www.devmedia.com.br
[versão para impressão]
Link original: http://www.devmedia.com.br/articles/viewcomp.asp?comp=26008

Executando consultas ao MySQL com PHP e AJAX

Veja neste artigo como executar consultas rápidas e dinâmicas a bancos de dados MySQL utilizando PHP e AJAX.

Olá pessoal, nesse artigo será demonstrado como executar consultas em um banco de dados MySQL utilizando AJAX e PHP. Exibir resultados de consultas utilizando PHP é bem comum, mas nesse exemplo vamos incrementar a nossa consulta utilizando AJAX.

Mas o que é AJAX?

Resposta: Ajax é uma sigla para Asynchronous JavaScript and XML. Basicamente é o uso metodológico de tecnologias como JavaScript e XML, providas por navegadores, para tornar páginas Web mais interativas com o usuário, utilizando-se de solicitações assíncronas de informações.

Observação: AJAX não é uma linguagem, mas uma forma de construir um site utilizando diversas tecnologias.

Para desenvolver esse artigo foram utilizadas as seguintes ferramentas:

- WAMPSERVER (Apache, MySQL, PHP5)
- Netbeans 7.2 (Você pode utilizar o editor de sua preferência)
- MySQL Query Browser (Gerenciamento do banco de dados)

** Todas as ferramentas podem ser adquiridas gratuitamente na internet.*

Vamos criar um banco de dados "Agenda" contendo apenas uma tabela de "Contato". Para criar a base de dados usamos a ferramenta MySQL Query Browser, segue abaixo script:

Listagem 1: Script para criar o banco "Agenda".

```
CREATE DATABASE `Agenda`;
CREATE TABLE `Contato` (
  `ID` int(11) NOT NULL AUTO_INCREMENT,
  `NOME` varchar(100) DEFAULT NULL,
  `FONE` varchar(15) NOT NULL,
  `CELULAR` varchar(15) NOT NULL,
  `EMAIL` varchar(50) DEFAULT NULL,
  PRIMARY KEY (`ID`)
) ENGINE=InnoDB AUTO_INCREMENT=2 DEFAULT CHARSET=utf8;
```

Agora vamos criar três arquivos:

- index.html - vai ser nossa página de interação com o usuário;
- ajax.js - esse arquivo vai possuir toda a lógica para as requisições;
- contato.php - onde será feita a conexão com o banco e executada as consultas.

Aqui não será feito uso de CSS ou personalização de componentes na página, esse não é o foco do artigo.

O arquivo index.html

Listagem 2: Código HTML da página index.html

```
<!DOCTYPE html>
<html>
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
        <title></title>
    </head>
    <body>
        <script type="text/javascript" src="ajax.js"></script>
        <div id="Container">
            <h1>Agenda de Contatos utilizando AJAX</h1>
            <hr/>

            <h2>Pesquisar Contato:</h2>
            <div id="Pesquisar">
                Infome o nome:
                <input type="text" name="txtnome" id="txtnome"/>
                <input type="button" name="btnPesquisar" value="Pesquisar" onclick="getDados();"/>
            </div>
            <hr/>

            <h2>Resultados da pesquisa:</h2>
            <div id="Resultado"></div>
            <hr>

        </div>
    </body>
</html>
```

