09

Templates dinâmicos

Transcrição

Nessa etapa, aprenderemos como o Marko ajuda na geração de *templates* dinâmicos.

Anteriormente, conseguimos utilizar o `resp` para chamar o método `marko()`, que recebe a página a ser mostrada no navegador. Além desse parâmetro, o método também pode receber um objeto JavaScript contendo as informações que queremos enviar para a tela. Para isso, só precisamos definir as chaves e os valores desse objeto.

Por exemplo, se queremos fazer uma listagem de livros, podemos definir uma chave chamada `livros`, passando, como valor, um objeto ou um array JavaScript. Nesse caso, preparamos um array com dois livros, cada um com seu `id` e `titulo`:

```
app.get('/livros', function(req, resp) {
    resp.marko(
        require('../views/livros/lista/lista.marko'),
        {
            livros: [
                {
                    id: 1,
                    titulo: 'Fundamentos do Node'
                },
                {
                    id: 2,
                    titulo: 'Node Avançado'
                }
            ]
        }

    );
});
```

Agora precisamos recuperar essa informação no nosso template, e a linguagem Marko possui uma sintaxe que nos auxiliará nesse processo.

Na tag `<tr>` de `lista.marko`, na qual havíamos definido o livro estático, criaremos um `for ()` recebendo `data`, uma variável disponibilizada pelo Marko cujo valor representará o objeto JavaScript que passamos no segundo parâmetro da função `marko()`. Como esse objeto tem uma propriedade livros, podemos acessá-la com o auxílio de uma variável auxiliar `for (data.livros)`.

Agora queremos imprimir o `id` e o `titulo` da variável `livro`. Para isso, usaremos uma *marko expression*, que é uma forma de referenciar uma variável em nossa página. Ela é utilizada com `${}`, passando a referência dentro dessa expressão:

```
<html>
    <head>
        <meta charset="utf-8">
    </head>
    <body>
```

```
        <h1> Listagem de livros </h1>

        <table>
            <tr>
                <td>ID</td>
                <td>Título</td>
            </tr>
                <tr for (livro in data.livros)>
                <td>${livro.id}</td>
                <td>${livro.titulo}</td>
            </tr>
        </table>
    </body>
</html>
```

Feito isso, podemos salvar os arquivos alterados e rodar novamente nossa aplicação. Na URL http://localhost:3000/livros (http://localhost:3000/livros), teremos a seguinte resposta:

> Listagem de livros
>
> ID Título
>
> 1 Fundamentos do Node
>
> 2 Node Avançado

Antes de finalizarmos essa etapa dos nossos estudos, vamos nos atentar a alguns detalhes.

Primeiro, como já aprendemos anteriormente, o método `require()` (que utilizamos para importar a página de listagem para o método `marko()`) na realidade importa módulos Node. Porém, nesse caso, estamos importando um arquivo `.marko`. Estranho, não?

Repare que, quando executamos nossa aplicação, o Marko transforma `lista.marko` em um arquivo `lista.marko.js`, que é um módulo do Node, possibilitando essa importação.

Além disso, você se lembra que, nas aulas anteriores, aprendemos a utilizar o script `npm start` para inicializar nossa aplicação? Apesar disso, nosso instrutor continuou fazendo essa inicialização com `node server.js`! Isso aconteceu porque o Marko e o `nodemon`, a ferramenta instalada para automatizarmos a atualização da aplicação, não funcionam bem em conjunto sem, antes, fazermos algumas configurações.

Quer saber mais sobre o assunto e saber que configurações são essas? Confira o "Para saber mais: Nodemon e Marko (https://cursos.alura.com.br/course/nodejs-fundamentos/task/47652) antes de prosseguir com seus estudos!

Agora que conseguimos passar uma listagem para nossa página, queremos fazer isso por meio de um banco de dados, de modo que nossa listagem fique realmente dinâmica. Começaremos a implementar essa nova funcionalidade a seguir!