Django Framework do Zero

Como Criar um Aplicativo Web

*Canal **youtube.com/@VamosCodar** - Wanderson Reis*

31 - Usar Níveis de Acesso

Requisitos

1) Aulas 01 a 30 - Django Framework do Zero - Playlist: https://www.youtube.com/playlist?list=PLFOqHo8oIjzewcT23HCxJV0xWO451CTJe

Introdução

Nesta aula vamos aprender a usar os níveis de acesso nativos do Django. O django implementa por padrão dois níveis de acessos básicos: o superusuário e o staff (membro de equipe) que são papéis que podem acessar a área administrativa.

Nesta aula:

1) Implementar um decorator de classe
2) Aplicar o decorator para limitar o acesso

Observação: Lembre-se de estar com o ambiente virtual ativado (venv).

Passo 1: Implementar um decorator de classe

Criar um novo arquivo em **django-do-zero/webapp** com o nome **decorators.py.** O objetivo é manter os decorators comuns ao sistemas (que modificam o comportamento das funções) dentro da pasta raiz do projeto (webapp)**.** O conteúdo do novo arquivo deve ser conforme indicado na imagem abaixo:

webapp > decorators.py > ...

```python
from django.utils.decorators import method_decorator


def class_view_decorator(function_decorator):
    def decorator(view_object):
        view_object.dispatch = method_decorator(function_decorator)(view_object.dispatch)
        return view_object

    return decorator
```

(**salve o arquivo**).

Passo 2: Implementar a view para exibir o formulário

Editar o arquivo **django-do-zero/polls/views.py** e modifique conforme indicado nas próximas imagens:

Na parte superior do arquivo

polls > views.py > ...

```python
from django.http import HttpResponse
from django.contrib.auth.decorators import login_required
1 from django.contrib import messages
from django.contrib.admin.views.decorators import staff_member_required

from polls.models import Question, Choice
2 from polls.forms import QuestionForm, QuestionImportForm
from webapp.decorators import class_view_decorator
```

Decorar a classe QuestionCreateView conforme ilustrado abaixo

polls > views.py > ...

Acrescentar o decorator

```python
@class_view_decorator(staff_member_required)
class QuestionCreateView(LoginRequiredMixin, CreateView): # view baseada em classe, usa herança
    model = Question # vincula o model a view para gerar o form HTML
    template_name = 'polls/question_form.html' # template que monta o form
    fields = ('question_text', 'pub_date', 'categoria', ) # campos que estarão disponíveis no form
    success_url = reverse_lazy('index') # URL para redirecionado em caso de sucesso
    success_message = 'Pergunta criada com sucesso.'

    # implementa o método que conclui a ação com sucesso (dentro da classe)
    def form_valid(self, form):
        form.instance.author = self.request.user # usuário logado
        messages.success(self.request, self.success_message)
        return super(QuestionCreateView, self).form_valid(form)
```

(**salve o arquivo**).

O decorator de classe vai aplicar à view o comportamento "**staff_member_required**" que que também é decorator que exige que o usuário logado na sessão tem o perfil de staff (membro de equipe).

Em seguida suba o servidor de desenvolvimento.

python manage.py runserver **(ENTER)**

Para testar, acesse a função de criar uma nova enquete e tente criar uma nova. Se necessário, acesse a área administrativa com um super usuário para atribuir o perfil (papel) de staff a um outro usuário que terá permissão / acesso para criar novas enquetes. Consulte a aula em vídeo para mais detalhes.