



Hailton David Lemos(hailton@terra.com.br).Tecnólogo em Internet e Redes, Bacharel em Administração de Empresas, Especialista em: Tecnologia da Informação, MBA em Planejamento e Gestão Estratégica, Matemática e Estatística. Trabalha com desenvolvimento de Sistema há mais de 20 anos, atualmente desenvolve sistemas especialistas voltados a Planejamento Estratégico, Tomada de Decisão e Gestão utilizando tecnologias Java, Perl, OWC, VBA, Openoffice, Office.

Trabalhando com JSP, Ajax, Servlet.

O objetivo deste artigo é propiciar ao leitor uma visão mais simplista da integração entre JSP, Ajax e Servlet, tornando mais agradável e fácil a utilização desta tecnologia, que não é nova, e que por vezes fica obscura dentro de pacotes e calhamaços de livros.

Mas, O que é AJAX?

AJAX (Asynchronous JavaScript and XML) é um termo criado para designar duas características dos browsers, efetuar pedidos ao servidor sem ter que fazer um refresh, e trabalhar com documentos XML, **Figura 01**. Entretanto esta segunda característica muitas vezes pode se tornar confusa para alguns, pode ser facilmente substituída por uma seqüência lógica mais simples, e com o mesmo resultado, e à medida que o usuário for se familiarizando com o XML poderá ir adotando como padrão.



Figura 01. Ajax

O Ajax não é uma nova linguagem de programação, mas sim uma técnica para criar melhores, mais rápidas, mais eficientes e interativas aplicações web. A técnica do Ajax faz com que as aplicações para Internet, fiquem menores, e tenha um uso mais amigável. Com o Ajax é possível fazer uma comunicação diretamente com o servidor, utilizando o objeto do JavaScript **XMLHttpRequest**. O objeto **XMLHttpRequest** é o responsável por criar a interface web.

As diretivas utilizadas no Ajax são muito bem definidas, e suportada pela maioria dos browsers disponíveis no mercado, (Internet Explorer 5.0+, Safári 1.2, Mozilla 1.0, Firefox, Opera 8+, e Netscape 7), porque o Ajax é uma aplicação independente de browser ou plataforma.

O Ajax trabalha com quatro princípios básicos: O navegador hospeda uma aplicação, e não conteúdo; O servidor fornece dado, e não conteúdo; A interação do utilizador com a aplicação pode ser flexível e continua; Real codificação requer disciplina, e isto requer a aplicação das boas praticas; e a técnica do Ajax é baseada em quatro pilares da Web: JavaScript; XML; HTML; CSS.

Trabalhando com Ajax, JSP e Servlet.

Chegou a hora, vamos fazer a tecnologia funcionar. O primeiro passo que temos que fazer é escolher a IDE, pode se utilizar o Netbeans, **figura 2**, o Eclipse ou qualquer outra IDE, fica por conta do leitor esta escolha, neste caso estarei utilizando o Netbeans, versão 5.0.

