



# Aplicações Web com AJAX

v. 2.1 - Setembro/2007



## Objetivo:

Esta apresentação tem por objetivo apresentar a tecnologia AJAX, realizando uma breve explanação sobre os conceitos envolvidos em sua utilização.

Como case, trataremos de um uso comum mas extremamente útil desta tecnologia: a interação entre elementos de formulário, mais precisamente entre uma caixa de seleção de estados e outra de cidades.



## Preparativos:

Primeiramente criaremos um formulário HTML com duas caixas de seleção: uma de estados e outra de cidades. Depois criaremos um gatilho que irá disparar nossa função AJAX.

Para isto utilizaremos o evento **onChange** da caixa de seleção de estados:

```
<select name='estados' onChange='buscaCidades();'>
```

Que, por sua vez agirá sobre a caixa de seleção de cidades, que originalmente terá apenas uma opção:

```
<select name='cidades'>  
  <option value='x' selected>Aguardando seleção de estado</option>  
</select>
```



## 1. O que é AJAX:

# **A**synchronous **J**avaScript **A**nd **X**ML

ou “JavaScript Assíncrono e XML”. AJAX não é uma nova linguagem, mas apenas a utilização de duas características mais recentes da linguagem JavaScript:

- 1) A possibilidade de trabalhar com requisições HTTP
- 2) A habilidade de trabalhar com documentos XML

combinadas com uma terceira que já é velha conhecida dos desenvolvedores: a manipulação direta do DOM HTML.

## Requisições HTTP

São as requisições feitas para um servidor web.

Um exemplo simples de uma requisição HTTP é quando você está navegando em um website. Cada clique em um hiperlink dispara uma requisição para o servidor:

Usuário / Cliente



**Requisição HTTP**

Provedor / Servidor



Retorno (HTML)





## Documentos XML

São documentos criados na linguagem de marcação XML, que permite a descrição e armazenamento de informações.

Vejam um exemplo simplificado do arquivo XML que utilizaremos:

```
<?xml version="1.0" encoding="iso-8859-1"?>
<idades>
  <idade>
    <codigo>1</codigo>
    <nome>Porto Alegre</nome>
  </idade>
  <idade>
    <codigo>2</codigo>
    <nome>Canoas</nome>
  </idade>
</idades>
```



## 2. Trabalhando com a requisição

Quando trabalhamos com uma requisição HTTP devemos ter em mente os seguintes passos:

- 1 – Criação da requisição;**
- 2 – Definição de uma função que tratará a resposta desta requisição;**
- 3 – Abertura da requisição;**
- 4 – Definição do tipo de conteúdo da requisição;**
- 5 – Envio da requisição.**



## Criando uma requisição HTTP

Uma requisição HTTP é um objeto que é criado de acordo com o navegador onde a aplicação irá rodar:



```
http_request = new XMLHttpRequest("Msxml2.XMLHTTP"); // IE 6+  
http_request = new XMLHttpRequest("Microsoft.XMLHTTP"); // IE 5.5-
```



```
http_request = new XMLHttpRequest();
```



## Definindo a função de tratamento da resposta

Neste passo definimos uma função que será responsável por tratar a resposta de nossa requisição, ou seja, esta função será o passo final de nossa aplicação.

Para que esta função possa desempenhar seu papel dois fatores precisam ser checados:

1. O Estado da requisição (**readyState**): Uma requisição HTTP passa pelos seguintes estados:

- 0 – Não inicializada
- 1 – Carregando
- 2 – Carregada
- 3 – Processando
- 4 – Pronta

2. O código de Status HTTP retornado pelo servidor (**status**): Caso este código seja 200 (HTTP OK) tudo correu bem. Qualquer outro código pode significar um erro retornado pelo servidor.



## Definindo a função de tratamento da resposta (cont.)

A definição da função é feita através do atributo **onreadystatechange** do objeto da requisição. Veremos aqui como definir uma função “*on-the-fly*”:

```
http_request.onreadystatechange = function ()
    {
        if (http_request.readyState == 4) {
            if (http_request.status == 200) {
                // Código
            } else {
                alert (“O servidor retornou um erro.”);
            }
        }
    }
}
```



## Abrindo a requisição

A abertura da requisição é feita através do método **open**. Este método aceita 3 parâmetros:

- 1 – A forma de envio dos dados: GET ou POST
- 2 – A URL do script server-side
- 3 – Se esta requisição será assíncrona

Vejam os exemplos:

```
http_request.open('GET', 'http://localhost/script.php?iduf=' +  
document.forms[0].estados.value, true);
```



## Definindo o tipo de conteúdo da requisição

Utilizando o método **setRequestHeader**, definiremos o tipo de conteúdo que está sendo enviado por nossa requisição.

Como utilizaremos um formulário para enviá-la, precisamos definir um tipo especial de conteúdo:

```
http_request.setRequestHeader('Content-Type',  
'application/x-www-form-urlencoded');
```



## Enviando a requisição

Para finalmente enviarmos a requisição usamos o método **send**.

Como estamos trabalhando com o método de envio GET, não enviaremos dados, mas apenas o valor especial **null**.

Os dados, quando utilizamos este método, são enviados através de um *query string* na própria URL, como vimos no método **open**.

```
http_request.send(null);
```



### 3. Retornando XML

Veremos agora como gerar um retorno em XML através de um script server-side programado em PHP.

