



Curso PHP

Do XLSX ao CMS (aula 11)

Variáveis globais



Variáveis globais

Com Twig, temos variáveis locais, que são criadas no contexto do *controller*, imediatamente na invocação da *view* e não podem ser acessadas de forma externa.

As variáveis globais, por outro lado, são comuns a todas as *views* e podem ser acessadas de qualquer ponto da *view*, sem que sejam definidas no *controller*, diminuindo a repetição de código.



Variáveis globais

O código ao lado mostra a forma de injetar variáveis globais nas *views*.

Como dito antes, as *views* apenas recebem os dados passados a elas, evitando ao máximo realizar processamentos e tomadas de decisão.

Não uma é regra, mas usamos a escrita em CAIXA ALTA para diferenciar variáveis de contexto local do contexto global.

```
// adiciona variáveis globais ao contexto da view
View::addGlobals([
    'APP_LOCALE' => str_replace('_', '-', env('APP_LOCALE')),
    'CURRENT_URI' => $request->getPathInfo(),
]);
```

Estilização semântica

Faça mais com menos.



Pico CSS

Um kit inicial minimalista e leve que prioriza a sintaxe semântica, tornando cada elemento HTML responsivo e elegante por padrão.

Escreva HTML, adicione o Pico CSS e o Voilà!

([Pico CSS](#))

😊 Pico CSS

```
<form>
  <input type="text">
  <button type="submit">Action</button>
</form>
```

😓 Utility CSS Framework

```
<div class="container display-flex my-md mx-sm">
  <form class="form shadow-md my-md mx-sm align-center">
    <div class="input-wrapper border-radius-sm">
      <input type="text" class="input text-color-gray placeholder-c
    </div>
    <div class="button-wrapper border-radius-sm">
      <button type="submit" class="button bg-color-blue text-color-
        Action
      </button>
    </div>
  </form>
</div>
```




Layout base

Tem base um trem desse?

—



Layout base

O layout base a parte comum de HTML a todas as “telas” do sistema.

Neste arquivo, será inserido todas as funcionalidades básicas, como CSS, navegação global e rodapé.

```
<!doctype html>
<html lang="{{ APP_LOCALE }}">

  <head>
    <meta charset="utf-8">
    <meta name="color-scheme" content="light dark">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <link rel="icon" href="/favicon.ico" type="image/x-icon">
    <link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/@picocss/pico@2/css/pico.min.css">
    <title>Plano de aulas</title>
  </head>

  <body>
    <header>
    </header>
    <main>
      <div class="container">
        {% block main %}{% endblock %}
      </div>
    </main>
    <footer>
      <div class="container">
        <p>&copy; 2025</p>
      </div>
    </footer>
  </body>

</html>
```



Twig: block

Ao usar a marcação **block**, nós definimos algo chamado de área reservada.

Não é regra, mas nomear os blocos de forma semântica é mais fácil de associar do que dá nomes que nos fazem pensar mais.

Usaremos **main** como área reservada.

Nesta área, podemos injetar o conteúdo a partir de outro lugar, o que dá sentido ao uso de um layout base.

```
{% extends "base.twig" %}

{% block main %}
<h1>Agendamentos</h1>
<ul>
    {% for agendamento in agendamentos %}
    <li>
        <a href="/agendamentos/{{ agendamento.id }}">#{{ agendamento.id }}</a>: {{ agendamento.data }}
    </li>
    {% endfor %}
</ul>
{% endblock %}
```

Usando template

Páginas de erro

A meme featuring Woody and Buzz Lightyear from the movie Toy Story. Woody is on the left, looking concerned. Buzz is on the right, looking excited and pointing his finger. The background is a simple room with a door and a window.

ERRORS

ERROR EVERYWHERE

makeameme.org



Páginas de erro

Atualmente, nosso sistema lança uma *exception* para erros como 404 e 501.

Contudo, em ambiente de produção, isto não é recomendado, pois mostra ao usuário (principalmente ao mal intencionado) informações sobre o sistema.