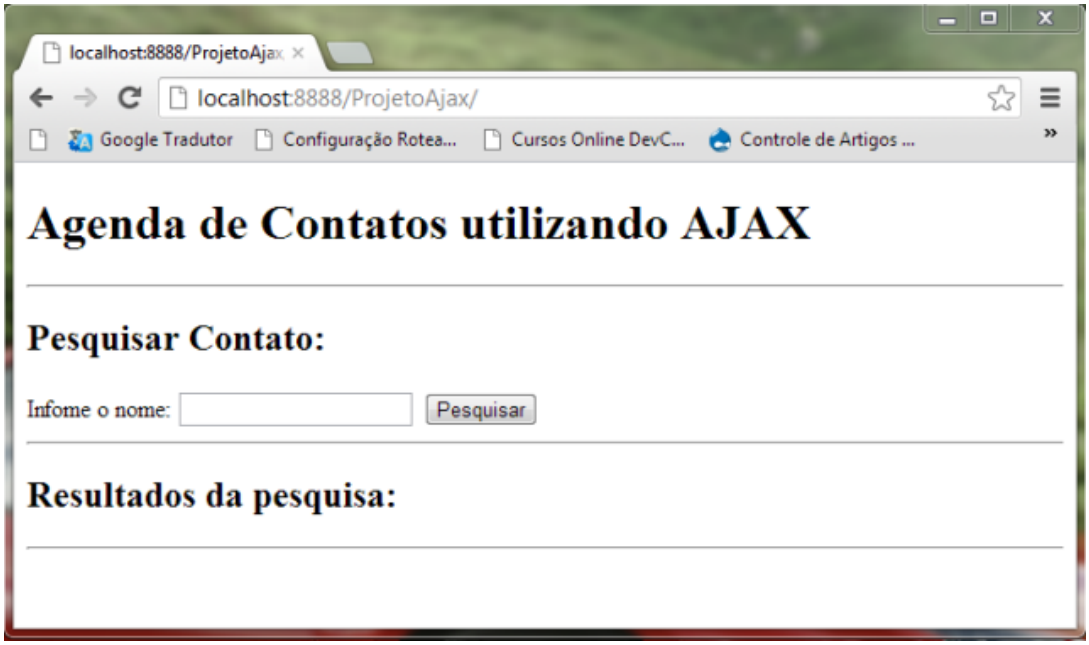

Figura 1: Página no navegador

O código HTML não tem muito segredo, mas vale destacar duas linhas que merecem importância nesse código.

Estamos importando no arquivo que vai executar toda a mágica do AJAX:

No evento onclick do botão estamos chamando a função getDados() que foi escrita no arquivo ajax.js:

O arquivo ajax.js

Como estamos falando em arquivo ajax.js então vamos criar o código, como segue abaixo:

Listagem 3: Conteúdo do arquivo ajax.js.

```
/**
  * Função para criar um objeto XMLHTTPRequest
  */
 function CriaRequest() {
     try{
         request = new XMLHttpRequest();
     }catch (IEAtual){

         try{
             request = new ActiveXObject("Msxml2.XMLHTTP");
         }catch(IEAntigo){

             try{
                 request = new ActiveXObject("Microsoft.XMLHTTP");
             }catch(falha){
                 request = false;
             }
         }
     }

     if (!request)
```

```
        alert("Seu Navegador não suporta Ajax!");
    else
        return request;
}

/**
 * Função para enviar os dados
 */
function getDados() {

    // Declaração de Variáveis
    var nome   = document.getElementById("txtnome").value;
    var result = document.getElementById("Resultado");
    var xmlreq = CriaRequest();

    // Exibi a imagem de progresso
    result.innerHTML = '<img src="Progresso1.gif"/>';

    // Iniciar uma requisição
    xmlreq.open("GET", "Contato.php?txtnome=" + nome, true);

    // Atribui uma função para ser executada sempre que houver uma mudança de ado
    xmlreq.onreadystatechange = function(){

        // Verifica se foi concluído com sucesso e a conexão fechada (readyState=4)
        if (xmlreq.readyState == 4) {

            // Verifica se o arquivo foi encontrado com sucesso
            if (xmlreq.status == 200) {
                result.innerHTML = xmlreq.responseText;
            }else{
                result.innerHTML = "Erro: " + xmlreq.statusText;
            }
        }
    };
    xmlreq.send(null);
}
```

O código está comentado, mas vale ressaltar que para trabalhar com requisições AJAX temos que criar um objeto do tipo XMLHTTPRequest, mas esse objeto não é padrão para todos os navegadores, então a função "CriaRequest()" executa uma série de verificações sobre o tipo de navegador e retorna um objeto XMLHTTPRequest. Basicamente Firefox, Opera, Safari e Chrome utilizam o mesmo tipo, mas para o Internet Explorer é utilizado ActiveXObject que muda conforme a versão do I.E, então é feita outra verificação para criar um objeto compatível com a versão do navegador em questão. Podem existir navegadores que não suportem AJAX então nesse caso será emitida uma mensagem avisando o usuário desse problema.

Na função "getDados()" é onde ocorre toda a mágica do AJAX, basicamente essa função trabalha com requisições e respostas, nada muito diferente do conceito do desenvolvimento WEB, a diferença é que nesse caso não será necessária uma nova "carga" na página para carregar as respostas, aquele famoso refresh.

