08

Usando a taglib core

Transcrição

Continuaremos o desenvolvimento do nosso projeto, ainda focados no JSP.

Aprendemos que não é interessante escrever scriptlets no HTML, afinal isso poderá gerar problemas de manutenção no futuro. Procuramos outras formas de escrever um código mais amigável, e encontramos a *expression language* e a biblioteca de tags JSTL.

O JSP é, portanto, uma página HTML com JSTL e expression language. A partir desse ponto, continuaremos usando essa biblioteca JSTL dessa maneira. Claro, existem algumas tags úteis que serão apresentadas no curso, mas não todas. Em todo caso, existe um curso específico na plataforma Alura para este fim.

Antes, faremos um breve resumo dos conteúdos aprendidos.

JSTL (Java Standard Tag Library)

core - controle de fluxo

fmt - formatação / i18n (internacionalização)

sql - executar SQL

xml - gerar XML

Falamos sobre o *core*, que realiza o controle de fluxo, mas na verdade a biblioteca possui mais tags com outros focos. Um exemplo é o **fmt**, que serve para a formatação de datas, números e i18n, isto é internacionalização. Quando escrevemos uma página JSP, ela pode atender no idioma português, inglês ou francês, varia de acordo com o seu interesse, e é justamente para isso que serve esta biblioteca.

Teremos, ainda, mais duas sub-bibliotecas: **sql** e **xml**. Elas existem, mas são raramente utilizadas dentro de um projeto e executadas dentro de um arquivo JSP, afinal, trata-se de uma má prática portanto não gastaremos muito tempo com isso. Gerar arquivos XML é importante, assim como executar sql, mas não no JSP.

Para utilizarmos o **core** e o **fmt** sempre teremos de ter as seguintes declarações no início da página - lembrando que o prefixo pode ser escolhido livremente:

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

fmt - formatação/i18n (internacionalização) `<%@ taglib uri = "http://java.sun.com/jsp/jstl/fmt" prefix = "fmt"%>`

Veremos mais alguns exemplos, pois essas são bibliotecas importantes e muito utilizadas, e é interessante dominá-las.

De volta ao Eclipse, acionaremos o atalho "Ctrl + Shift + Tab" para fechar todos os arquivos abertos. É importante lembrar que para utilizarmos JSP não podemos utilizar um arquivo `.html`, precisamos renomear o arquivo para `.jsp`. Faremos exatamente isso com `formnNovaEmpresa.html`, que passará a ser `formnNovaEmpresa.jsp` e o abriremos:

```
<!DOCTYPE html>
<html>
<head>
```

```
<meta charset="ISO-8859-1">
<title>Insert title here</title>
</head>
<body>

    <form action="/gerenciador/novaEmpresa" method="post">

        Nome: <input type="text" name="nome"  />

        <input type="submit" />
    </form>

</body>
</html>
```

COPIAR CÓDIGO

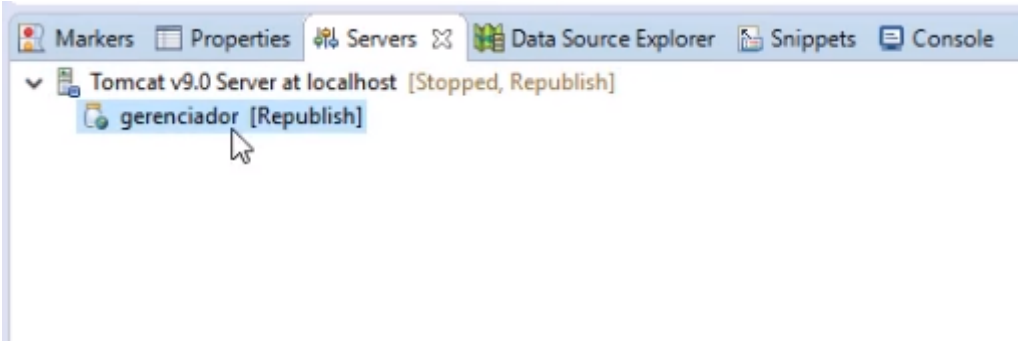
Já temos um caso em que a biblioteca core pode ser útil. Usaremos o cabeçalho de importação `<%@ taglib uri = "http://java.sun.com/jsp/jstl/core" prefix = "c"%>` em `formNovaEmpresa.jsp` , dessa forma conseguiremos interagir com as tags da biblioteca.

```
<%@ taglib uri = "http://java.sun.com/jsp/jstl/core" prefix = "
<!DOCTYPE html>
<html>
<head>
<meta charset="ISO-8859-1">
<title>Insert title here</title>
</head>
<body>

    <form action="/gerenciador/novaEmpresa" method="post">

        Nome: <input type="text" name="nome"  />

        <input type="submit" />
    </form>
```

```
</body>
</html>
```

