AULA 05

FORMULÁRIOS E VALIDAÇÃO DE DADOS

O QUE VEREMOS HOJE

01 FORMULÁRIOS NO DJANGO

02 MODELFORMS

03 RENDERIZAÇÃO DE FORMULÁRIOS

04 CSRF TOKEN

05 VALIDAÇÃO DE DADOS

FORMULÁRIOS NO DJANGO

Formulários no Django são uma ferramenta para coletar dados de usuários e processá-los no servidor. Eles permitem a criação de interfaces intuitivas e seguras, garantindo a validação dos dados antes de serem armazenados no banco de dados.

Através dos formulários, é possível criar campos de entrada, dropdowns, checkboxes e outros controles, facilitando a interação do usuário com a aplicação.

Os formulários do Django são utilizados em diversos cenários, como cadastro de usuários, sistemas de login, páginas de contato, coleta de feedback, processamento de pedidos em e-commerce e muito mais

FORMULÁRIOS HTML X DJANGO FORMS

Formulários HTML	Django Forms
Necessidade de escrever todo o código HTML manualmente	Geração automática de HTML a partir de classes Python
Validação dos dados via JavaScript (manualmente)	Validação automática dos dados
Maior vulnerabilidade a ataques se não for bem implementado	Proteção automática contra ataques CSRF e validação de dados integrada.
Necessidade de implementar manualmente a lógica para salvar e manipular dados no banco,	Integração direta com models para salvar, editar ou validar dados no banco de forma eficiente.

DJANGO FORMS

Uma ferramenta do Django que facilita a criação e o gerenciamento de formulários HTML.

O Django Forms é baseado em classes Python. Cada formulário é representado por uma classe, cujos atributos correspondem aos campos do formulário.

Por padrão, os formulários são definidos no arquivo **forms.py** dentro do app correspondente.

Há duas formas principais de criar formulários:

- **Formulários Simples**: Independentes do model, usados para coletar dados que não necessariamente serão vinculados diretamente ao banco de dados.
- **ModelForms**: Baseados em um model, usados para criar ou editar registros diretamente no banco de dados.

FORMULÁRIOS SIMPLES

São formulários criados manualmente usando o Django Forms e que não possuem conexão com o banco de dados.

Para criar, só precisamos da classe e colocar como atributos da classe os campos do formulário.

Os tipos de campos disponíveis no Django estão disponíveis na documentação.

```python
from django import forms

class ContatoForm(forms.Form):
    nome = forms.CharField(label='Nome', max_length=100)
    email = forms.EmailField(label='E-mail')
    mensagem = forms.CharField(label='Mensagem', widget=forms.Textarea)
```

MODELFORMS

Os ModelsForms são formulários baseados em um model no Django.

Os campos são gerados automaticamente com base nos campos do model e esses formulários são úteis para criar ou editar registros no banco de dados.

```python
from django import forms
from .models import Produto

class ProdutoForm(forms.ModelForm):
    class Meta: # A classe Meta é uma classe interna que define metadados sobre o formulário
        model = Produto # O atributo model é uma referência ao modelo que o formulário irá utilizar
        fields = ['nome', 'descricao', 'preco', 'estoque'] # O atributo fields é uma lista de campos que serão exibidos no formulário
        labels = {
            'nome': 'Nome do Produto',
            'descricao': 'Descrição',
            'preco': 'Preço',
            'estoque': 'Estoque'
        }
```

RENDERIZAÇÃO DO FORMULÁRIO

Para utilizar o formulário no HTML, é necessário passar o formulário como contexto para que ele seja acessível no template.

