02

Serialização com JAXB

A gente já viu como serializar, isso é, como pegar aquele nosso recurso e transformar ele em uma representação, tanto no lado servidor como no cliente, usando o que? Xstream, usando json, poderia usar qualquer biblioteca. Mas por padrão, o JAX-RS já suporta o JAXB, o serializador padrão da plataforma Java SE. Vamos usar o JAXB?

Para usar o JAXB, o que a gente vai fazer é: pegar aqui o nosso carrinho resource e, em vez do que, devolver o:

```
return carrinho.toXML();
```

Simplesmente devolva o carrinho. Devolve o carrinho, acabou:

```
return carrinho;
```

Devolve o carrinho, feliz e contente. Só que ele precisa saber que esse carrinho é um cara suportado pelo nosso colega JAXB. Então a gente vai pegar o nosso carrinho e vai anotar com:

```
@XmlRootElement
```

Quando a gente fala `XmlRootElement` , a gente está falando, olha, esse elemento aqui é um elemento válido do XML do JAXB. A gente vai anotar também falando que o:

```
@XmlAccessorType
```

ele é:

```
@XmlAccessorType(XmlAccessType.FIELD)
```

Então quer dizer que todos os campos serão serializados por padrão. Legal, vou fazer a mesma coisa no produto. No produto eu também vou colocar as duas anotações.

Tenho que tomar um cuidado. O JAXB vai pedir pra mim um construtor sem argumentos. Então eu vou colocar aqui o construtor sem argumentos. A mesma coisa no carrinho. O construtor sem argumentos, no caso do carrinho, já existe. Então, legal, já temos o construtor sem argumentos aqui. O que a gente faz agora? A gente vem no nosso `CarrinhoResource` e vai devolver o carrinho. Vamos testar? Como é que a gente faz isso? Roda o nosso servidor, estou rodando ele, e agora eu vou lá no meu navegador e tento acessar a `http://localhost:8080/carrinhos/1`

Maravilha, continua funcionando! Por que? Porque ele passou a serizalizar o meu carrinho utilizando o JAXB. A única diferença é que no padrão do jaxJAXBb, o nome dos campos é diferente.

```
<carrinho>
<produtos>
<produtos>
```

```
</produtos>
</carrinho>
```

Então ele usa um padrão diferente por padrão do que o Xstream. É claro, tanto no caso do JAXB, no caso do Xstream, você vai querer customizar esse XML, esse JSON, seja lá o que for, na maneira que você tem interesse. E aí você configura a vontade.

Legal, consegui devolver o carrinho. Mas e a nosso API de cliente? Você lembra a nossa classe de teste? A classe de teste, ela também trabalha com carrinho, e a gente serializou via Xstream. Hum...

A gente queria usar o JAXB aqui, né? Então, o que eu posso fazer? Eu posso, na hora que eu falo que eu estou pegando esse cara, em vez de devolver pra mim uma String, me devolve, por favor, um carrinho.

```
Carrinho carrinho = target.path("/carrinhos/1").request().get(Carrinho.class);
```

Pronto, ele vai fazer a mesma coisa, ele vai perceber que o carrinho já tem a anotação. Que anotação? A anotação do JAXB e vai falar, opa, usa o JAXB para testar esse cara aqui, para desserializar esse cara.

Vamos rodar só este teste? Clico da direita, *Run As* > *JUnit Test*, estou rodando só este teste. Legal, verde, ele desserializou usando JAXB. E agora, na hora de criar um novo carrinho? Quando eu crio um novo carrinho, eu tenho dois passos. O segundo passo é o passo que a gente já conhece, o passo em que a gente traz o carrinho de volta. Então eu posso pegar o carrinho aqui.

```
Carrinho carrrinho = cliente.target(location).request().get(Carrinho.class);
```

Igual a esse carrinho. Legal, aí eu posso assertar que é:

```
Assert.assertEquals("Microfone", carrinho.getProdutos().get(0).getNome());
```

O nome do primeiro produto tem que ser “Microfone”, legal? Então eu estou fazendo esse meu GET e vou pegar ele aqui.

A única coisa é que já existe uma variável chamada carrinho ali atrás e esse daqui é o `carrinhoCarregado`.

Legal, só que qual XML que eu envio? Aqui, lembra, eu estou usando o Xstream, eu não quero usar o Xstream, eu quero que ele automaticamente use o JAXB para serializar. Mas ainda falta fazer o post sem ter que serializar usando o `toXml`.

Não quero fazer `toXml`, não quero usar o Xstream, quero usar o padrão, que é o JAXB. Se eu quero usar o padrão, eu não passar aqui XML, eu vou passar direto o carrinho, quem nem a gente fez no GET. Recebeu direto o carrinho, então eu vou passar direto o carrinho. Em vez de ser uma entidade de `String`, vai ser uma:

```
Entity<Carrinho> entity = Entity.entity(carrinho, MediaType.APPLICATION_XML);
```

Claro. Então esse é meu código do cliente. Só lá no servidor, vamos dar uma olhadinha no nosso post, no nosso servidor, a gente está recebendo uma `String conteudo` e lendo através de XML Xstream. Não é pra ler através de XML Xstream, certo? A gente vai ler através do padrão, que é o JAXB, então:

```
public Response adiciona(Carrinho carrinho) {
   new CarrinhoDAO().adiciona(carrinho);
```

Recebi direto o meu carrinho. Legal? Então vou salvar esse cara, e agora eu posso rodar o meu teste. Vou lá, executo, executo os testes, os dois testes estão verdes.

Então, no cliente, bastou eu utilizar direto o carrinho sem ter que serializar. No servidor, bastou eu usar direto o carrinho sem ter que ficar desserilizando e serializando, porque eu estou usando por padrão as anotações `@XmlRootElement` , `@XmlAccessorType` para que ele serialize e desserialize os meus fields, os meus campos. Lembrando que eu preciso daquele construtor, mesmo que seja `default` , não precisa ser público, sem argumentos, porque senão o JAXB vai reclamar por padrão. Legal?

Por fim, aqui embaixo no nosso resource, a gente tem o `alteraProduto` , que também recebe `String conteudo` . Mas ele desserializa pra que? Pra um produto, então eu vou usar aqui:

```
Produto produto
```

Legal. Então lembra, como o produto já está anotado com a anotação do JAXB, basta eu receber o produto no mesmo formato que a coisa funciona bonitinha. Legal? Então assim eu passei a usar o JAXB, que é o conversor padrão do JAX-RS do Java SE. Mas você viu também como você pode fazer para usar o Xstream, se você preferir, ou o JSON, ou qualquer outro serializador, desserializador que você tem interesse. Vamos pros exercícios?