Foram Declaradas 3 variáveis: a primeira é "nome" que vai receber o conteúdo do componente "txtnome" para pesquisar, a segunda é "result" que recebe o local da página onde queremos que seja carregada a resposta da requisição e a terceira recebe um objeto XMLHTTPRequest que é retornado da função CriaRequest().

Notem que foi atribuída uma animação (Progresso1.gif) para a variável result, ou seja, essa animação será carregada no início da requisição, uma espécie de progresso enquanto usuário aguarda o retorno da pesquisa.

Uma vez instanciado o objeto XMLHTTPResquest temos que abrir uma requisição chamando o método open(). Esse método requer 3 parâmetros:

- Primeiro argumento define qual método de envio deverá ser utilizado (GET ou POST).
- Segundo argumento especifica a URL do script no servidor.
- Terceiro argumento especifica que a requisição deverá ser assíncrona.

O método send() envia a requisição para o servidor. A requisição pode retornar dois tipos de resposta, responseText e responseXml. Vamos usar nesse artigo o tipo responseText.

Depois vamos ficar monitorando a propriedade readyState através do evento onreadystatechange, até o seu valor ser igual a 4 (requisição finalizada). A segunda condição verifica se o status é igual a 200 (“OK”), se essas duas condições forem atendidas é só carregar o conteúdo da resposta no local previamente definido na variável result (

), senão carrega a mensagem de erro.

A imagem abaixo demonstra como funciona uma requisição normal e outra usando AJAX.

