11

(Desafio) XmlAdapter

Todas as mensagens SOAP enviadas para o serviço de cadastro de ítens, agora precisam possuir o *Token* para autenticação. Esse token, que é enviado no Header da mensagem, possui um código além de uma data de validade. Por exemplo:

```
<soapenv:Header>
      <ws:tokenUsuario soapenv:mustUnderstand="1">
         <token>AAA</token>
         <dataValidade>2015-12-31T00:00:00</dataValidade>
      </ws:tokenUsuario>
   </soapenv:Header>
```

No entanto, estamos enviando a data no formato UTC que é pouco amigável para humanos. Como podemos melhorar este formato, fazendo com que o JAX-B consiga entender uma data no estilo `dd/MM/yyyy` ?

```
<soapenv:Header>
      <ws:tokenUsuario soapenv:mustUnderstand="1">
         <token>AAA</token>
         <dataValidade>31/12/2015</dataValidade>
      </ws:tokenUsuario>
   </soapenv:Header>
```

Precisamos ensinar ao JAX-B que quando ele encontrar o valor do atributo `dataValidade` no formato `dd/MM/yyyy` , este valor deve ser convertido em um date (unmarshal).

Ou seja, vamos criar uma classe que fará essa adaptação para nós.

```
public class DateAdapter {

    private String pattern = "dd/MM/yyyy";

    public Date unmarshal(String dateString) throws Exception {
        return new SimpleDateFormat(pattern).parse(dateString);
    }

}
```

De forma inversa, devemos ensinar o JAX-B a converter um Date em uma String do tipo `dd/MM/yyyy` (marshal).

```
public String marshal(Date date) throws Exception {
   return new SimpleDateFormat(pattern).format(date);
}
```

Nossa classe `DateAdapter` deverá ficar algo como:

```
public class DateAdapter {
```

```
    private String pattern = "dd/MM/yyyy";

    public Date unmarshal(String dateString) throws Exception {
       return new SimpleDateFormat(pattern).parse(dateString);
    }

    public String marshal(Date date) throws Exception {
       return new SimpleDateFormat(pattern).format(date);
    }

}
```

Devemos agora dizer na nossa classe `TokenUsuario` que gostariamos de usar o Adapter que criamos para o atributo `dataValidade`. Para isso, usaremos a anotação `@XmlJavaTypeAdapter`:

```
@XmlAccessorType(XmlAccessType.FIELD)
public class TokenUsuario {

    @XmlElement(required=true)
    private String token;

    @XmlJavaTypeAdapter(DateAdapter.class)
    @XmlElement(required=true)
    private Date dataValidade;

    // código omitido
```

Para finalizar, precisamos dizer que essa classe é um adapter do JAX-B. Para isso, iremos extender a classe abstrata `XmlAdapter`.

Como ficará a classe `DateAdapter`?

Responda

INSERIR CÓDIGO		FORMATAÇÃO