02

Trabalhando com JSON

Transcrição

Um recurso interessante nas atuais aplicações é a possibilidade de integração entre as mesmas. Em muitas aplicações encontramos o recurso de permitir a leitura de dados dos produtos ou serviços em formatos diferentes do HTML, por exemplo, o nome e o preço de um item.

Geralmente, o formato usado é o **JSON**, mas encontraremos o **XML**, além de outros. Usaremos este recurso em nossa aplicação. O usuário ao acessar a página de produtos, passando um determinado `id` no caminho `/produtos/{id}`, deverá receber os dados do produto no formato `JSON`.

Como se trata de uma representação dos detalhes de um determinado produto, duplicaremos o método `detalhe` da classe `ProdutosController`.

```
@RequestMapping("/detalhe/{id}")
public ModelAndView detalhe(@PathVariable("id") Integer id){
    ModelAndView modelAndView = new ModelAndView("/produtos/detalhe");
    Produto produto = produtoDao.find(id);
    modelAndView.addObject("produto", produto);
    return modelAndView;
}
```

Não precisaremos mais do `ModelAndView`, então, removeremos todas as linhas referentes a este objeto. Iremos retornar o resultado da chamada do método `produtoDao.find(id)`. O caminho que o método atende ficará apenas `/{id}` e chamaremos este método de `detalheJSON`. Com estas modificações, teremos:

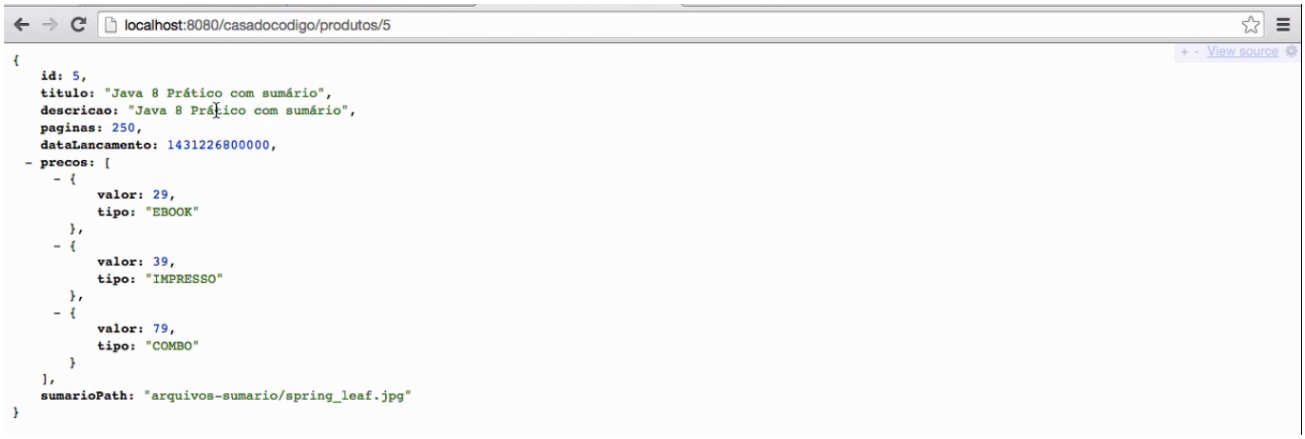
```
@RequestMapping("/{id}")
public Produto detalheJSON(@PathVariable("id") Integer id){
    return produtoDao.find(id);
}
```

Note que o retorno do método em sua assinatura mudou, não estamos mais retornando um objeto `ModelAndView`, mas sim um `Produto`. Podemos testar e ver que algo estranho acontece. Ao reiniciar o servidor e tentar acessar `localhost:8080/casadocodigo/produtos/5` um erro `404` é exibido.

O *Spring* deveria retornar o JSON do livro, no entanto, ele está buscando uma `view` com o mesmo `id` do livro. Não é o que queremos. Para que o *Spring* consiga responder a requisição da forma que queremos neste caso, usaremos a anotação `ResponseBody` no método `detalheJSON` da classe `ProdutosController`.

```
@RequestMapping("/{id}")
@ResponseBody
public Produto detalheJSON(@PathVariable("id") Integer id){
    return produtoDao.find(id);
}
```

Dessa forma, estamos dizendo para o *Spring* que o retorno daquele método será o corpo da resposta da requisição. Se acessarmos novamente `localhost:8080/casadocodigo/produtos/5`, teremos:



Funcionou! Temos um livro no formato JSON como resposta da requisição!

É importante notar que só funcionou, porque o `Jackson` está configurado no projeto. Se não tivéssemos configurado a biblioteca anteriormente no primeiro módulo do curso, o recurso não teria funcionado. Teríamos problema também, caso mais de uma biblioteca estivesse configurada para transformar objetos em `JSON`, no projeto. Haveria conflito e teríamos que realizar novas configurações para obter o mesmo resultado.