AJAX:Como começar

Este artigo guia o leitor através das bases do AJAX e oferece dois exemplos práticos simples para poder começar.

Conteúdo

[esconder]

[editar]

O que é AJAX

AJAX (Asynchronous JavaScript and XML) é um termo criado recentemente para duas características poderosas dos browsers que existem há anos mas tem sido ignoradas por muitos criadores de páginas web até recentemente, quando aplicações como Gmail, Google suggest e Google Maps foram lançadas.

As duas principais características são a possibilidade de :

- efectuar pedidos ao servidor sem ter de recarregar a página
- analisar gramaticalmente e trabalhar com documentos XML

[editar]

1º passo - dizer "Por favor!" ou Como fazer um pedido HTTP

Para se fazer um pedido HTTP ao servidor usando JavaScript, você precisa de uma instância de uma classe que disponibiliza essa funcionalidade. Tal classe foi primeiro introduzida no Internet Explorer sob a forma de um objecto ActiveX chamado `XMLHTTP`. Então o Mozilla, o Safari e outros browsers seguiram-se, implementando uma classe de nome `XMLHttpRequest` que suportava os métodos e as propriedades do objecto ActiveX original da Microsoft.

Como resultado, em ordem para criar uma instância (objecto) multi-plataformas da classe pretendida, você pode fazer:

```
if (window.XMLHttpRequest) { // Mozilla, Safari, ...
    http_request = new XMLHttpRequest();
} else if (window.ActiveXObject) { // IE
    http_request = new ActiveXObject("Microsoft.XMLHTTP");
}
```

(só a título de exemplo, o código acima é uma versão simplificada do código a ser usado para a criação de uma instância XMLHTTP. Para um exemplo mais "vida real", dê uma olhadela ao 3º passo deste artigo.)

Algumas versões de alguns browsers Mozilla não irão funcionar bem se a resposta do servidor não possuir um cabeçalho mime-type XML. Para satisfazer isto, você pode usar uma chamada extra a um método para ultrapassar o cabeçalho enviado pelo servidor, só no caso de não ser no formato `text/xml`.

```
http_request = new XMLHttpRequest();
http_request.overrideMimeType('text/xml');
```

A próxima coisa a ser feita é decidir o que quer fazer após receber a resposta do servidor ao seu pedido. Nesta etapa só precisa de dizer ao objecto pedido HTTP que função JavaScript irá processar a resposta. Isto é feito definindo a propriedade `onreadystatechange` do objecto ao nome da função JavaScript que pretende utilizar, tipo assim:

```
http_request.onreadystatechange = nameOfTheFunction;
```

Note-se que não existem chavetas após o nome da função e não são passados parâmetros. Também, em vez de dar um nome duma função, você pode usar a técnica JavaScript de definir funções "à pressão" e definir as acções que vão processar a resposta logo, tipo assim:

```
http_request.onreadystatechange = function(){
    // do the thing
};
```

De seguida, após ter declarado o que vai acontecer mal receba a resposta, você precisa de consumar o pedido. Precisa de chamar os métodos `open()` e `send()` da classe pedido HTTP, tipo assim:

```
http_request.open('GET', 'http://www.example.org/some.file',
true);
http_request.send(null);
```

- O primeiro parâmetro da chamada do método `open()` é o método pedido HTML – GET, POST, HEAD ou outro método qualquer que queira usar e que seja suportado pelo seu servidor. Mantenha o nome do método em maiúsculas para obedecer às normas HTTP senão certos browsers (como o Firefox) podem não processar o pedido. Para obter mais informação sobre os possíveis métodos pedido HTTP pode dar uma olhadela em W3C specs
- O segundo parâmetro é a URL da página que está a pedir. Como medida de segurança, não pode efectuar pedidos de páginas de domínios externos. Certifique-se que usa o nome exacto do domínio em todas as suas páginas ou irá receber um erro "Permissão Negada" quando efectua uma chamada `open()`. Um erro comum é aceder ao seu domínio através de domínio.tld ao mesmo tempo que tenta chamar páginas com www.domínio.tld.
- O terceiro parâmetro define se o pedido é assíncrono. Se `TRUE`, a execução da função JavaScript irá continuar enquanto que a resposta do servidor ainda não foi recebida. Isto é o A de AJAX.

O parâmetro do método `send()` pode ser costituido por quaisquer dados que pretenda enviar ao servidor ao enviar (POST) o pedido. Os dados devem estar sob a forma de uma linha de texto de pergunta, tipo:

`name=value&anothername=othervalue&so=on`

Note-se que se pretende enviar (POST) dados, você deve alterar o tipo MIME do pedido usando a seguinte linha:

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

De outra forma o servidor irá ignorar os dados (post).

[editar]

2º Passo - "Ora aqui está!" ou Lidar com a Resposta do Servidor

Lembre-se que quando estava a enviar o pedido, você providenciou o nome de uma função JavaScript que é criada para lidar com a resposta.

