02

Removendo recursos e subrecursos

O que queremos fazer agora no nosso servidor é ser capaz de remover um produto de um carrinho. Eu tenho um videogame mas agora quero tirar este produto do meu carrinho. Isto é, tenho uma URI que representa o meu carrinho. Eu quero remover, deletar, esse carrinho? Não, eu quero remover um produto de dentro do carrinho, deletar um produto de dentro do carrinho.

O videogame tem o id 6237 dentro do carrinho 1. Como quero remover somente este produto, preciso de uma URI que identifique somente este produto dentro deste carrinho. Se o carrinho está na URI `/carrinhos/1` , que tal usarmos a URI `/carrinhos/1/produtos/6237` ?

É comum chamarmos isso de subrecurso, como se fosse um subrecurso dentro de outro. Não há problemas em chamar essas URIs de subrecurso, mas é importante lembrar que a estrutura da URI não significa nada. A URI é só um atalho, um apontador, um indicador de onde se encontra algo. O path de uma URI não possui estrutura, portanto para o navegador, para o cliente, para a internet, `/carrinhos/1/produtos/6237` e `/98hr2f0j/1/198jfef/6237` ambos identificam recursos, e nenhum significa que é recurso e subrecurso. Como seres humanos sabemos que essa URI está tentando nos indicar algo, mas o padrão da internet - e todos os softwares - não tem como saber isso.

Escolhemos nossa URI `/carrinhos/1/produtos/6237` , que é um padrão interessante para facilitar nosso trabalho, um padrão que somente o servidor e o cliente conhecerão, a internet desconhece hierarquia, subrecursos etc. Gostaria de fazer então um DELETE nesse produto, ao invés de um GET ou POST, quero DELETE este produto. Vamos no nosso servidor e adicionamos um novo método para essa URI que recebe os dois ids:

```
@Path("{id}/produtos/{produtoId}")
@DELETE
public Response removeProduto(@PathParam("id") long id, @PathParam("produtoId") long produto

}
```

Quero devolver um status de sucesso, então devo utilizar o `Response.ok()` :

```
@Path("{id}/produtos/{produtoId}")
@DELETE
public Response removeProduto(@PathParam("id") long id, @PathParam("produtoId") long produto
    return Response.ok().build();
}
```

Quero agora remover, para isso usarei o banco de dados, lembre-se: sempre faça a sua lógica de negócios, não importa qual ela seja. Vou primeiro buscar o carrinho:

```
Carrinho carrinho = new CarrinhoDAO().busca(id);
```

E agora remover o produto baseado no id dele:

```
        Carrinho carrinho = new CarrinhoDAO().busca(id);
        carrinho.remove(produtoId);
        return Response.ok().build();
```

Maravilha, dessa maneira eu busquei o carrinho, removi o produto, salva (no nosso caso é tudo na memória, ao remover, ele automaticamente 'é salvo'). O que fizemos então? Criamos um método que suporta DELETE ao invés de GET ou POST, ele tem uma URI com dois parâmetros ao invés de um (não há limites), busca o carrinho, remove o produto e retorna 200 OK. O DELETE e o número de parâmetros são as novidades técnicas aqui, além da criação de um subrecurso via URI ser a nova teoria.

Levantamos o servidor e buscamos o carrinho 1, ele se mostra como o de sempre. Para remover o carrinho, usamos o `curl` com a opção de DELETE:

```
curl -v -X DELETE http://localhost:8080/carrinhos/1/produtos/6237
```

E recebemos o OK:

```
< HTTP/1.1 200 OK
< Date: Thu, 24 Apr 2014 17:59:35 GMT
< Content-Length: 0
```

Verificamos o carrinho, que agora não possui mais o videogame:

```
<br.com.alura.loja.modelo.Carrinho>
  <produtos>
    <br.com.alura.loja.modelo.Produto>
      <preco>60.0</preco>
      <id>3467</id>
      <nome>Jogo de esporte</nome>
      <quantidade>2</quantidade>
    </br.com.alura.loja.modelo.Produto>
  </produtos>
  <rua>Rua Vergueiro 3185, 8 andar</rua>
  <cidade>S?o Paulo</cidade>
  <id>1</id>
</br.com.alura.loja.modelo.Carrinho>
```

Além do verbo GET para trazer representações de um recurso, POST para criar recursos, temos outros verbos HTTP, inclusive um que remove, o DELETE, que diferente do POST e do GET, não recebe nem envia a representação, simplesmente remove o recurso identificado por aquela URI, no nosso caso um produto de dentro de um carrinho.

Se eu executar duas vezes a mesma requisição DELETE o que acontece? A primeira retorna 200 e a segunda o que? Ele é idempotente?

Vamos aos exercícios?