Uso de servlets nas aplicações web desenvolvidas para a plataforma java [1]

Resumo

Em uma interação feita em uma aplicação Web existem tarefas que devem ser executadas no servidor, tais como processamento de formulários de pedidos e acessar banco de dados residente no servidor. Uma classe Java chamada de servlet pode ser usada para essas tarefas. Um servlet é uma classe Java compilada. Um objeto instanciado a partir dessa classe é executado no servidor quando solicitado a partir de um documento XHTML que está sendo exibido pelo browser. Um servlet produz um documento XHTML como resposta, onde algumas partes são estaticas e geradas por declarações de saída simples, enquanto outras partes são criadas dinamicamente quando o servlet for chamado.

Abstract

There are many computational tasks in a Web interaction that must occur on the server, such as processing order forms and accessing server-resident database. A Java class called a servlet can be used for these applications. A servlet is a compiled Java class, an object of which is executed on the server system when requested by the XHTML document being displayed by the browser. A servlet produces an XHTML document response, some parts of which are stattic and are generated by simple output statements, while other parts are created dynamically when the server is called.

1 – O ambiente de processamento de uma aplicação web

As aplicações projetadas para serem utilizadas via internet são chamadas de aplicações web. Elas são executadas em um ambiente distribuído, onde cada elemento da aplicação pode estar localizado em diferentes computadores conectados via internet. As aplicações web também são chamadas de sites.

Um servidor web é um software (programa) que permite a execução de uma aplicação web. O computador que armazena o software servidor web também é chamado de servidor web, isto é, o software que ele armazena é o que o torna "especial", definindo o seu nome. O computador usado para acessar a aplicação web é chamado de cliente. A interface dessa aplicação é implementada por meio de um programa chamado browser (navegador). Assim, o usuário utiliza o browser para solicitar serviços e receber as respostas do software web server.

O fato de a aplicação web ser executada em um ambiente distribuído modifica uma série de conceitos de programação de computadores: elas usam uma arquitetura multinível em que os serviços executados pelos programas que compõem a aplicação web podem estar distribuídos em uma rede de computadores. A aplicação web usa um infraestrutura de rede cujo padrão foi adotado pela internet.

Os elementos que compõem a aplicação web estão organizados da seguinte forma: de um lado está a máquina usada para acessar a aplicação web chamada máquina cliente e do outro está a máquina que armazena o software servidor web que executa a aplicação web.

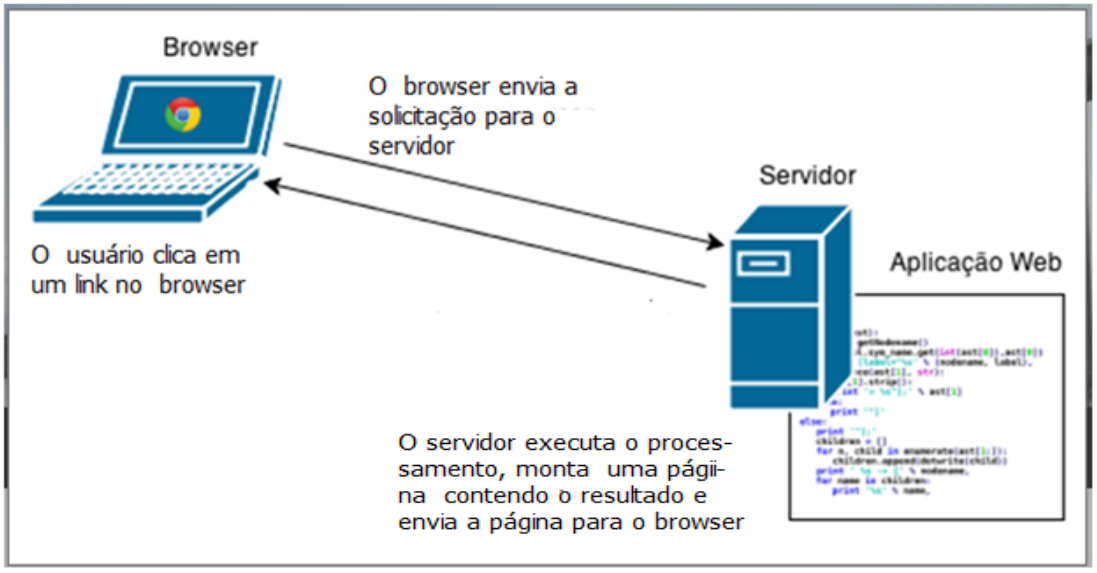
2 Função da máquina cliente

---

[1] Mestre em Engenharia. Professor do curso de Sistemas de Informação das Faculdades Interadas "Campos Salles".