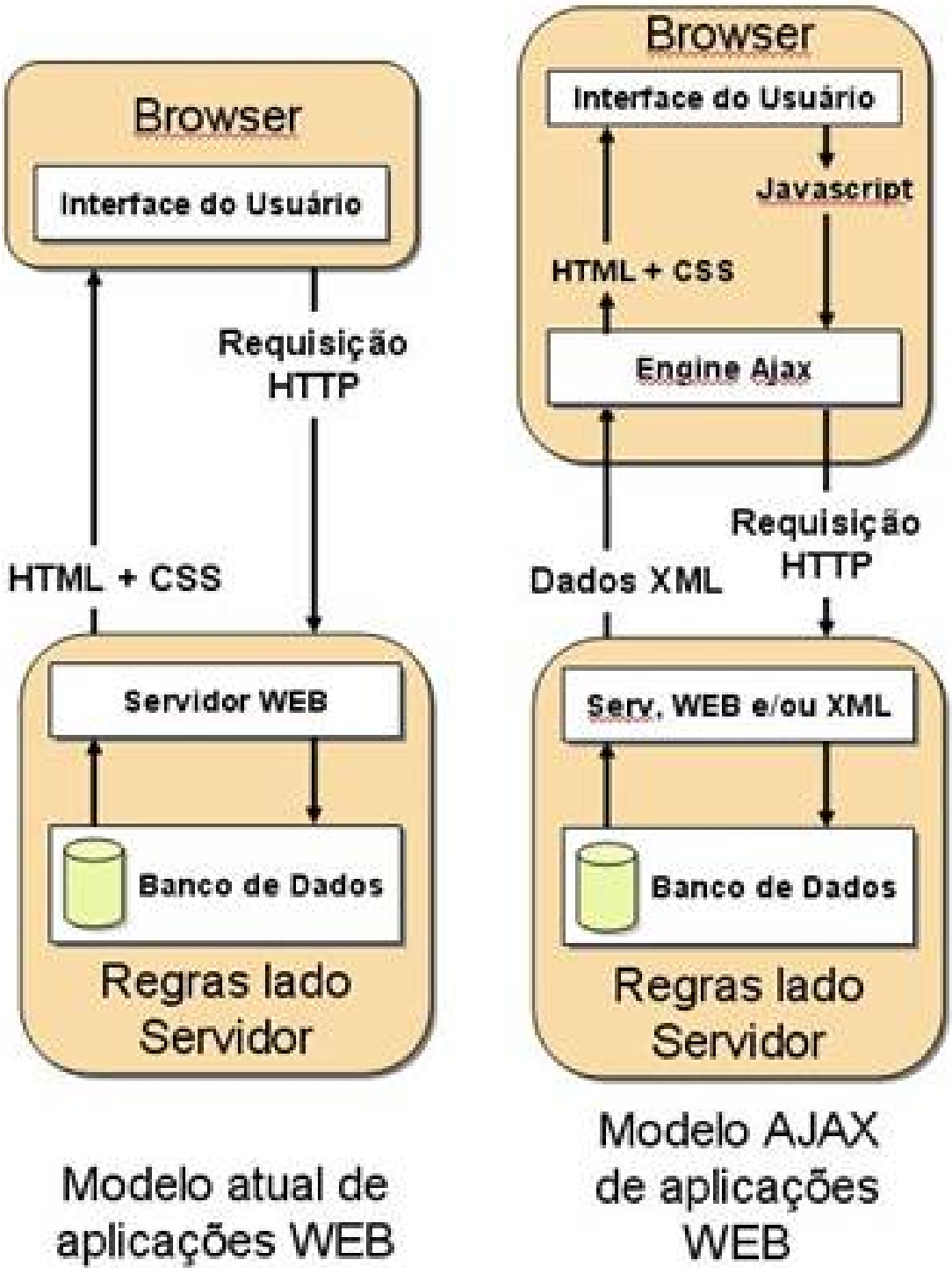

Figura 2: Fluxo de trabalho de uma página WEB

O arquivo contato.php

Nesse arquivo vamos conectar ao banco e executar nossa consulta de verdade, até agora preparamos o ambiente para receber nossa requisição e carregar nossa resposta.

Listagem 4: Código php para executar consulta.

```
<?php
// Verifica se existe a variável txtnome
if (isset($_GET["txtnome"])) {
    $nome = $_GET["txtnome"];
    // Conexao com o banco de dados
    $server = "localhost";
    $user = "root";
    $senha = "011224";
    $base = "agenda";
    $conexao = mysql_connect($server, $user, $senha) or die("Erro na conexão!");
    mysql_select_db($base);
    // Verifica se a variável está vazia
    if (empty($nome)) {
        $sql = "SELECT * FROM contato";
    } else {
        $nome .= "%";
        $sql = "SELECT * FROM contato WHERE nome like '$nome'";
    }
    sleep(1);
    $result = mysql_query($sql);
    $cont = mysql_affected_rows($conexao);
    // Verifica se a consulta retornou linhas
    if ($cont > 0) {
        // Atribui o código HTML para montar uma tabela
        $tabela = "<table border='1'>
                    <thead>
                        <tr>
                            <th>NOME</th>
                            <th>TELEFONE</th>
                            <th>CELULAR</th>
                            <th>EMAIL</th>
                        </tr>
                    </thead>
                    <tbody>
                    <tr>";
        $return = "$tabela";
        // Captura os dados da consulta e inseri na tabela HTML
        while ($linha = mysql_fetch_array($result)) {
            $return.= "<td>" . utf8_encode($linha["NOME"]) . "</td>";
            $return.= "<td>" . utf8_encode($linha["FONE"]) . "</td>";
            $return.= "<td>" . utf8_encode($linha["CELULAR"]) . "</td>";
            $return.= "<td>" . utf8_encode($linha["EMAIL"]) . "</td>";
            $return.= "</tr>";
        }
        echo $return."</tbody></table>";
    } else {
        // Se a consulta não retornar nenhum valor, exibi mensagem para o usuário
        echo "Não foram encontrados registros!";
    }
}
?>
```

O Código está comentado, mas alguns pontos merecem destaque: logo no início verificamos se existe a variável "txtnome", caso exista o código php será executado, usamos o a função "sleep(1)" para congelar por 1 segundo a execução do código e exibir a nossa animação (Progresso1.gif). A conexão com o banco e a forma de executar as instruções SQL para consulta não tem nenhum segredo. Verifica se a variável "$nome" que recebeu o conteúdo de "txtnome" está vazia, caso esteja executamos uma consulta padrão sem nenhum tipo de filtro, senão executamos outra instrução SQL mas passando o conteúdo de "$nome" como parâmetro para pesquisar por nome usando LIKE, assim não será necessário digitar o nome completo do contato, somente as iniciais do primeiro nome, por exemplo.

Outro detalhe bacana nesse código é que retornamos os resultados da consulta dentro de uma tabela HTML. Para isso criamos uma variável "$tabela", atribuimos o início da tabela e depois, usando um loop, atribuímos os resultados da consulta. Desse modo as informações já serão carregadas na página como uma tabela HTML.

