

# O que é AJAX e como aplicá-la com PHP, parte 1

Por Alfred Reinold Baudisch

Nesse artigo, entenda o que é AJAX e como usá-la facilmente como o PHP através da classe CPAINT, com explicações por meio de vários exemplos.

O uso de AJAX está se proliferando cada vez mais pela internet, justamente por trazer mais interatividade e usabilidade para as aplicações – ou até mesmo, mais “glamour”. Junto disso, surgem inúmeras maneiras de implementar essa combinação (AJAX não é uma tecnologia, é um uso conjunto de tecnologias), algumas complicadas e com recursos demais, outras simplificadas, mas deficientes de recursos. A classe CPAINT é uma classe completa e permite incluir AJAX eficientemente em suas aplicações web. Nessa série de artigos, entenda o que é AJAX, as características da CPAINT, bem como uma explicações por meio de códigos exemplo.

## 1. INTRODUÇÃO

### 1.1 O que é AJAX?

Bom, não irei explicar detalhadamente o que se trata o AJAX. Existem vários artigos por aí que tratam do assunto. Mas, irei mostrar pelo menos a idéia base para que os que desconhecem do assunto possam entender o conceito principal.

**AJAX** - *Asynchronous JavaScript and XML* (JavaScript e XML Assíncronos) – é uma técnica usada para criar aplicações web mais interativas, usando uma combinação de tecnologias:

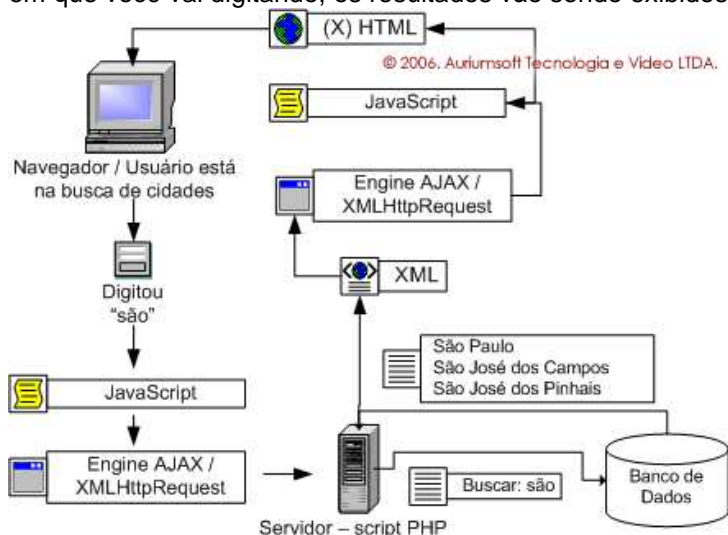
- (X)HTML e CSS (para a boa apresentação dos dados, ou pelo menos organizada)
- DOM + JavaScript (separação correta dos dados)
- XML (intercâmbio de informações)
- XMLHttpRequest (solicitações assíncronas de informações CLIENTE <-> SERVIDOR sem necessidade de dar refresh na página)
- Uma linguagem do lado de servidor (PHP, ASP.NET, etc) para enviar os dados (no caso o XML).

Você deve estar pensando: “Ok, que legal todos esses nomes e tecnologias. Eu sei que elas existem e já uso há um bom tempo JavaScript + HTML + CSS + XML + PHP, etc... Então o que esses caras estão querendo com esse tal de AJAX? Apenas mais um nome para gravarmos?”. Quase isso.

O conceito “tradicional” da web é de: o usuário digita um endereço, recebe uma página que contém formulário, imagens, links, etc... Vamos a um rápido exemplo: se um usuário quer efetuar uma busca, ele digita algo no formulário buscar, pressiona o botão “OK”, o navegador envia requisição para o servidor, que processa um script PHP, e devolve ao navegador uma nova página que é totalmente aberta “do zero” no navegador do usuário. Requisição concluída.

