Ajax para quem só ouviu falar

Introdução

Sempre que um assunto entra em voga, aparece um monte de gente perguntando as mesmas perguntas de sempre. E, quando eu já estudei alguma coisa sobre o assunto, eu sempre tento, inutilmente, é verdade, escrever um artigo que elimine as dúvidas iniciais e mostre o "camino das pedras" para quem só ouviu falar naquilo e quer saber o que é, pra que serve, e ter uma idéia de como se faz.

Este artigo é uma tentativa dessas. O assunto é Ajax. Ajax é a sigla para "Asynchronous Javascript and XML". Se você entende inglês, pode ler uma excelente explicação do assunto no artigo de Jesse James Garrett: Ajax: A New Approach to Web Applications. Não é uma coisa nova, e eu já falei um pouco sobre isso no meu blog. A idéia é utilizar Javascript para transformar suas páginas em aplicações, de modo que não precise recarregar a tela cada vez que o usuário clicar em alguma coisa. Você pode recarregar apenas a área que precisa ser alterada pela ação realizada.

Há alguns excelentes exemplos do que estou falando. Você pode dar uma olhada neles para ter idéia do que isso é capaz de fazer:

- Backbase - Esse serve para impressionar os amigos. E, eu acho, só para isso.
- Google Suggest - Digite devagar, você vai ver...
- Google Maps - Fantástico, fabuloso.
- Gmail - O melhor webmail que eu já vi.
- Start.com - Sim, é da Microsoft, por quê? Muito bom também.

Claro, não quero te ensinar neste artigo a construir um GMail ou coisa parecida. Aliás, nem todo mundo tem a oportunidade de trabalhar em projetos de grandes aplicações client-side. Se você resolver estudar o assunto mais a fundo, eu posso garantir, vale a pena! É bastante divertido trabalhar com isso.

Acabo de entregar uma aplicação de tamanho razoável, o novo Prodo, o CMS leve da Atípico, a empresa onde eu orgulhosamente passo os dias me divertindo. Infelizmente, ele não está disponível para test-drive. Mas há outra aplicação, um pouco menor, mas de tamanho já considerável, que eu fiz há algum tempo e você pode ver: o editor do blog do Tableless. Há uma senha de demonstração para isso, usuário "ajax", senha "ajax". Pode mexer à vontade, o conteúdo não é o real (é uma cópia do conteúdo real atualizada a cada 30 minutos.)

Objetivos

Nosso objetivo com esse artigo é contruir uma aplicação Ajax muito simples, que te faça entender os conceitos básicos por trás disso tudo. A primeira coisa importante a ser dita aqui é a respeito de acessibilidade. É possível, e bastante desejável, construir aplicações Ajax que tenham algumas características:

- Funcionar sem Javascript
- Funcionar sem CSS
- Ao clicar, mudar a url na barra de endereços e o título da página
- Manter funcionais os botões de avançar, voltar e recarregar
- Permitam salvar links, adicionar a bookmarks, etc.

O Backbase e o Gmail, por exemplo, têm as três últimas características. O editor do Tableless tem as duas primeiras e, dependendo do seu navegador, a quarta. Cada uma delas envolve mais ou menos trabalho para implementar, dependendo da complexidade de sua aplicação.

As cinco características são desejáveis e, claro, a decisão de implementá-las é sua, considerando o tipo de aplicação que está desenvolvendo. E sempre há a possibilidade de, como o GMail, oferecer uma versão mais simples de sua aplicação para navegadores não suportados.

Nosso objetivo neste artigo é construir um aplicativo Ajax simples, completamente funcional em navegadores não compatíveis. Para isso, trabalhamos o Ajax como última camada, a ser aplicada ao final de todo o restante. Nessa abordagem, que eu tenho usado bastante, seu código é construído de acordo com as seguintes camadas:

1. Conteúdo HTML
2. Programação Server-Side (fazer funcionar)
3. CSS (esta etapa e a anterior podem mudar de ordem)
4. Javascript e Ajax

Fazendo assim, ao terminar a camada 2, você tem uma aplicação completamente funcional, em qualquer navegador, até no Lynx. A camada 3 melhora (muito) a experiência para quem tem navegadores mais atuais. Sim, você já ouviu esse papo todo, é por isso que esse site exite. Agora faremos a mesma coisa, com Ajax. É uma camada, acima do CSS, que melhora ainda mais a experiência de quem tem navegadores capazes de vê-la. Mas as coisas continuam funcionando normalmente no Lynx.

