Instituto Federal Goiano - Campus Ceres
Bacharelado em Sistemas de Informação
Prof. Me. Ronneesley Moura Teles

Daniel Moreira Cardoso
Davi Ildeu de Faria
Fernando Maciel
Igor Justino Rodrigues
José Gilvan Jacinto Junior
Paulo Henrique Rodrigues Araujo

O que é AJAX? Como se programa?

Sumário

O que é AJAX? Como se programa?

1 Introdução

O Ajax é um conjunto de tecnologias de desenvolvimento web que quando reunidas permite o desenvolvimento de uma aplicação melhor elaborada, intuitiva, e de melhor desempenho. Ajax Asynchronous JavaScript and XML(JavaScript assíncrono e XML) foi gerado por Jesse James Garret e o nome foi escolhido justamente para facilitar a explicação do modelo de interação diferente entre o navegador e o servidor web.

Asynchronous(assíncrono) é um tipo de comunicação que não ocorre exatamente ao mesmo tempo, não-simultânea. Ajax é o ato de carregar e renderizar uma página usando scripts rodando pelo lado do cliente carregando os dados em plano de fundo sem que haja a necessidade de atualização da página. Ajax não é uma tecnologia, mas sim um conjunto de tecnologias.

2 Como o AJAX trabalha?

Enquanto uma aplicação web convencional busca as informações no servidor e retorna ao cliente o AJAX não necessita dessa ação agilizando o processo de carregamento de dados, com o AJAX toda a lógica e processamento é passado para o usuário, quando o usuário faz uma requisição quem busca e trás essas informações é o JavaScript de forma assíncrona evitando o famoso "reload" na tela. O tratamento dos dados, seu formato e exibição fica toda por conta do script que foi carregado inicialmente quando se acessou a página.

O processo inicial de carregamento é mais lento que de uma aplicação comum, pois muitas informações são pré-carregadas. Mas depois, somente os dados são carregados, tornando assim o site mais rápido.

3 Aplicação

A partir de um CRUD feito em PHP, para realizar as interações, realizamos um exemplo para conectar ao banco de dados e inserir sem a necessidade de atualização da página

```
1  CREATE DATABASE  IF NOT EXISTS `ajax`;
2
3  USE `ajax`;
4
5  DROP TABLE IF EXISTS `usuario`;
6
7  CREATE TABLE `usuario` (
8    `id` int(11) NOT NULL AUTO_INCREMENT,
9    `nome` varchar(45) DEFAULT NULL,
10   `sexo` varchar(45) DEFAULT NULL,
11   PRIMARY KEY (`id`)
12 ) ENGINE=InnoDB AUTO_INCREMENT=0 DEFAULT CHARSET=latin1;
```

recursos/codigo/banco.sql

```php
<?php
  switch(isset($_GET['opcao']) ? $_GET['opcao'] : "apagar"){
    case 'inserir':
      inserir();
    break;
    case 'apagar':
      apagar();
    break;
  }

  function conectar(){
    $con = new mysqli("localhost", "root", "123456", "ajax");
    if($con->connect_errno){
      die("Nao foi possivel conectar, n do erro: " . $con->
    connect_errno . ", mensagem: " . $con->connect_error);
    }

    return $con;
  }

  function apagar(){
    $conexao = conectar();
    if(isset($_GET['id'])){
      $id = $_GET['id'];

      $conexao->query("delete from usuario where id = $id");
      $conexao->close();
    } else {

      $conexao->query("delete from usuario where id > 0");
      $conexao->close();
    }
  }

  function inserir(){
    if(isset($_POST['nome'])){
      $nome = $_POST['nome'];
    }
    if(isset($_POST['sexo'])){
      $sexo = $_POST['sexo'];
    }
    $conexao = conectar();

    $conexao->query("insert into usuario (nome, sexo) values ('$nome','
    $sexo')");

    $conexao->close();
  }
?>
```