O Django diferencia as requisições que chegam na página:

- se for uma requisição GET, o formulário será exibido vazio;
- se for uma POST, significa que o formulário foi enviado pelo usuário e precisamos processar os dados.

```python
from django.shortcuts import render
from .forms import ProdutoForm

def criar_produto(request):
    # Se o método da requisição for POST, o formulário foi submetido para envio dos dados
    if request.method == 'POST':
        form = ProdutoForm(request.POST) # Cria um formulário com os dados da requisição
        if form.is_valid(): # Verifica se o formulário é válido com base nas regras definidas no formulário
            form.save() # Salva os dados do formulário no banco de dados
            form = ProdutoForm() # Cria um novo formulário em branco
    else:
        # Se o método da requisição não for POST, o formulário será exibido em branco
        form = ProdutoForm()
    return render(request, 'criar_produto.html', {'form': form}) # Renderiza o template criar_produto.html com o formulário
```

RENDERIZAÇÃO DO FORMULÁRIO

Para formulários que não exigem conexão com banco de dados, não é necessário diferenciar o método HTTP, mas o que será feito com os dados do formulário vai depender de cada caso.

Ao validar o formulário com o método is_valid(), o Django disponibiliza os dados limpos e validados no atributo cleaned_data. Esse atributo é um dicionário que garante que os dados recebidos estão seguros e prontos para uso.

```python
from django.shortcuts import render
from django.core.mail import send_mail
from .forms import ProdutoForm, ContatoForm

def contato(request):
    if request.method == 'POST':
        form = ContatoForm(request.POST)
        if form.is_valid():
            # Captura os dados do formulário
            nome = form.cleaned_data['nome']
            email = form.cleaned_data['email']
            mensagem = form.cleaned_data['mensagem']

            # Envia o e-mail
            send_mail(
                f"Mensagem de {nome}",  # Assunto
                f"De: {nome} <{email}>\n\n{mensagem}",  # Corpo do e-mail
                'seu_email@exemplo.com',  # Remetente
                ['destinatario@exemplo.com'],  # Destinatário
            )

            # Redefine o formulário após envio
            form = ContatoForm()
    else:
        form = ContatoForm()
    return render(request, 'contato.html', {'form': form })
```

FORMULÁRIOS NO TEMPLATE

Para usar o formulário no template é preciso usar a variável passada como contexto na view.

Os formulários podem ser exibidos de várias formas com os campos organizados automaticamente pelo Django ou personalizados com HTML.

```html
<body>
    <!-- Template com Renderização Básica  -->
    <form method="post">
        {% csrf_token %}
        {{ form.as_p }} <!-- Renderiza o formulário utilizando a tag p -->
        <button type="submit">Enviar</button>
    </form>

    <!-- Template com Renderização personalizada -->
    <form method="post">
        {% csrf_token %}
        <div>
            <label for="nome">Nome:</label>
            {{ form.nome }}
        </div>
        <div>
            <label for="preco">Preço:</label>
            {{ form.preco }}
        </div>
        <div>
            <label for="descricao">Descrição:</label>
            {{ form.descricao }}
        </div>
        <button type="submit">Enviar</button>
    </form>
</body>
```

CSRF TOKEN

No Django, é necessário adicionar o token CSRF ({% csrf_token %}) em formulários que utilizam o método POST.

Esse token é uma medida de segurança que protege a aplicação contra ataques do tipo Cross-Site Request Forgery (CSRF), que podem forjar envios de dados sem o consentimento do usuário.

O Cross-Site Request Forgery (CSRF) é um tipo de ataque feito entre sites em que um site malicioso tenta enganar o navegador de um usuário para enviar dados a um site legítimo, sem que o usuário tenha intenção de fazê-lo.

O Django gera automaticamente um token CSRF único para cada sessão do usuário. Esse token é validado no momento do envio do formulário. Se o token estiver ausente ou inválido, o envio será rejeitado

Então em todo formulário te que ter o {% csrf_token %}.

ATIVIDADE PRÁTICA

No App de tarefas crie um formulário para permitir a criação de tarefas diretamente na interface do usuário (não pelo admin)

- Crie um form a partir do Model
- Crie a view de cadastro da tarefa
- Exiba o formulário no template e faça um cadastro
- Verifique se o dado cadastrado aparece no Admin

VALIDAÇÃO DE DADOS

A validação de dados é um processo fundamental para garantir a integridade e a confiabilidade das informações inseridas em um formulário.