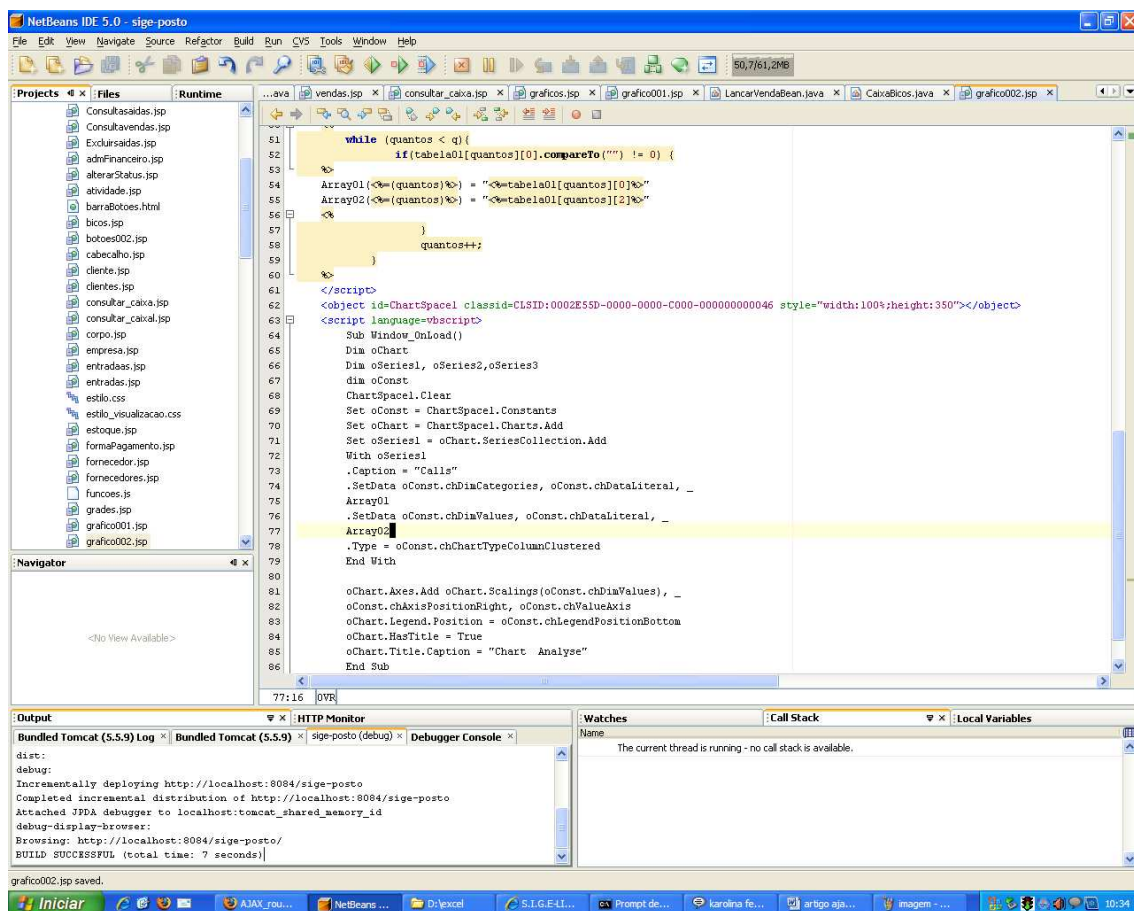


Figura 2. Netbeans 5.0

Uma vez definida a IDE que será utilizado, é hora de criar o projeto. O nome do projeto será Ajax. Após a criação do projeto na IDE, deverá ser criado também um pacote, que irá armazenar o objeto Cidades, **listagem 2**, bem como o Servlet,

listagem 3, e o nome deste pacote será exemplo, a **figura 3** contempla como ficara a IDE após a execução destes passos.

