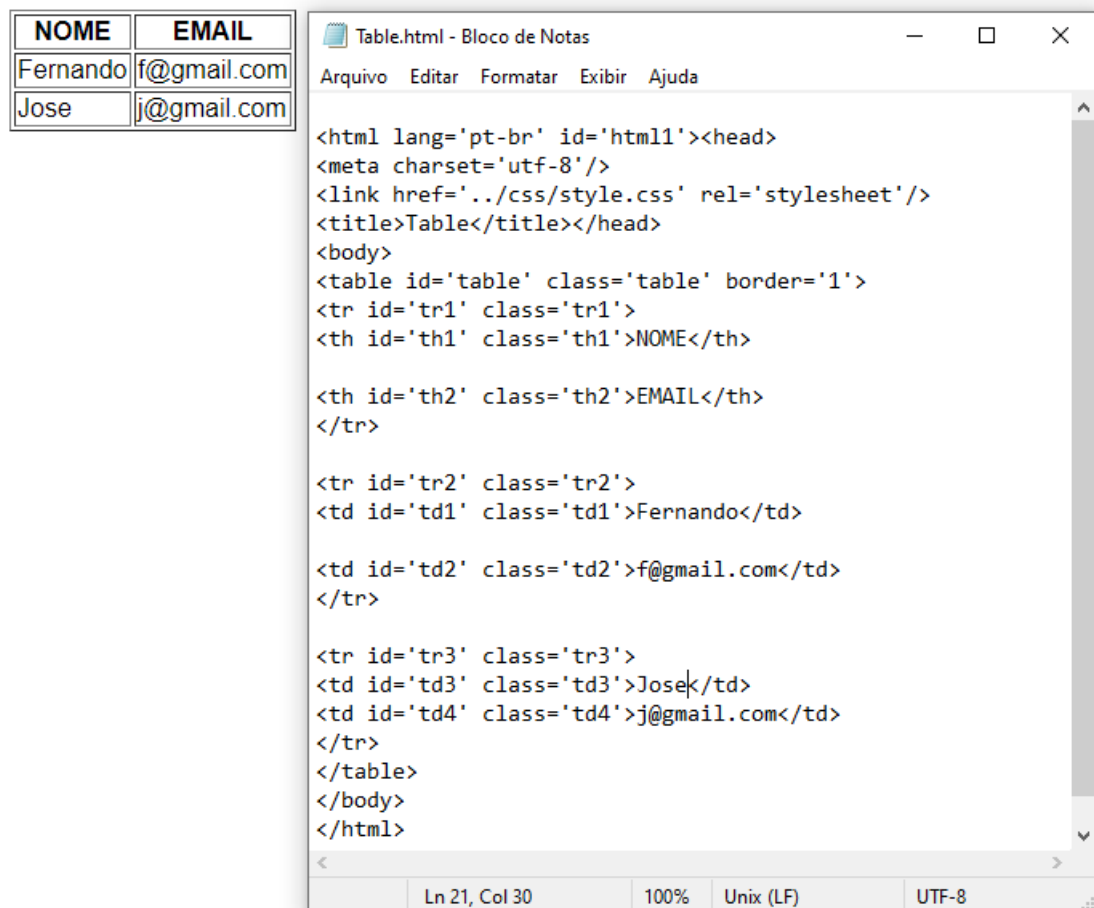
<body>
<form id='form' class='form' action='cadastro.html' method='post'>
<input id='nome' class='nome' type='text' value='Fernando' name='nome'>

<input id='email' class='email' type='text' value='f@gmail.com' name='email'>

<button id='button' class='button' type='submit'>Enviar</button>
</form>
</body>
</html>

Arquivo Cadastro.html gerado com sucesso
```

**Figura 8:** Código de um formulário gerado pela ferramenta.  
Fonte: Criado pelo autor.



**Figura 9:** Arquivo de uma tabela gerado pela ferramenta.

Fonte: Criado pelo autor.

```
<html lang='pt-br' id='html1'><head>
<meta charset='utf-8' />
<link href='../css/style.css' rel='stylesheet' />
<title>Table</title></head>
<body>
<table id='table' class='table' border='1'>
<tr id='tr1' class='tr1'>
<th id='th1' class='th1'>NOME</th>

<th id='th2' class='th2'>EMAIL</th>
</tr>

<tr id='tr2' class='tr2'>
<td id='td1' class='td1'>Fernando</td>

<td id='td2' class='td2'>f@gmail.com</td>
</tr>

<tr id='tr3' class='tr3'>
<td id='td3' class='td3'>José</td>

<td id='td4' class='td4'>j@gmail.com</td>
</tr>
</table>
</body>
</html>
```

Arquivo Table.html gerado com sucesso

**Figura 10:** Código de uma tabela gerado pela ferramenta.  
Fonte: Criado pelo autor.

## 4 *Conclusão*

A ferramenta mostrou muita confiabilidade para a criação de páginas *html* por meios interativos com o usuário, ou seja, sem o usuário precisar de entender sobre programação.

Sem as estruturas de padrão de projeto, não teria como desenvolver a ferramenta em um espaço de tempo tão curto, também seria mais complicado realizar algumas operações de composição de objetos sem a utilização da estrutura do *Composite*.

O *Flyheight* possuiu um papel fundamental para a estrutura do projeto, muitas vezes, um sistema web depende de várias telas, e cada tela, pode repetir algumas características, como o *Header*, *Sidebar* e o *Footer*, logo a economia de recursos se torna muito satisfatória.

Portanto, é possível concluir que, antes de iniciar qualquer projeto, por mais simples que seja, sempre deve procurar por uma estrutura de padrão de projeto para facilitar e organizar o desenvolvimento da solução.

## ***Referências Bibliográficas***

DEVDOCS. **HTML**. 2020. Disponível em: <<https://devdocs.io/html/>>.