Curso PHP

Do XLSX ao CMS (aula 12)

Dividir para conquistar...

Twig Components
Macros

Infelizmente, a opção de Twig Components está disponível somente no framework Symfony, mas podemos usar macros.

Macros

Macros são comparáveis com funções em linguagens de programação regulares.

Eles são úteis para reutilizar fragmentos de modelos para não se repetir.

(Twig)

```
{% macro pagination(links) %}
<nav>
    <span aria-hidden="true"><!-- placeholder --></span>
    <div>
        {% for link in links %}
        {% set label = loop.first ? '&larr; Anterior' : (loop.last ? 'Próxima &rarr;' : link.label) %}
        <a {{ attr({
            href: link.url ?: 'javascript:;',
            class: link.active ? '' : 'outline',
            role: 'button',
            disabled: link.active or not link.url,
        })|raw }}>{{ label|raw }}</a>
        {% endfor %}
    </div>
    <span aria-hidden="true"><!-- placeholder --></span>
</nav>
{% endmacro %}
```

```
{% from "components.twig" import pagination %}

{{ pagination(links) }}
```

Agora, é hora daquele CRUD básico.

Criação

Criação

Para centralizar nosso formulário, iremos criar um arquivo que possa ser reaproveitado: **fieldset.twig**.

```
{% from "components.twig" import select %}

<fieldset class="grid">
    <label>
        Atividade
        {{ select('Atividade', 'atividade_id', atividades) }}
    </label>
    <label>
        Disciplina
        {{ select('Disciplina', 'disciplina_id', disciplinas) }}
    </label>
    <label>
        Data
        <input type="date" name="data" value="">
    </label>
</fieldset>
<fieldset class="grid">
    <label>
        Conteúdo
        <textarea name="conteudo" rows="15"></textarea>
    </label>
</fieldset>
```

Criação: UI

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

{% block main %}
<hgroup role="heading">
    <h3>Cadastrar agendamento</h3>
</hgroup>
<nav>
    <ul>
        <li>
            <a href="{{ url('/agendamentos') }}">&larr; Voltar para listagem</a>
        </li>
    </ul>
</nav>
<form action="{{ url('/agendamentos') }}" method="post">
    {% include "agendamentos/fieldset.twig" %}
    <input type="submit" value="&#128190; Cadastrar">
</form>
{% endblock %}
```

Criação: controller

```
public function cadastrar()
{
    $atividades = Atividade::toOptGroup();
    $disciplinas = Disciplina::toOptGroup();

    return View::render('agendamentos/cadastrar', compact('atividades', 'disciplinas'));
}

public function salvar(Request $request)
{
    $agendamento = Agendamento::create([
        'atividade_id' => $request->get('atividade_id'),
        'disciplina_id' => $request->get('disciplina_id'),
        'conteudo' => $request->get('conteudo'),
        'data' => $request->get('data'),
    ]);

    return new RedirectResponse('/agendamentos');
}
```

Edição

Edição

A edição, é similar ao processo de criação, porém, precisamos injetar na *view* os dados do m*odel* providos pelo *controller*.

Lembre que também usamos esta *view* para criação, ou seja, não existe um *model* para fornecer estes dados.

Assim, precisamos prepara-la de forma condicional usando [null coalescing operator](#).

```
{% from "components.twig" import select %}

<fieldset class="grid">
    <label>
        Atividade
        {{ select('Atividade', 'atividade_id', atividades, agendamento.atividade_id ?? '') }}
    </label>
    <label>
        Disciplina
        {{ select('Disciplina', 'disciplina_id', disciplinas, agendamento.disciplina_id ?? '') }}
    </label>
    <label>
        Data
        <input type="date" name="data" value="{{ (agendamento.data ?? '')|date('Y-m-d') }}">
    </label>
</fieldset>
<fieldset class="grid">
    <label>
        Conteúdo
        <textarea name="conteudo" rows="15">{{ agendamento.conteudo ?? '' }}</textarea>
    </label>
</fieldset>
```

Edição: UI

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

{% block main %}
<hgroup role="heading">
    <h3>Editar agendamento</h3>
</hgroup>
<nav>
    <ul>
        <li>
            <a href="{{ url('/agendamentos') }}">&larr; Voltar para listagem</a>
        </li>
    </ul>
</nav>
<form action="{{ url('/agendamentos/{id}', {id: agendamento.id}) }}" method="post">
    {% include "agendamentos/fieldset.twig" %}
    <input type="hidden" name="_method" value="put">
    <input type="submit" value="&#128190; Salvar">
</form>
{% endblock %}
```

É POST ou PUT?

Os formulário HTML suportam somente os métodos GET e POST.

Contudo, para que haja semântica e “preparar o terreno” para implementação do padrão REST, precisamos habilitar a sobreescrita de método HTTP.

```php
// captura requisição
$request = Request::createFromGlobals();

// habilita reescrita de método (PUT, PATCH, DELETE)
$request->enableHttpMethodParameterOverride();
```

Edição: controller

```php
public function editar(Agendamento $agendamento)
{
    $atividades = Atividade::toOptGroup();
    $disciplinas = Disciplina::toOptGroup();

    return View::render('agendamentos/editar', compact('atividades', 'disciplinas', 'agendamento'));
}

public function atualizar(Request $request, Agendamento $agendamento)
{
    $agendamento->update([
        'atividade_id' => $request->get('atividade_id'),
        'disciplina_id' => $request->get('disciplina_id'),
        'conteudo' => $request->get('conteudo'),
        'data' => $request->get('data'),
    ]);

    return new RedirectResponse('/agendamentos');
}
```

Exclusão

Exclusão

A exclusão é realizada por meio do envio de uma requisição HTTP DELETE.

Antes de enviar a requisição, interpelamos o usuário se ele deseja de fato excluir o item.

```
<form action="{{ url('/agendamentos/{id}', {id: agendamento.id}) }}" method="post">
    <input type="hidden" name="_method" value="delete">
    <a href="javascript:;"
       onclick="if (confirm('Deseja excluir este item?')) this.closest('form').submit()">
        <small>&#128465; Excluir</small>
    </a>
</form>
```

Exclusão: controller

```
public function excluir(Agendamento $agendamento)
{
    $agendamento->delete();

    return new RedirectResponse('/agendamentos');
}
```

Agora, temos um CRUD completo...

...mas e se o usuário inserir uma dado que não existe?
E se a data não for de fato uma data?
E se...?

Validar dados é o assunto para nossa próxima aula.