escola britânica de artes criativas & tecnologia

Desenvolvedor Full Stack Python

Django Views

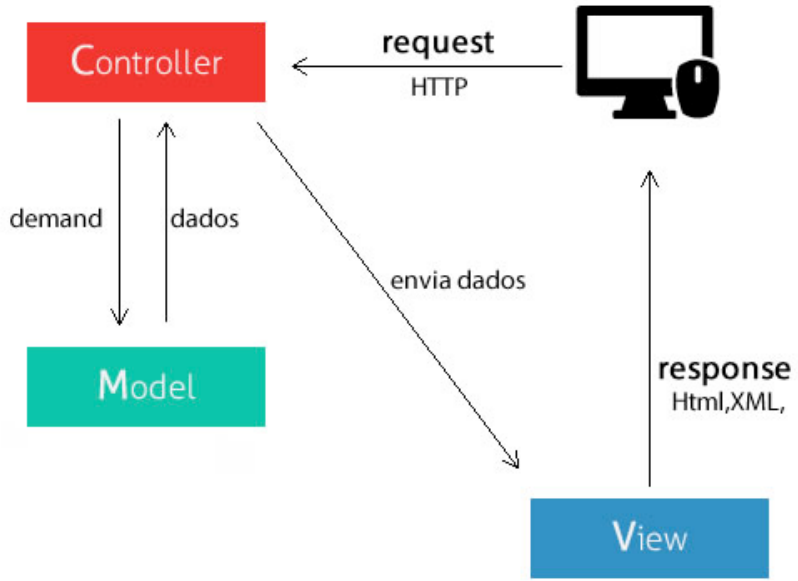
escola britânica de artes criativa & tecnologia

AGENDA

escola britânica de artes criativas & tecnologia

- **Introdução a Arquitetura MVC e MVT**
- **Views em Django**
- **Exercício**

Introdução a MVC

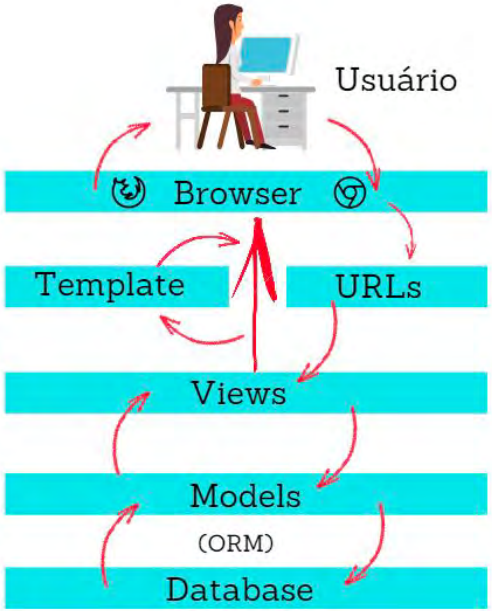


O que é MVC

MVC é nada mais que um padrão de arquitetura de software, separando sua aplicação em 3 camadas.

A camada de interação do usuário(**view**),
a camada de manipulação dos dados(**model**) e
a camada de controle(**controller**).

Introdução a MVT

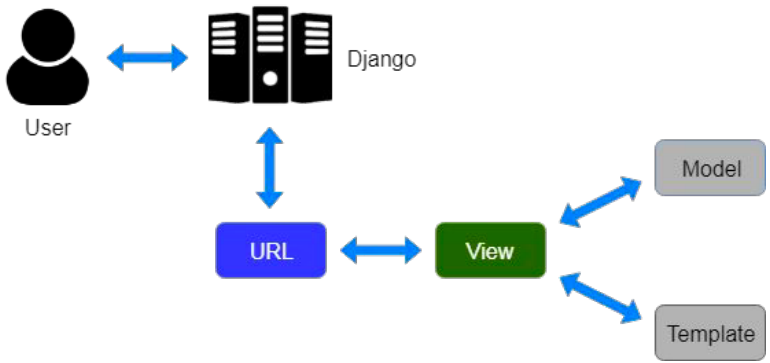


O que é MVT e a arquitetura Django

A arquitetura do Django é relativamente simples.

