Transcrição

Até este ponto do curso, criamos três Servlets, aprendemos a configuração básica, construímos um pequeno modelo de cadastro para empresas e definimos o conteúdo a ser devolvido para o navegador. É exatamente este último ponto que iremos trabalhar.

Em `NovaEmpresaServlet` utilizamos HTML dentro do código Java, afinal esta é a linguagem compreendida pelo navegador. Esse tipo de abordagem é possível em um projeto pequeno, mas ao construirmos uma página mais sofisticada essa abordagem seria inviável.

```
@WebServlet("/novaEmpresa")
public class NovaEmpresaServlet extends HttpServlet {

    private static final long serialVersionUID = 1L;

    protected void doPost(HttpServletRequest request, HttpServletResponse respo
            throws ServletException, IOException {
        System.out.println("Cadastrando nova empresa");

        String nomeEmpresa = request.getParameter("nome");
        Empresa empresa = new Empresa();
        empresa.setNome(nomeEmpresa);

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

        PrintWriter out = response.getWriter();
        out.println("<html><body>Empresa " + nomeEmpresa + " cadastrada com suc
    }

}
```

Na página da **Alura**, por exemplo, ao analisarmos o código fonte veremos uma quantidade muito grande de informações HTML. Precisaríamos acionar `out.println()` incontáveis vezes para exibirmos o conteúdo necessário no navegador. **Não é uma boa prática possuir código de interface e visualização HTML dentro de uma classe**. Para isso, existe outro recurso que utilizaremos: as páginas HTML do projeto, como `formNovaEmpresa.html`:

```
<!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>
```

Se inserirmos esse código em Java, a manutenção será muito grande e teremos ilegibilidade em sua estrutura. A meta é extrair o código HMTL presente em `NovaEmpresaServlet` e armazená-lo em uma página HTML. Com o botão direito, clicaremos sobre a pasta `WebContent` e selecionaremos as opções "New > File", e criaremos o arquivo `novaEmpresaCriada.html`.

Neste novo arquivo, armazenaremos a estrutura HTML antes presente no Servlet `NovaEmpresaServlet`.

```
<html><body>
Empresa " + nomeEmpresa + " cadastrada com sucesso!
</body></html>
```

Teremos um problema: estávamos concatenando `nomeEmpresa`, e nesta página não possuímos esta variável. Aliás, não podemos programar dessa maneira dentro de um arquivo HTML. Para isso, usaremos outra tecnologia relacionada ao Servlet que transformará a página HTML em algo dinâmico, que permite algumas ações de programação.

O que faremos é uma **página dinâmica Java** ou *Java server page*, também conhecida por JSP. Renomearemos o arquivo - atalho F2 - `novaEmpresaCriada.htlm` para `novaEmpresaCriada.jsp`. Perceba que o arquivo não mudou de aspecto, continua com a estrutura inicial:

```
<html><body>
Empresa " + nomeEmpresa + " cadastrada com sucesso!
</body></html>
```

Porém, neste novo formato podemos inserir códigos Java, desde que eles sejam marcados por `<% %>`. Chamamos essa inserção de código Java um arquivo jsp de *scriptlet*. Inseriremos uma variável `NomeEmpresa`, de uma empresa fixa `Alura`. Sem seguida, usaremo so `System.out.println()` para averiguarmos se o código está funcional.

```
<%
    String nomeEmpresa = "Alura";
    System.out.println(nomeEmpresa);
%>

<html><body>
Empresa " + nomeEmpresa + " cadastrada com sucesso!
</body></html>
```

Ainda não teremos todos os problemas resolvidos com a inserção desse novo recurso, mas vamos por etapas. No navegador, digitaremos a URL `localhost:8080/gerenciador/novaEmpresaCriada.jsp`. Teremos a seguinte mensagem exibida na página:

```
Empresa " +nomeEmpresa+" cadastrada com sucesso!
```

A variável `nomeEmpresa` ainda não foi compreendida. Ainda não estamos realizando um *scriplet*, muito embora a parte do código Java que escrevemos foi executada no lado do servidor, e poderemos ver o resultado `Alura` impresso na console. Por ora, conseguimos um pequeno avanço: executar códigos Java no HTML.

Resolveremos a concatenação, transformando `nomeEmpresa` em um scriplet por meio da marcação `<%`. Utilizaremos ainda a variável automática `out`, referência para o `getWriter()` como vimos em `NovaEmpresaServlet`. Sem seguida, usaremos o `println()` para imprimir a informação.

```
<%
    String nomeEmpresa = "Alura";
    System.out.println(nomeEmpresa);
%>

<html><body>
Empresa " + <% out.println(nomeEmpresa) %> + " cadastrada com sucesso!
</body></html>
```

Essa parte do código (`<% out.println(nomeEmpresa) %>`) será interpretada no servidor, e o resultado dessa formulação deveria ser `Alura`, mas vamos testar isso no navegador. Quando alteramos um arquivo JSP não precisamos reiniciar o servidor, portanto vamos simplesmente escrever o URL `localhost:8080/gerenciador/novaEmpresaCriada.jsp` no Chrome.

Tereremos uma mensagem de erro `HTTP Status 500 - Internal Server Error`. Isso quer dizer que não foi interpretado o código Java que fizemos no lado do servidor, isto é, foi feito algo errado no scriplet que sua leitura não foi possível. Pode ser uma sintaxe, uma variável ou algo do gênero. Neste casso, o erro foi simples: a falta do sinal `;`.

```
<%
    String nomeEmpresa = "Alura"
    System.out.println(nomeEmpresa);
%>

<html><body>
Empresa " + <% out.println(nomeEmpresa); %> + " cadastrada com sucesso!
</body></html>
```

Feita a inserção do sinal, iremos testar novamente a aplicação no navegador. Dessa vez, teremos a seguinte mensagem:

```
Empresa Alura cadastrada com sucesso!
```

Conseguimos criar um código dinâmico com o auxílio do JSP, uma combinação de HTML e Java. Graças a este recurso, podemos fazer a lista de empresas. O `out.println()` em `navaEmpresaCadastrada.jsp`, imprimiu a informação e a devolveu como resposta para o navegador, mas existe outra forma de executar a ação: substituirmos o `out.println()` por `=`, é a variável que significa que queremos imprimir o conteúdo.

```
<%
    String nomeEmpresa = "Alura";
    System.out.println(nomeEmpresa);
%>

<html><body>
Empresa " + <%= (nomeEmpresa); %> + " cadastrada com sucesso!
</body></html>
```

Trata-se de uma simplificação que não altera o funcionamento do código. Nas próximas aulas, seguiremos melhoralhando nosso código HTML.