Isso tem vantagens muito semelhantes ao uso de CSS para construção de layouts tableless, você separa seu conteúdo do comportamente, e pode alterar um sem precisar mexer no outro. E, é claro, você pode escolher não trabalhar assim, e se tiver uma aplicação realmente fantástica, como o Google Maps, as pessoas vão agradecê-lo. Depende do que você vai desenvolver, e como isso vai ser usado.

A aplicação que queremos desenvolver é um site simples, de conteúdo comum. É exatamente este site que você está vendo, desenvolvido com PHP e Ajax.

Apenas um site comum

Para começar, vamos criar as primeiras camadas, do jeito mais simples possível para não confundir você com coisas que não importam agora. Vamos trabalhar com PHP porque é uma linguagem muito conhecida (embora eu odeie PHP.) O código é tão simples que eu não me animei em desenvolver versões Python ou ASP dele. Mas, se alguém quiser gastar cinco minutos para fazê-lo, nos avise que publicaremos aqui o link do seu trabalho para os interssados.

Há duas páginas php no processo, a index.php, em que você navega, e a funcoes.php, que contém as funções PHP necessárias e será também a página que será requisitada pelo Ajax. Comecemos pela index.php. Há um trecho assim no começo:

```
include("funcoes.php");
```

```
//Lê o parâmetro i (índice do conteúdo)
$i=1;
if(isset($_GET["i"]))$i=intval($_GET["i"]);
```

Aqui incluímos o arquivo de funções e lemos o valor i recebido na querystring. Se i não estiver presente, seu valor será 1. No menu os links são construídos assim:

```
<li><a href="?i=1"<?classi(1)?> title="Parte 1">Introdução</a></li>
<li><a href="?i=2"<?classi(2)?> title="Parte 2">Objetivos</a></li>
<li><a href="?i=3"<?classi(3)?> title="Parte 3">Site comum</a></li>
<li><a href="?i=4"<?classi(4)?> title="Parte 4">Parte 4</a></li>
<li><a href="?i=5"<?classi(5)?> title="Parte 5">Parte 5</a></li>
```

As chamadas à função classi servem para que o código `class="selected"` seja inserido no link atual (é isso o que deixa marcado o link atual no menu). O último código de interesse:

```
<div id="conteudo">
    <?=leconteudo($i)?>
</div>
```

Aqui dizemos que, dentro do div conteudo, o PHP vai colocar o conteúdo (escolhido pelo valor de i na querystring). Agora vamos dar uma olhada no funcoes.php. O trecho que nos interessa agora:

```
/*
Lê o conteúdo de índice n. Aqui estou lendo de arquivos
html no disco, para não perdermos tempo com coisas que
fogem ao escopo do artigo. No mundo real, geralmente você
vai ler isso aqui do banco de dados, ou usar uma função
pronta disponibilizada por seu CMS.
*/
function leconteudo($n){
    return file_get_contents("$n.html");
}

//Insere class="selected" se n=i
function classi($n){
    global $i;
    if($n==$i)echo ' class="selected"';
}
```

As funções são as mesmas que chamamos no index, de modo que acho que apenas os comentários já elucidam tudo (se você não entende PHP, contente-se em ler os comentários, é simples.)

Com isso já temos um site, funcionando.

Como obter o conteúdo

Há duas maneiras de um navegador requisitar informações do servidor sem descarregar a página atual. A primeira é a arqueológica técnica de esconder um frame (hoje em dia prefiro trabalhar com um iframe, claro) e fazer suas requisições nele. Assim, digamos que quiséssemos obter uma lista de cidades do servidor, e passar essa lista para a função javascript fazAlgumaCoisaComAsCidades. Poderíamos requisitar uma página num iframe, cujo código seria:

```
<script type="text/javascript">
parent.fazAlgumaCoisaComAsCidades([
  "Belo Horizonte",
  "Rio de Janeiro",
```

```
  "Porto Alegre",
  "Curitiba"
])
</script>
```

Essa solução era usada há muito tempo, desde o século passado. Esconde-se o iframe com CSS e tudo resolvido. Apesar disso, é um jeito um tanto deselegante de resolver o problema, principalmente porque vai fazer aquele sonzinho de "clic" ao navegar no Internet Explorer, mas também porque vai interferir no botão de voltar e entrar no histórico do sujeito.