Agora imagina uma WEB funcionando como um software, em que ao mesmo tempo em que você vai fornecendo dados, as informações vão sendo exibidas e executadas de acordo com o que é fornecido, tudo em tempo real.

Exemplos: um formulário de cadastro numa loja virtual, em que você digita seu CEP e ao terminar de digitar, o campo Endereço, Cidade e Estado são preenchidos automaticamente sem qualquer tipo de refresh na página. Ou como no exemplo da “web tradicional”, um sistema de busca: ao mesmo tempo em que você vai digitando, os resultados vão sendo exibidos, SEM QUALQUER TIPO DE REFRESH.



Parece impossível, não? Ou sem tanto exagero, parece algo que tomará meses de programação! Que nada! O AJAX é exatamente isso – esse novo conceito de aplicações web.

## 1.2 Como funciona?

Vamos usar como exemplo uma calculadora. Imaginemos um formulário com uma caixa de texto para o usuário digitar um número e um botão para cada uma das operações básicas. Ao apertar um dos botões um método JavaScript vai obter o valor da caixa de texto, validar os dados e chamar sua engine AJAX, que por sua vez é uma encapsulação para o método XMLHttpRequest.

O método XMLHttpRequest envia os dados fornecidos (nesse caso, o número digitado) em tempo real para um script do lado do servidor definido por você (no nosso caso, um script PHP), que irá processá-los a gosto e retornar outros dados, seja em texto puro ou XML (o principal). Aos novos dados terem retornado, um novo método JavaScript é chamado para processá-los e atualizar a página sem dar refresh, normalmente inserindo em elementos XHTML DIV e SPAN.

Mas, você não precisa saber como funciona o XMLHttpRequest nem mesmo como implementá-lo. Para isso, basta usar um dos inúmeros frameworks de AJAX prontos que possuem na internet. O escolhido por mim para as aplicações que desenvolvo é o framework **CPAINT** (*Cross-Platform Asynchronous INterface Toolkit* - <http://cpaint.wiley14.com/>). Ele permite uma RÁPIDA - FÁCIL E EFICIENTE inserção de AJAX. E o melhor, não apenas para PHP, mas também ASP, .NET e CFM!

E é exatamente a abordagem AJAX com CPAINT e PHP que é abordada nesse e nos próximos artigos da série.

**OBS:** Faça o download da CPAINT aqui:

[http://sourceforge.net/project/showfiles.php?group\\_id=141041](http://sourceforge.net/project/showfiles.php?group_id=141041)

**1.3 Código exemplo** – a calculadora acima expressada. Para cada valor digitado, o usuário apertará uma operação e o resultado atualizará em tempo real (resposta do PHP).

**OBS 1:** Exemplo e código muito simples, mas servem apenas para ilustrar qual a lógica base do AJAX.

**OBS 2:** Código propositalmente compactado para não ocupar muito espaço do artigo.

**OBS 3:** Você deve estar se perguntando: mas uma calculadora dessas eu faço muito mais rapidamente apenas com JavaScript. Claro, eu concordo. Acontece que por meio de um modelo desses, você já pode imaginar centenas de aplicações mais interessantes, como busca no banco de dados, validação de formulário, etc, tudo através dessa comunicação em tempo real com o lado do servidor (PHP no caso).

