02

# Definindo comportamentos

### Transcrição

No último vídeo , nós conseguimos criar um conversor que exibe o conteúdo das nossas tags em um HTML. Entretanto, queremos que esse arquivo seja formatado de maneira mais legível, o que pode ser feito com o auxílio do nosso arquivo XSL, descrevendo qual comportamento deve ser adotado para cada tag XML encontrada.

Em `xmlParaHtml.xsl` , criaremos uma tag `<xsl:template>` com o atributo `match` que deverá ser aplicado à tag `venda` , que é a primeira do nosso documento. Quanto essa tag for encontrada, imprimiremos um título ( `<h2>` ) com o texto "Venda".

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">]
    <xsl:template match="venda">
        <h2>Venda</h2>
    </xsl:template>

</xsl:stylesheet>
```

Para termos certeza de que a conversão está funcionando, imprimiremos uma mensagem "Conversão concluída" no método `main()` de `ConverserParaHtml.java` .

```
public class ConversorParaHtml {

    public static void main(String[] args) throws Exception {
        InputStream xsl = new FileInputStream("src/xmlParaHtml.xsl");
        StreamSource xslSource = new StreamSource(xsl);

        InputStream dados  = new FileInputStream("src/vendas.xml");
        StreamSource xmlSource = new StreamSource(dados);

        StreamResult saida = new StreamResult("src/vendas.html");

        Transformer transformer = TransformerFactory.newInstance().newTransformer(xslSource);
        transformer.transform(xmlSource, saida);

        System.out.println("Conversão concluída");
    }

}
```

Após executarmos o código, nosso arquivo `vendas.html` trará o seguinte retorno:

```
<?xml version="1.0" encoding="UTF-8"?><h2>Venda</h2>
```

Nosso título foi adicionado com sucesso, mas os outros valores não foram carregados. A partir do momento que definimos um template para uma tag, o comportamento padrão (de imprimir somente os valores) deixa de funcionar

para essa tag e seus filhos. Como `<Venda>` engloba todas as outras tags do nosso XML, nenhum dos seus valores é convertido.

O próximo passo será exibirmos a `<formaDePagamento>`. Com `<p>`, iniciaremos um parágrafo com o texto "Forma de pagamento:" seguido do valor dentro da tag `<formaDePagamento>`. Para isso, usaremos a tag `<xsl:value-of>` com o atributo `select`.

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">]
    <xsl:template match="venda">
        <h2>Venda</h2>
        <p>Forma de pagamento: <xsl:value-of select="formaDePagamento" /></p>
    </xsl:template>

</xsl:stylesheet>
```

Nosso HTML então será atualizado com sucesso:

```
<?xml version="1.0" encoding="UTF-8"?><h2>Venda</h2><p>Forma de pagamento: Cartão</p>
```

Repetiremos o processo, dessa vez definindo como queremos que cada `<produto>` apareça. Criaremos uma nova tag `xsl:tempalte>` com o atributo `match` que deverá ser aplicado à `produto`. Como comportamento, mostraremos o valor do campo `nome` em uma título `<h3>`, que é de menor importância. Por fim, em uma tag de parágrafo, adicionaremos também o valor dentro de `preco` precedido por "R$:", e separaremos cada produto com uma linha horizontal `<hr />`.

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">]
    <xsl:template match="venda">
        <h2>Venda</h2>
        <p>Forma de pagamento: <xsl:value-of select="formaDePagamento" /></p>
    </xsl:template>
    <xsl:template match="produto">
        <h3><xsl:value-of select="nome" /></h3>
        <p>R$: <xsl:value-of select="preco" /></p>
    </xsl:template>
    <hr />

</xsl:stylesheet>
```

Se executarmos nosso conversor novamente, nada mudará em `vendas.html`. Isso acontece porque o conversor já encontrou um template, ignorando o resto. Para solucionarmos tal problema, usaremos a tag `<xsl:apply-templates>` com o atributo `select` aplicado em `produtos`.

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">]
    <xsl:template match="venda">
        <h2>Venda</h2>
        <p>Forma de pagamento: <xsl:value-of select="formaDePagamento" /></p>
        <xsl:apply-templates select="produtos"/>
    </xsl:template>
```

```
    <xsl:template match="produto">
        <h3><xsl:value-of select="nome" /></h3>
        <p>R$: <xsl:value-of select="preco" /></p>
    </xsl:template>
    <hr />

</xsl:stylesheet>
```

Também nos esquecemos de definir o comportamento para a tag `<produtos>`. Repetindo o procedimento que já conhecemos, criaremos outro template que irá simplesmente repassar a execução para a tag `<produto>`, respeitando a hierarquia.

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">]
    <xsl:template match="venda">
        <h2>Venda</h2>
        <p>Forma de pagamento: <xsl:value-of select="formaDePagamento" /></p>
        <xsl:apply-templates select="produtos"/>

    <xsl:template match="produtos">
    <xsl:apply-templates select="produto"/>
    </xsl:template>

    </xsl:template>
    <xsl:template match="produto">
        <h3><xsl:value-of select="nome" /></h3>
        <p>R$: <xsl:value-of select="preco" /></p>
        <hr />
    </xsl:template>


</xsl:stylesheet>
```

Executando novamente o conversor, o HTML será atualizado com as novas informações:

```
<?xml version="1.0" encoding="UTF-8"?><h2>Venda</h2><p>Forma de pagamento: Cartão</p>
        <h3>Livro de xml</h3><p>R$: 29.90</p><hr/>
        <h3>Livro de O.O. java</h3><p>R$: 29.90</p><hr/>
```

Assim, conseguimos converter um arquivo XML para o formato HTML!