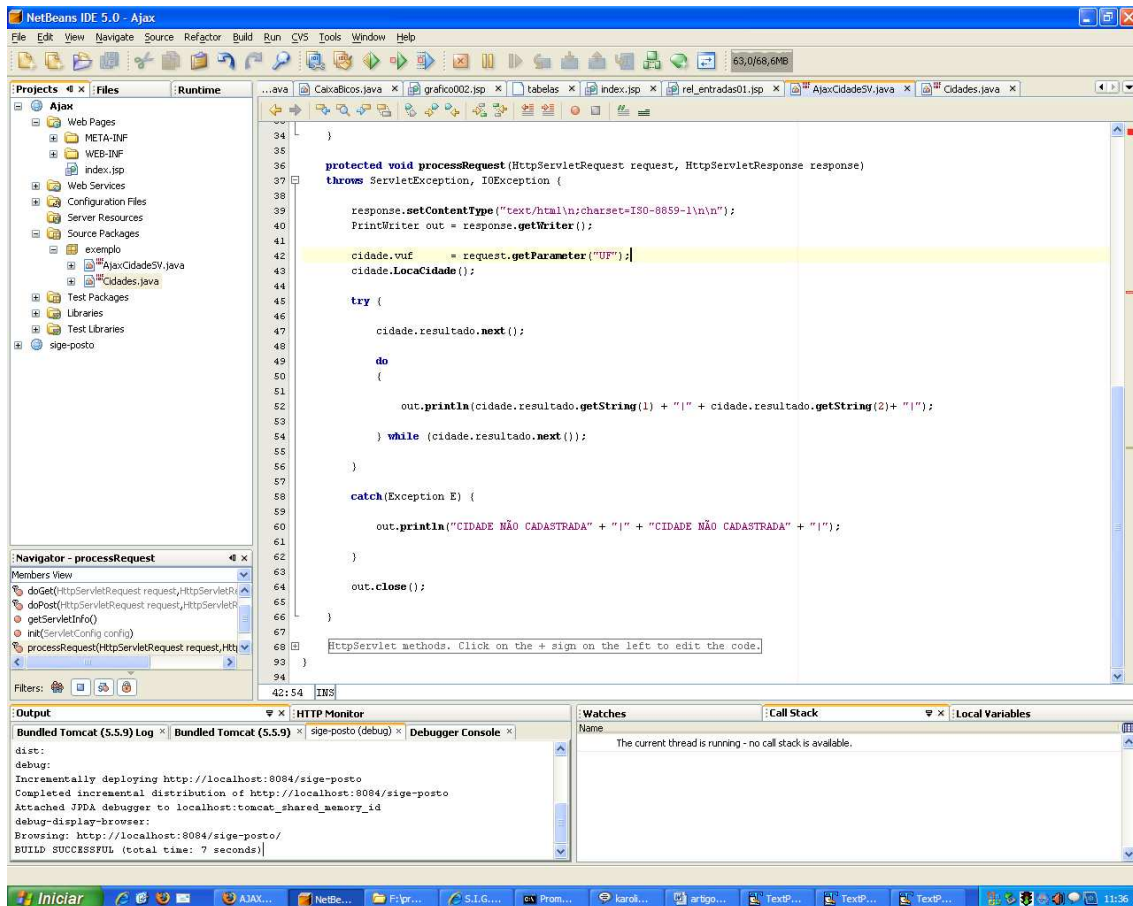


Figura 3. Estrutura da IDE

Quando a aplicação é gerada a IDE já cria o programa `index.jsp`, neste caso iremos alterá-lo para que possa atender as necessidades do exemplo que esta sendo desenvolvido, **listagem 1**.

```

<%@page contentType="text/html"%>
<%@page pageEncoding="ISO-8859-1"%>
<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1" /
  >
  <script language="javascript">
    var url = "";
    var xmlhttp = "";
    function obterObjRequisicao(UF){
      url = "<%=request.getContextPath()%>/"+ "AjaxCidadeSV?UF=" + UF;
      if (window.XMLHttpRequest) {
        xmlhttp = new XMLHttpRequest();
      }
      else {
        try{
          xmlhttp = new ActiveXObject("Msxml2.XMLHTTP");
        }
        catch (e) {
          try {

```

```

xmlhttp = new ActiveXObject("Microsoft.XMLHTTP");
}
catch (E) {
xmlhttp = false;
}
}
}
if(!xmlhttp && window.createRequest) {
try {
xmlhttp = window.createRequest();
}
catch (e) {
xmlhttp=false;
}
}
try {
xmlhttp.open("GET", url, true);
xmlhttp.onreadystatechange = function() {
if (xmlhttp.readyState == 4){
if(xmlhttp.status == 200){
processar();
}
}
}
try {
xmlhttp.send(null);
}
catch(e){
xmlhttp="";
}
}
catch (e) {
try {
xmlhttp.send(null)
xmlhttp="";
}
catch(e){
xmlhttp="";
}
}
function processar(){
var result = xmlhttp.responseText;
var quebra = result.split("\n");
var tamanha = quebra.length - 1;
var select = document.getElementById("cidades");
select.innerHTML = "";
for(i = 0; i < tamanha; i++) {
var quebre = quebra[i].split("|");
var oSelect = document.getElementsByName('cidades')[0].options.length
nextOptIndex = quebre[0];
newOpt = document.createElement('option');
newOpt.text = quebre[1];
newOpt.value = nextOptIndex;
document.getElementsByName('cidades')[0].options.add(newOpt, oSelect)
}
}
}

```

```

</script>
<title>
  Cidades
</title>
</head>
<body>
  <jsp:include page="cabecalho.jsp" />
  <form method="post" name="form" action="xxxx.jsp ">
    <table align="center">
      <tr>
        <td align="right">
          Estado/Cidade
        </td>
        <td>
          <select name="estados" id="estados"
onChange="obterObjRequisicao(this.value);">
            <option selected value="NDA">-- Estados --</option>
            <option value="GO">Goiás</option>
            <option value="MG">Minas Gerais</option>
            <option value="SP">São Paulo</option>
          </select>
          <select name="cidades" id="cidades">
            <option>-- Escolha uma Cidade --</option>
          </select>
        </td>
      </tr>
    </table>
  </form>
</body>
</html>