### calculadora.html

```
=====
<!-- INCLUI A CLASSE CPAINT - AJAX -->
<script type="text/javascript" src="cpaint2.inc.compressed.js"></script>

<script type="text/javascript">
// Valor atual da nossa conta
var Total = 0;

// Cria objeto CPAINT
var cp = new cpaint();
cp.set_transfer_mode('POST');
cp.set_response_type('TEXT');

//
// Funções da calculadora
//
function Adicionar() { chamaAjax('Adicionar'); }
function Subtrair() { chamaAjax('Subtrair'); }
function Multiplicar() { chamaAjax('Multiplicar'); }
function Dividir() { chamaAjax('Dividir'); }
function CE() { chamaAjax('CE'); }

//
// Função que faz a comunicação JavaScript X PHP
//
function chamaAjax(Operacao)
```

```

{
    // Obtém valor digitado
    valor = document.getElementById('valor').value;
    // Limpa campo
    document.getElementById('valor').value = '';
    // Retorna foco para o campo
    document.forms[0].valor.focus();

    // Só chama se foi preenchido algo, ou se for a operação de limpeza
    // pode chamar em qualquer caso.
    if(valor != '' || Operacao == 'CE')
    {
        // PRINCIPAL MÉTODO (call) = Chama o PHP e obtém o retorno
        cp.call('ajaxCalculadora.php', Operacao, imprimirResultado, Total, valor);
    }
}

//
// Função usada pela CPAINT para imprimir o resultado dado pelo PHP
//
function imprimirResultado(Retorno)
{
    Total = Retorno;
    document.getElementById('resultado').innerHTML = Retorno;
}
</script>

<div id="corpo">
<form>
    <div>Valor: <input type="text" id="valor" name="valor" size="4" /></div>

    <div>Operação:
        <input type="button" value=" + " size="3" onclick="Adicionar();" />
        <input type="button" value=" - " size="3" onclick="Subtrair();" />
        <input type="button" value=" x " size="3" onclick="Multiplicar();" />
        <input type="button" value=" / " size="3" onclick="Dividir();" />
        <input type="button" value=" CE " size="3" onclick="CE();" />
    </div>

    <div>Resultado: <span id="resultado"><script>imprimirResultado(Total);</script></
span></div>
</form>
</div>

```

## ajaxCalculadora.php

```

=====
<?php

// Importa a Cpaint
require 'cpaint2.inc.php';

// Instancia Cpaint
$CPaint = new cpaint();

// Registra funções
$CPaint->register(array('Calculadora', 'Adicionar'));
$CPaint->register(array('Calculadora', 'Subtrair'));
$CPaint->register(array('Calculadora', 'Dividir'));
$CPaint->register(array('Calculadora', 'Multiplicar'));
$CPaint->register(array('Calculadora', 'CE'));

// Inicia Cpaint e retorno de dados
$CPaint->start('ISO-8859-1');
$CPaint->return_data();

class Calculadora
{
    function Adicionar($Total, $Valor)
    {
        $Total += $Valor;
        Calculadora::EnviaValor($Total);
    }

    function Subtrair($Total, $Valor)

```

```

    {
        $Total -= $Valor;
        Calculadora::EnviaValor($Total);
    }

function Dividir($Total, $Valor)
{
    $Total /= $Valor;
    Calculadora::EnviaValor($Total);
}

function Multiplicar($Total, $Valor)
{
    $Total *= $Valor;
    Calculadora::EnviaValor($Total);
}

function CE() { Calculadora::EnviaValor(0); }

/**
 * Essa função retorna um valor para o CPAINT que por sua
 * vez retornará para a página
 */
function EnviaValor($Total)
{
    global $CPaint; $CPaint->set_data($Total);
}
}

?>

```

### Explicação do Código

1) calculadora.html é a página que você abrirá no navegador e ajaxCalculadora.php é o script que executa as operações e retorna valor para a calculadora.html

2) A primeira tarefa é incluir a CPAINT nas páginas que a utilizam, e isso deve ocorrer em duas frentes: no lado do usuário (via JavaScript) e no lado do servidor (via PHP), para tanto ocorre a chamada:

```
<script type="text/javascript" src="cpaint2.inc.compressed.js"></script>
```

No script HTML e:

```
require 'cpaint2.inc.php';
```

No script PHP.