recursos/codigo/CRUD.php

E pela página HTML enviamos os dados:

```html
<!DOCTYPE html>
<html>
  <head>
    <meta charset="UTF-8">
    <title>AJAX | Dados</title>
  </head>
  <body>
```

```
8     <form>
9       <label>Digite seu nome: </label>
10      <input type="text" id="nome" />
11      <label>Digite seu sexo: </label>
12      <input type="text" id="sexo" />
13      <input id="Inserir" type="button" style="cursor: pointer; text-
    decoration: underline" value="Inserir" />
14      <label id="status"><label>
15    </form>
16    <script type="text/javascript">
17      (function() {
18        var httpRequest;
19        document.getElementById("Inserir").onclick = function() {
20          var nome = document.getElementById("nome").value;
21          var sexo = document.getElementById("sexo").value;
22          makeRequest('CRUD.php?opcao=inserir', [nome, sexo]);
23        };
24
25        function makeRequest(url, data) {
26          if (window.XMLHttpRequest) { // Mozilla, Safari, ...
27            httpRequest = new XMLHttpRequest();
28          } else if (window.ActiveXObject) { // IE
29            try {
30              httpRequest = new ActiveXObject("Msxml2.XMLHTTP");
31            } catch (e) {
32              try {
33                httpRequest = new ActiveXObject("Microsoft.XMLHTTP");
34              } catch (e) {}
35            }
36          }
37
38          if (!httpRequest) {
39            alert('Desistindo :( Nao e possivel criar uma instancia
    XMLHTTP');
40            return false;
41          }
42
43          httpRequest.onreadystatechange = alertContents;
44          httpRequest.open('POST', url);
45          httpRequest.setRequestHeader('Content-Type', 'application/x-
    www-form-urlencoded');
46
47
48          httpRequest.send('nome=' + encodeURIComponent(data[0]) +'&
    sexo=' + encodeURIComponent(data[1]));
49        }
50
51        function alertContents() {
52          if (httpRequest.readyState === 4) {
53            if (httpRequest.status === 200) {
54              document.getElementById("status").innerHTML = "Registro
    Inserido com sucesso!";
55            } else {
56                alert('Houve um problema com o pedido.');
57            }
58          } else {
59            document.getElementById("status").innerHTML = "Carregando
    ...";           }
60        }
61      })();
```

```
    </script>
  </body>
</html>
```

recursos/codigo/requisicao_banco_dados.html

Já no próximo exemplo, informamos o nome e é retornado um alert com as horas, minutos e segundos atuais:

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="UTF-8">
    <title>AJAX | Dados</title>
  </head>
  <body>
    <label>Digite seu nome:
      <input type="text" id="nome" />
    </label>
    <span id="ajaxButton" style="cursor: pointer; text-decoration:
    underline">
       Fazer Pedido
    </span>
        <script type="text/javascript">
      (function() {
        var httpRequest;
        document.getElementById("ajaxButton").onclick = function() {
          var nome = document.getElementById("nome").value;
          makeRequest('dados.php', nome);
        };

        function makeRequest(url, nome) {
          if (window.XMLHttpRequest) { // Mozilla, Safari, ...
            httpRequest = new XMLHttpRequest();
          } else if (window.ActiveXObject) { // IE
            try {
              httpRequest = new ActiveXObject("Msxml2.XMLHTTP");
            } catch (e) {
              try {
                httpRequest = new ActiveXObject("Microsoft.XMLHTTP");
              } catch (e) {}
            }
          }

          if (!httpRequest) {
            alert('Desistindo :( Nao e possivel criar uma instancia
    XMLHTTP');
            return false;
          }

          httpRequest.onreadystatechange = alertContents;
          httpRequest.open('POST', url);
          httpRequest.setRequestHeader('Content-Type', 'application/x-
    www-form-urlencoded');


          httpRequest.send('nome=' + encodeURIComponent(nome));
        }

        function alertContents() {
          if (httpRequest.readyState === 4) {
```

```
            if (httpRequest.status === 200) {
              alert(httpRequest.responseText);
            } else {
                alert('Houve um problema com o pedido.');
            }
          }
        }
      })();
    </script>
  </body>
</html>
```

recursos/codigo/requisicao_dados.html

Alert em PHP:

```
<?php
  $name = (isset($_POST['nome'])) ? $_POST['nome'] : 'Nao ha nome';

  $mensagem = "Ola ";
  $mensagem .= $name . ", agora sao ";
  $mensagem .= date("H") . " horas " . date("i") . " minutos e " . date
    ("s") . " segundos ";

  echo $mensagem;
?>
```

recursos/codigo/dados.PHP

4 Manipulando a resposta do servidor

Lembre-se que quando você estava enviando a requisição, você forneceu o nome de uma função JavaScript que foi projetada para lidar com a resposta.