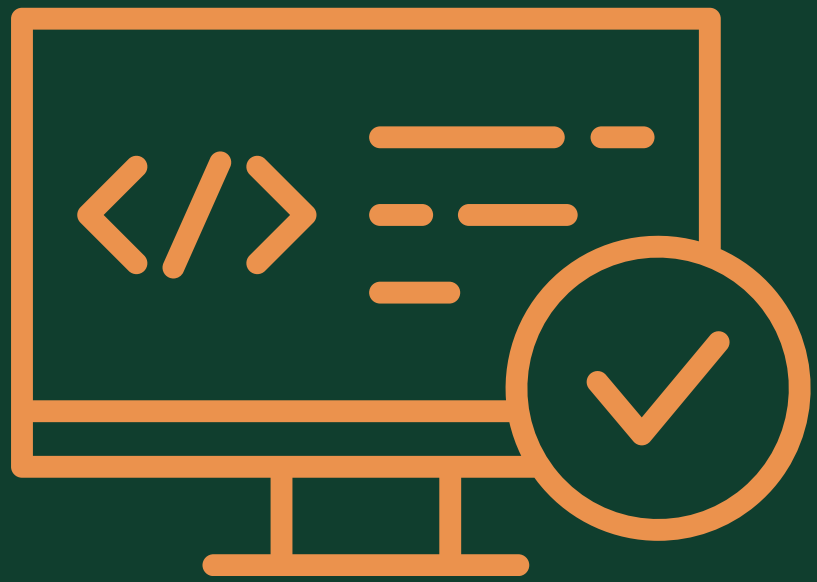
Ela envolve a verificação dos campos de entrada, a aplicação de regras de validação e o tratamento adequado de erros.

No Django, a validação de dados é realizada através de recursos nativos da framework, como os campos de formulário e a validação personalizada.

TIPOS DE VALIDAÇÕES

- **Validações automáticas**: O Django possui validações embutidas através do tipo do campo nos formulários. Então, campos como CharField e EmailField já possuem validações para verificar se o campo é obrigatório ou se o email é válido.

- **Validações personalizadas**: É possível criar regras específicas para cada campo ou para o formulário inteiro utilizando métodos.



VALIDAÇÕES PERSONALIZADAS

No Django, as validações personalizadas são implementadas dentro do próprio formulário por meio de funções específicas. Essas funções recebem o argumento self, que representa a instância do formulário que está sendo processada.

Usamos o self.cleaned_data para acessar os dados preenchidos, pois é onde o Django armazena os dados que foram enviados pelo formulário e passaram por validações básicas.

```python
from django import forms
from .models import Produto

class ProdutoForm(forms.ModelForm):
    class Meta:
        model = Produto
        fields = ['nome', 'descricao', 'preco', 'estoque']
        labels = {
            'nome': 'Nome do Produto',
            'descricao': 'Descrição',
            'preco': 'Preço',
            'estoque': 'Estoque'
        }

    def clean_nome(self):
        nome = self.cleaned_data.get('nome')  # Captura o valor do campo 'nome'
        if len(nome) < 3:
            raise forms.ValidationError("O nome deve ter pelo menos 3 caracteres.")
        return nome  # Retorna o valor validado

    # Validação específica para o campo 'preco'
    def clean_preco(self):
        preco = self.cleaned_data.get('preco')
        if preco <= 0:
            raise forms.ValidationError("O preço deve ser maior que zero.")
        return preco
```

ATIVIDADE PRÁTICA

No App de Gerenciamento de Tarefas, crie uma validação para o nome da tarefa e para a descrição da tarefa (se preenchida).

O nome deve ter pelo menos 5 caracteres.

A descrição deve ter pelo menos 10 caracteres, se for preenchida.