3) Agora focando no calculadora.html. É iniciada a variável Total, que armazenará o resultado do cálculo em andamento, e essas 3 linhas iniciam a CPAINT:

```
var cp = new cpaint();
cp.set_transfer_mode('POST');
cp.set_response_type('TEXT');
```

- Recomendo sempre usar POST como meio de transferência (set\_transfer\_mode), caso seja GET, as URLs de requisições aparecerão no histórico do navegador, além de ser aberto a falhas de segurança, dependendo dos dados a serem transmitidos.

- Quanto ao tipo de resposta (set\_response\_type), a CPAINT permite TEXT, OBJECT e XML. No modo TEXT você retorna exatamente o conteúdo novo que se inserirá na página. Já os demais, permite serem separados, para um melhor tratamento, mas ambos serão tratados mais tarde.

- Note que cp é apenas o nome da variável e você pode usar qualquer outro, exemplo: obj, cpaint, ajax, etc.

4) Em Funções da Calculadora são apenas 5 funções atalho, que chamam outra, a chamaAjax(), passando como argumento o nome da operação a ser executada.

5) A função chamaAjax() é a mais importante no calculadora.html. Essa função que faz a chamada do script PHP, bem como faz o retorno do resultado. No primeiro momento, ela obtém o valor digitado pelo usuário na caixa valor. E assim, validado o valor, ela faz chamada ao método call do seu objeto cpaint (cp nesse caso).

```
cp.call('ajaxCalculadora.php', Operacao, imprimirResultado, Total, valor);
```

O método call() necessita de no mínimo 3 argumentos:

a) [string] Script a ser chamado, esse script deve conter uma implementação e retorno da CPAINT

(`ajaxCalculadora.php` no nosso caso).

- b) [string] Nome da função remota a ser executada. No nosso caso eu passei a variável JavaScript `Operacao`, que contém uma string contendo o nome da operação a ser executada. Como você vai ver mais para frente, essa operação nada mais é que uma função definida em `ajaxCalculadora.php`.
- c) [callback] Nome da função JavaScript que processará o resultado enviado pelo PHP. No nosso caso o nome da função é `imprimirResultado`.
- d) Os demais argumentos são opcionais, eles são os valores que você quer enviar para o PHP junto com a chamada a função. Eu estou enviando o valor `Total` do cálculo, bem como o novo `valor` digitado pelo usuário. Você pode enviar infinitos valores, bastando os separar por vírgulas.

Apesar dos próprios argumentos explicarem o funcionamento da função `call`, eu irei repetir: *Ela chama o script dado, enviando os argumentos, e executa a função passada. Após o script retornar, ela chama a função nomeada para tratamento do retorno, passando como argumento o resultado.*

**6)** A próxima função é a `imprimirResultado`. Como viram acima, ela é chamada pela `call` após o PHP retornar o resultado. Você sempre deve permitir um argumento para a função, pois esse argumento que contém o resultado da operação. Tudo que eu faço é simplesmente preencher o `span` resultado com o valor que foi retornado.

Esse valor poderia ter HTML / CSS / XHTML dentro e isso iria ser inserido no `span` resultado sem qualquer problemas. Você está livre para retornar QUALQUER tipo de texto.

Faça o seguinte, mude depois a `imprimirResultado` para:

```
function imprimirResultado(Retorno)
{
    Total = Retorno;
    alert(Retorno);
}
```

Ao invés do resultado ser colocado no HTML, será exibido um alerta para você. Apenas para que você conheça que está livre para tudo. E quando eu falo TUDO é TUDO mesmo. Claro que somente retornando valores separados via XML, mas isso será na segunda parte dessa série de artigos.

**7)** O corpo é apenas `div`s com os campos: `valor` (onde você digitará os valores para serem colocados no cálculo), botões para as operações, que fazem chamada às funções referentes e o lugar para exibir o resultado.

**8)** Agora vamos para a segunda parte principal: o script PHP. Conforme falado acima, primeiro importa a classe `CPAINT`. Em segundo faz a sua instância.

