Urls, Views (class), Templates

Yuri Kaszubowski Lopes

Especialização em Tecnologia Python para Negócios

Atualize models.py

server/seriados/models.py

```
1 from django.db import models
2 from django.conf import settings
3 from django.utils.translation import gettext_lazy as _
4
5 class Serie(models.Model):
6     nome = models.CharField(max_length=70)
7
8     def __str__(self):
9         return self.nome
10
11
12 class Temporada(models.Model):
13     numero = models.IntegerField()
14     serie = models.ForeignKey(Serie, on_delete=models.CASCADE)
15
16     def __str__(self):
17         return f"{self.serie.nome}: {self.numero}"
```

- Adicionamos o método `__str__` em **`class`** `Temporada`

Atualize models.py

server/seriados/models.py

```
1  class Episodio(models.Model):
2      data = models.DateField()
3      titulo = models.CharField(max_length=200, verbose_name="Título")
4      temporada = models.ForeignKey(Temporada, on_delete=models.CASCADE)
5
6      def __str__(self):
7          nome_serie = self.temporada.serie.nome
8          return f"{nome_serie} - {self.temporada.numero}: {self.titulo}"
9
10     def get_absolute_url(self):
11         from django.urls import reverse
12         return reverse('seriados:episodio_detalhes', kwargs={'pk' : self.pk})
13
14     def eh_antigo(self):
15         import datetime
16         if self.data < datetime.date(2000, 1, 1):
17             return True
18         return False
```

- na **class** `Episodio`:
  - Adicionamos o argumento `verbose_name` no campo `titulo`
  - Reescrevemos o método `__str__`

Atualize models.py

server/seriados/models.py

```
1  class Revisor(models.Model):
2      user = models.ForeignKey(settings.AUTH_USER_MODEL, on_delete=models.CASCADE)
3      reviews_episodios = models.ManyToManyField(Episodio, through='ReviewEpisodio')
4
5
6  class ReviewEpisodio(models.Model):
7      NOTA_A = 'A'
8      NOTA_B = 'B'
9      NOTA_C = 'C'
10     NOTAS_CHOICES = [
11         (NOTA_A, _("Excelente")),
12         (NOTA_B, _("Bom")),
13         (NOTA_C, _("Ruim")),
14     ]
15     episodio = models.ForeignKey(Episodio, on_delete=models.CASCADE)
16     revisor = models.ForeignKey(Revisor, on_delete=models.CASCADE)
17     nota = models.CharField(
18         max_length=1,
19         choices=NOTAS_CHOICES,
20         default = NOTA_B
21     )
```

- Nenhuma alteração nos modelos Revisor e ReviewEpisodio

URL

server/seriados/urls.py

```
1 from django.urls import path, re_path, include, register_converter
2
3 from . import views
4
5 app_name = 'seriados'
6
7 urlpatterns = [
8     path('series/', views.series_list, name='series_list'),
9     path('series/<int:pk>/', views.series_details, name='series_details'),
10
11    path('episodios/', views.episodio_list, name='episodio_list'),
12    path('episodios/<int:pk>/', views.episodio_details, name='episodio_details'),
13
14    path('episodios/nota/<str:nota>/', views.episodio_nota_list,
15        name='episodio_nota_list'),
16 ]
```

- Reescrevemos as **todas** as urls e adicionamos novas

Views: preparando dados

- O código abaixo prepara dados e metadados para **templates genéricos**

server/seriados/views.py

```
1  from django.http import HttpResponse
2  from django.shortcuts import render, get_object_or_404
3  from django.forms.models import model_to_dict
4
5  from .models import Serie, Temporada, Episodio
6
7  def prepare_data_list(objects, fields_name):
8      labels = list()
9      for field_name in fields_name:
10         field = objects.model._meta.get_field(field_name)
11         labels.append(field.verbose_name)
12
13     rows = list()
14     for _object in objects:
15         row = dict()
16         rows.append(row)
17         row['pk'] = _object.pk
18         row['data'] = list()
19         for field_name in fields_name:
20             row['data'].append(getattr(_object, field_name))
21
22     return labels, rows
```

Views: preparando dados

server/seriados/views.py

```
1 def prepare_data_detail(_object, fields_name):
2     data = model_to_dict(_object)
3     rows = list()
4     for field_name in fields_name:
5         field = _object._meta.get_field(field_name)
6         rows.append({'label': field.verbose_name, 'value': data[field_name]})
7     return rows
```

Views: usando templates genéricos

server/seriados/views.py

```
1  def series_list(request):
2      objects = Serie.objects.all()
3      labels, rows = prepare_data_list(objects, ['nome'])
4      context = {
5          'title': "Series",
6          'labels': labels,
7          'rows': rows,
8          'detail_url': 'seriados:series_details',
9          }
10     return render(request, 'list.html', context)
```

