Aula 05: Entendendo o MVT do Django

Introdução ao MVT

O Django é um framework web que segue o padrão MVT (Model-View-Template). Esse padrão é semelhante ao MVC (Model-View-Controller), mas com algumas diferenças específicas. Vamos explorar cada componente do MVT e como eles interagem entre si.

Model (Modelo)

O Model é responsável pela definição da estrutura dos dados e pela interação com o banco de dados. No Django, os modelos são definidos como classes Python que herdam de `django.db.models.Model`. Cada atributo da classe representa um campo no banco de dados.

Exemplo de um modelo simples:

```
from django.db import models

class Aluno(models.Model):
    nome = models.CharField(max_length=100)
    email = models.EmailField(unique=True)
    data_nascimento = models.DateField()

    def __str__(self):
        return self.nome
```

View (Visão)

A View é responsável por processar as requisições e retornar as respostas apropriadas. No Django, as views são funções ou classes que recebem uma requisição HTTP e retornam uma resposta HTTP. Elas podem acessar os modelos para obter dados e renderizar templates para gerar a resposta.

Exemplo de uma view simples:

```
from django.shortcuts import render
from .models import Aluno

def home(request):
    alunos = Aluno.objects.all()
    return render(request, 'home.html', {'alunos': alunos})
```

Template (Modelo de Apresentação)

O Template é responsável pela apresentação dos dados. No Django, os templates são arquivos HTML que podem conter tags especiais do Django Template Language (DTL) para exibir dados dinâmicos.

Exemplo de um template simples (`home.html`):

```
<!DOCTYPE html>
<html>
<head>
    <title>Lista de Alunos</title>
</head>
<body>
    <h1>Lista de Alunos</h1>
    <ul>
        {% for aluno in alunos %}
            <li>{{ aluno.nome }} - {{ aluno.email }}</li>
        {% endfor %}
    </ul>
</body>
</html>
```

Conclusão

O padrão MVT do Django ajuda a organizar o código de forma clara e modular, separando a lógica de negócios (Model), a lógica de apresentação (View) e a interface do usuário (Template). Compreender como esses componentes interagem é fundamental para desenvolver aplicações web eficientes e escaláveis com Django.