**9)** Para cada função que você queira usar remotamente, você deve registrá-la na `CPAINT`, caso contrário ela não saberá o que você está chamando. O interessante que você possui 3 métodos de registro:

- Apenas função
- Classe e função
- Objeto e função

Se o seu sistema for estruturado em funções, sem problemas. Se for um sistema modular e orientado a objetos, também sem problemas. No caso do exemplo, eu usei a abordagem Classe e Função:

```
// Registra funções
$CPaint->register(array('Calculadora', 'Adicionar'));
$CPaint->register(array('Calculadora', 'Subtrair'));
$CPaint->register(array('Calculadora', 'Dividir'));
$CPaint->register(array('Calculadora', 'Multiplicar'));
$CPaint->register(array('Calculadora', 'CE'));
```

Ou seja, no registro deve ser passado um array, onde o primeiro elemento é o nome da classe e o segundo é o nome da função. Note que o nome da função é exatamente o mesmo das strings que você chama em `calculadora.html`. TEM de ser o mesmo, caso contrário o que a `CPAINT` chamará?

Para registrar uma função, apenas passe o nome dela como uma string:

```
$CPaint->register('Adicionar'); (isso funciona no caso de existir uma função solta no código chamada Adicionar, por exemplo).
```

E para registrar um objeto, passe da mesma maneira via array, mas o primeiro elemento deve ser referência ao objeto, exemplo:

```
$objCalculadora = new Calculadora;
$CPaint->register(array(&$objCalculadora, 'Multiplicar'));
```

OBS: A classe Calculadora e essas funções estão logo abaixo dessas linhas de registro ali no código.

**10)** Em seguida a CPAINT é autorizada a iniciar e retornar data. Na verdade o método start() faz chamada ao método solicitado via JavaScript e o return\_data() retorna para o JavaScript o que o PHP retornou.

```
$CPaint->start('ISO-8859-1');  
$CPaint->return_data();
```

**11)** Para que o PHP retorne algo, você deve enviar isso para a CPAINT via o método set\_data(). É o valor passado a set\_data() que será retornado para sua página, no nosso caso a função imprimirResultado que receberá esse valor.

Na classe Calculadora do exemplo todas as funções de operação chamam outra função chamada EnviarValor:

```
function EnviaValor($Total)  
{  
    global $CPaint; $CPaint->set_data($Total);  
}
```

O que ela faz é simplesmente chamar o objeto CPAINT criado e chamar a set\_data() enviando o resultado que quero que apareça na página.

Você pode simplesmente chamar isso diretamente na sua função, exemplo:

```
function Buscar($Nome)  
{  
    global $CPaint;  
    if($Nome == 'A')  
        $CPaint->set_data('Antonio');  
    else  
        $CPaint->set_data('Sem nome');  
}
```

#### 1.4 Finalizando

Teste esses scripts em seu computador (não se esqueça de que deve ser executado no seu localhost!)! Faça mudanças, tente outros tipos de funcionalidades, exemplo: o usuário digita uma frase e o php retorna ela em maiúsculas.

A próxima parte do artigo abordará o uso do CPAINT retornando dados mais completos, bem como XML, para aplicações reais no dia-a-dia, como por exemplo validação de formulário.

Até o próximo!  
Alfred Reinold Baudisch

#### Jornada Imperial

[www.auriumsoft.com.br/blog/](http://www.auriumsoft.com.br/blog/)

#### THE Mobile Developer

<http://mobiledeveloper.auriumsoft.com>

#### O Desenvolvedor PHP

<http://desenvolvedorphp.auriumsoft.com>

#### Referências:

**Wikipedia.** Ajax (Programming) - <http://en.wikipedia.org/wiki/AJAX>

**Adaptive Path.** Ajax: A New Approach to Web Applications - <http://www.adaptivepath.com/publications/essays/archives/000385.php>

**Documentação CPAINT** - <http://cpaint.wiley14.com/doc/>