A máquina cliente executa um programa chamado browser (navegador). Esse programa implementa a interface da aplicação web. Usando essa interface, o usuário pode entrar com solicitações ao servidor web ou simplesmente servidor. Ele processa a solicitação e envia o resultado para a interface onde foi feita a solicitação.

Nessas condições, o browser (Internet Explorer, Google Chrome, Mozila, etc.) é o software que sabe se "comunicar" com o servidor, isto é, software servidor web. O ambiente de processamento de uma aplicação web está ilustrado na Figura 1.



Basicamente, a "comunicação" entre cliente e servidor consiste no envio e no recebimento de arquivos chamados páginas web, ou páginas. Cada arquivo ou página armazena uma "folha" que pode conter um texto, figuras, vídeos etc. A disposição desses elementos nessa folha é chamada de formatação.

Na Figura 2 (a) tem-se o conteúdo de uma página, e na Figura 2 (b) tem-se o resultado da impressão do texto.

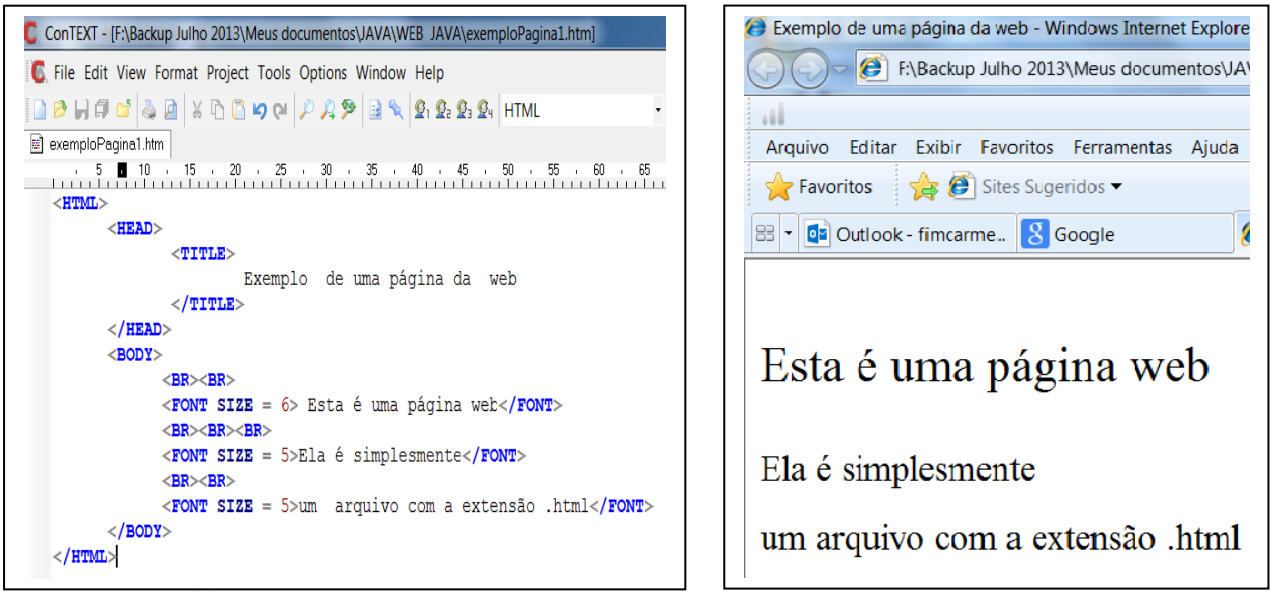


Figura 2 - (a) Página web escrita na linguagem HTML (b) e seu respectivo código.

Na página estão os elementos que a compõem e sua posição na interface, isto é, a tela do computador. Para que o browser possa "entender" como deverá ser feita a impressão, foi criada a linguagem declarativa HTML.

Na Figura 2 (a) tem-se as instruções que o browser deve executar para produzir a página da Figura 2 (b). Esse conjunto de instruções é chamado de código ou programa, expresso na linguagem HTML.

Um site corresponde a um conjunto de páginas interligadas. O browser também é um programa: sua função é interpretar o código HTML de cada página do site e fazer a sua impressão, isto é, navegar pelo site imprimindo as suas páginas.

3 Função da máquina servidor

Ao navegar em um site de uma loja, uma pessoa pode acessar determinada página a fim de comprar determinado produto na loja virtual: ela digita o seu número, o código do produto e a quantidade comprada. A seguir, clica em um botão da página que contém a palavra submit para submeter os dados digitados ao processamento no servidor.

