02

Quando o dispatcher não atende

Transcrição

Continuaremos a explorar as possibilidades do Servlet e a melhorar nossa aplicação, lembrando que muito do que aprendemos aqui também se aplica aos *frameworks* de mais alto nível.

Estamos conseguindo cadastrar empresas e datas por meio do formulário. Ao cadastrarmos uma empresa, visualizamos uma mensagem que confirma esse cadastramento, por exemplo "*Empresa Alura cadastrada com sucesso!*".

No entanto, não queremos mais receber essa mensagem, e sim, sermos direcionados automaticamente para a página de lista de empresas - afinal, esse é um comportamento comum nas aplicações de forma geral.

Em `NovaEmpresaServlet.java`, não chamaremos o JSP `/novaEmpresaCriada.jsp`, mas sim, `listaEmpresas.jsp`.

```
        Banco banco = new Banco();
        banco.adiciona(empresa);

        //chamar o JPS
        RequestDispatcher rd = request.getRequestDispatcher("/
        request.setAttribute("empresa", empresa.getNome());
        rd.forward(request, response);
    }
```

COPIAR CÓDIGO

Em `listaEmpresas.jsp`, adicionaremos o bloco `if` - similar ao que encontramos em `novaEmpresaCriada.jsp` - dentro da tag `<body>`. Dessa forma conseguiremos exibir a mensagem apenas se `empresa` não estiver vazia.

```
<!DOCTYPE html>
<html>
<head>
<meta charset="ISO-8859-1">
<title>Java Standard Taglib</title>
</head>
<body>

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


    Lista de empresas: <br />

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

            <li>${empresa.nome } - <fmt:formatDate value="${em
        </c:forEach>
    </ul>

</body>
</html>
```

COPIAR CÓDIGO

Ao acessarmos a URL http://localhost:8080/gerenciador/novaEmpresa (http://localhost:8080/gerenciador/novaEmpresa), teremos a seguinte mensagem exibida na tela:

Empresa Alura cadastrada com sucesso! Lista de empresas:

A mensagem foi exibida, mas não a lista de empresas cadastradas. Vamos averiguar o que aconteceu.

Ao analisarmos o código fonte no navegador, veremos que o laço começou a ser feito, mas não foi concluído.

```
<!DOCTYPE html>
<html>
<head>
<meta chartset="ISO-8859-1">
<title>Java Standard Taglib</title>
</head>
<body>

        Empresa Alura cadastrada com sucesso!

    Lista de empresas: <br />

    <ul>

    </ul>

</body>
</html>
```

COPIAR CÓDIGO

Na verdade, a variável `empresas` não existe. Repare que chamamos `NovaEmpresaServlet` mas não estamos utilizando `Banco` para carregar todas as empresas. Na verdade, o que estamos fazendo é reimplementando o Servlet. Toda a lógica (que ainda é muito simples) de carregar o banco e as empresas, inserir a lista de empresas dentro da requisição e depois evocar o JSP, deve ser repetida dentro de `NovaEmpresaServlet` .

Poderíamos até mesmo aproveitar o código utilizado em `ListaEmpresasServlet` , contudo, repetir código nunca é uma boa prática e

quase sempre gera erros inesperados.

Para resolver essa questão, existe outro percurso que podemos tomar: ao enviarmos a requisição e cairmos no Servlet, chamaremos outro Servlet, e não o arquivo JSP. Isto é, podemos despachar a requisição para um outro Servlet, que por sua vez ira despacha-la para o JSP.

Nós incluiremos um novo Servlet na cadeia de fluxo da requisição: de `novaEmpresaServlet`, ela será dispachada para `ListaEmpresaServlet` e, por fim, será enviada para o arquivo JSP. Então, as empresas serão listadas.

Em `NovaEmpresaServlet`, chamaremos `listaEmpresas`, por meio de `getRequestDispatcher()`. Dessa forma, dois Servlets estão em intercomunicação.

```
        Banco banco = new Banco();
        banco.adiciona(empresa);

        //chamar o JPS
        RequestDispatcher rd = request.getRequestDispatcher("/
        request.setAttribute("empresa", empresa.getNome());
        rd.forward(request, response);
    }
```

COPIAR CÓDIGO

Por sua vez, `listaEmpresas.Servlet` irá chamar o arquivo `/listaEmpresas.jsp`.

```
Banco banco = new Banco();
List<Empresa> lista = banco.getEmpresas();

request.setAttribute("empresas", lista);

RequestDispatcher rd = request.getRequestDispatcher("/listaE
rd.forward(request, response);
```

COPIAR CÓDIGO

Vamos reiniciar a aplicação e testá-la no navegador.

Preencheremos o formulário com um nome da empresa e data aleatórios. Ao tentarmos cadastra-los, receberemos a mensagem de erro:

> HTTP Status 405 - Method Not Allowed.

Já vimos essa mensagem em outro momento do curso, lembra? Ao analisarmos o código fonte do nosso formulário, perceberemos que está sendo enviada uma requisição do tipo POST:

```
<!DOTYPE 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" />
        Data Abertura: <input type="text" name="data" />

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

</body>
</html>
```

COPIAR CÓDIGO

A requisição será enviada para `NovaEmpresaServlet` que oferece suporte para o tipo POST. Em seguida, como sabemos, será chamado outro Servlet (`ListaEmpresasServlet`), que por sua vez oferece suporte para requisições do tipo GET. Por isso, teremos uma incompatibilidade entre os Servlets.

```
@WebServlet("/listaEmpresas")
public class ListaEmpresasServlet extends HttpServlet {
    private static final long serialVersionUID = 1L;

    protected void doGet(HttpServletRequest request, HttpServle
            throws ServletException, IOException {
```

COPIAR CÓDIGO

Para fazermos com que tudo opere perfeitamente, em `ListaEmpresaServlet` , ao invés da classe suportar a apenas a requisição GET, adicionaremos o termo `service` , que abarca também POST.

```
@WebServlet("/listaEmpresas")
public class ListaEmpresasServlet extends HttpServlet {
    private static final long serialVersionUID = 1L;

    protected void service(HttpServletRequest request, HttpSer
            throws ServletException, IOException {
```

COPIAR CÓDIGO

Salvaremos as modificações, reiniciaremos o Tomcat e partiremos para o teste da aplicação no navegador.

Na página do formulário, cadastraremos a empresa "Alura" e a data "01/01/2009". Em seguida, veremos na tela a lista de empresas cadastradas como o esperado.

Mas ainda temos um problema. Perceba que, ao reenviarmos a mesma requisição do tipo POST, isto é, com os mesmos dados cadastrais, teremos registros idênticos em nosso banco de dados. O fato da nossa aplicação permitir múltiplos registros iguais é uma má prática, e veremos nas próximas aulas como solucionar este ponto.