06

Mais URLs dinâmicas

Transcrição

Na nossa aplicação, sempre que fazemos um redirecionamento para uma rota, estamos trabalhando com essa rota por escrito, ou seja, rigidamente codificada. Em alguns casos, estamos até mesmo passando uma *query string* ou interpolando para gerar uma rota.

Isso pode gerar dificuldades se tivermos que mudar alguma rota, e o Flask nos fornece uma maneira melhor de gerarmos essas URLs de forma dinâmica: a função `url_for()`, que utilizamos para acessar nossa pasta de estáticos.

Antes de tudo, precisamos importar a função `url_for()` do pacote Flask. Em seguida, começaremos a trabalhar com o primeiro ponto em que temos um redirecionamento, que é na função `novo()`:

```
@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')
```

A função `url_for()` nos permite acessar um recurso da aplicação diretamente pelo nome da função. Se o redirecionamento é para `/login`, por exemplo, podemos passar `url_for('login')`. No caso da rota `/novo`, ainda precisamos passar uma informação, que é a variável `proxima`, com o valor `novo`. Podemos fazer isso da mesma forma que faríamos em um `render_template()`, passando essa variável após uma vírgula (`,`).

Como o valor `novo` seria usado para acessar outra rota, é mais lógico utilizarmos novamente a função `url_for()`. Dessa forma, ficamos ligados a um método, e não a um trecho de código codificado rigidamente. Como o método que estamos chamando é `novo`, basta passarmos esse parâmetro para a função:

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

Nosso próximo redirecionamento está na função `criar()`:

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

Essa `/`, na verdade, é a nossa função `index()`. Portanto, repetiremos o processo anterior para essa função:

```
@app.route('/criar', methods=['POST',])
def criar():
    nome = request.form['nome']
    categoria = request.form['categoria']
    console = request.form['console']
    jogo = Jogo(nome, categoria, console)
    lista.append(jogo)
    return redirect(url_for('index'))
```

Em `autenticar()`, quando o usuário loga com sucesso, escrevemos uma interpolação para juntar o valor de `proxima_pagina` com a `/`. Como já estamos passando o endereço completo de `proxima_pagina` (`/novo`) na função `novo()`, podemos passar somente essa variável. Já se o usuário não loga com sucesso, ele é redirecionado para `/login`, o que pode ser feito passando apenas `url_for(login)`:

```
@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(proxima_pagina)
    else :
        flash('Não logado, tente de novo!')
        return redirect(url_for('login'))
```

Agora basta aplicarmos essa alteração para a função `logout()`:

```
@app.route('/logout')
def logout():
    session['usuario_logado'] = None
    flash('Nenhum usuário logado!')
    return redirect(url_for('index'))
```

E terminamos de implementar as URLs dinâmicas na nossa aplicação. Porém, ainda precisamos alterar alguns pontos do nosso HTML. Em `login.html`, estamos fazendo um `POST` para `/autenticar`. Além disso, a variável `proxima` tem os valores `proxima or '/'`. Vamos alterar esses campos para os métodos correspondentes:

```
<h1>Faça seu login</h1>
<form method="POST" action="{{ url_for('autenticar') }}">
    <input type="hidden" name="proxima" value="{{ proxima or url_for('index') }}">
```

Já em `novo.html`, teremos que alterar apenas o campo `form action = "criar" method="post">:

```
<form action="{{ url_for('criar') }}" method="post">
```

Agora, podemos acessar as URLs da nossa aplicação para verificar se elas estão funcionando corretamente. Quando tentamos acessar a página `/novo` sem estarmos logados, somos redirecionados para http://127.0.0.1:5000/login?

proxima=%2Fnovo (http://127.0.0.1:5000/login?proxima=%2Fnovo). Esse `%2Fnovo` é justamente a `/` que foi transformada para ser carregada na URL.

Já que tudo está correto, nossas páginas estarão funcionando como gostaríamos. Inclusive, podemos incluir novos jogos na lista.

Conseguimos melhorar o nosso código, evitando problemas de manutenção no futuro. E agora, o que falta? Bom, a nossa lógica de login ainda está um pouco estranha, já que temos uma senha `mestra` que é utilizada para todos os usuários. Na verdade, queremos ter múltiplos usuários, cada um com uma senha específica.

Mas, como armazenar os dados de vários usuários na nossa aplicação? Estudaremos como, a seguir.