Caso não seja encontrado nenhum registro no banco com o conteúdo da variável "$nome" é retornada uma mensagem sem formatação nenhuma.

Foram cadastrados alguns contatos previamente no banco, segue abaixo os registros obtidos através de consulta utilizando MySQL Query Browser.

	NOME	FONE	CELULAR	EMAIL
1	WILLIAM	2222	4444	willfl@ig.com.br
2	KATIA	223242	3223242	wewr@ig.com.br
3	JUAN	32432	676767	wfdd@ig.com.br
5	JOÃO ALVES	6665	464646	odfdofd@ig.com.br
6	MARIA DE ALMEIDA	6665	464646	dssd@ig.com.br
7	MARIO SOUZA	54894	4515	asa@ig.com.nbt
8	LUIS	43434	1211	saswe@ig.com.br
9	CARLA	525	99669	qwqw@ig.com.br
10	CASSIO	89898	45454	wewe@rrrrrr
12	DANIEL	526252	51414	fdf@hotmail.com

Figura 3: Todos os contatos gravados no banco

Abaixo temos o resultado da pesquisa no navegador. Para exibir esse resultado não foi necessário digitar nada no campo, pois codificamos no arquivo contato.php que caso não seja passado nenhum valor será carregada uma consulta com todos os registros do banco.