Iremos tratar estes erros depois, por enquanto, apenas uma apresentação diferente será mostrada.

```
$this->response->setStatusCode(Response::HTTP_NOT_IMPLEMENTED);

return View::render('erro_501', [
    'controller' => $controller,
    'method' => $method,
]);

{% extends "base.twig" %}

{% block main %}
<hgroup>
    <h3>Erro 501: Não implementado</h3>
    <p>URI: <mark>{{ CURRENT_URI }}</mark></p>
</hgroup>
<p><kbd>{{ controller }}@{{ method }}</kbd></p>
{% endblock %}
```


Plano de aulas

Período: --

[Agendamentos](#) [Períodos](#) [Disciplinas](#) [Atividades](#)

Erro 501: Não implementado

URI: /agendamentos

```
App\Controllers\AgendamentoController@metodoNaoExiste
```

© 2025

Exemplo de erro 501

Listando dados



Listagem: scope

Nossa regra de negócio para exibição de agendamentos diz:

- A partir de hoje
- Ordenação crescente por data

Graças a funcionalidade de *escopo do Eloquent*, podemos criar um método para tornar esta regra menos repetitiva.

```
public function scopePrevistos(Builder $query): void
{
    $query->whereDate('data', '>=', today());
}

$agendamentos = Agendamento::query()
    ->previstos()
    ->oldest('data')
    ->with('atividade', 'disciplina')
    ->get();

return View::render('agendamentos/index', [
    'agendamentos' => $agendamentos,
]);
```



Listagem: view

Agora que temos nossos dados, precisamos mostra-los na tela.

Porém, caso não haja dados, também precisamos mostrar a informação sobre isso na tela, para que o usuário saiba o que está acontecendo.

Usando o bloco `{% else %}` dentro do bloco `{% for %}`, criamos uma verificação de array vazio facilmente.

```
{% for agendamento in agendamentos %}
<tr style="vertical-align: top">
  <td>
    {{ agendamento.data|date('d/m') }}<br />
    <strong>{{ agendamento.disciplina.nome|upper }}</strong><br />
    <small>{{ agendamento.atividade.nome }}</small>
  </td>
  <td>{{ agendamento.conteudo }}</td>
</tr>
{% else %}
<tr>
  <td colspan="2">Não há agendamentos previstos</td>
</tr>
{% endfor %}
```

Paginando dados

composer require illuminate/pagination

Instala a dependência



Paginação: setup

Para que o Eloquent, nosso ORM, enxergue o parâmetro na URL que indica a paginação e também a URI (path) atual, precisamos entregar estes dados a ele.

```
// paginação
Illuminate\Pagination\Paginator::currentPageResolver(
    fn ($pageName) => $request->get($pageName)
);

// paginação
Illuminate\Pagination\Paginator::currentPathResolver(
    fn () => $request->getPathInfo()
);
```



Paginação

Para criar a paginação de dados com Eloquent, usa-se o método **paginate**.

De forma automática, o Eloquent implementa:

- Condições *limit* e *offset*
- Quantidade total de registros
- Links para navegação de paginas
- Indicação de página atual

Agora, ao invés de passar a variável **\$agendamentos** para a *view*, passamos todo o paginador, pois ela contém todas as variáveis que precisamos.

```
$data = Agendamento::query()
    ->previstos()
    ->oldest('data')
    ->with('atividade', 'disciplina')
    ->paginate(5)
    ->toArray();

return View::render('agendamentos/index', $data);
```




Paginação: result

Resultado de um objeto de paginação.

```
array (size=13)
  'current_page' => int 1
  'data' =>
    array (size=0)
      empty
  'first_page_url' => string '/agendamentos?page=1' (length=20)
  'from' => int 1
  'last_page' => int 2
  'last_page_url' => string '/agendamentos?page=2' (length=20)
  'links' =>
    array (size=4)
      0 =>
        array (size=3)
          'url' => null
          'label' => string 'Previous' (length=8)
          'active' => boolean false
      1 =>
        array (size=3)
          'url' => string '/agendamentos?page=1' (length=20)
          'label' => string '1' (length=1)
          'active' => boolean true
      2 =>
        array (size=3)
          'url' => string '/agendamentos?page=2' (length=20)
          'label' => string '2' (length=1)
          'active' => boolean false
      3 =>
        array (size=3)
          'url' => string '/agendamentos?page=2' (length=20)
          'label' => string 'Next' (length=4)
          'active' => boolean false
  'next_page_url' => string '/agendamentos?page=2' (length=20)
  'path' => string '/agendamentos' (length=13)
  'per_page' => int 5
  'prev_page_url' => null
  'to' => int 5
  'total' => int 9
```

Filtrando dados



Pesquisa

A pesquisa de dados ou filtros, nos permite, como o nome sugere, filtrar dados para encontrar o que queremos de forma mais fácil.