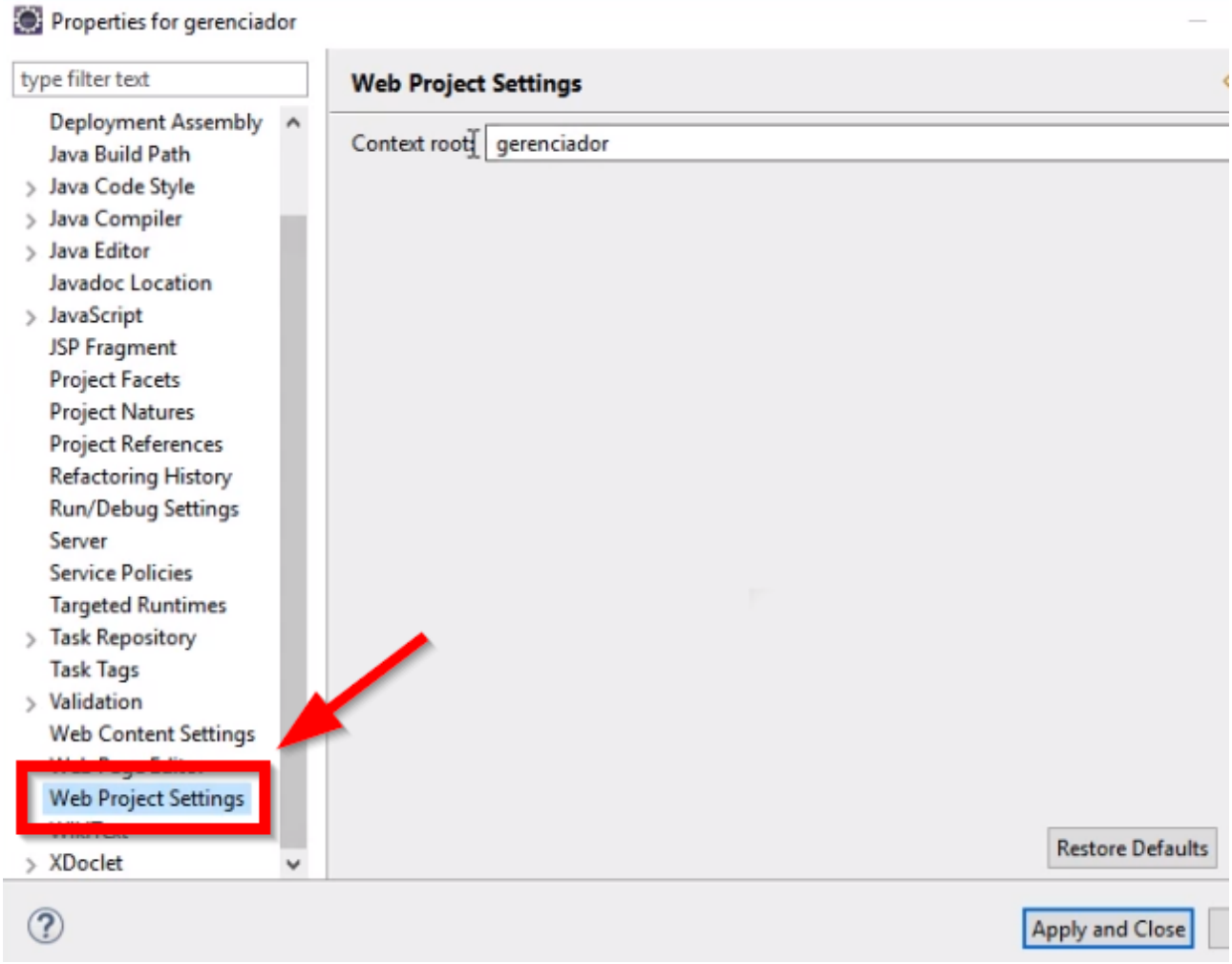
COPIAR CÓDIGO

Dentro da tag `<form>` temos `/gerenciador/novaEmpresa`, para que possamos executar a requisição e chegar em nosso Servlet. Contudo, existe um problema em utilizar este "caminho" até o Servlet. Removeremos o projeto `gerenciador` do Tomcat, clicando sobre a aba "Servers" no Eclipse. Em seguida, clicando com o botão direito sobre o nome do projeto, escolheremos a opção "Remove".