- Note que renomeamos as funções e templates das `views`
- Além dos dados passamos informações úteis para construir o template:
  - title, labels, detail_url

Template genérico

server/seriados/templates/list.html

```
1 {% extends "base.html" %}
2
3 {% block content %}
4 <h1>Lista de {{title}}</h1>
5
6 <table border="1px">
7     <tr>
8     {% for label in labels %}
9         <th>{{ label }}</th>
10    {% endfor %}
11    <th>Detalhes</th>
12    </tr>
13
14    {% for row in rows %}
15        <tr>
16        {% for value in row.data %}
17            <td>{{ value }}</td>
18        {% endfor %}
19        <td><a href={% url detail_url pk=row.pk %}>Ver...</a></td>
20        </tr>
21    {% endfor %}
22 </table>
23 {% endblock %}
```

Views: usando templates genéricos

server/seriados/views.py

```
1 def series_details(request, pk):
2     _object = get_object_or_404(Serie, pk=pk)
3     context = {
4         'title': "Serie",
5         'data': prepare_data_detail(_object, ['nome']),
6     }
7     return render(request, 'details.html', context)
```

Template genérico

server/seriados/templates/details.html

```
1 {% extends "base.html" %}
2
3 {% block content %}
4 <h1>Detalhes de {{title}}</h1>
5
6 {% for row in data %}
7     <p><b>{{ row.label }}</b>: {{ row.value }}</p>
8 {% endfor %}
9 {% endblock %}
```

Views: usando templates genéricos

server/seriados/views.py

```
1  def episodio_list(request):
2      objects = Episodio.objects.all()
3      labels, rows = prepare_data_list(objects, ['titulo', 'data'])
4      context = {
5          'title': "Episódios",
6          'labels': labels,
7          'rows':rows,
8          'detail_url': 'seriados:episodio_detalhes',
9          }
10     return render(request, 'list.html', context)
11
12
13 def episodio_details(request, pk):
14     _object = get_object_or_404(Episodio, pk=pk)
15     context = {
16         'title': "Episódio",
17         'data': prepare_data_detail(_object, ['titulo', 'data', 'temporada']),
18     }
19     return render(request, 'details.html', context)
```

Formato de uma url

- Considere a url:
- `http://127.0.0.1:8000/seriados/episodios/?search=The`
- Temos as partes:
  - `http://`: Esquema
  - `127.0.0.1`: Domínio (esse pode ser em mais partes):
    - Em `www.exemplo.com.br`:
    - `www`: subdomínio
    - `exemplo`: domínio
    - `com.br`: domínio de nível superior (top level domain)
  - `8000`: Porta
  - `/seriados/episodios/`: caminho (path)
  - `?`: separador da string de consulta (QS)
  - `search=The`: parâmetro da QS
- Podemos ter vários parâmetros e também um fragmento:
  - `...?search=the&order=asc#partedois`

Formato de uma url

- Podemos fazer um casamento de padrão com partes do caminho (path) da URL
  - Atribuimos nomes para o casamento
  - São argumentos para a função da view

urls.py

```
1     path('episodios/<int:pk>/', views.episodio_details,
2         name='episodio_details'),
3
4     path('episodios/nota/<str:nota>/', views.episodio_nota_list,
5         name='episodio_nota_list'),
```

views.py

```
1     def episodio_details(request, pk):
2         ...
3
4     def episodio_nota_list(request, nota):
5         ...
```

- E os parâmetro da QS?

Usando parâmetro da QS para filtro

- E os parâmetro da QS?
  - Está no objeto request

server/seriados/views.py

```
1  def episodio_list(request):
2      search = request.GET.get('search', "")
3      objects = Episodio.objects.filter(titulo__contains=search)
4      labels, rows = prepare_data_list(objects, ['titulo', 'data'])
5      context = {
6          'title': "Episódios",
7          'labels': labels,
8          'rows':rows,
9          'detail_url': 'seriados:episodio_details',
10         }
11     return render(request, 'list.html', context)
```

- Acesse: `http://127.0.0.1:8000/seriados/episodios/?search=The`
- Troque `titulo__contains` por `titulo__startswith`
- Veja a documentação em https://docs.djangoproject.com/en/4.0/topics/db/queries/#field-lookups

Class Views

- O Django permite especificar as views usando classes
- Há várias views já prontas (generics)

server/seriados/urls.py

```
1 from django.urls import path, re_path, include, register_converter
2 from django.views.generic import TemplateView
3
4 from . import views
5
6 app_name = 'seriados'
7
8 urlpatterns = [
9     path('series/', views.series_list, name='series_list'),
10    path('series/<int:pk>/', views.series_details, name='series_details'),
11
12    path('episodios/', views.episodio_list, name='episodio_list'),
13    path('episodios/<int:pk>/', views.episodio_details, name='episodio_details'),
14
15    path('episodios/nota/<str:nota>/', views.episodio_nota_list,
16        name='episodio_nota_list'),
17
18    path('sobre/', TemplateView.as_view(template_name="about.html"), name='about'),
19    path('contato/', views.Contact.as_view(), name='contact'),
20 ]
```

