Programação de Sistemas para Internet

Prof. Diego Cirilo

Aula 12: Mensagens/Paginação

Mensagens

- O Django possui uma ferramenta para notificações ou mensagens para o usuário;
- Essas mensagens servem de *feedback* sobre ações no sistema;
- Ex. erros nas ações, etc;
- Esse tipo de mensagem também é chamado de *toast* ou *flash message*.

Mensagens

- Existem 5 tipos de mensagem padrão:
  - `debug`, `info`, `success`, `warning` e `error`
- Devemos importar `from django.contrib import messages` nas *views*
- Para enviar uma notificação:

```
messages.success(request, 'A ação foi realizada com sucesso!')

messages.error(request, 'Houve um erro na solicitação.')
```

Mensagens nos templates

- O Django se responsabiliza apenas pelo *back-end* das mensagens
- A *interface* é responsabilidade do *front-end*
- Para que as mensagens sejam exibidas para os usuários, devemos adicioná-las ao template:

```
{% if messages %}
    <div>
        {% for message in messages %}
            <div class="alert alert-{{ message.tags }}">
                {{ message }}
            </div>
        {% endfor %}
    </div>
{% endif %}
```


- O `message.tags` retorna a tipo da mensagem

Mensagens nos templates

- É necessário implementar com CSS/JS funcionalidades extras, como fechar a notificação;
- No *Bootstrap* podemos usar o `alert` e `toast`;
- Existem também bibliotecas CSS/JS específicas;
- Ex. Toastr

Paginação

- Utilizamos a paginação para dividir os resultados de uma *query* ao BD;
- Muitos dados de uma só vez:
  - Podem deixar o sistema lento
  - Dificultam a visualização

Paginação do Django

- O Django possui uma ferramenta para paginação;
- É necessário apenas implementar o *front-end* de navegação;
- Para usar importamos a classe *Paginator*;
- `from django.core.paginator import Paginator`;
- Podemos passar o número da página desejada na URL como uma *querystring*;
- Ex. `/posts?pagina=1`.

Exemplo

- `views.py`

```
from django.shortcuts import render
from .models import Post
from django.core.paginator import Paginator

def post_list(request):
    posts = Post.objects.all()  # Pega todos os posts
    paginator = Paginator(posts, 10)  # Separa em páginas de 10 posts
    numero_da_pagina = request.GET.get('pagina')  # Pega o número da página da URL
    posts_paginados = paginator.get_page(numero_da_pagina)  # Pega a página específica
    context = {
        'posts': posts_paginados,
    }
    return render(request, 'post_list.html', context)
```

Exemplo

- No template

```
<ul>
    {% for post in posts %}
        <li>{{ post.title }}</li>
    {% endfor %}
</ul>

<!-- Navegação da paginação -->
<div class="pagination">
    <span class="step-links">
        {% if posts.has_previous %}
            <a href="?pagina=1">&laquo; Início</a>
            <a href="?pagina={{ posts.previous_page_number }}">anterior</a>
        {% endif %}

        <span class="current">
            Página {{ posts.number }} de {{ posts.paginator.num_pages }}.
        </span>

        {% if posts.has_next %}
            <a href="?pagina={{ posts.next_page_number }}">próxima</a>
            <a href="?pagina={{ posts.paginator.num_pages }}">última &raquo;</a>
        {% endif %}
    </span>
</div>
```

Dúvidas?