06

Serviço com XML

Transcrição

Bem vindo de volta! No vídeo anterior, criamos nosso primeiro Web Service em Java usando os Servlets. Isso prova que não é necessário usar bibliotecas sofisticadas para criar um Web Service, já que eles somente utilizam o protocolo HTTP e devolvem os dados - normalmente em JSON ou XML. Essa devolução poderia ser feita em outro formato, mas esses são os que dominaram o mercado.

Existem serviços mais sofisticados com Web Services, mas a simplicidade faz mais sucesso. Por esse motivo, serviços de grandes empresas, como Google, Amazon e Microsoft, trabalham com HTTP e JSON.

Evidentemente, se estivéssemos trabalhando com um controlador como o Spring MVC, não precisaríamos fazer manualmente a concatenação para JSON, pois ele já tem essa funcionalidade embutida.

Dessa vez, queremos que a devolução seja feita em XML. O melhor é que esse processo é praticamente o mesmo que fizemos para o JSON: nós copiaremos uma biblioteca que sabe fazer esse trabalho, utilizaremos essa biblioteca para transformar nosso objeto em XML e o devolveremos na resposta.

Existem duas bibliotecas muito populares para esse processo. Uma delas é o JAXP, que faz parte dos padrões Java. Porém, hoje em dia é necessário configurá-lo, pois o módulo JAXP não é carregado automaticamente, o que gera problemas com o compilador JRE.

Por isso, utilizaremos outra biblioteca, chamada XStream, que pode ser baixada aqui (https://caelum-online-public.s3.amazonaws.com/1001-servlets-parte2/06/xstream-1.4.10-jars.zip). Se quiser conhecê-la melhor, saiba que temos um curso de XStream (https://cursos.alura.com.br/course/xstream) aqui na Alura!

Agora basta extrair os arquivos `xmlpull-1.1.3.1.jar` e `xstream-1.4.10.jar` do `xstream-1.4.10-jars.zip` e copiá-los para dentro da pasta "WEB-INF/lib". Em seguida, vamos copiar e colar o código que escrevemos na aula anterior para a concatenação de JSON, alterando os parâmetros para se adequarem à nossa biblioteca `XStream` e à nossa variável `xml` :

```
XStream xstream = new XStream();
String xml = xstream.toXML(empresas);

response.setContentType("application/xml");
response.getWriter().print(xml);
```

COPIAR CÓDIGO

Dica: não se esqueça de comentar o código anterior!

Viu como é simples? Quando reiniciamos o servidor e testamos no navegador, obtemos:

```
This XML file does not appear to have any style information as
<list>
  <br.com.alura.gerenciador.modelo.Empresa>
    <id>1</id>
    <nome>Alura</nome>
    <dataAbertura>2018-11-01 18:15:04.842 UTC</dataAbertura>
  </br.com.alura.gerenciador.modelo.Empresa>
  <br.com.alura.gerenciador.modelo.Empresa>
    <id>2</id>
```

```
    <nome>Caelum</nome>
    <dataAbertura>2018-11-01 18:15:04.842 UTC</dataAbertura>
  </br.com.alura.gerenciador.modelo.Empresa>
</list>
```

COPIAR CÓDIGO

Conseguimos receber um XML corretamente. Repare que o XML também têm elementos que abrem e fecham, como `<list></list>`. Porém, ele precisa definir o nome do elemento. Isso é fácil quando temos atributos como `id`, mas em `<br.com.alura.gerenciador.modelo.Empresa>` ele retornou o *full qualified name* da classe, e não o nome simples.

Para melhorarmos a apresentação do nosso XML, vamos configurar essa exibição por meio do método `alias()`:

```
XStream xstream = new XStream();
xstream.alias("empresa", Empresa.class);
String xml = xstream.toXML(empresas);
```

COPIAR CÓDIGO

Dessa forma, toda vez que a biblioteca encontrar um objeto do tipo `Empresa`, ele escreverá apenas `empresa`:

```
This XML file does not appear to have any style information as
<list>
  <empresa>
    <id>1</id>
    <nome>Alura</nome>
    <dataAbertura>2018-11-01 18:15:04.842 UTC</dataAbertura>
  </empresa>
  <empresa>
    <id>2</id>
    <nome>Caelum</nome>
    <dataAbertura>2018-11-01 18:15:04.842 UTC</dataAbertura>
```

```
    </empresa>
</list>
```

COPIAR CÓDIGO

Pronto, nosso XML está mais expressivo.

Agora vamos estabelecer uma nova motivação: queremos deixar nosso Web Service flexível, de modo que o cliente (o navegador, o celular ou uma outra aplicação Java) possa pedir o JSON ou o XML na requisição. A partir dessa informação, o servidor devolverá o formato correto.

Faremos isso no próximo vídeo. Até lá!