Class Views

- Na url sempre usar `as_view`
- `TemplateView` Simplesmente renderiza o template passado pelo argumento `template_name`
- Nossa view `Contact` deviva de `TemplateView`

server/seriados/views.py

```
1 from django.views import View
2 from django.views.generic import TemplateView, ListView
3
4 ...
5
6 class Contact(TemplateView):
7     template_name = 'contact.html'
```

Templates about.html e contact.html

server/seriados/templates/about.html

```
1 {% extends "base.html" %}
2
3 {% block content %}
4
5     Sistema para cadastro e reviews de Seriados!<br>
6     <b>Curso de Especialização em Python para Negócios</b>
7     <p><a href="{% url 'seriados:contact' %}">Contato</a></p>
8
9 {% endblock %}
```

server/seriados/templates/contact.html

```
1 {% extends "base.html" %}
2
3 {% block content %}
4
5     <h3>Contato</h3>
6     <p><b>email:</b> contato@exemplo.com.br</p>
7     <p><a href="{% url 'seriados:about' %}">Sobre</a></p>
8
9 {% endblock %}
```

Class Views

- Adicione a seguinte URL e view

server/seriados/urls.py

```
1     path('', views.HomeView.as_view(), name='home'),
```

server/seriados/views.py

```
1 class HomeView(View):
2     def get(self, request):
3         return render(request, 'home.html', {})
```

Template home.html

server/seriados/templates/home.html

```
1 {% extends "base.html" %}
2
3 {% block content %}
4
5 <p><a href={% url 'seriados:home' %}>Home</a></p>
6
7 <p><a href={% url 'seriados:series_list' %}>Series</a></p>
8 <p><a href={% url 'seriados:episodio_list' %}>Episódios</a></p>
9
10 <p>
11     <a href={% url 'seriados:episodio_nota_list' nota='A'%}>Episódios Nota A</a> -
12     <a href={% url 'seriados:episodio_nota_list' nota='B'%}>Episódios Nota B</a> -
13     <a href={% url 'seriados:episodio_nota_list' nota='C'%}>Episódios Nota C</a>
14 </p>
15
16 <p><a href={% url 'seriados:temporadas' %}>Temporadas</a></p>
17
18 <p><a href={% url 'seriados:about' %}>Sobre</a></p>
19 <p><a href={% url 'seriados:contact' %}>Contato</a></p>
20
21 <p><a href={% url 'admin:index' %}>ADMIN</a></p>
22
23 {% endblock %}
```

Generic Views: ListView

- Adicione a seguinte URL e view

server/seriados/urls.py

```
1     path('temporadas/', views.TemporadaListView.as_view(), name='temporadas'),
```

server/seriados/views.py

```
1 class TemporadaListView(ListView):
2     template_name = 'temporada_list.html'
3     model = Temporada
```

Template temporada_list.html

server/seriados/templates/temporada_list.html

```
1 {% extends "base.html" %}
2
3 {% block content %}
4 <h1>Lista de {{title}}</h1>
5
6 <table border="1px">
7     <tr>
8         <th>Série</th>
9         <th>Temporada</th>
10    </tr>
11
12    {% for object in object_list %}
13        <tr>
14            <td>{{ object.serie.nome }}</td>
15            <td>{{ object.numero }}</td>
16        </tr>
17    {% endfor %}
18 </table>
19 {% endblock %}
```

- A preferência é um template único para cada view de cada modelo
- Os templates genéricos podem ajudar para coisas simples e um MVP rápido

Javascript no template home.html

- vamos usar o parâmetro `search` da URL que lista episódios
- Adicione o código abaixo no final do block (antes de `endblock`)

server/seriados/templates/home.html

```
<h3>Procurar episódio</h3>

<input id='input_search' type="text" placeholder="Search..">
<button id="btn_search">Procurar Episódio</button>

<script src="https://code.jquery.com/jquery-3.6.0.min.js"
    integrity="sha256-/xUj+3OJU5yExlq6GSYGSHk7tPXikynS7ogEvDej/m4="
    crossorigin="anonymous">
</script>

<script type="text/javascript">
    $(document).ready(function() {
        $("#btn_search").click(function() {
            search = $("#input_search").val();
            url = '{% url 'seriados:episodio_list' %}' + '?' +
                $.param({search:search});
            window.location.href = url;
        });
    });
</script>
```

Urls, Views (class), Templates

Yuri Kaszubowski Lopes

Especialização em Tecnologia Python para Negócios