```
httpRequest.onreadystatechange = nomeDaFuncao;
```

recursos/codigo/funcao.html

Vamos ver o que essa função deve fazer. Primeiro, a função precisa checar o estado da requisição. Se o estado da requisição tem o valor igual a "4", significa que a resposta do servidor foi recebida por completo e está tudo OK para continuar o processo.

```
if (httpRequest.readyState === 4) {
// evento de sucesso
} else {
// Aguardando
}
```

recursos/codigo/funcao2.html

A lista completa dos valores readyState é a seguinte:

- 0 (não inicializado);
- 1 (carregando);
- 2 (carregado);
- 3 (interativo);

- 4 (completo).

*** readyState é algo como "estado de prontidão", mostra qual é o status do processo que está sendo executado e se está sendo executado.**

A próxima coisa a se checar é o código de resposta do servidor HTTP. No exemplo a seguir, nós tratamos do retorno bem sucedido ou mal sucedido da requisição HTTP do AJAX, verificando se o código de resposta for 200.

```
1 if (httpRequest.status === 200) {
2 // OK!
3 } else {
4 // Erro
5 }
```

recursos/codigo/funcao3.html

Os principais Códigos de Requisição são:

- 200: "OK"
- 403: "Proibido"
- 404: Página Não Encontrada"

Agora, após você ter checado o estado da requisição e o código de status do HTTP da resposta, caberá a você fazer o que quiser com os dados que o servidor lhe enviou. Você tem duas opções para acessar esses dados:

- httpRequest.responseText – retorna a resposta do servidor como uma string de texto
- httpRequest.responseXML – retorna a resposta do servidor como um objeto XMLDocument no qual você poderá percorrer usando as funções DOM do JavaScript.

Note que os passos acima são válidos somente se você usou uma solicitação assíncrona (terceiro parâmetro de open() foi definido como true). Se você usou um pedido síncrono você não precisa especificar uma função, você pode acessar o retorno de dados pelo servidor diretamente depois de chamar send(), porque o script irá parar e esperar pela resposta do servidor.

5 Vantagens e Desvantagens

5.1 Vantagens

- Uma das principais vantagens do uso do Ajax é o desempenho do site. O Ajax permite buscar dados no servidor e mostrar em apenas uma parte da página assim não se fazendo necessário recarregar toda a página;
- Outra vantagem bastante interessante sobre o Ajax é a compatibilidade. Não importa se a aplicação é em Java, PHP, Python ou outra linguagem de programação. (Sem o uso de frameworks);
- Uma vantagem para o usuário é a interface. Normalmente o uso do Ajax proporciona uma melhor navegabilidade e facilidade de uso para o usuário.

5.2 Desvantagens

- Há uma boa compatibilidade com o Ajax e as linguagens de programação, porém quando usamos frameworks para implementar o Ajax podemos perder essa compatibilidade pois as vezes um framework pode interferir no funcionamento do outro. Como por exemplo a função $ no Jquery e Prototype.

6 Conclusão

Com o desenvolvimento da pesquisa e implementação dos códigos Ajax foi possível perceber a real necessidade da utilização de tecnologias como esta em um mercado cada vez mais exigente e competitivo como é o nosso. Ainda contribuindo para em ganhos em desempenho e facilidade de uso (para cliente e desenvolvedor).

7 Referências Bibliográficas

[**1**] https://imasters.com.br/artigo/10224/ajax/vantagens-e-desvantagens-do-uso-de-ajax-aspectos-praticos?trace=1519021197&source=single.

[**2**] https://developer.mozilla.org/pt-BR/docs/AJAX/Getting_Started

[**3**] http://www.devfuria.com.br/php/como-funcionam-os-metodos-get-e-post/

[**4**] http://www.devfuria.com.br/php/como-funcionam-os-metodos-get-e-post/