09

Content negotiation

Transcrição

Nossa aplicação já está retornando JSON e XML. Mas no XML, antes de mais nada, precisamos ver como ele está sendo gerado. Ele usou `<collection>` como tag principal e isso pode incomodar um pouco já que queremos um XML mais organizado. Para solucionar isso teremos que utilizar uma configuração do RESTEasy em cima do método `List<Livro>` :

```
//outras annotations

@Wrapped(element = "livros")
public List<Livros> ultimosLancamentosXml() {
    return dao.ultimosLancamentos();
}
```

O pacote do `@Wrapped` é o `org.jboss.resteasy.annotations.provider.wrapper` . Dentro dele podemos passar qual elemento raiz queremos chamar, no caso "livros". Reiniciamos o servidor e atualizamos a página. Veremos que, de fato, `<livros>` vira a tag principal:

```
<livros>
    <livro>
        <autores>
            <id>3</id>
            <nome>Guilherme Silveira</nome>
        </autores>
        <autores>
            <id>1</id>
            <nome>Paulo Silveira</nome>
        </autores>
        //...
    </livro>
</livros>
```

Modificamos um pouco para que possamos observar um erro: cada autor está sendo chamado de "autores". A estrutura ideal seria:

```
<autores>
    <autor>
        <id>3</id>
        <nome>Guilherme Silveira</nome>
    </autor>
    <autor>
        <id>1</id>
        <nome>Paulo Silveira</nome>
    </autor>
</autores>
```

Dentro de da tag `<autores>` temos a tag `<autor>` para cada autor. Nosso XML fica mais consiso dessa forma. Quando tratamos de integração devemos deixar o código o mais expressivo possível para que os sistemas que irão se integrar ao

nosso não "sofram" com qualquer tipo de configuração.

Para resolvermos esse problema como um todo, faremos algumas configurações do JAXB em cima da Classe `Livro`. A primeira anotação será a `@XmlAccessorType` informando que queremos o acesso através do campo (*field*):

```
//outras annotations
@XmlAccesorType(XmlAccessorType.FIELD.)
public class Livro {

    //...
}
```

Quando informamos que queremos o tipo de acesso do tipo *field*, será acessado diretamente o campo, como `titulo`, `descricao`, etc. Com isso também podemos colocar um `Wrapper` em cima do método dos autores, assim como indicar que o elemento é `autor`:

```
//outras annotations
@XmlElement(name="autor")
@XmlElementWrapper(name="autores")
private List<Autor> autores = new ArrayList<>();
```

Reiniciamos o servidor e testando novamente:

```
<livros>
    <livro>
        ....
        <autores>
            <autor>
                ...
            </autor>
            <autor>
            ...
            </autor>
        </autores>
    </livro>
</livros>
```

Perceba que agora sim que todas as tags `<autor>` estão dentro de `<autores>`.

Agora tanto o XML quanto o JSON estão funcionando perfeitamente. mas como fazemos para mudar de um para outro? XML para JSON? Neste momento temos um método para cada:

```
//JSON
@GET
@Path("json")
@Produces({MediaType.APPLICATION_JSON})
public List<Livro> ultimosLancamentosJson() {
    return dao.ultimosLancamentos();
}

//XML
```

```
@GET
@Path("xml")
@Produces({MediaType.APPLICATION_XML})
@Wrapped(element = "livros")
public List<Livros> ultimosLancamentosXml() {
    return dao.ultimosLancamentos();
}
```

Os códigos são praticamente iguais, fizemos um "Copy & Paste", o que é uma das piores implementação para se manter. Para melhorar isso primeiramente vamos passar o `Wrapper` para o método de cima e o método não será json nem xml, apagamos o método de baixo:

```
@GET
@Path("json")
@Produces({MediaType.APPLICATION_JSON})
@Wrapped(element = "livros")
public List<Livro> ultimosLancamentos() {
    return dao.ultimosLancamentos();
}
```

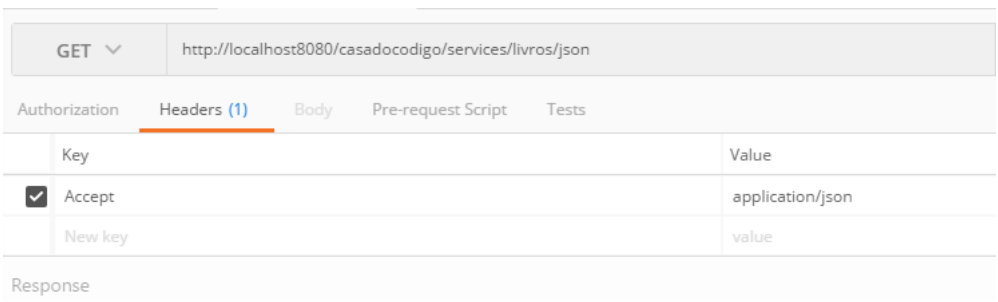
O `@Produces` já é um *array*, ou seja, podemos fazer:

```
@GET
@Path("json")
@Produces({MediaType.APPLICATION_JSON, MediaType.APPLICATION_XML})
@Wrapped(element = "livros")
public List<Livro> ultimosLancamentos() {
    return dao.ultimosLancamentos();
}
```

Dessa forma já conseguimos produzir os dois. Isso é entendido pelo JAX-RS. Se testarmos, por default é retornado o XML. Para pegarmos o JSON usaremos o protocolo HTTP que irá definir o tipo que iremos pegar. Primeiro vamos mudar o `@Path` :

```
@Path("lancamentos")
```

Agora usaremos o Postman, trocando o Accept daquela nossa requisição para "json":



Enviando um novo *Request* teremos um JSON. Pelo navegador a configuração seria bem mais complicada, por isso usamos o Postman.

É isso que faz a diferença, que faz a negociação do tipo de resultado que teremos. Por isso que ele se chama *Content Negociation*, ele é um conceito geral do REST no qual podemos ter dois formatos sendo atendidos pela aplicação. Quem precisa de JSON usa JSON, quem precisa de XML usa XML. Nossa aplicação fica flexível e elegante. Melhor ainda se não mantivermos no path da url qualquer informação desses dois. Se, por exemplo, o YML se tornar um formato padrão, será bem simples mudarmos.

Veja que o Content Negociation pode ficar tanto em cima do método, como fizemos, como em cima de toda a classe:

```
@Produces({MediaType.APPLICATION_JSON, MediaType.APPLICATION_XML})
public class LivroResource {

    //...
}
```