O browser envia esses dados para o servidor, solicitando o cálculo do valor a pagar.

Depois de receber a solicitação, o servidor web passa os dados para a aplicação web que irá executar o seu processamento: nesse caso, o processamento consiste em calcular o valor a ser pago e obter os dados do cliente no banco de dados da loja. Basicamente, pode-se ter os seguintes tipos de processamento: armazenar ou ler (obter) dados do banco de dados, executar cálculos, validações etc.

A seguir a aplicação web, usando a linguagem HTML, monta uma página contendo os dados do cliente e o valor a ser pago, isto é, o resultado do processamento. A seguir a aplicação web devolve o controle de execução da solicitação para o servidor web, que envia a página para o browser do cliente. O browser interpreta o código escrito em HTML e imprime a página na interface (tela do monitor).

4 Client side e server side

Os elementos que compõem uma aplicação web estão organizados de tal forma que de um lado está a máquina usada para acessar a aplicação web chamada de cliente e do outro, a máquina que armazena o software servidor web que executa a aplicação web.

Os recursos tecnológicos que foram desenvolvidos para a internet podem ser classificados em dois grandes grupos: recursos para rodar do lado do cliente (client side) e recursos para rodar no lado do servidor (server side). O browser, por exemplo, é um recurso do lado do cliente, o mesmo acontecendo com a aplicação escrita usando-se as linguagens HTML e CSS. Outro exemplo é a linguagem JavaScript, que é usada para dar conteúdo dinâmico a uma página web localizada no lado do servidor da aplicação web.

No caso das aplicações web que rodam do lado do servidor web, ou simplesmente servidor, também se deve ter um programa que execute o "papel" de servidor. Esses programas também são chamados de servidor web ou servidor de aplicação.

No mercado há vários browsers ou navegadores, tais como: Internet Explorer, Chrome, Mozila etc. Da mesma forma, há vários servidores web, tais como: IIS (Internet Information Server), PWS (Personal Web Server), que são produtos da Microsoft, portanto rodam somente programas ou apps escritos com as linguagens fornecidas por essa empresa, Apache Tomcat, WebLogic, Blazis etc., que rodam programas ou apps escritos com JSP na plataforma Java.

Neste artigo será usado o servidor web Apache Tomcat, um dos mais populares na plataforma Java.

5 O que é um servlet?

Servlets são "pequenos" programas escritos em Java que rodam do lado servidor da aplicação web. Um servlet é uma classe que permite a chamada dos seus métodos a partir de métodos escritos em JSP localizados em outras máquinas via protocolos e formatos de dados padronizados tais como HTTP e XML.

Serviço é um tipo específico de bem que se caracteriza por ser intangível, isto é, sem existência física. Um produto também é um bem, porém tem existência física, portanto não é um serviço.

O fornecimento de energia elétrica, o preenchimento da declaração de imposto de renda, a consulta a um médico são exemplos de serviços em geral prestados por uma empresa ou pessoa. Alguns desses serviços podem ser automatizados. Por exemplo, o cálculo do frete para o envio de um produto comprado via internet. O cliente entra no site da empresa, monta o pedido e através do seu CEP é possível calcular o valor do frete usando-se um programa que está armazenado em um computador que pode ser acessado via internet. Portanto, esse programa para o cálculo de fretes presta esse serviço para o site de compras, ou seja, é um programa que presta serviços para outros programas. Esse tipo de programa é chamado de servlet na plataforma Java.

Os servlets prestam serviços aos clientes das aplicações escritas para a plataforma Java. Os sevlet são compostos por vários métodos que podem ser usados por programas ou aplicações diferentes: eles não pertencem a "ninguém". Por exemplo, em um sistema projetado para a área de recursos humanos de uma empresa certamente existe um método para gravar os dados de um funcionário no banco de dados processado pela aplicação. Esse método é usado pelo programa que cadastra os dados de um funcionário recém-contratado, pelo programa que permite atualizar esses dados (grava os dados atualizados no banco de dados) etc. Em vez de ter que escrever esse método no programa de cadastramento, depois repetir seu código no programa de alteração de dados etc., tem-se a opção de implementar esse método como um servlet: nesse caso, ele será escrito somente uma vez, porém poderá ser usado por qualquer programa onde seja necessário gravar os dados de um funcionário no banco de dados da empresa, inclusive aqueles que compõem o sistema de recursos humanos da empresa.