Agora que o Tomcat está limpo, iremos na área "Project Explorer" e clicaremos com o botão direito sobre `gerenciador`, depois, selecionaremos a opção "*Properties*".

Teremos uma nova janela com uma série de configurações disponíveis, e uma delas é a "*Web Project Settings*".



Veremos a opção "*Context Root*", que nada mais é que o conteúdo que inserimos na raiz, isto é, tudo aquilo depois da IP. Do ponto de vista do servidor, estamos falando do contexto, e esse campo é editável. Atualmente a configuração é "gerenciador", e escreveremos "gerenciadorX". **Ao fazermos isso, não renomeamos nosso projeto, apenas o contexto.**

Feito isso, novamente adicionaremos o projeto `gerenciador` no Tomcat, o executaremos e testaremos as modificações no navegador.

Escreveremos a URL http://localhost:8080/gerenciador/listaEmpresas (http://localhost:8080/gerenciador/listaEmpresas). Ao tentarmos acessar o conteúdo, teremos uma mensagem de erro:

> HTTP Status 404 - Not Found.

Isso ocorreu porque mudamos o contexto, portanto precisaremos escrever a partir de agora http://localhost:8080/gerenciador/listaEmpresas (http://localhost:8080/gerenciador/listaEmpresas) para acessarmos a lista de empresas cadastradas.

O nome do contexto pode variar. Ao acessarmos o formulário via http://localhost:8080/gerenciadorX/formNovaEmpresa (http://localhost:8080/gerenciadorX/formNovaEmpresa), analisando o código fonte, perceberemos que o contexto a ser chamado ainda é `gerenciador` e não `gerenciadorX`.

```
<!DOCTYPE html>
<html>
<head>
<meta charset="ISO-8859-1">
<title>Insert title here</title>
</head>
<body>

        <form action="/gerenciador/novaEmpresa" method="post">

            Nome: <input type="text" name"nome" />

            <input type="submit" />
        </form>

</body>
</html>
```

COPIAR CÓDIGO

Logo, escrever o nome do contexto no arquivo JSP não é uma boa prática, uma vez que ele pode variar e gerar problemas em outras páginas. Para lidarmos com esse fluxo dinâmico, usaremos a biblioteca Taglib, mais especificamente, a tag `<c:url />`. Nela, inseriremos o atributo `value`, e, em seguida, iremos

inserir um único caminho possível para chegar até o Servlet. Apenas para testar esse recurso, escreveremos `"/novaEmpresa"` .

```
<%@ taglib uri = "http://java.sun.com/jsp/jstl/core" prefix = '
<!DOCTYPE html>
<html>
<head>
<meta charset="ISO-8859-1">
<title>Insert title here</title>
</head>
<body>

<c:url value="/novaEmpresa"/>

    <form action="/gerenciador/novaEmpresa" method="post">

        Nome: <input type="text" name="nome"  />

        <input type="submit" />
    </form>

</body>
</html>
```

COPIAR CÓDIGO

No navegador, acessaremos http://localhost:8080/gerenciadorX/formNovaEmpresa.jsp (http://localhost:8080/gerenciadorX/formNovaEmpresa.jsp). Será impresso na tela, além do formulário de cadastro, a linha `gerenciadorX/novaEmpresa` , justamente o caminho que precisamos utilizar para cessar o Servlet.

Primeiramente, iremos desfazer a modificação realizada na *Context Root*. Para isso, removeremos o projeto do Tomcat novamente, acessaremos as configurações de `gerenciador` e escolheremos a opção "*Web Project Settings*"

Então, renomearemos "gerenciadorX" para "gerenciador". Feito isso, incluiremos novamente o projeto no Tomcat.

Em seguida, passaremos a tag `<c:url>` para dentro de `<form>`, o que pode não ser muito elegante, mas é funcional.

```
<%@ taglib uri = "http://java.sun.com/jsp/jstl/core" prefix = "
<!DOCTYPE html>
<html>
<head>
<meta charset="ISO-8859-1">
<title>Insert title here</title>
</head>
<body>

    <form action="<c:url value="/novaEmpresa"/>" method="post"

        Nome: <input type="text" name="nome"  />

        <input type="submit" />
    </form>

</body>
</html>
```

COPIAR CÓDIGO

Reiniciaremos a aplicação e então estaremos prontos para testar as modificações no navegador.

