02

Cadastro de livros

Transcrição

Vamos começar a desenvolver a funcionalidade de cadastro de livros da nossa aplicação. De volta ao nosso código, criaremos, na pasta "livros", uma pasta "form" contendo um arquivo `form.marko`, no qual construiremos o formulário de cadastro.

Isso será feito com um HTML comum:

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

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

            <input type="hidden" id="id" name="id" />

            <div>
                <label for="titulo">Titulo:</label>
                <input type="text" id="titulo" name="titulo" placeholder="coloque o titulo" />
            </div>
            <div>
                <label for="preco">Preço:</label>
                <input type="text" id="preco" name="preco" placeholder="150.25" />
            </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>
```

Como podemos analisar, temos um cabeçalho informando que essa é a página de cadastro; um formulário definido pela tag `<form>`, contendo um `input` oculto (para a `id` do nosso livro), campos de texto para o título e para o preço, e um `textarea` para a descrição do livro; um botão para submeter as informações do formulário; e a `action` indicando a URL para qual esses dados serão enviados, que nesse caso é `/livros`.

Também temos a definição do método de envio dos nossos dados, que é POST, de modo eles sejam encapsulados no corpo da requisição e não apareçam na URL.

Agora precisaremos criar duas novas rotas na nossa aplicação. A primeira delas deve possibilitar ao usuário o acesso a esse formulário que acabamos de criar, o que é bem simples.

Em `rotas.js`, definiremos a URL `/livros/form` e uma função de *callback* que receberá `req, resp` e será executada sempre que o usuário fizer a requisição para essa rota. Em seguida, devolveremos o *template* do nosso formulário com `resp.marko(require('../views/livros/form/form.marko'))`.

```
app.get('/livros/form', function(req, resp) {
    resp.marko(require('../views/livros/form/form.marko'))
});
```

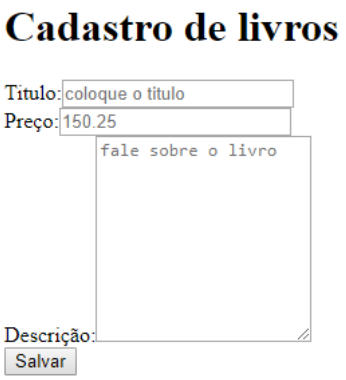
A nossa segunda rota será acessada sempre que o usuário fizer o envio dos dados do formulário. Essa rota, na verdade, já foi definida no nosso formulário:

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

Mas como construiremos uma rota para o método POST do HTTP? É bem simples: utilizaremos a mesma sintaxe que em `app.get()`, substituindo-a pelo método `post()`, mantendo a função *callback* que será chamada. No corpo da função, como queremos apenas imprimir os dados que vieram do formulário, faremos `console.log(req.body)`.

```
app.post('/livros', function(req, resp) {
    console.log(req.body);
});
```

Após salvarmos as alterações no nosso projeto, poderemos testá-lo no navegador. Para isso, executaremos a aplicação e acessaremos a URL http://localhost:3000/livros/form (http://localhost:3000/livros/form). O formulário simples que criamos em `form.marko` será exibido na tela:



Se preenchermos os campos e clicarmos no botão "salvar", o navegador carregará a página por tempo indeterminado. Por que isso acontece?

No arquivo `rotas.js`, quando criamos o método `post()`, definimos apenas que o Node faria a exibição, no console, das informações do corpo da requisição. Ou seja, ainda precisamos definir uma visualização a ser exibida para o usuário, e faremos isso em aulas posteriores.

Por enquanto, precisamos saber se os dados no corpo da requisição realmente foram recebidos. Abrindo o Prompt de Comando, encontraremos... "*undefined*"?

Isso significa que o Node não conseguiu trazer os dados do formulário para o corpo da requisição. Pode ser uma surpresa, mas esse é o comportamento padrão e esperado do Node! Quer resolver essa situação? Pois acompanhe os próximos capítulos!