6 Ambiente de execução de um servlet

Quando um servidor web como o Tomcat recebe uma solicitação para a execução de um dos métodos de um servlet, ele entrega a solicitação não ao servlet em si, mas a um programa (classe) que trabalha como um "intermediário" entre o servlet e o servidor web. Esse programa (classe) é chamado de contêiner. Essa classe tem métodos que permitem oferecer toda a infraestrutura de software necessária para rodar o servlet no servidor e atender as requisições de serviços feitas pelos programas ou apps clientes. Esses programas que solicitam a execução de serviços disponíveis no servlet podem ser um simples formulário web, uma aplicação desktop, uma aplicação móvel rodando em um smartphone etc.

Assim, a codificação de um servlet fica muito simplificada porque todos os detalhes de máquina e software já estão codificados na classe contêiner. Um servlet é executado nas seguintes condições:

1. O usuário clica em um link que tem uma URL para um servlet;
2. O contêiner recebe a solicitação, "vê" que ela é para um servlet e instancia (monta) dois objetos: HttpServletResponse e HttpServletRequest;
3. Tendo o URL do servlet, o contêiner localiza o serlet solicitado pelo programa ou app cliente e chama (ativa a execução) um método chamado service ( ) do servlet, passando como argumentos os objetos solicitação (HttpServletRequest) e resposta (HttpServletResponse) instanciados no passo número 2;

4. O método service ( ) identifica qual o método do servlet que será chamado em função do método HTTP (GET, POST etc.) enviado pelo cliente. Na verdade o cliente envia uma solicitação HTTP GET para que o método service ( ) chame o método doGet( ) do servlet, passando como argumentos os objetos solicitação e resposta instanciados no passo 2;

5. O servlet executa o processamento e usa o objeto resposta para devolver o resultado do processamento ao contêiner;

6. Quando o computador terminar de executar o método service ( ) a thread "morre" ou vai para um estoque de threads que também é gerenciado pelo contêiner. As variáveis de referência dos objetos solicitação e resposta saem do escopo, portanto elas se tornam lixos e serão coletados mais tarde.

> *Nota: Thread* em inglês significa "linha", então se diz que o computador está executando uma sequência ou "linha" de comandos. Entretanto, determinadas sintaxes como o Java permitem a execução de várias sequências de comandos ou thread ao "mesmo" tempo.

Esse ciclo de vida fornece uma visão geral daquilo que acontece quando uma aplicação usa os "recursos" disponibilizados por um servlet. Detalhando esse ciclo de vida com uma visão um pouco mais técnica, tem-se:

a) A classe contêiner carrega a classe servlet;

b) O contêiner instancia um objeto a partir da classe servelet usando o seu construtor padrão;

c) O contêiner inicializa o servlet ativando seu método init ( ) antes que ele passe a atender qualquer solicitação enviada através do método HTTP(GET, POST etc) e respectiva chamada (doGet( ), doPost ( ) etc. da classe servlet.

d) O contêiner chama o método service ( ) a fim de atender às solicitações do cliente doGet ( ), do Post ( ) etc., e cada solicitação roda em um thread diferente.

e) O contêiner chama o método destroy ( ), permitindo que o servlet se limpe antes de ser destruído, isto é, sua variável de referência perde o foco e ele vira lixo.

7 API que compõe a arquitetura dos servlets

As classes e intefaces que compõem a API servlet estão dispostas em dois pacotes, a saber:

. javax.servlet usada para a construção de servlets genéricos

. javax.servlet.http usada para a construção de servlets específicos para rodar na web

Considerando os objetivos deste artigo, serão estudadas as classes e interfaces do pacote javax.servlet.http, isto é, construção de servlets que rodem em ambiente web e atendam a requisições enviadas via protocolo HTTP (protocolo de transferência de hipertexto).

Para construir esse tipo de servlet basta estender (criar uma subclasse) a classe javax.servlet.http.HttpServlet. Essa classe implementa a interface javax.servlet.Servlet e contém as funcionalidades adequadas para o desenvolvimento de aplicações que rodem na web.

Assim, pode-se afirmar que um servlet para rodar em ambiente web é uma subclasse da classe javax.servlet.http.HttpServlet que contém os métodos definidos no projeto da aplicação.

