05

Conhecendo JSTL

Transcrição

Nosso objetivo é melhorar o laço de scriptlet em `listaEmpresas.jsp` :

```
<!DOCTYPE hmtl>
<html>
<head>
<meta charset="ISO-8859-1">
<title>Insert title here</title>
</head>
<body>
    Lista de empresas: <br />
    <ul>
    <%
        List<Empresa> lista = (List<Empresa>)request.getAttribu
        for (Empresa empresa : lista) {
    %>
        <li><%= empresa.getNome() %></li>
    <%
        }
    %>
    </ul>

</body>
</html>
```

COPIAR CÓDIGO

Já conseguimos melhorar o aspecto de `NovaEmpresaCriada.jsp` , eliminando o scriptlets por meio da *expression language*, marcada pelo uso de `${ }` . A

expressão a ser interpretada por essa linguagem pode ser um cálculo matemático ou o nome de uma variável. No caso de nosso código, utilizamos o nome `empresa`, que será buscada pela *expression language* como atributo na requisição.

```
<html>
    <body>
        Empresa ${ empresa } cadastrada com sucesso!
    </body>
</html>
```

COPIAR CÓDIGO

Essa linguagem possui algumas outras funções, mas seu cerne é interpretar expressões e imprimir o resultado dessa interpretação na tela, isto é, conseguiremos ver o nome da empresa cadastrada no navegador.

Com isso, aprendemos uma nova forma mais simples de escrever código Java, e aplicaremos esse conhecimento em `listaEmpresas`, escrevendo um laço de maneira mais amigável com o mundo HTML.

Repare que no mundo HTML utilizamos *tags* para realizar marcações de elementos do código.

```
<!DOCTYPE hmtl>
<html>
<head>
<meta charset="ISO-8859-1">
<title>Insert title here</title>
</head>
<body>
    Lista de empresas: <br />
    <ul>
    <%
        List<Empresa> lista = (List<Empresa>)request.getAttr
        for (Empresa empresa : lista) {
```

```
        %>
            <li><%= empresa.getNome() %></li>
        <%
            }
        %>
        </ul>

    </body>
    </html>
```

COPIAR CÓDIGO

O que faremos é justamente marcar nosso laço, declarando seu começo e fim por meio de tags.

Para realizamos essa ação, precisaremos de uma biblioteca, afinal, por padrão, o Tomcat não possui essa capacidade. [A biblioteca que usaremos pode ser baixada neste link (https://caelum-online-public.s3.amazonaws.com/986-servlets-parte1/05/jstl-1.2.jar)](https://caelum-online-public.s3.amazonaws.com/986-servlets-parte1/05/jstl-1.2.jar).

Trata-se de um arquivo chamado `jstl-1.2.jar`. Ao terminar de realizar o download, copiaremos este arquivo e o colaremos na pasta "WebContent > WEB-INF > lib". Todas as aplicações Java web possuem a pasta `lib` para armazenar bibliotecas.

Usaremos a tag `<forEach>` para realizar o laço. Contudo, o Eclipse ainda não saberá que este `<forEach>` é da nossa nova biblioteca anexada.

```
<!DOCTYPE hmtl>
<html>
<head>
<meta charset="ISO-8859-1">
<title>Insert title here</title>
</head>
<body>
    Lista de empresas: <br />
```

```
    <ul>
        <forEach>

        </forEach>
    </ul>
```

COPIAR CÓDIGO

Precisaremos importar a nova biblioteca. Normalmente as importações são agrupadas no começo da página, e seguiremos esse padrão. A declaração de importação é parecida com a de uma classe ou pacote, usamos as marcações `<%@ %>` e a expressão `taglib`, que está relacionada a uma biblioteca de tags.

Na teoria, essa expressão pode ser incluída por meio do atalho "Ctrl + Space", mas em geral o editor JSP do Eclipse não é muito bom, e às vezes algumas expressões podem estar sublinhadas de vermelho sem qualquer razão, portanto é sempre bom desconfiar.

O nome do mundo de bibliotecas de *tags* é definido por meio de do atributo `uri`. Adicionaremos o nome da tag lib `http://java.sun.com/jsp/jstl/core`. Trata-se apenas de um apelido, que faz referência a uma biblioteca de mesmo nome.

```
<% @taglib uri="http://java.sun.com/jsp/jstl/core" %>

<!DOCTYPE hmtl>
<html>
<head>
<meta charset="ISO-8859-1">
<title>Insert title here</title>
</head>
<body>
    Lista de empresas: <br />

    <ul>
```

```
        <forEach>

        </forEach>
    </ul>
```

COPIAR CÓDIGO

Estamos dentro do um `jsp`, e a biblioteca de tags chama-se `jstl`, isto é, `Java Standard Taglib`, que adicionaremos na tag `<title>`. Das bibliotecas dentro no nosso arquivo, a mais importante é a `core`.