Ao analisarmos o código fonte de http://localhost:8080/gerenciador/formNovaEmpresa.jsp (http://localhost:8080/gerenciador/formNovaEmpresa.jsp), notaremos as atualizações do código aplicadas corretamente:

```
<%@ taglib uri = "http://java.sun.com/jsp/jstl/core" prefix = '
<!DOCTYPE html>
<html>
<head>
<meta charset="ISO-8859-1">
<title>Insert title here</title>
</head>
<body>

<c:url value="/novaEmpresa"/>

    <form action="/gerenciador/novaEmpresa" method="post">

        Nome: <input type="text" name="nome"  />

        <input type="submit" />
    </form>

</body>
</html>
```

COPIAR CÓDIGO

Para aprendermos uma configuração de código mais elegante, usaremos a tag `<c:url>` no começo do código de `formNovaEmpresa.jsp`, e declararemos uma variável `linkServletNovaEmpresa`. Em seguida, incluiremos a variável em `<form>`, utilizando a *expression language*.

```
<%@ taglib uri = "http://java.sun.com/jsp/jstl/core" prefix = '
<c:url value="/novaEmpresa" var="linkServletNovaEmpresa"/>

<!DOCTYPE html>
<html>
<head>
<meta charset="ISO-8859-1">
<title>Insert title here</title>
```

```
</head>
<body>
    <form action="${linkServletNovaEmpresa}" method="post">

        Nome: <input type="text" name="nome"  />

        <input type="submit" />
    </form>

</body>
</html>
```

COPIAR CÓDIGO

Testaremos novamente as modificações no navegador, e veremos que as o código fonte continua limpo e sem informações extras, mostrando que tudo ocorreu como o esperado.

A tag `<c:url>` é muito importante, e você a encontrará em todos os projetos que utilizam JSP. Portanto, aprendemos um conceito fundamental nesta aula. Vamos seguir trabalhando em `novaEmpresaCriada.jsp` :

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

COPIAR CÓDIGO

Ao acessarmos http://localhost:8080/gerenciador/novaEmpresa.jsp (http://localhost:8080/gerenciador/novaEmpresa.jsp), teremos a mensagem:

> Empresa cadastrada com sucesso!

Isso significa que não há o nome de uma empresa, pois não enviamos por meio do formulário o nome de uma empresa específica. Dessa forma, o Servlet não foi chamado e o nome não foi inserido na requisição, que, por sua vez, não chegou até o JSP.

Portanto, não é interessante que haja essa mensagem quando o JSP for chamado diretamente, ou seja, na condição específica de não haver qualquer empresa cadastrada. O ideal é criamos uma outra mensagem para essas situações.

Para criarmos uma nova mensagem, usaremos a biblioteca core. Escreveremos a tag `<c:if>`, e inseriremos em seu corpo a condição. Por fim, precisaremos definir a condição por meio do atributo `test`. Em seguida, utilizaremos novamente a *expression language*, e através dela avaliaremos se a variável `empresa` existe ou não por meio de `not empty`.

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

<html>
    <body>
        <c:if test= "${not empty empresa}">
            Empresa ${ empresa } cadastrada com sucesso!
        </c:if>
    </body>
</html>
```

COPIAR CÓDIGO

Em seguida, criaremos uma outra condição: caso a variável esteja vazia, será impressa a mensagem "Nenhuma Empresa cadastrada".

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

<html>
    <body>
```

```
            <c:if test= "${not empty empresa}">
                Empresa ${ empresa } cadastrada com sucesso!
            </c:if>

            <c:if test= "${empty empresa}">
                Nenhuma empresa cadastrada!
            </c:if>


        </body>
    </html>
```

COPIAR CÓDIGO

Com as duas condições definidas, podemos testar as alterações diretamente no navegador.

Ao acessarmos diretamente http://localhost:8080/gerenciador/novaEmpresaCriada.jsp (http://localhost:8080/gerenciador/novaEmpresaCriada.jsp), teremos a mensagem "*Nenhuma empresa cadastrada!*", que é o comportamento esperado da nossa aplicação. Ao cadastrarmos uma empresa utilizando o formulário, a mensagem de cadastramento também é exibida corretamente.