Django Framework do Zero

Como Criar um Aplicativo Web

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

10 - CRUD Básico - CREATE

Requisitos

1) Aulas 01 a 09 - Django Framework do Zero - Playlist: https://www.youtube.com/playlist?list=PLFOqHo8oIjzewcT23HCxJV0xWO451CTJe

Introdução

CRUD é um acrônimo para Create, Read, Update e Delete. São as operações básicas que todo sistema que gerencia dados realiza. Com o Framework Django é possível criar estas operações de forma muito rápida usando o recurso das views genéricas (CreateView, ListView, DetailView, UpdateView e DeleteView). Na prática são Class-based views, isto é, são classes de views implementadas no formato de classes ao invés de usar o formato de função. As views genéricas são pré-modeladas e usadas como base para as suas próprias views. A principal vantagem é usar o recurso de herança (da Orientação à Objetos) para reduzir a quantidade de esforço na implementação. O essencial já vem pronto e já se conecta automaticamente com o Model (reconhecendo os campos: tipos, formatos e comportamentos). Nesta aula vamos implementar exemplos de views baseadas em funções e em classes.

Nesta aula vamos implementar o CREATE (criação de perguntas). Para explicitar e diferenciar as views baseadas em classe e em funções vamos criar duas views, desta forma será possível visualizar qual você prefere implementar. As views baseadas em classe são mais modernas e recomendadas se quiser se beneficiar mais da Orientação a Objetos do Django.

Programar qualquer operação CRUD basicamente se resume a quatro etapas: 1) Criar o form django que monta o form HTML (com base no Model ou personalizado), 2) Criar a view que processa a resposta, 3) Criar o template HTML que formata a resposta e 4) Definir uma rota que para receber a requisição.

Nesta aula:

1) Criar a classe QuestionForm
2) Implementar as views
3) Criar o template para exibir o formulário
4) Definir as rotas para views
5) Controle de acesso às rotas

Passo 1: Criar a classe QuestionForm

Vamos começar criando a classe **QuestionForm**() em um novo arquivo chamado **forms.py** dentro de polls/. Implementar o form é obrigatório se for usar view baseada em função, contudo será muito comum

precisar criar o form para personalizar a exibição (montagem) do formulário HTML mesmo se usar views baseado em classes.

Criar o arquivo **django-do-zero/polls/forms.py** e contendo o seguinte conteúdo:

```
# na parte superior do arquivo
from django import forms # importa a classe form do django
from polls.models import Question # importa Question de models.py

# na parte inferior do arquivo
# cria o ModelForm
class QuestionForm(forms.ModelForm):
    class Meta:
        model = Question # vincula o model ao form
        fields = ('question_text', 'pub_date') # campos a exibir no formulário
```

(**salve o arquivo**).

Passo 2: Implementar as views

Agora vamos implementar as views **QuestionCreateView**() e **question_create**(), ambas terão a responsabilidade de criar novas perguntas.

Editar o arquivo **django-do-zero/polls/views.py** e acrescente a view baseada em classe da seguinte forma (**atenção apenas ao conteúdo novo**):

```
# na parte superior do arquivo
from django.shortcuts import render, redirect
from django.views.generic.edit import CreateView
from django.urls import reverse_lazy

from polls.models import Question
from polls.forms import QuestionForm # importa a classe QuestionForm

# na parte inferior do arquivo
# view baseada em classe (genérica)
class QuestionCreateView(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', ) # 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):
```

```
        messages.success(self.request, self.success_message)
        return super(QuestionCreateView, self).form_valid(form)


# view baseada em função
def question_create(request):
    context = {} # dados para o contexto do form
    # cria o objeto form
    form = QuestionForm(request.POST or None, request.FILES or None)
    context['form'] = form

    if request.method == "POST": # processa se o método de requisição for POST
        if form.is_valid(): # verifica se o form é válido
            form.save() # salva os dados do form no modelo
            messages.success(request, 'Pergunta criada com sucesso.')
            return redirect("index") # redireciona em caso de sucesso

    return render(request, 'polls/question_form.html', context) # carrega o form no template
```

(**salve o arquivo**).

Passo 3: Criar o template para exibir o formulário

O template HTML seguirá a convenção de nome do django e será utilizado para ambas views criados no passo anterior.

Crie o arquivo de template chamado **question_form.html** dentro da pasta **django-do-zero/templates/polls/** com uma estrutura similar ao exemplo abaixo:

Arquivo templates/polls/question_form.html:

```
{% extends '_layout1.html' %} {# herda tudo que já tem em _layout1.html #}

{% block content %} {# substitui  o bloco content em _layout1.html #}
<div class="card">
    <div class="card-header">
        <h5 class="card-title">Criando uma Pergunta</h5>
    </div>
    <div class="card-body">
        <form method="post">
            {% csrf_token %} {# obrigatório para segurança do formulário #}
            {{ form.as_p }} {# imprime o objeto form formato em tags HMTL #}
            <button type="submit" class="btn btn-success">Salvar</button>
        </form>
    </div>
</div>
{% endblock %} {# fecha o bloco content #}
```

(**salve o arquivo**).

Passo 4: Definir as rotas para views

Edite o arquivo **django-zero/polls/urls.py** e acrescente a nova rota seguindo o exemplo abaixo (**atenção apenas ao conteúdo novo**):

```
# na parte superior do arquivo
from django.urls import path
from polls.views import index, ola, QuestionCreateView, question_create

urlpatterns = [
    path('index/', index, name="index"),
    path('ola/', ola, name="ola"),
    path('enquete/add', QuestionCreateView.as_view(), name="poll_add"),
    path('pergunta/create', question_create, name="poll_create"),
]
```

(**salve o arquivo**).

Para completar vamos incluir o link para acesso rápido para criar novas perguntas (enquetes).

Edite o template **index.html** dentro da pasta **django-zero/polls/templates** e incluir dentro do bock “content” uma tag similar ao exemplo abaixo:

```
<a href="{% url 'poll_add' %}" class="btn btn-lg btn-primary font-weight-bold">
  + Pergunta
</a>
```

(**salve o arquivo**).

Em seguida suba o servidor de desenvolvimento e acesse as novas rotas diretamente ou pelo link incluido (que aparece na rota /ola).

python manage.py runserver (ENTER)

No navegador acesse:

http://127.0.0.1:8000/enquete/add

e/ou

http://127.0.0.1:8000/pergunta/create

O resultado será o mesmo, a diferença é somente no nome da rota e na view que processa a requisição.

Passo 5: Controle de acesso às rotas

Existem diversas formas de implementar o controle de acesso. Se for realizada diretamente nas views, temos dois formatos básicos: 1) view baseada em funções, 2) Views baseadas em classe (inclui as views genéricas que utilizamos no CRUD). Vamos implementar os dois exemplos:

Edite o arquivo **django-do-zero/polls/views.py** e inclua as modificações conforme o exemplo a seguir (**atenção apenas ao conteúdo novo**):

```
# (...manter tudo o que já existe…)
# na parte superior do arquivo
from django.contrib.auth.decorators import login_required
from django.contrib.auth.mixins import LoginRequiredMixin

# localizar as views e modificá-las
@login_required # controle de acesso usando o decorador de função
def question_create(request):
    # (...manter tudo o que já existe…)

# controle de acesso usando o mecanismo de herança que torna o login obrigatório
class QuestionCreateView(LoginRequiredMixin, CreateView):
    model = Question
    # (...manter tudo o que já existe…)
```

(**salve o arquivo**).

Feita as alterações acima, as rotas para criação de novas perguntas / enquetes serão acessíveis somente para usuários logados. Acesse as novas rotas para testar.