Basicamente, um projeto Django possui como padrão de projeto o MTV (Model, Template, View).

A sigla **MTV** representa uma camada e cada camada é composta por uma coleção de componentes.



Model

Mapeamento do banco de dados para o projeto.

Template

Páginas para visualização de dados. Normalmente, é aqui que fica o HTML que será renderizado nos navegadores.

View

Lógica de negócio. É aqui que determinamos o que irá acontecer em nosso projeto.

Views em Django

Uma função view, ou somente view, é simplesmente uma função Python que recebe uma requisição Web e retorna uma resposta Web.

Base View

A classe genérica master. Todas as outras classes herdam dessa classe. O fluxo básico de execução dessa classe quando recebe uma requisição é:

- dispatch()
- http_method_not_allowed()
- options()

```
from django.http import HttpResponse
from django.views.generic import View

class MyView(View):

    def get(self, request, *args, **kwargs):
    return HttpResponse ('Hello, World!')
```

Template View

Renderiza um template. O fluxo básico de execução dessa classe quando recebe uma requisição é:

- dispatch()
- http_method_not_allowed()
- get_context_data()

Template View

Quando você precisa apenas renderizar uma página para o usuário essa com certeza é a melhor CBV para o caso. Você pode editar o contexto que o template recebe sobrescrevendo a função **get_context_data():**

```
from django.views.generic base import TemplateView

from aticles.models import Article

class HomePageView(TemplateView):

        template_name = “home.html”

        def get_context_data(self, **kwargs):
                context = super(HomePageView, self).get_context_data(**kwargs)
                context[‘latest_articles’] = Article.objects.all()[:5]
                return context
```

List View

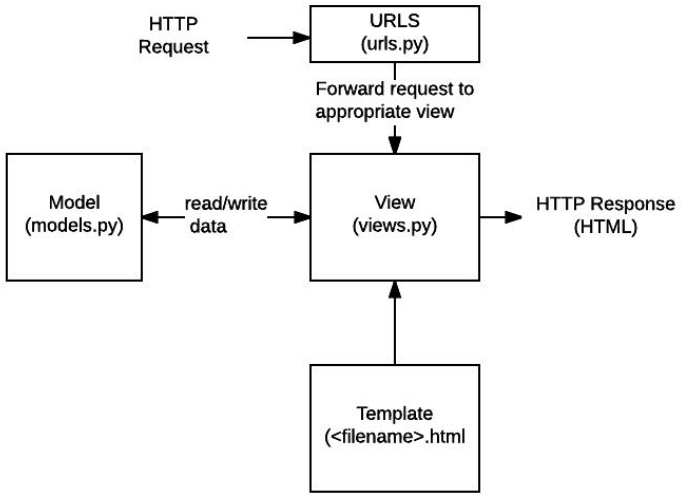
Uma página que representa uma lista de objetos. Enquanto essa view está executando a variável self.object_list vai conter a lista de objetos que a view está utilizando.

```
from django.views.generic list import ListView
from django.utils import timezone

from articles.models import Artigo

class ArticleListView(ListView):

        model = Artigo

        def get_queryset(self, **kwargs):

                return Artigo.objects.filter(status= 'publicado')
```

```
<h1>Articles</h1>
<ul>
{% for article in object_list %}
        <li>{{ article.pub_date|date }} - {{ article.headline }}</li>
{%empty %}
        <li>No articles yet.</li>
{% endfor %}
</ul>
```

URLS

O primeiro passo para escrever views é montar sua estrutura de URLs.
Você faz isso criando um módulo em Python, chamado de URLconf.
URLconfs são como o Django associa uma URL a um código em Python.



Resumo

- Aprendemos o que é MVC
- Entendemos qual é a arquitetura usada no Django
- Django Views

Exercício

escola britânica de artes criativa & tecnologia

Nesse exercício vamos dar inicio a construção das nossas Views.

Para esse exercício crie uma branch de **views**, adicione os views de **PostDetail** e **PostList**, e crie um **Pull Request** adicionando os professores da EBAC como revisores do código.