`http_request.onreadystatechange = nameOfTheFunction;`

Vamos a ver o que é que esta função deve fazer. Primeiro, a função precisa de verificar o estado do pedido. Se o estado possui o valor 4, isso significa que a totalidade da resposta do servidor foi recebida e que pode continuar a processá-la à vontade.

```
if (http_request.readyState == 4) {
    // everything is good, the response is received
} else {
    // still not ready
}
```

A lista completa dos valores `readyState` é a seguinte:* 0 (uninitialized)

- 1 (a carregar)
- 2 (carregado)
- 3 (interactivo)
- 4 (completo)

(Source)

A próxima coisa a verificar é o código do estado da resposta HTTP do servidor. Todos os códigos possíveis estão listados na página W3C. Para os nossos objectivos nós só estamos interessados na resposta `200 OK`.

```
if (http_request.status == 200) {
    // perfect!
} else {
    // there was a problem with the request,
    // for example the response may be a 404 (Not Found)
    // or 500 (Internal Server Error) response codes
}
```

Depois de verificar o estado do pedido e o código do estado HTTP da resposta, compete-lhe a si fazer aquilo que quer fazer com os dados que o servidor lhe enviou. Tem duas opções para aceder a esses dados:

- `http_request.responseText` – irá devolver a resposta do servidor como uma linha de texto
- `http_request.responseXML` – irá devolver a resposta do servidor como um objecto `XMLDocument` que pode percorrer usando as funções DOM de JavaScript.

[editar]

Passo 3 – "Agora todos juntos!" - Um exemplo simples

Vamos agora pôr tudo junto e efectuar um simples pedido HTTP. O nosso JavaScript vai pedir um documento HTML, `teste.html`, que contém o texto "Sou um teste." e então vamos `alert()` os conteúdos do ficheiro `teste.html`.

```
<script type="text/javascript" language="javascript">

    var http_request = false;

    function makeRequest(url) {

        http_request = false;

        if (window.XMLHttpRequest) { // Mozilla, Safari,...
            http_request = new XMLHttpRequest();
            if (http_request.overrideMimeType) {
                http_request.overrideMimeType('text/xml');
                // See note below about this line
            }
        } else if (window.ActiveXObject) { // IE
            try {
                http_request = new
ActiveXObject("Msxml2.XMLHTTP");
            } catch (e) {
                try {
                    http_request = new
ActiveXObject("Microsoft.XMLHTTP");
                } catch (e) {}
            }
        }

        if (!http_request) {
            alert('Giving up :( Cannot create an XMLHTTP
instance');
```

```
        }
        http_request.onreadystatechange = alertContents;
        http_request.open('GET', url, true);
        http_request.send(null);

    }

    function alertContents() {

        if (http_request.readyState == 4) {
            if (http_request.status == 200) {
                alert(http_request.responseText);
            } else {
                alert('There was a problem with the
request.');
            }
        }

    }
</script>
<span
    style="cursor: pointer; text-decoration: underline"
    onclick="makeRequest('test.html')">
        Make a request
</span>
```

Neste exemplo:

- O utilizador clicka no atalho "efectuar pedido" no browser;
- Isto chama a função `makeRequest()` com um parâmetro -- o nome `teste.html` de um ficheiro HTML na mesma directoria;
- O pedido é feito e então (`onreadystatechange`) a execução é passada a `alertContents()`;
- `alertContents()` verifia se a resposta foi recebida e se é um OK e então alerta (`alert()`) os conteúdos do ficheiro `test.html`.

Você pode testar o exemplo aqui e pode ver o ficheiro de teste aqui.

Nota: A linha `http_request.overrideMimeType('text/xml');` acima irá causar erros de Consola JavaScript no Firefox 1.5b tal como está documentado em https://bugzilla.mozilla.org/show_bug.cgi?id=311724 se a página chamada por `XMLHttpRequest` não for XML válido (por exemplo, se for texto simples).

Se obter Erro de Sintaxe ou Erro Mal Formado naquele browser e não está a tentar carregar um apágina XML a partir de `XMLHttpRequest`, retire aquela linha do seu código.

Nota 2: se estiver a enviar um pedido a um pedaço de código que irá retornar XML em vez de a um ficheiro estático de XML então precisa de definir algum cabeçalho de resposta se a sua página precisa de funcionar no Internet Explorer além do Mozilla. Se não definir o cabeçalho `Content-Type: application/xml`, o Internet Explorer irá lançar um erro JavaScript 'Esperado Objecto' após a linha onde tenta aceder a um elemento XML. Se não define o cabeçalho `Cache-Control: no-cache` o browser irá guardar a resposta e nunca re-enviar o pedido, tornando o depuramento de erros "um desafio".

[editar]

Passo 4 – "Os Ficheiros-X" ou Trabalhar com a Resposta XML

No exemplo anterior, após termos recebido a resposta ao pedido HTTP, nós usamos a propriedade `reponseText` do objecto de pedido e continha os conteúdos do ficheiro `test.html`. Agora vamos experimentar a propriedade `responseXML`.

Antes de tudo, vamos criar um documento XML válido que vamos pedir mais à frente. O documento (test.xml) contém o seguinte:

```
<?xml version="1.0" ?>
<root>
    I'm a test.
</root>
```

No guião só precisamos de alterar a linha do pedido com:

```
...
onclick="makeRequest('test.xml')">
...
```

Então em `alertContents()` nós precisamos de substituir a linha de alerta (`alert(http_request.responseText);`) com:

```
var xmldoc = http_request.responseXML;
var root_node = xmldoc.getElementsByTagName('root').item(0);
alert(root_node.firstChild.data);
```

Assim pegamos no objecto `XMLDocument` fornecido por `responseXML` e usamos os métodos DOM para aceder a alguns dados obtidos no documento XML. Pode ver o ficheiro `test.xml` aqui e o guião de testes actualizado aqui.