Agora, para indicar que o `<forEach>` vem da biblioteca que importamos, adicionaremos um prefixo (`prefix`), definido como `c`, fazendo uma indicação à `core`.

```
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %

<!DOCTYPE hmtl>
<html>
<head>
<meta charset="ISO-8859-1">
<title>Java Standard Taglib</title>
</head>
<body>
    Lista de empresas: <br />

    <ul>
        <forEach>

        </forEach>
    </ul>
```

COPIAR CÓDIGO

Desse modo, a importação está finalizada. Repetiremos o prefixo dentro na tag `<forEach>`, ou seja `<c:forEach>`. Contudo, alguns atributos ainda são necessários, sendo os mais importantes `items` e `var` - `items` corresponde

aos elementos que usaremos para realizar o laço, neste caso nossa `lista`, que por sua vez provém de `request.getAttribute()`.

```
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %

<!DOCTYPE hmtl>
<html>
<head>
<meta charset="ISO-8859-1">
<title>Java Standard Taglib</title>
</head>
<body>
    Lista de empresas: <br />

    <ul>
        <c:forEach>

        </forEach>
    </ul>
    <ul>
    <%
        List<Empresa> lista = (List<Empresa>)request.getAttribu
        for (Empresa empresa : lista) {
    %>
        <li><%= empresa.getNome() %></li>
    <%
        }
    %>
    </ul>
```

COPIAR CÓDIGO

Já sabemos que esta linha (`List<Empresa> lista = (List<Empresa>)request.getAttribute("empresas")`) pode ser traduzida por meio da *expression language*, afinal podemos acessar o atributo `empresas` escrevendo simplesmente `${empresas}`.

Podemos fazê-la operar em conjunto com a tag `<forEach>`, dessa forma podemos fazer um laço baseado na *taglib* em cima dos itens contidos na variável `empresas`, que por sua vez acessa o atributo da requisição.

```
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %

<!DOCTYPE hmtl>
<html>
<head>
<meta charset="ISO-8859-1">
<title>Java Standard Taglib</title>
</head>
<body>
    Lista de empresas: <br />

    <ul>
        <c:forEach items="${empresas}">

        </forEach>
    </ul>
```

COPIAR CÓDIGO

Por fim, resta definirmos uma variável(`var`) para o laço, e esta variável se chamará `empresa` .

```
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %

<!DOCTYPE hmtl>
<html>
<head>
<meta charset="ISO-8859-1">
<title>Java Standard Taglib</title>
</head>
<body>
    Lista de empresas: <br />
```

```
    <ul>
        <c:forEach items="${empresas}" var="empresa">

        </c:forEach>
    </ul>
```

COPIAR CÓDIGO

Ainda não terminamos toda a ação, mas já começamos a perceber que a nova forma de definir um laço por meio de tags combina muito mais com o mundo HTML, e não precisamos mais sair do contexto utilizando Java via scriptlet. Dessa forma nosso código fica um pouco mais elegante.

Resta trabalharmos na segunda parte do laço, que se refere à linha `<li><%= empresa.getNome() %></li>` .

```
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %

<!DOCTYPE hmtl>
<html>
<head>
<meta charset="ISO-8859-1">
<title>Java Standard Taglib</title>
</head>
<body>
    Lista de empresas: <br />

    <ul>
        <c:forEach items="${empresas}" var="empresa">

        </c:forEach>
    </ul>
    <%
        List<Empresa> lista = (List<Empresa>)request.getAttribu
        for (Empresa empresa : lista) {
    %>
```

```
        <li><%= empresa.getNome() %></li>
    <%
        }
    %>
    </ul>
```

COPIAR CÓDIGO

Queremos imprimir o nome da empresa, e para isso utilizaremos novamente a *expression language* para acessar a variável `empresa`. Desta vez, queremos chamar o método `getNome()`, então seria natural pensarmos em uma construção tal como `${empresa.getNome()}`.

Contudo, a linguagem de expressão não aceita a sintaxe Java, portanto a construção que usaremos será `${empresa.nome}`. De forma oculta, o método `getNome()` continuará a ser executado.

```
<html>
<head>
<meta charset="ISO-8859-1">
<title>Java Standard Taglib</title>
</head>
<body>
    Lista de empresas: <br />

    <ul>
        <c:forEach items="${empresas}" var="empresa">
            <li>${empresa.nome }</li>
        </c:forEach>
    </ul>
```

COPIAR CÓDIGO

Dessa forma, podemos excluir de nosso código toda forma antiga que estávamos utilizando para realizar o laço. Depois da refatoração, teremos como resultado a seguinte configuração:

```
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %

<!DOCTYPE hmtl>
<html>
<head>
<meta charset="ISO-8859-1">
<title>Java Standard Taglib</title>
</head>
<body>
    Lista de empresas: <br />

    <ul>
        <c:forEach items="${empresas}" var="empresa">
            <li>${empresa.nome}</li>
        </c:forEach>
    </ul>

</body>
</html>
```

COPIAR CÓDIGO

Refizemos nosso código de uma forma mais amigável, tendo em vista o mundo HTML. Tivemos algum trabalho inicialmente ao realizar a importação, definimos o prefixo e assim por diante.

Ao acionarmos o atalho "Ctrl + Space", poderemos verificar a existência de uma série de outras tags. Inclusive aqui na **Alura** temos um curso específico focado apenas na biblioteca JSTL (https://cursos.alura.com.br/course/jstl).

Agora iremos reiniciar nosso Tomcat e testar a aplicação. No navegador acessaremos a URL http://localhost:8080/listaEmpresas (http://localhost:8080/listaEmpresas). Teremos a seguinte mensagem exibida na tela:

Lista de empresas:

Alura

Caelum

São as duas empresas que estão cadastradas em nosso banco de dados. Repare que, ao investigarmos o código fonte, teremos um HTML limpo, não teremos nenhuma declaração dos taglibs ou do laço, afinal ele foi executado no Tomcat. Desse modo, o navegador apenas recebeu o HTML puro.

```
<!DOCTYPE html>
<html>
<head>
<meta charset="ISO-8859-1">
<title>Java Standard Taglib</title>
</head>
    Lista de empresas: <br />

    <ul>

        <li>Alura</li>
        <li>Caelum</li>

    </ul>

</body>
</html>
```

COPIAR CÓDIGO

Neste ponto, já temos um JSP bem profissional, embora ainda haja algumas informações que aprenderemos nas próximas aulas.