Esta é a lógica que seguiremos:

- 1 – Criamos uma variável que guardará o código XML
- 2 – Testamos se o 'id' recebido é o que esperamos
- 3 – Em caso positivo alimentamos nossa variável com mais XML
- 4 – Definimos novamente um tipo de conteúdo, desta vez referente ao retorno de nosso script.
- 5 – Damos saída direta no código XML, que será posteriormente capturado por nossa função de tratamento em JavaScript.



## Retornando XML (cont.)

Aplicada em código, nossa lógica resultará no seguinte:

```
<?php
$retorno = '<?xml version="1.0" encoding="iso-8859-1"?>';
$retorno .= '<idades>';

if ($_GET['iduf'] == 1) {
    $retorno .= <<<FIM
    <idade>
        <codigo>1</codigo>
        <nome>Porto Alegre</nome>
    </idade>
    <idade>
        <codigo>2</codigo>
        <nome>Canoas</nome>
    </idade>
    FIM;
}

$retorno .= '</idades>';
header('Content-type: application/xml; charset=iso-8859-1');
echo $retorno;
?>
```



## 4. O Grand Finale: JavaScript + XML + DOM

A primeira coisa que faremos é “zerar” a combobox de cidades, de forma à evitar que ela seja repetidamente populada. Feito isto, trataremos o XML retornado.

Isto será feito acessando-se o atributo **responseXML** de nossa requisição. Utilizando este atributo realizaremos uma manipulação do DOM de forma à realizar quatro etapas:

- 1 – Utilizando o método DOM **getElementsByTagName** iremos capturar cada cidade presente no retorno XML.
- 2 – Para cada uma destas cidades capturaremos os elementos 'codigo' e 'nome', utilizando o mesmo método DOM.
- 3 – Através do método DOM **createElement**, criaremos dinamicamente opções (options de formulário).
- 4 – Adicionaremos, com o método **add**, estas opções ao combobox de cidades.



## O Grand Finale: JavaScript + XML + DOM (cont.)

O código a seguir deve ser colocado no lugar indicado em nossa função de tratamento de retorno:

```
document.forms[0].cidades.options.length = 1;
retorno = http_request.responseXML;
cidades = retorno.getElementsByTagName('cidade');

for (x = 0; x < cidades.length; x++) {
    codigo = cidades[x].getElementsByTagName('codigo');
    descricao = cidades[x].getElementsByTagName('nome');

    novaOp = document.createElement('option');

    novaOp.value = codigo[0].firstChild.nodeValue;
    novaOp.text = descricao[0].firstChild.nodeValue;

    document.forms[0].cidades.options.add(novaOp);
}
```



## Como fica o código HTML / JavaScript Final:

```
<html>
  <head>
    <script type="text/javascript">
      var http_request;

      function buscaCidades()
      {
        http_request = new XMLHttpRequest();
        http_request.onreadystatechange = function ()
        {
          if (http_request.readyState == 4) {
            if (http_request.status == 200) {
              document.forms[0].cidades.options.length = 1;
              retorno = http_request.responseXML;
              cidades = retorno.getElementsByTagName('cidade');
              for (x = 0; x < cidades.length; x++) {
                codigo = cidades[x].getElementsByTagName('codigo');
                descricao = cidades[x].getElementsByTagName('nome');

                novaOp = document.createElement('option');

                novaOp.value = codigo[0].firstChild.nodeValue;
                novaOp.text = descricao[0].firstChild.nodeValue;

                document.forms[0].cidades.options.add(novaOp);
              }
            }
          }
        }
      }
    </script>
  </head>
</html>
```



## Como fica o código HTML / JavaScript Final (cont.)

```
        } else {
            alert("Possível erro na requisição");
        }
    }
}

    http_request.open('GET', 'http://localhost/script.php?iduf=' +
document.forms[0].estados.value, true);
    http_request.setRequestHeader('Content-Type',
'application/x-www-form-urlencoded');
    http_request.send(null);
}
</script>
</head>
<body>
    <form>
        Estado:
        <select name='estados' onChange='buscaCidades();'>
            <option value="0">Selecione:</option>
            <option value="1">RS</option>
        </select><br />
        Cidade:
        <select name="cidades">
            <option value='x' selected>Aguardando seleção de estado</option>
        </select>
    </form>
```



## Como fica o código HTML / JavaScript Final (cont.)

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



## Sobre o autor:

Er Galvão Abbott trabalha há mais de dez anos com programação de websites e sistemas corporativos com interface web.

Autodidata, teve seu primeiro contato com a linguagem HTML em 1995, quando a internet estreava no Brasil.

Além de lecionar em cursos, escrever artigos e ministrar palestras, tem se dedicado ao desenvolvimento de aplicações web, tendo nas linguagens PHP, Perl e JavaScript suas principais paixões.

É o fundador e líder do UG PHP RS, além de trabalhar como consultor e desenvolvedor para uma empresa norte-americana da área de segurança.



## Contatos:

Contatos com o autor:

Web:

<http://www.galvao.eti.br>

<http://blog.galvao.eti.br/>

<http://www.phprs.com.br>



E-mail:

[galvao@galvao.eti.br](mailto:galvao@galvao.eti.br)

IM:

[ergalvao@hotmail.com](mailto:ergalvao@hotmail.com) (MSN)

[er.galvao@gmail.com](mailto:er.galvao@gmail.com) (GoogleTalk)