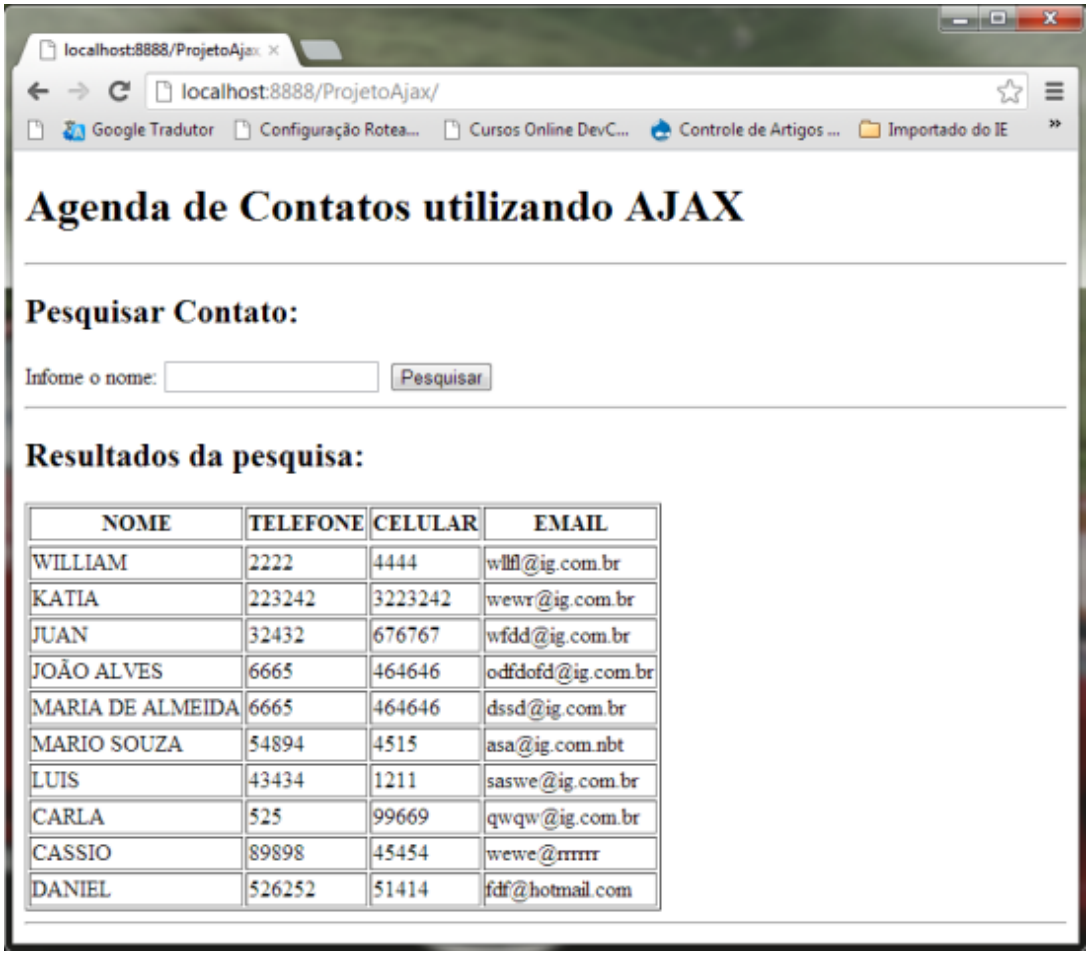

Figura 4: Tabela com todos os contatos

Agora vamos pesquisar todos os contatos que começam com a letra "C":

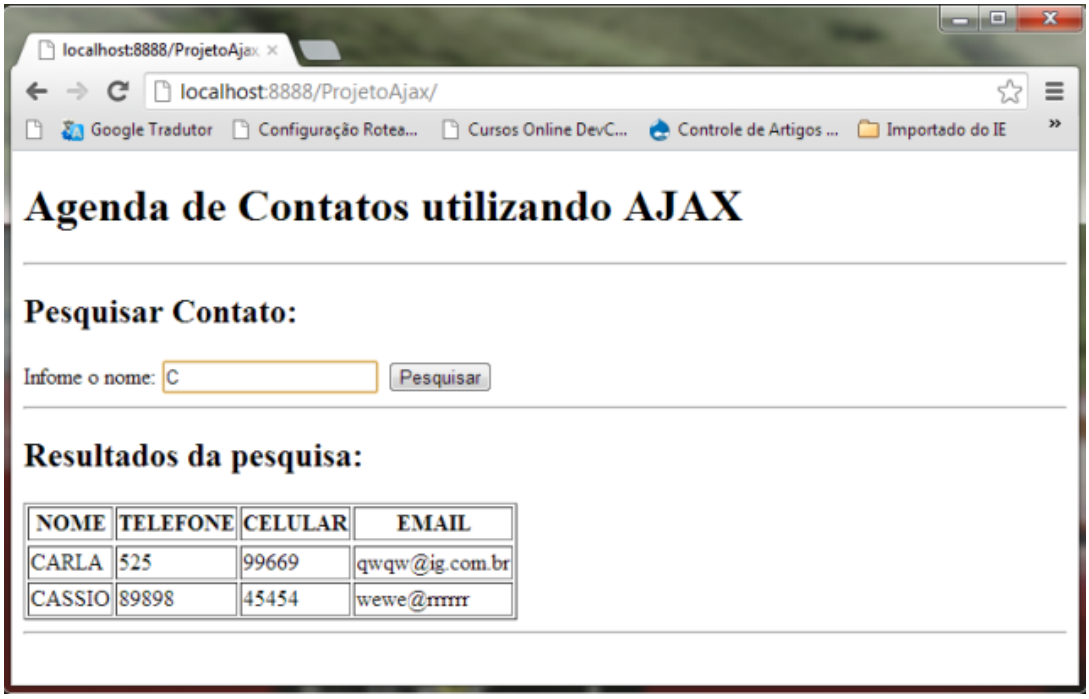

Figura 5: Contatos que começam com a letra C

Como último teste, pesquisamos usando a letra "N". Como já foi mostrado acima, não existem contatos começando com essa letra, então será emitida uma mensagem avisando o usuário.

Figura 6: Mensagem para o usuário

Bom pessoal, foi demonstrado nesse artigo como pode ser simples trabalhar com AJAX e PHP, sendo que a função "CriaRequest()" descrita no arquivo ajax.js é quase que padrão, pois ali conseguimos prever a maioria das situações que podem ocorrer em quase todos os browsers atualmente usados.

Mas é sempre bom ressaltar que devemos usar AJAX com cautela, onde podemos citar algumas situações. Numa aplicação Ajax transfere-se muito do processamento do servidor para o cliente, essa mudança tem custos porque estaremos a delegar ao cliente a responsabilidade de realizar determinadas operações para as quais não estaria inicialmente destinado. Podemos sobrecarregar o cliente caso não se tomem as devidas precauções durante a fase de desenvolvimento deste tipo de aplicação Web.

Mas no geral sabendo usar o AJAX ele pode trazer muito mais vantagens que desvantagens.

Até o próximo artigo!

William