Uma abordagem mais atual para o problema é o uso do objeto XMLHttpRequest, que faz parte do padrão ECMA e está presente em todas as boas versões do Javascript. Os navegadores que suportam XMLHttpRequest hoje incluem:

- Opera 8
- Mozilla e Firefox
- Konqueror
- Safari

Além disso o Internet Explorer, desde a versão 5, suporta o Microsoft XMLHTTP, um substituto para o XMLHttpRequest bom o suficiente para que possamos usá-lo (e não mais que isso.)

Assim, nosso primeiro desafio é criar o objeto XMLHttpRequest ou XMLHTTP, dependendo do navegador. Isso não é tão complicado:

```
try{
    xmlhttp = new XMLHttpRequest();
}catch(ee){
    try{
        xmlhttp = new ActiveXObject("Msxml2.XMLHTTP");
    }catch(e){
        try{
            xmlhttp = new ActiveXObject("Microsoft.XMLHTTP");
        }catch(E){
            xmlhttp = false;
        }
    }
}
```

No final desse código, teremos em xmlhttp o objeto apropriado para nossa conexão ou, no caso de um navegador sem suporte a nenhum dos objetos acima, teremos false, que podemos usar para não mexer no conteúdo em navegadores onde o Ajax vai funcionar.

Vejamos um exemplo de como usar esse objeto:

```
xmlhttp.open("GET", "http://www.tableless.com.br/ajax/",true);
xmlhttp.onreadystatechange=function() {
    if (xmlhttp.readyState==4){
        alert(xmlhttp.responseText)
    }
}
xmlhttp.send(null)
```

O objeto xmlhttp é executado de maneira assíncrona, numa outra thread. Por isso não lemos o retorno depois de chamar o método send, mas criamos uma função onreadystatechange, que será executada pelo objeto quando ele terminar sua navegação.

Contruindo a aplicação

Vamos agora construir a página PHP que será solicitada pelo objeto XMLHttpRequest, em seguida faremos as função Javascript em que o navegador usará está página. Vamos usar a própria funcoes.php, inserindo ao final:

```
/*
Essa aqui é a parte necessária para o Ajax. Se este
arquivo for chamado sozinho, recebendo um parâmetro
n, ele retorna o texto de índice n. Passa pela
função urlencode por causa dos bugs do MSXML com
acentos (valeu mais uma vez, Bill!)
*/
if(isset($_GET["n"])){
    $t=leconteudo(intval($_GET["n"]));
    echo(urlencode($t));
}
```

Note que estamos usando a leconteudo, função que já tínhamos. DRY. Esse é um bom conselho. Se for preciso, quebre as funções que você já tem em pedaços menores, para que você possa usar para gerar o código para Ajax as mesmas funções que usa para o código normal. Isso vai salvar sua vida, principalmente na manutenção.

Usamos urlencode porque o componente da Microsoft tem um grave problema com acentos. Não consegui até hoje fazer o Internet Explorer 5 requisitar corretamente páginas com acentos de jeito nenhum. Então usamos urlencode no servidor e depois, no navegador, usaremos seu inverso em javascript: unescape. Agora a mágica:

```
atual=0
function carrega(n){

    //Exibe o texto carregando no div conteúdo
    var conteudo=document.getElementById("conteudo")
    conteudo.innerHTML='<div class="carregando">carregando...</div>'

    //Guarda a página escolhida na variável atual
    atual=n

    //Abre a url
    xmlhttp.open("GET", "funcoes.php?n="+n,true);

    //Executada quando o navegador obtiver o código
    xmlhttp.onreadystatechange=function() {

        if (xmlhttp.readyState==4){

            //Lê o texto
            var texto=xmlhttp.responseText

            //Desfaz o urlencode
            texto=texto.replace(/\+/g," ")
            texto=unescape(texto)

            //Exibe o texto no div conteúdo
            var conteudo=document.getElementById("conteudo")
            conteudo.innerHTML=texto

            //Obtém os links do menu
            var menu=document.getElementById("menu")
            var links=menu.getElementsByTagName("a")

            //Limpa as classes do menu
            for(var i=0;i<links.length;i++)
                links[i].className=""

            //Marca o selecionado
            links[atual-1].className="selected"
        }
    }
    xmlhttp.send(null)
```

```
}
```