```

Listagem 1.

```

/*
 * Cidades.java
 *
 * Created on 6 de Junho de 2005, 15:07
 */
package Negocio;
import java.sql.*;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.SQLException;
/**
 *
 * @author Hailton David Lemos
 */
public class Cidades
{
  protected String vid;
  protected String vcidade;
  protected String vuf;
  protected Connection conexao;
  protected PreparedStatement declaracao;
  protected Statement stm;
  protected ResultSet resultado;
  protected String comando;

```

```

protected String vmensagem;
public Cidades ()
{
    vid      = "";
    vcidade  = "";
    vmensagem = "";
    vuf      = "";
}

protected boolean GravaCidade (int opcao)
{
    try
    {
        Class.forName ("com.mysql.jdbc.Driver").newInstance ();
    }
    catch(Exception E)
    {
        vmensagem = "Erro na abertura do Cidades";
        return false;
    }

    try
    {
        try
        {
            conexao.close ();
            stm.close ();
        }
        catch(Exception sqle )
        {

        }

        conexao = DriverManager.getConnection ("jdbc:mysql://localhost/exemplo
", "root","");
        stm      = conexao.createStatement ();
        if(opcao == 1)
        {
            comando = "insert into cidade (";
            comando = comando + "cidade, ";
            comando = comando + "uf) ";
            comando = comando + "values (";
            comando = comando + "" + vcidade + ", ";
            comando = comando + "" + vuf + ")";
        }
        else if (opcao == 2)
        {
            comando = "update cidade set ";
            comando = comando + ", cidade = " + vcidade;
            comando = comando + ", uf = " + vuf;
            comando = comando + " where id = " + vid + """;
        }
        else if (opcao == 3)
        {
            comando = "delete from cidade where id = " + vid + """;
        }
        else if (opcao == 4)
        {

```

```

        comando = "select * from cidade where uf = '" + vuf + "' order by
cidade";
    }
    else if (opcao == 5)
    {
        comando = "select * from cidade";
    }
    else if (opcao == 6)
    {
        comando = "select cidade from where cidade = '" + vcidade + "'";
    }
    else if (opcao == 7)
    {
        comando = "select * from cidade where id = '" + vid + "'";
    }
    if(opcao <= 3 || opcao == 8 || opcao == 9)
    {
        stm.executeUpdate (comando);
    }
    else
    {
        resultado = stm.executeQuery (comando);
    }
    if(opcao == 5 || opcao == 7)
    {
        try
        {
            resultado.next ();
            vid          = resultado.getString (1);
            vcidade      = resultado.getString (2);
            vuf          = resultado.getString (3);
        }
        catch(SQLException E1)
        {
            vmensagem = "nao consegui executar o query" + comando;
        }
    }
    else if (opcao == 6)
    {
        try
        {
            resultado.next ();
        }
        catch(SQLException E1)
        {
            vmensagem = "nao consegui executar o query de busca " +
comando;
        }
    }
}
catch(Exception E)
{
    vmensagem = comando + " ERRO";
    return false;
}
return true;
}
}

```

```
public boolean IncCidade ()
{
    if(GravaCidade(1))
    {
        vmensagem = "A inclusao efetuada com sucesso\n";
        return true;
    }
    else
    {
        return false;
    }
}
public boolean AltCidade ()
{
    if(GravaCidade(2))
    {
        vmensagem = "A alteraçã foi efetuada com sucesso\n";
        return true;
    }
    else
    {
        return false;
    }
}
public boolean ExcCidade ()
{
    if(GravaCidade(3))
    {
        vmensagem = "A exclusã foi efetuada com sucesso\n";
        return true;
    }
    else
    {
        return false;
    }
}
public boolean LocaCidade ()
{
    if(GravaCidade (4))
    {
        return true;
    }
    else
    {
        return false;
    }
}
public boolean BuscaCidade ()
{
    if(GravaCidade (6))
    {
        return true;
    }
    else
    {
        return false;
    }
}
```

```

    }
    public boolean PsqCidade ()
    {
        if(GravaCidade (7))
        {
            return true;
        }
        else
        {
            return false;
        }
    }
}