7.1 Pacote javax.servlet.http

A Tabela 1 apresenta as principais interfaces que compõem o pacote javax.servlet.http
Tabela 1 - Principais interfaces do pacote javax.serlet.http

Interface	**Finalidade**
HttpServletRequest	Estender a interface ServletRequest visando fornecer informações sobre requisições direcionadas a servlets usando o protocolo HTTP. Representa uma requisição recebida por um servlet.
HttpServletRespost	Estender a interface ServletReSPOST visando fornecer informações sobre o envio de respostas usando o protocolo HTTP. Representa uma resposta enviada por um servlet.
HttpSession	Permitir criar uma forma de identificar um usuário a partir de mais de uma página e armazenar as informações relativas à comunicação com o usuário.

Na Tabela 2 tem-se as principais classes do pacote javax.servlet.http

Tabela 2 - Principais classes do pacote javax.servlet.http

Classe	**Finalidade**
HttpServlet	É estendida a fim de criar um servlet que se comunique com seus clientes via HTTP
Cookie *	Usada para representar um cookie

* Cookie é um bloco de informações que é enviado do servidor para o navegador no cabeçalho de uma página. A partir daí, e em função do tempo de validade do cookie, o navegador reenvia a informação para o servidor a cada nova requisição. Opcionalmente, o cookie pode ser armazenado na máquina do cliente e quando houver nova visita ao site o cookie é enviado para o servidor, fornecendo as informações do cliente. Os cookies são importantes na identificação dos usuários e no acompanhamento de sua navegação na web, principalmente quando ele resolve fazer uma compra via web.

Exemplo:

Na Listagem 1 tem-se o código de uma página onde o usuário entra com a sua identificação (usuário e senha); na Figura 3 é feita a validação da digitação desses dados.

Basicamente, tem-se um formulário onde o cliente informa o seu usuário e senha: se a sua identificação estiver correta, o programa chama a execução do servlet servSaudacao que irá imprimir uma mensagem para o cliente. Considerar que esse trecho de programa pertence ao projeto ControleVendas.

A implementação desse programa ou app será feita usando o servidor web Tomcat e o ambiente de desenvolvimento NetBeans. Deve-se executar os seguintes passos:

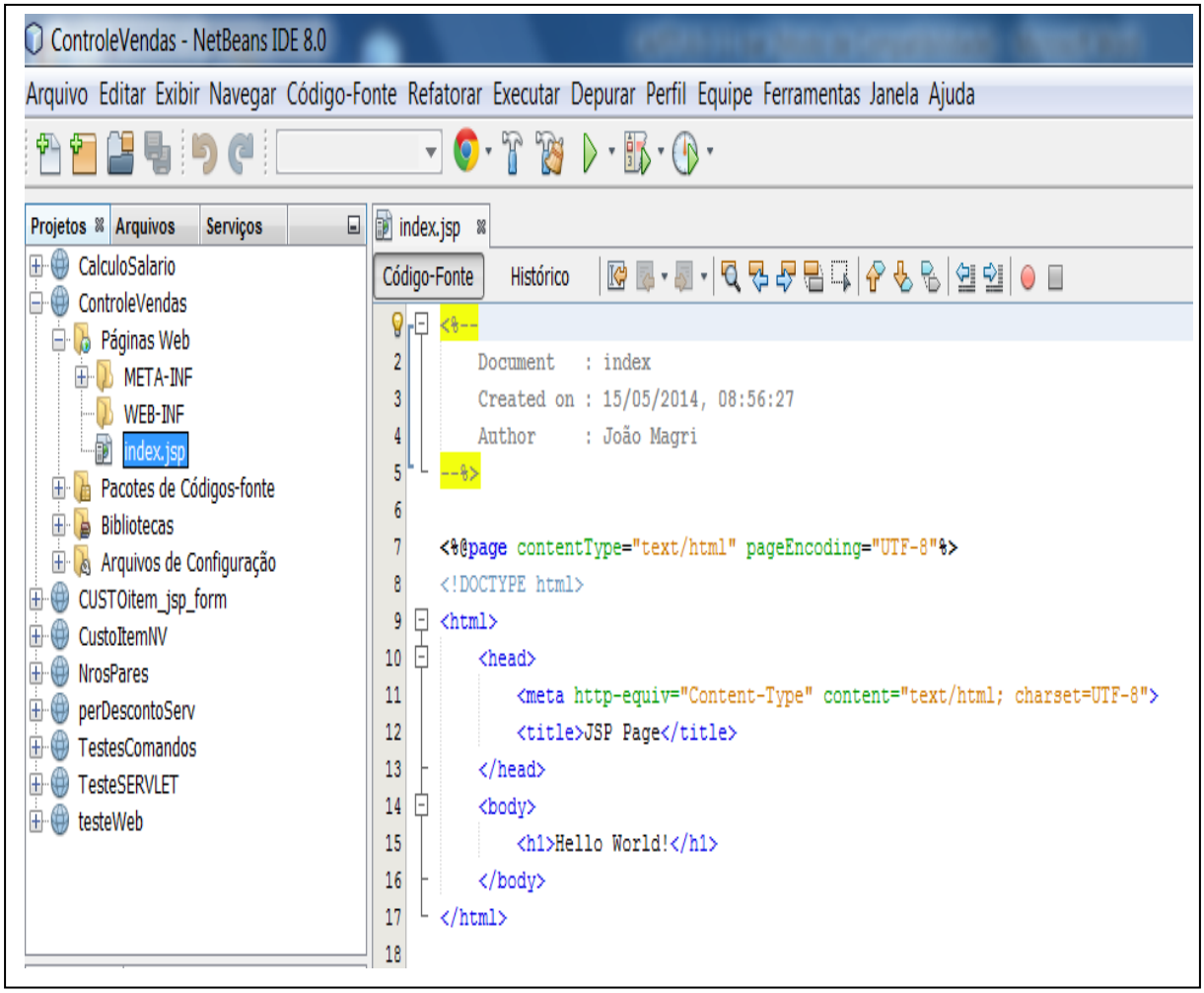
1) Ativar a execução do Tomcat, caso não esteja em execução

