02

Atualizando recursos e o PUT

O próximo passo que nosso cliente pede é implementar o um serviço REST para alterar a quantidade de um produto em nosso carrinho. Não quero fazer o DELETE, quero outra coisa. Lembra de nosso primeiro carrinho? Ele tem um videogame e dois jogos de esporte. Não quero dois jogos de esporte, quero alterar para um único jogo, como fazer isso? Como trocar de dois para um, como trocar parte do produto.

Ao invés de fazer um GET para trazer informações, POST para criar um recurso, DELETE para apagar, vou fazer uma alteração, trocar, PUT. PUT, troca o recurso por um outro recurso.

Se eu fizer um PUT para '/carrinhos/1/produtos/3467' (que identifica nossos jogos de esporte), eu posso passar uma representação do meu produto que será colocada para substituir o produto em questão. Como fazer isso? Passo exatamente o XML que substituirá o recurso de nosso produto, mas agora com quantidade 1:

```
<br.com.alura.loja.modelo.Produto>
  <preco>60.0</preco>
  <id>3467</id>
  <nome>Jogo de esporte</nome>
  <quantidade>1</quantidade>
</br.com.alura.loja.modelo.Produto>
```

Vou no meu servidor e crio um método que é acessado via PUT para a mesma URI do produto, recebendo os parâmetros que indicam o carrinho, o produto e o conteúdo XML:

```
@Path("{id}/produtos/{produtoId}")
@PUT
@Consumes(MediaType.APPLICATION_XML)
public Response alteraProduto(@PathParam("id") long id, @PathParam("produtoId") long produto
    return Response.ok().build();
}
```

Buscamos nosso carrinho:

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

E alteramos a quantidade de um produto. Como o cliente enviou a representação (e não um mero int... lembre-se isto é REST, são representações que são enviadas de um lado para o outro), devemos deserializar o produto:

```
Carrinho carrinho = new CarrinhoDAO().busca(id);
Produto produto = (Produto) new XStream().fromXML(conteudo);
return Response.ok().build();
```

Falta agora trocar a quantidade do produto, isto é, pego o meu carrinho e chamo o método de troca:

```
    @Path("{id}/produtos/{produtoId}")
    @PUT
    @Consumes(MediaType.APPLICATION_XML)
    public Response alteraProduto(@PathParam("id") long id, @PathParam("produtoId") long produto
        Carrinho carrinho = new CarrinhoDAO().busca(id);
        Produto produto = (Produto) new XStream().fromXML(conteudo);
        carrinho.troca(produto);
        return Response.ok().build();
    }
```

Faremos agora um teste com um PUT para esta URI passando esse produto com a quantidade um, uma vez que ele inicia com quantidade dois. Para conferir levantamos o servidor e fazemos o get inicial, temos dois jogos:

```
<br.com.alura.loja.modelo.Carrinho>
  <produtos>
    <br.com.alura.loja.modelo.Produto>
      <preco>4000.0</preco>
      <id>6237</id>
      <nome>Videogame 4</nome>
      <quantidade>1</quantidade>
    </br.com.alura.loja.modelo.Produto>
    <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>São Paulo</cidade>
  <id>1</id>
</br.com.alura.loja.modelo.Carrinho>
```

Vamos executar agora o método (`-x`) PUT, com os dados do produto de quantidade um:

```
curl -v -X PUT -H "Content-Type: application/xml" -d "<br.com.alura.loja.modelo.Produto>      <
```

O resultado é o 200:

```
< HTTP/1.1 200 OK
< Date: Fri, 25 Apr 2014 14:37:16 GMT
< Content-Length: 0
```

E se fizemos o GET novamente temos que nosso recurso foi trocado com sucesso:

```
<br.com.alura.loja.modelo.Carrinho>
  <produtos>
    <br.com.alura.loja.modelo.Produto>
```

```
      <preco>4000.0</preco>
      <id>6237</id>
      <nome>Videogame 4</nome>
      <quantidade>1</quantidade>
    </br.com.alura.loja.modelo.Produto>
    <br.com.alura.loja.modelo.Produto>
      <preco>60.0</preco>
      <id>3467</id>
      <nome>Jogo de esporte</nome>
      <quantidade>1</quantidade>
    </br.com.alura.loja.modelo.Produto>
  </produtos>
  <rua>Rua Vergueiro 3185, 8 andar</rua>
  <cidade>São Paulo</cidade>
  <id>1</id>
</br.com.alura.loja.modelo.Carrinho>
```

Então usamos o GET para receber a representação de um recurso, POST para criar, DELETE para remover e PUT para substituir. Mas claro que temos um problema aqui, o cliente, o usuário final pode alterar o preço do produto na hora de enviar a representação do produto, isso pois ele é *obrigado* pela definição do PUT a enviar *toda* a representação, ele deve enviar o preço e tudo mais. Não podemos falar no PUT para enviar somente a quantidade e o ID, isso quebra o protocolo HTTP.

Então o que quero fazer agora é substituir somente um pedaço do meu recurso, não todo o recurso. Poderia criar então um recurso que representa somente a quantidade de um produto dentro de um carrinho. Ao invés de utilizar a URI `/carrinhos/{id}/produtos/{produtoId}` para atualizar o produto inteiro dentro de um carrinho, crio uma URI `/carrinhos/{id}/produtos/{produtoId}/quantidade` que representa a quantidade de um produto dentro do carrinho... agora sim levaremos em conta somente o campo quantidade, mais nenhum campo. Também alteramos a chamada ao método `troca` para `trocaQuantidade` para que usemos somente a quantidade, ignorando qualquer outro campo:

```
    @Path("{id}/produtos/{produtoId}/quantidade")
    @PUT
    @Consumes(MediaType.APPLICATION_XML)
    public Response alteraProduto(@PathParam("id") long id, @PathParam("produtoId") long produto
        Carrinho carrinho = new CarrinhoDAO().busca(id);
        Produto produto = (Produto) new XStream().fromXML(conteudo);
        carrinho.trocaQuantidade(produto);
        return Response.ok().build();
    }
```

Essa é uma das maneiras de seguir o PUT ao pé da letra: ao invés de trocar o recurso inteiro, troco somente a parte que representa a quantidade, 'criando' um recurso para ela. Testamos novamente, agora com a nova URI:

```
curl -v -X PUT -H "Content-Type: application/xml" -d "<br.com.alura.loja.modelo.Produto> <id>346
```

E temos o mesmo resultado de sucesso! Sempre que usamos o PUT devemos pensar com calma: quero trocar o recurso inteiro ou somente uma parte dele? Se é o recurso inteiro, utilize a URI que o representa, caso contrário, utilize uma nova URI que representa a parte a ser trocada.