

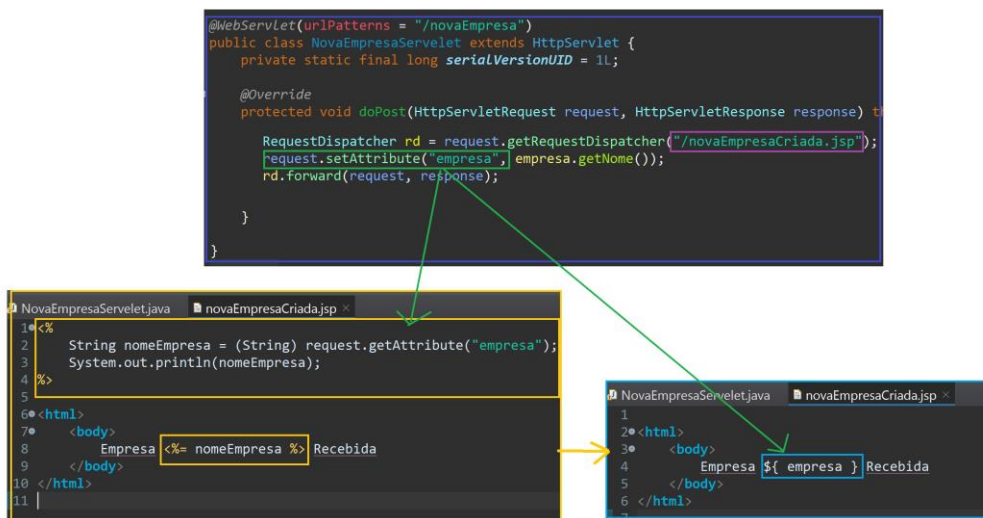


Java – Servlet – Expression Language

Eu achei os **Scriptlets** bem esquisitos e pelo jeito eu não era o único. Visando simplificar um pouco da complexidade que é o **scriptlets** foi introduzido o uso de **“expression language”**.

A **sintaxe da expression language** é bem comum em várias linguagens na real, ela é aquele role de usar **“\${}”**, onde **tudo que estiver dentro das chaves** pode tentar ser interpretado e mostrado na tela, isso vale para **operações simples ou mostrar o valor de variáveis**.

A **expression language** ajuda muito a reduzir a complexidade/verbosidade que uma JSP pode conter, por exemplo :



Java – Servlet – Expression Language - JSTL

Expression language já ajuda a escrever códigos mais amigáveis em arquivos **JSP**, no entanto ela não tem o poder de fazer o mesmo com iterações (*na verdade, ta mais pro tomcat*). Isso acaba gerando umas **iteraões** bem estranhas como essa, onde eu preciso intercalar **scriptlets** com **tags HTML**:

```
<body>
<h1>Empresas cadastradas</h1>
<ul>
<%
List<Empresa> empresas = (List<Empresa>) request.getAttribute("empresas");
for(Empresa empresa : empresas){
%>
<li><%= empresa.getNome() %></li>
<% } %>
</ul>
</body>
</html>
```

Obviamente a comunidade também não curtiu isso, então surgiu **algumas bibliotecas** para facilitar a escrita de **código java** em **JSP**, na verdade, deixá-lo mais amigável. Uma **dessas lib** foi a **JSTL**, que como uma biblioteca precisa **ser baixada no projeto** e **importada no JSP**:

```

1 <%@ page language="java" contentType="text/html; charset=ISO-8859-1"
2   pageEncoding="ISO-8859-1"%>
3 <%@ page import="java.util.List, br.com.gerenciador.servelet.Empresa" %>
4 <%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>
5 <!DOCTYPE html>
6 <html>
7 <head>
8   <meta charset="ISO-8859-1">
9   <title>Insert title here</title>
10 </head>
11 <body>

```

Java – Servlet – JSTL iteração

Com a **JSTL** importada eu posso fazer um **“for”** usando uma **tag html** combinando com **expression language**, o que vai transformar todo aquele **scriptlet** zuado em algo mais semântico e simples. Onde eu referencio o **prefixo da JSTL** com uma tag específica:

```

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

<h1>Empresas cadastradas</h1>

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

</body>
</html>

```

Java – Servlet – JSTL definir caminhos

Outra tag bem útil do **JSTL** é a de **url**, nela podemos **definir um caminho** e **armazenar em uma variável**, dessa maneira eu posso controlar o fluxo de algo sem precisar me preocupar com o contexto (**geralmente a pasta root do projeto é o contexto**).

Por exemplo, esse **formulário** chama um método **post** no **caminho da url armazenada**:

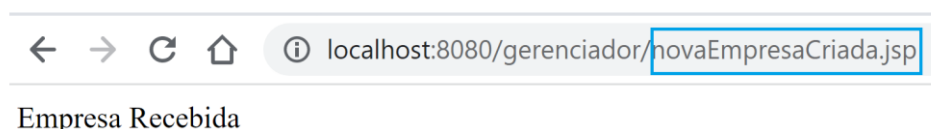
```

1 <%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>
2
3 <c:url value="/novaEmpresa" var="linkServletNovaEmpresa" /></c:url>
4 <!DOCTYPE html>
5 <html>
6 <head>
7   <meta charset="ISO-8859-1">
8   <title>Insert title here</title>
9 </head>
10 <body>
11
12 <form action="${linkServletNovaEmpresa}" method="post">
13   Nome: <input type="text" name="nome"/>
14   <input type="submit" />
15 </form>
16
17
18 </body>
19 </html>

```

Java – Servlet – JSTL condicionais

Dependendo do fluxo de interação da aplicação, algumas informações precisam ser checadas. Por exemplo, se eu acessar a **página de novaEmpresa** diretamente sem passar nenhuma empresa, eu vou ter o seguinte resultado:



Mas não era isso que eu queria, então através do **core do JSTL** eu posso definir um comportamento **baseado em uma condição**:

