08

Reescrita de métodos HTTP

Transcrição

Precisamos implementar uma forma de efetivamente fazer a edição dos nossos livros, ao invés de adicionar um novo em nossa lista toda vez que editamos as informações de um dos elementos.

Sabemos que esse problema ocorre pois, quando submetemos os dados no nosso formulário, a rota `/livros` é ativada para o método POST do HTTP. Consequentemente, o método `adiciona()` é executado.

Quando estamos criando um livro novo, o objeto `livro` retornado para a rota `/livros` não tem `id` nenhum. Em contrapartida, quando fazemos o carregamento do formulário para edição, passamos um livro que tem um `id`, pois vem diretamente do nosso banco de dados.

Portanto, uma estratégia válida seria, na rota do método POST para `/livros`, pegar o `livro` na requisição com `req.body`. No entanto, antes do método `adicionar()`, teríamos que escrever uma condição para que, se tivéssemos um `id` diferente de vazio, seria executado um código referente à edição do livro; caso contrário, o método `adiciona()` seguiria executando normalmente.

Porém, isso faria com que a rota POST de `/livros` acumulasse muitas responsabilidades. Seria mais interessante utilizássemos um método específico do protocolo HTTP para editar informações que já existem, chamado PUT.

Para isso, repetiremos o código criado para a rota POST, trocando os seguintes campos:

- `app.post()` para `app.put()`
-
- `livroDao.adiciona()` para `livroDao.atualiza()`

Ainda existe um detalhe: no `form.marko`, que representa o formulário de cadastro de livros, o método de envio das informações é sempre POST.

```
<html>
    <body>
        <h1>Cadastro de livros</h1>

        <form action="/livros" method="post">

            <input type="hidden" id="id" name="id" value="${data.livro.id}" />

            <div>
                <label for="titulo">Titulo:</label>
                <input type="text" id="titulo" name="titulo" value="${data.livro.titulo}" placel
            </div>
            <div>
                <label for="preco">Preço:</label>
                <input type="text" id="preco" name="preco" value="${data.livro.preco}" placehol
            </div>
            <div>
                <label for="descricao">Descrição:</label>
                <textarea cols="20" rows="10"  id="descricao" name="descricao" placeholder="fal
```

```
            </div>

            <input type="submit" value="Salvar" />
        </form>
    </body>
</html>
```

Dessa maneira, precisamos encontrar uma forma de filtrar todas as requisições que cheguem na aplicação, verificando se, nelas, existe ou não um valor para `id`. Ou seja, dada uma condição específica, queremos sobrescrever o método de envio da nossa requisição.

Felizmente, já existe um *middleware* que faz justamente isso, chamado `method-override` (https://github.com/expressjs/method-override).

Para implementarmos esse *middleware*, depois de utilizarmos o `bodyParser`, precisaremos declarar o `method-override`, como no exemplo da documentação:

```
app.use(bodyParser.urlencoded({extended: true}));
app.use(methodOverride(function (req, res) {
  if (req.body && typeof req.body === 'object' && '_method' in req.body) {
    // look in urlencoded POST bodies and delete it
    var method = req.body._method
    delete req.body._method
    return method
  }
}));
```

Nesse código, estamos verificando se o corpo da requisição é do tipo `object` e se, nele, existe um `_method`. Em caso positivo, o valor passado no `<input>` será salvo em uma variável `method`. Ao final, esse novo método será retornado.

Sempre que quisermos que um método seja sobrescrito, também precisaremos adicionar ao formulário um `<input>` do tipo `hidden` com o nome `_method` e passando como valor o método que será utilizado para sobrescrevê-lo:

```
<input type="hidden" name="_method" value="DELETE">
```

Antes de prosseguirmos, precisamos instalar esse *middleware*. No terminal, digitaremos `npm install method-override@3.0.0 --save-exact`, instalando-o no nosso `package.json`.

No arquivo `custom-express.js`, colaremos, logo após o `bodyParser`, o código encontrado na documentação:

```
app.use(methodOverride(function (req, res) {
    if (req.body && typeof req.body === 'object' && '_method' in req.body) {
      var method = req.body._method;
      delete req.body._method;
      return method;
    }
  }))
```

Também precisamos declarar a constante `methodOverride`:

```
const methodOverride = require('method-override');
```

Por último, no `form.marko`, colocaremos o `<input>` referente ao método PUT. Porém, temos que nos atentar ao fato de que queremos utilizar esse método somente quando estivermos editando um livro - ou seja, quando a propriedade `livro` tiver um `id` válido.

Com a sintaxe do Marko JS, faremos isso facilmente, declarando uma `<div>` que só será exibida se `data.livro.id` for válido. Tudo que estiver dentro dessa `<div>` (ou seja, os *inputs* ocultos) só será renderizado caso tenhamos esse `id` válido - exatamente o que acontece quando estamos editando um livro.

```
<html>
    <body>
        <h1>Cadastro de livros</h1>

        <form action="/livros" method="post">
            <div if(data.livro.id)>
                <input type="hidden" name="_method" value="PUT">
                <input type="hidden" id="id" name="id" value="${data.livro.id}" />
            </div>
            <div>
                <label for="titulo">Titulo:</label>
                <input type="text" id="titulo" name="titulo" value="${data.livro.titulo}" placel
            </div>
            <div>
                <label for="preco">Preço:</label>
                <input type="text" id="preco" name="preco" value="${data.livro.preco}" placehol
            </div>
            <div>
                <label for="descricao">Descrição:</label>
                <textarea cols="20" rows="10"  id="descricao" name="descricao" placeholder="fal
            </div>

            <input type="submit" value="Salvar" />
        </form>
    </body>
</html>
```

Após salvarmos todas essas alterações, acessaremos novamente a URL http://localhost:3000/livros (http://localhost:3000/livros) e escolheremos um dos livros na lista para editar. Na página seguinte, bastará efetuarmos qualquer mudança nos campos e clicarmos em "Salvar". Dessa vez, tudo ocorrerá corretamente da maneira que previámos... exceto a descrição do livro!

Isso acontecerá pois o valor de `<textarea>` não é passado com a propriedade `value`, mas sim entre as *tags* desse elemento HTML! Portanto, faremos essa pequena alteração:

```
<html>
    <body>
        <h1>Cadastro de livros</h1>

        <form action="/livros" method="post">
            <div if(data.livro.id)>
                <input type="hidden" name="_method" value="PUT">
```

```
                <input type="hidden" id="id" name="id" value="${data.livro.id}" />
            </div>
            <div>
                <label for="titulo">Titulo:</label>
                <input type="text" id="titulo" name="titulo" value="${data.livro.titulo}" placel
            </div>
            <div>
                <label for="preco">Preço:</label>
                <input type="text" id="preco" name="preco" value="${data.livro.preco}" placehol
            </div>
            <div>
                <label for="descricao">Descrição:</label>
                <textarea cols="20" rows="10"  id="descricao" name="descricao" placeholder="fal
            </div>

            <input type="submit" value="Salvar" />
        </form>
    </body>
</html>
```

Dessa vez, todas as alterações que fizermos editando os livros serão salvas corretamente!