03

Montando um formulário no flask

Transcrição

O conteúdo do nosso formulário está sendo exibido abertamente na **query string**, e não queremos que isso aconteça. No HTTP, quando passamos um conteúdo às claras na **query string**, estamos fazendo uma requisição (ou método) GET. Para resolvermos esse problema, basta mudarmos o método para POST.

Antes de prosseguirmos, vamos entender como funciona o fluxo da criação de um novo jogo.

Quando mandamos um *request* HTTP para o servidor Flask, ele precisa receber essa request, tratar essa informação e enviar algum retorno — no caso, algum HTML que mostre que a criação do novo jogo funcionou.

No arquivo `novo.html`, vamos alterar o `<form>` para que ele tenha as informações necessárias e consiga enviar informação para o servidor — já que, até o momento, ele está somente renderizando novamente a tela exibida. Precisamos, primeiramente, indicar para onde irá a requisição.

Criaremos uma `action` que representará uma URL ou rota para onde essa requisição será enviada. No caso, um formulário com as características de um novo jogo. Vamos definir essa rota como `/criar` — ela ainda não existe, mas a criaremos daqui a pouco.

Também precisaremos passar um método `post` para que os dados não sejam expostos. Assim, teremos:

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <title>Jogoteca</title>
</head>
<body>
    <div class="container">
      <div class="page-header">
          <h1>{{ titulo }}</h1>
      </div>
      <form action="criar" method="post">
//...
```

Na aplicação **jogoteca**, criaremos a rota `/criar`. Nela, definiremos o que deve ser feito com o dado recebido na requisição. Como estamos recebendo um novo jogo, é interessante colocarmos esse jogo na lista, e é isso que faremos na função `criar()`:

```
@app.route('/criar')
def criar():
```

Em seguida, importaremos um *helper* do Flask chamado `request`, que nos fornece, por meio de uma variável global, todas as informações recebidas no request, inclusive a informação do `<form>`.

Perceba que, no nosso formulário HTML, temos as seguintes informações no `<input type>`:

```
<input type="text" id="nome" name="nome" class="form-control">
```

O campo `name` é o nome de um `input`, ou seja, ele é uma das características do nosso `<form>`. Essas características ficam armazenadas no dicionário da requisição, e podem ser acessadas a partir do `name` atribuído, por exemplo, fazendo `request.form['nome']`. Repetiremos esse processo para as outras informações:

```
@app.route('/criar')
def criar():
    nome = request.form['nome']
    categoria = request.form['categoria']
    console = request.form['console']
```

Agora que pegamos esses valoes, precisaremos criar o novo jogo na nossa lista — o que é bem simples: basta instanciarmos o jogo, passando as variáveis que recebemos no request:

```
@app.route('/criar')
def criar():
    nome = request.form['nome']
    categoria = request.form['categoria']
    console = request.form['console']
    jogo = Jogo(nome, categoria, console)
```

Nesse momento, só precisamos adicionar o novo jogo à nossa lista. Porém, perceba que a nossa lista está dentro da função `ola()`, e não temos acesso a ela a partir da função `criar()`. Além disso, sempre que rodamos a aplicação, o Flask cria a lista novamente, ou seja, nossa aplicação não está dinâmica.

Para resolvermos isso, vamos colocar a lista fora do escopo da função `ola()`:

```
app = Flask(__name__)

class Jogo:
    def __init__(self, nome, categoria, console):
        self.nome = nome
        self.categoria = categoria
        self.console = console

jogo1 = Jogo('Super Mario', 'Acao', 'SNES')
jogo2 = Jogo('Pokemon Gold', 'RPG', 'GBA')
jogo3 = Jogo('Mortal Kombat', 'LutA', 'SNES')
lista = [jogo1, jogo2, jogo3]

@app.route('/inicio')
def ola():
    return render_template('lista.html', titulo='Jogos',
                           jogos=lista)
```

Dessa forma, nossas funções conseguirão ter acesso à lista, já que ela está operando de forma global nesse pacote. Na função `criar()`, utilizaremos `lista.append()` para adicionar um novo item à lista — no caso, o `jogo` que acabamos de criar.

Ainda precisamos retornar algo ao navegador para mostrar que o jogo foi salvo na lista, e nada mais adequado nesse ponto do que exibir a própria lista. Para isso, retornaremos um `render_template()` de `lista.html`, passando o título `Jogos` e atribuindo `jogos=lista`:

```
@app.route('/criar')
def criar():
    nome = request.form['nome']
    categoria = request.form['categoria']
    console = request.form['console']
    jogo = Jogo(nome, categoria, console)
    lista.append(jogo)
    return render_template('lista.html', titulo='Jogos',
                           jogos=lista)
```

Para testar, vamos rodar a aplicação novamente e tentar cadastrar um novo jogo em http://127.0.0.1:5000/novo (http://127.0.0.1:5000/novo). Obteremos a seguinte resposta:

> **Method Not Allowed**
>
> The method is not allowed for the requested URL.

Isso significa que, por algum motivo, o método POST não está sendo aceito pela função que recebe a requisição. Daqui a pouco, verificaremos como resolver isso. Até já!