2) Carregar o NetBeans

3) Selecionar as seguintes opções:

Arquivo / Novo projeto / Java web / Aplicação web / clicar no botão Próximo / Digitar o nome do projeto: ControleVendas (sem espaço...) / Clicar no botão Próximo / Clicar no botão Próximo

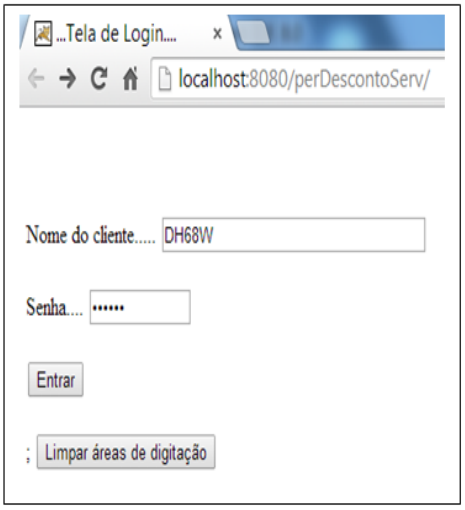
O NetBeans responde com a janela da Figura 4, onde está sendo apresentado o código da página index.jsp



Nessa página está digitado o código do programa ou app para ler o nome do cliente e sua senha, isto é, a identificação do cliente. Apagar o código gerado pelo NetBeans (vide Figura 4) e digitar o código da Listagem 1.

```
<%--
    Document   : index
    Created on : 19/02/2014, 22:34:23
    Author     : João Magri
--%>

<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
    <head>
    <title>...Tela de Login....</title>
    </head>
    <body>
      <form action="controle.jsp">
        <br /><br /><br />
        Nome do cliente.....
        <input type="text" size="35" name="login" /><br /><br />
        Senha....
        <input type="password" size="10" name="senha" /><br /><br />
        <input type="Submit" value="Entrar" /> <br> <br> ;
        <input type="reset" value="Limpar áreas de digitação" />
      </form>
    </body>
</html>
```

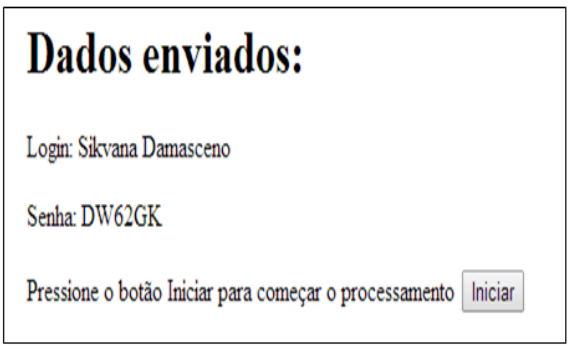


Listagem 1 - Código da página para ler a identificação do usuário.