Esta tarefa gera muitas condicionais e como *controllers* devem ser métodos simples, passaremos a responsabilidade disso para o *model* – responsável pela consulta à base de dados.

```
$data = Agendamento::toSearch([
    'atividade_id' => $request->get('atividade_id'),
    'disciplina_id' => $request->get('disciplina_id'),
]);

return View::render('agendamentos/index', $data);
```

Amostra final

Agora, nossa tela possui todos os componentes que precisamos:

- Listagem
- Paginaçao
- Filtros

Plano de aulas

Período: --

[Agendamentos](#) [Períodos](#) [Disciplinas](#) [Atividades](#)

Agendamentos

Existem 10 agendamentos previstos

Atividade ▾ Disciplina ▾ [Filtrar](#)

Atividade	Conteúdo
09/02 OMNIS Prova	Aut in vel temporibus saepe aperiam vero voluptatem. Facere exercitationem ducimus quibusdam. Saepe id fuga aut dolorum corrupti. Quia necessitatibus qui quos vel ad vero ut. Est ut a et sunt. Ut eum dolorem labore. Veritatis quasi autem tempora iure. Voluptatem velit id eius quia molestias. Quidem iure veritatis cum consequatur magni unde eaque. Quasi veniam et aut.
11/02 CONSEQUUNTUR Debate	Est et voluptatem illo quam omnis laudantium quo. Voluptatum et dignissimos cupiditate ducimus. Dolor ut voluptas molestiae maxime voluptatum dolores assumenda. Enim culpa nam dignissimos animi delectus et maxime. Et corporis ea ea tenetur. Voluptatem adipisci numquam ab et. Debitis magni odio soluta blanditiis. Est qui asperiores a doloremque officia. Omnis placeat enim tempora voluptatem rem est qui.
16/02 OMNIS Prova	Consectetur consequuntur non modi magni. Esse minima nemo accusamus enim perferendis cumque omnis explicabo. Fugiat officis aut sit voluptatem ut fugiat et quam. Impedit fugit vitae labore odio. Omnis dignissimos quia consequatur sequi sunt hic. Sequi est cum consequatur consequatur omnis voluptatibus ut. Voluptatibus velit corrupti cumque odio. Enim doloribus aliquam minus adipisci dolorum error. Qui sunt veritatis tenetur voluptatem voluptatum inventore. Soluta velit molestiae ad illum non.
01/03 CONSEQUUNTUR Debate	Placeat minima est unde et. Aut sint est at illo. Aliquid ut nulla qui magnam. Iusto non sit aut consequuntur ut consequatur ea ipsa. Quasi architecto cum quasi. Corporis aliquid error ratione aut aut. Ut iusto odit vel ullam sunt. Voluptatem repudiandae hic vel voluptate sunt. Nulla enim soluta sunt nisi. Voluptatum rerum laborum quod. Maiores nihil excepturi doloribus vero exercitationem dignissimos. Rem et nisi voluptatibus magnam. Maiores ex nobis sit.
03/03 OMNIS Seminário	Labore saepe architecto quos maxime officia nemo repellendus. Eveniet suscipit aut et laborum eum commodi rerum. Non placeat cum nobis dolores. Laboriosam id aut delectus. Accusamus est voluptas in explicabo accusamus sed. Possimus culpa quaerat eos magni mollitia. Vel repellat rem officia dolore hic. Ut qui voluptatem nobis sed ipsa eum. Dolorem occaecati sed incidunt magni incidunt nobis excepturi. Nemo perspiciatis sapiente architecto aut qui. Corrupti voluptates libero soluta voluptate.

Anterior 1 2 Próxima

**Como fazer o mesmo para as
outras áreas?**



Já ouviu falar da iniciativa *Twig Components*?