```

Listagem 2. Cidades.java

```

/*
 * AjaxCidadeSV.java
 *
 * Created on 4 de Outubro de 2008, 10:17
 */
package exemplo;
import java.io.*;
import java.net.*;
import javax.servlet.*;
import javax.servlet.http.*;
/**
 *
 * @author Hailton David Lemos
 * @version
 */
public class AjaxCidadeSV extends HttpServlet {

    protected Cidades cidade;

    public void init(ServletConfig config) throws ServletException {

        cidade = new Cidades();
        super.init(config);

    }
    public void destroy() {
        super.destroy();
    }
    protected void processRequest(HttpServletRequest request, HttpServletResponse
response)
throws ServletException, IOException {
        response.setContentType("text/html\n; charset=ISO-8859-1\n\n");
        PrintWriter out = response.getWriter();
        cidade.vuf = request.getParameter("UF");
        cidade.LocaCidade();
        try {
            cidade.resultado.next();
            do
            {
                out.println(cidade.resultado.getString(1) + "|" +
cidade.resultado.getString(2)+ "|");

```

```

        } while (cidade.resultado.next());
    }
    catch(Exception E) {
        out.println("CIDADE NÃO CADASTRADA" + "|" + "CIDADE NÃO
CADASTRADA" + "|");
    }
    out.close();
}
// <editor-fold defaultstate="collapsed" desc="HttpServlet methods. Click on the
+ sign on the left to edit the code.">
/** Handles the HTTP <code>GET</code> method.
 * @param request servlet request
 * @param response servlet response
 */
protected void doGet(HttpServletRequest request, HttpServletResponse
response)
throws ServletException, IOException {
    processRequest(request, response);
}
/** Handles the HTTP <code>POST</code> method.
 * @param request servlet request
 * @param response servlet response
 */
protected void doPost(HttpServletRequest request, HttpServletResponse
response)
throws ServletException, IOException {
    processRequest(request, response);
}
/** Returns a short description of the servlet.
 */
public String getServletInfo() {
    return "Short description";
}
// </editor-fold>
}

```

Listagem 3. AjaxCidadeSV.java

O segundo passo é criar um banco de dados com uma tabela, que conterá os dados que serão processados pelo Servlet e retornados a pagina dinamicamente. Para este exemplo, o nome sugerido para o banco é `aprendizado` e para a tabela é `idades`, e a versão do Mysql que esta sendo utilizada é a versão 4.0.12.

A estrutura da tabela será bastante simples, e para criar toda a estrutura iremos utilizar a linha de comando, e digitar as instruções SQL diretamente no prompt do Mysql, **Listagem 1**. Se o leitor optar por utilizar alguma ferramenta gráfica para executar os passos não há problema algum, fica a critério do leitor a forma de gerar o banco bem como também a tabela.

Iremos também popular a tabela com algumas informações, uma vez que o objetivo do artigo é trabalhar o Ajax e não a construção de uma aplicação como um todo.

Comandos SQL:

```
Create database aprendizado;
```

```
Use aprendizado;
```

```
Create table cidades(  
Id int not null auto_increment primary key,  
Cidade varchar(50),  
Estado varchar(2),  
);
```

```
Insert into cidades(Cidade, Estado) values("Goiania", "GO");  
Insert into cidades(Cidade, Estado) values("Guapo", "GO");  
Insert into cidades(Cidade, Estado) values("Crominia", "GO");  
Insert into cidades(Cidade, Estado) values("Barro Alto", "GO");  
Insert into cidades(Cidade, Estado) values("São Paulo", "SP");  
Insert into cidades(Cidade, Estado) values("São Vicente", "SP");  
Insert into cidades(Cidade, Estado) values("Campinas", "SP");  
Insert into cidades(Cidade, Estado) values("Belo Horizonte", "MG");  
Insert into cidades(Cidade, Estado) values("Passos", "MG");  
Insert into cidades(Cidade, Estado) values("Guaxupe", "MG");
```

Listagem 4. Instruções SQL.

Entendendo o funcionamento.

Uma vez montada toda estrutura, é hora de fazer o deploy do projeto e testá-lo.

Conclusão

Como pode ser visto a tecnologia Ajax pode ser utilizada de uma forma bem simplista para resolver problemas simples, e até mesmo problemas mais complexos.