Na Listagem 2 tem-se o código da página para validar os valores digitados e ativar a execução do servlet.

```
<%--
    Document   : LIVRO
    Created on : 19/02/2014, 01:07:07
    Author     : João Magri
--%>


<html>
 <head>
 <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
 <title>Controle JSP</title>
 </head>
   <body>
   <%
      String login = request.getParameter("login");
      String senha = request.getParameter("senha");
      byte fl = (byte)(0);
      if (login  ==  "" )
        {
          out.println("<p> Login  está  vazio </p>");
           fl = (1);
        }
      if (senha   ==  "")
        {
          out.println("<p> Senha   está  vazia </p>");
      fl = (1);
        }
      if ( fl == 0)
        {
          out.println("<h1>Dados enviados:</h1>");
          out.println("<p> Login: " + login + "</p>");
          out.println("<p> Senha: " + senha + "</p>");
        }
         else
```



```
35              out.println("Por favor informe todos os dados </p>");
36          %>
37            <form action = "servSaudacao" method = "POST">
38            <p>
39               Pressione o botão Iniciar para começar o processamento
40               <input type = "submit" value = "Iniciar" />
41            </p>
42            </form>
43        </body>
44    </html>
45
```

Listagem 2 - Código da página para validar a digitação e ativar a execução do Servlet.

Nas linhas de 37 a 44 tem-se o código do formulário que é impresso após a validação dos dados. Observar que na linha 37 tem-se a propriedade action da cláusula form: o valor dessa propriedade é o nome da classe que contém o servlet e não o nome do arquivo.

Na Listagem 3 tem-se o código do servlet que imprime a mensagem de saudação do cliente.

```
1   /*¶
2   ·*·To·change·this·license·header,·choose·License·Headers·in·Project·Properties.¶
3   ·*·To·change·this·template·file,·choose·Tools·|·Templates¶
4   ·*·and·open·the·template·in·the·editor.¶
5   ·*/¶
6   ¶
7   import·java.io.IOException;¶
8   import·java.io.PrintWriter;¶
9   import·javax.servlet.ServletException;¶
10  import·javax.servlet.http.HttpServlet;¶
11  import·javax.servlet.http.HttpServletRequest;¶
12  import·javax.servlet.http.HttpServletResponse;¶
13  ¶
14  /**¶
15  ·*¶
16  ·*·@author·João·Magri¶
17  ·*/¶
18  public·class·servSaudacao·extends·HttpServlet·{¶
19  ¶
20  ····/**¶
21  ·····*·Processes·requests·for·both·HTTP·<code>GET</code>·and·<code>POST</code>¶
22  ·····*·methods.¶
23  ·····*¶
24  ·····*·@param·request·servlet·request¶
25  ·····*·@param·response·servlet·response¶
26  ·····*·@throws·ServletException·if·a·servlet-specific·error·occurs¶
27  ·····*·@throws·IOException·if·an·I/O·error·occurs¶
28  ·····*/¶
29  ····protected·void·processRequest(HttpServletRequest·request,·HttpServletResponse·response)¶
30  ··········throws·ServletException,·IOException¶
31  ····{¶
32  ·····response.setContentType("text/html;charset=UTF-8");¶
33  ·····PrintWriter·out·=·response.getWriter();¶
34  ·····try·{····¶
```

```
35          ··········out.println("<!DOCTYPE·html>");¶
36          ··········out.println("<html>");¶
37          ··········out.println("<head>");¶
38          ··········out.println("<title>Executando·o·Servlet·servSaudacao</title>");··········¶
39          ··········out.println("</head>");¶
40          ··········out.println("<body>");¶
41          ··········out.println("<h1>·<br><br>·Olá!··Bem-vindo·a·Loja·virtual·Moura·S.·A.·</h1>");¶
42          ··········¶
43          ··········out.println("<br><br><h2>Servlet·servSaudacao·at·"·+·request.getContextPath()·+·"</h2>");¶
44          ··········out.println("</body>");¶
45          ··········out.println("</html>");¶
46          ··········¶
47          ·····}¶
48          ·····finally¶
49          ·····{¶
50          ··········out.close();¶
51          ·····}¶
52          ·····¶
53          ····}¶
54          ¶
55          ·····/**¶
56          ·····*·Handles·the·HTTP·<code>GET</code>·method.¶
57          ·····*¶
58          ·····*·@param·request·servlet·request¶
59          ·····*·@param·response·servlet·response¶
60          ·····*·@throws·ServletException·if·a·servlet-specific·error·occurs¶
61          ·····*·@throws·IOException·if·an·I/O·error·occurs¶
62          ·····*/¶
63          ····¶
64          ····protected·void·doGet(HttpServletRequest·request,·HttpServletResponse·response)¶
65          ··········throws·ServletException,·IOException·{¶
66          ·····processRequest(request,·response);¶
67          ····}¶
68          ¶
69          ····/**¶
```

```
69          ····/**¶
70          ·····*·Handles·the·HTTP·<code>POST</code>·method.¶
71          ·····*¶
72          ·····*·@param·request·servlet·request¶
73          ·····*·@param·response·servlet·response¶
```

```
74          ·····*·@throws·ServletException·if·a·servlet-specific·error·occurs¶
75          ·····*·@throws·IOException·if·an·I/O·error·occurs¶
76          ·····*/¶
77          ¶
78          ····protected·void·doPost(HttpServletRequest·request,·HttpServletResponse·response)¶
79          ··········throws·ServletException,·IOException·{¶
80          ·····processRequest(request,·response);¶
81          ····}¶
82          ¶
83          ····/**¶
84          ·····*·Returns·a·short·description·of·the·servlet.¶
85          ·····*¶
86          ·····*·@return·a·String·containing·servlet·description¶
87          ·····*/¶
88          }¶
```

