04

Melhorando o fluxo de login

Transcrição

Conseguimos produzir uma autenticação na nossa aplicação. Agora, sempre que fazemos uma requisição para a rota `/novo`, o servidor verifica se o usuário está autenticado. Caso positivo, ele é redirecionado para `/`; do contrário, ele cai no fluxo de `/login` e precisa fazer outra requisição com os dados de login para que esse usuário seja autenticado.

Precisamos fazer com que o servidor redirecione o usuário para a página de origem da requisição, ou seja, se ele for barrado na página `/novo`, deve retornar a ela após o login.

O HTTP não guarda estado entre as requisições, então como faremos para guardar esses dados? Uma maneira fácil de passarmos essa informação — considerando que ela não precisa ser segura — é por meio de uma *query string*.

Por exemplo, com `/login?proxima=novo`, estaremos informando que `/novo` é o caminho a seguir depois que o usuário fizer o login. Ou seja, esse dado deve ser passado para frente até o momento em que o usuário se autenticar.

```
@app.route('/novo')
def novo():
    if 'usuario_logado' not in session or session['usuario_logado'] == None:
        return redirect('/login?proxima=novo')
    return render_template('novo.html', titulo='Novo jogo')
```

Agora, precisamos pegar essa informação na rota `/login`. Para isso, criaremos uma variável `proxima` que receberá o argumento que chamamos de `proxima`, por meio do método `get()` da requisição. Para não perdermos essa informação, precisamos passá-la para `render_template()`, criando uma variável `proxima` que recebe o mesmo argumento que pegamos na requisição.

```
@app.route('/login')
def login():
    proxima = request.args.get('proxima')
    return render_template('login.html', proxima=proxima)
```

No formulário de login (`login.html`), sabemos que, em algum momento, precisaremos fazer o `POST` da requisição, e o `POST` carrega os dados do formulário. Em algum momento, precisamos também passar a informação de qual é a próxima página. Para isso, podemos criar um campo, por exemplo, `<input type ="text">` para guardar o texto do formulário para onde queremos ir.

Porém, na verdade queremos que essa informação fique escondida. Nesses casos, é muito comum utilizar `<input type="hidden">`, um tipo específico de `input`. Esse tipo é guardado na tela, junto aos dados do formulário, mas nenhuma informação é mostrada para o usuário.

Nesse tipo, temos que ter um `name` (que faz com que essa informação possa ser acessada, depois que é feito o `POST`) e um `value`. Como a informação que queremos está acessível por meio de uma variável chamada `proxima`, podemos acessá-la usando a notação `{{ }}` do Jinja2.

```
<h1>Faça seu login</h1>
<form method="POST" action="/autenticar">
    <input type="hidden" name="proxima" value="{{ proxima }}">
```

Isso é tudo que precisaremos fazer em `login.html`. Agora, vamos trabalhar com a rota `/autenticar`.

Nela, estamos sempre redirecionando o usuário para `/`. Porém, se o usuário tentou acessar a rota `/novo`, faz sentido que ele seja redirecionado de volta a ela. Como essa informação está no nosso formulário, podemos pegá-la da mesma forma que fizemos com as outras informações.

Criaremos uma variável `proxima_pagina`, que recebe `request.form[]`, passando a informação que queremos pegar do formulário, no caso, `proxima`.

Para passarmos essa informação corretamente para `return redirect()`, precisamos formar a rota `/novo`. Considerando que definimos a `proxima_pagina` como `novo`, teremos que interpolar `/`, formatando a string `proxima_pagina` e passando o valor resultante para `{}`:

```
@app.route('/autenticar', methods=['POST',])
def autenticar():
    if 'mestra' == request.form['senha']:
        session ['usuario_logado'] = request.form['usuario']
        flash(request.form['usuario'] + ' logou com sucesso!')
        proxima_pagina =  request.form['proxima']
        return redirect('/{}'.format(proxima_pagina))
    else :
        flash('Não logado, tente de novo!')
        return redirect ('/login')
```

Feita essa implementação, se acessarmos `/login` por meio da rota `/novo`, seremos corretamente redirecionados para `/novo`, após o login ter sido bem sucedido. Porém, se tentarmos acessar `/login` diretamente e fizermos o login, seremos redirecionados para `http://127.0.0.1:5000/None`, **uma página que não existe**.

Isso acontece porque, quando não estamos acessando `/login` por meio de outra página, a variável `proxima` tem o valor `None`, ou seja, vazio. Antes de tudo, precisamos verificar se o valor está vazio ou não.

Para resolvermos isso, em `login.html`, podemos passar as variáveis `proxima or '/'`, já que `/` também é uma string. Dessa forma, se `proxima` estiver com valor vazio, seremos redirecionados para `/`, que é a nossa lista de jogos.

Porém, temos alguns problemas no nosso código, como essa string `/` e o fato de estarmos passando diretamente strings com as rotas da nossa aplicação (por exemplo, `/login`), chegando até mesmo a fazer uma interpolação para gerar a rota desejada, tudo de forma bastante rígida.

O Flask nos ajuda a fazer isso de maneira mais fácil, e é isso que estudaremos adiante.