Essa função carrega o texto selecionado por n e o coloca no div conteudo, além de marcar o link selecionado no menu com a classe selected. Se você tiver dificuldades com ele, sugiro uma lida no meu tutorial de DHTML Crossbrowser, principalmente nas seções "Maquiagem" e "Conteúdo".

Fazendo acontecer

O truque agora é o seguinte: precisamos fazer com que chamadas à função carrega sejam atribuídas aos cliques nos links do menu. O jeito pré-histórico de se fazer isso é inserir `onclick="carrega(1);return false"` em cada um dos links. Mas isso vai gerar uma série de dores de cabeça. O primeiro problema é que assim acoplamos o HTML ao Javascript, de modo que se um dia removermos ou mudarmos o javascript vamos ter que mexer de novo no HTML. Para você acostumado com tableless, isso é o equivalente a usar `align="center" bgcolor="red"`, por exemplo. O segundo problema é que em navegadores sem suporte a XMLHttpRequest a função vai ser chamada assim mesmo, gerando erros de javascript. Então o ideal é que usemos event listeners, de modo a não ter que inserir mais nada em nosso HTML. Isso ficaria assim:

```
function menuclick(e){

    //Correção para eventos quebrados da Microsoft
    if(typeof(e)=='undefined')var e=window.event
    source=e.target?e.target:e.srcElement
    //Correção para o bug do Konqueror/Safari
    if(source.nodeType==3)source=source.parentNode

    //Obtém o número quebrando a url
    n=source.getAttribute("href").replace(/.*=/g,"")

    //Chama o carrega
    carrega(parseInt(n))

    //Cancela o click (evita a navegação)
    return false
}

function init(){

    //Obtém os links do menu
    var menu=document.getElementById("menu")
    var links=menu.getElementsByTagName("a")

    //Atribui o evento
    for(var i=0;i<links.length;i++)
        links[i].onclick=menuclick
}

if(xmlhttp)window.onload=init
```

Aqui também o código é bastante simples, e você deve entender apenas lendo os comentários. Os pontos que podem gerar discussão são as correções para IE e Konqueror no começo da menuclick e a expressão regular que quebra a url.

As correções para navegadores foram ensinadas no tutorial de DHTML Crossbrowser, na seção "Comportamento", e a expressão regular é bastante simples, ela apenas elimina do endereço do link tudo o que vem antes do sinal de igual (Para saber mais sobre a fantástica ferramenta que é o uso de expressões regulares recomendo o guia do Aurélio.)

O passo seguinte é colocar todo o conteúdo em um arquivo javascript, que eu chamei de ajax.js, e inserir uma chamada para esse arquivo no cabeçalho do index.php. E está pronta nossa aplicação.

Conselhos finais

Espero que você tenha entendido basicamente do que se trata o tal Ajax. Espero também seu feedback. Se você encontrar algo errado, por favor avise. Se tem mais idéias e quer extender o código aqui, sinta-se à vontade.

Se quer se aprofundar mais no assunto e aprender, por exemplo, como fazer mudar o endereço na barra de endereços e habilitar os botões de voltar, avançar e recarregar, há uma excelente referência aqui.

Este site foi testado em Internet Explorer 5.0 e 6.0, Firefox 1.0.4, Opera 8.01 e Konqueror 3.4.1 e, surpreendentemente, funcionou. Obviamente, é completamente acessível em navegadores em que o Javascript falhar (testei no Lynx.)

Fique ligado, depois de muito tempo em que o Javascript era mal visto pela nata dos padrões web, o movimento em direção a um uso inteligente e acessível está apenas começando. Sinto bons ventos.

Elcio Ferreira * Tableless.com.br