Listagem 3 - Servlet para imprimir a mensagem de saudação ao cliente.

Na Figura 4 tem-se a saída impressa pelo servlet servSaudação.

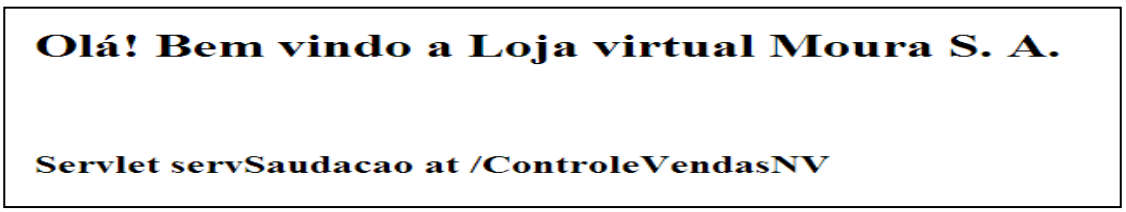


Figura 4 - Resultado da impressão a partir do servlet servSaudacao

Observar que o código do servlet é o código da subclasse da classe HttpServlet: na linha 18 tem-se o comando que estende essa superclasse ou classe base. A classe servlet tem sempre três métodos: doGet ( ) (início ou assinatura na linha 64) , doPost ( ) (início ou assinatura na linha 78) e processRequest ( ) (início ou assinatura na linha 29). O método processRequest ( ) monta a página de resposta nas linhas 32 a 45. Nesse trecho o código foi colocado dentro de uma estrutura try.. catch (linhas 34 até 51) visando ao rastreamento de erros de execução específicos do servlet e erros de entrada e saída.

Resumo

Esta pesquisa analisou o conceito de servlet web e sua aplicação na implementação de sistemas quando for usada a plataforma Java. Foi possível perceber a importância da utilização do ambiente de desenvolvimento NetBeans no desenvolvimento de uma aplicação do porte de um servlet: boa parte do código foi gerada por essa ferramenta,

Referências bibliográficas

FREEMAN, E. Use a cabeça HTML. Rio de Janeiro: Alta Books, 2006.
GONÇALVES, E. Desenvolvendo aplicações web. Rio de Janeiro: Ciência Moderna, 2007.
LERVIK, E.Java the UML way.Chichester: Willey, 2000.
MAGRI, J. A. Programação C#. 1. Ed. São Paulo: Editora Érica, 2014.
MAGRI, J. A. Criando e usando web services. Revista Acadêmica Augusto Guzzo. (eletrônica) n.11, S. Paulo: Ed. Campos Salles, 2013.
http://www.fics.edu.br/index.php/augusto_guzzo/article/view/160/232
_____. Acesso a coleções de dados usando LINQ. **Revista Acadêmica Augusto Guzzo** (eletrônica) . n. 9,S. Paulo: Ed. Campos Salles, 2012.
http://www.fics.edu.br/index.php/augusto_guzzo/article/view/33/18
_____. Arquitetura Dirigida a Modelos (MDA): utilizando Modelos no Desenvolvimento de Sistemas. **Revista Acadêmica Augusto Guzzo (eletrônica)** . n.8, S. Paulo,: Ed. Campos Salles, 2008. http://www.fics.edu.br/index.php/augusto_guzzo/article/view/42
_____. e RODRIGUES, V. B. **Princípios da arquitetura dirigida a modelos (MDA).** Trabalho de Conclusão de Curso (Graduação em Ciência da Computação). Fundação Instituto Tecnológico de Osasco, São Paulo, 2007.
SEBESTA, R. **Programming the world wide web**. Boston: Addison-Wesley, 2010.
SIERRA, K.**Use a cabeça Java**. Rio de Janeiro: Alta Books, 2005.