02

Subindo imagens para o servidor

Transcrição

Na página "Novo Jogo", em vez de simplesmente colocarmos os dados do jogo, queremos salvar também uma imagem - por exemplo, a capa do jogo. Inicialmente, tentaremos implementar o upload de arquivos no servidor da nossa aplicação.

O primeiro passo é incluirmos um campo de upload no formulário "Novo". Para isso, antes do `fieldset>` , colocaremos a informação de um `input` do tipo `file` - ou seja, o *input* de um arquivo. Isso significa que teremos um selecionador de arquivos que nos permitirá acessar os diretórios do computador.

Também definiremos um `name="arquivo"` , já que precisaremos acessar esse arquivo no servidor a partir da requisição, e um `accept=".jpg"` para limitar o tipo de arquivo que nossa aplicação aceita.

> Note que essa validação não é muito forte, e somente mostrará de primeira os arquivos `.jpg` para que o usuário faça o upload.

Com isso, ganharemos um botão "Escolher arquivo" (ou "Choose File") na página http://127.0.0.1:5000/novo (http://127.0.0.1:5000/novo), mas ainda não implementamos o upload no nosso servidor.

Para essa implementação, trabalharemos com a função `criar()` . Nela, depois que o jogo é salvo no banco de dados, iremos salvar a imagem que foi enviada ao servidor (já que as informações do jogo são mais importantes que a imagem).

Diferentemente das informações do `Jogo` , o Flask não utiliza o `form` para pegar informações de arquivo, mas sim outro tipo de atributo: `files` . Com ele, podemos acessar uma posição - nesse caso, a posição `arquivo` que definimos no nome do nosso *input*. Esse objeto recebido da requisição será salvo em uma variável `arquivo` .

Nesse momento, a variável será do tipo `file` . Isso facilitará nosso trabalho, pois o tipo `file` possui um método `save()` que precisa somente do destino em que salvaremos o arquivo. Por enquanto, utilizaremos o diretório raiz/padrão que está sendo executado, passando `arquivo` e o atributo `filename` :

```
@app.route('/criar', methods=['POST',])
def criar():
    nome = request.form['nome']
```

```
    categoria = request.form['categoria']
    console = request.form['console']
    jogo = Jogo(nome, categoria, console)
    jogo_dao.salvar(jogo)

    arquivo = request.files['arquivo']
    arquivo.save(arquivo.filename)
    return redirect(url_for('index'))
```

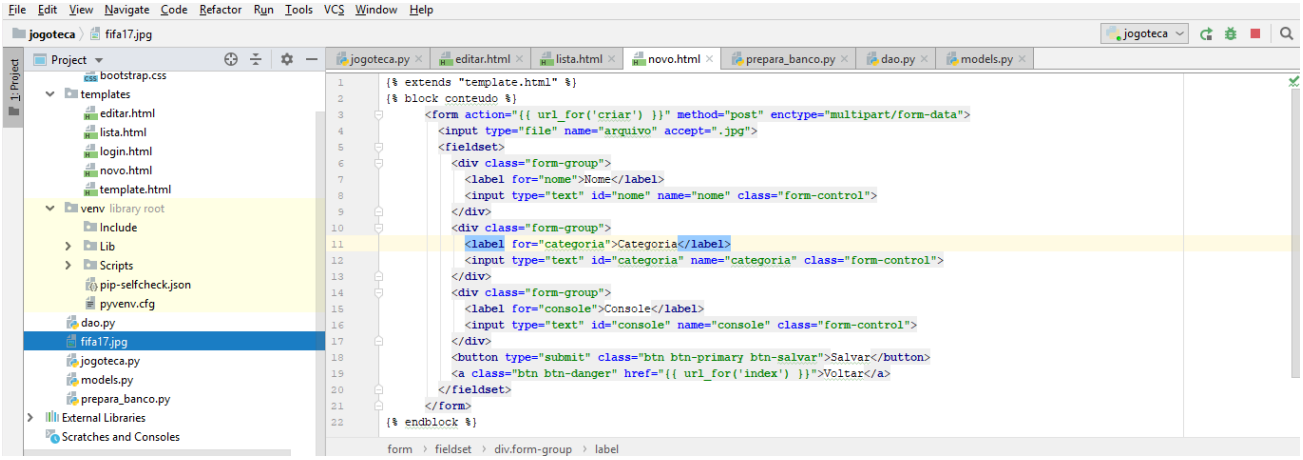
Se tentarmos fazer o upload de um arquivo na página "Novo" (por exemplo, `fifa17.jpg`), receberemos o seguinte erro:

> flask.debughelpers.DebugFilesKeyError flask.debughelpers.DebugFilesKeyError: You tried to access the file "arquivo" in the request.files dictionary but it does not exist. The mimetype for the request is "application/x-www-form-urlencoded" instead of "multipart/form-data" which means that no file contents were transmitted. To fix this error you should provide enctype="multipart/form-data" in your form.
>
> The browser instead transmitted some file names. This was submitted: "pati.jpg"

Aparentemente, faltou algo no nosso formulário. Existem muitas formas de trafegar arquivos, e uma delas é por meio de múltiplas partes - que é o que estamos tentando fazer. Para isso funcionar, precisamos indicar explicitamente no formulário que temos um `enctype="multipart/form-data"`.

```
{% extends "template.html" %}
{% block conteudo %}
      <form action="{{ url_for('criar') }}" method="post" enctype="multipart/form-data">
        <input type="file" name="arquivo" accept=".jpg">
        <fieldset>
          <div class="form-group">
            <label for="nome">Nome</label>
            <input type="text" id="nome" name="nome" class="form-control">
          </div>
          <div class="form-group">
            <label for="categoria">Categoria</label>
            <input type="text" id="categoria" name="categoria" class="form-control">
          </div>
          <div class="form-group">
            <label for="console">Console</label>
            <input type="text" id="console" name="console" class="form-control">
          </div>
          <button type="submit" class="btn btn-primary btn-salvar">Salvar</button>
          <a class="btn btn-danger" href="{{ url_for('index') }}">Voltar</a>
        </fieldset>
      </form>
{% endblock %}
```

Atualizando a página e refazendo a operação, não receberemos nenhum erro. Mas será que o upload funcionou? No próprio **PyCharm**, veremos que um novo arquivo surgiu no nosso diretório padrão, que é justamente o arquivo que salvamos.



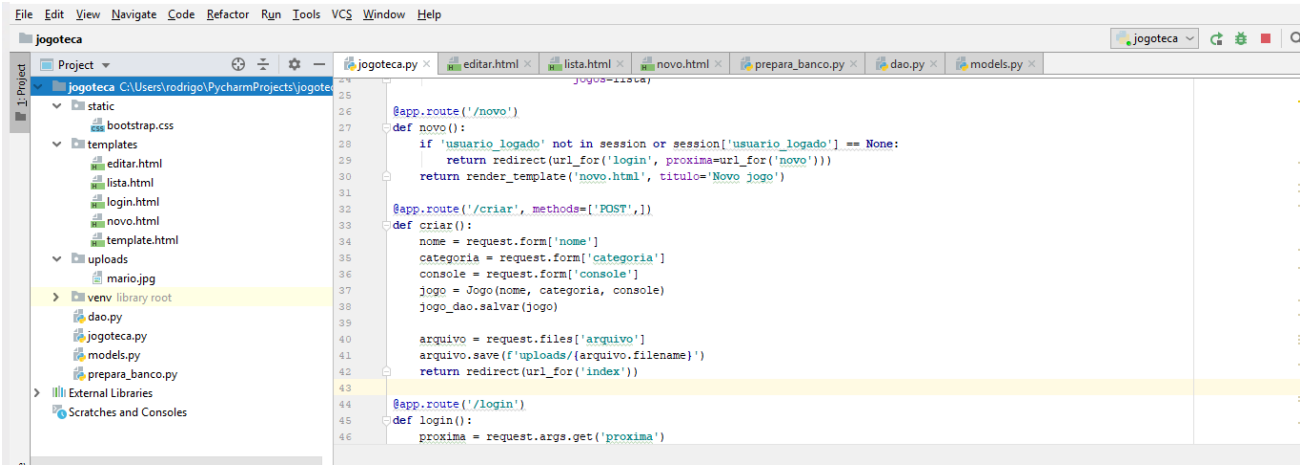
Isso funcionou como teste, mas não queremos misturar os arquivos que são enviados ao servidor com os arquivos do nosso projeto. Portanto, criaremos uma pasta "uploads" para guardar todas as imagens que forem salvas pelo usuário, e passaremos esse diretório para a função `save()` do objeto `arquivo`.

Como estamos usando o Python 3.6, podemos fazer isso com uma formatação um pouco diferente, passando um *format* (`f`), o diretório `uploads` e o `arquivo.filename`. Existem várias formas de formatar strings no Python, e dessa forma alcançamos nosso objetivo escrevendo bem pouco código.

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

    arquivo = request.files['arquivo']
    arquivo.save(f'uploads/{arquivo.filename}')
    return redirect(url_for('index'))
```

Retornando à aplicação e fazendo um novo upload de imagem, ela será salva no diretório correto que passamos.



Porém, estamos passando o diretório por meio de um caminho relativo, o que não é uma boa prática para a nossa aplicação. Nos próximos vídeos iremos resolver esse problema e melhoraremos ainda mais a usabilidade da nossa aplicação. Até lá!