AJAX Desmistificado

Marcio Belo R. Silva[1]

[1]Instituto Superior de Tecnologia em Ciência da Computação – Quintino – FAETEC
Rio de Janeiro – RJ – Brasil

mbelo.br@gmail.com

Abstract.. *The centralized processing model found in the first Internet applications, here referred as 1st generation, represented a step backward in the computational evolution, back to mainframe´s time and theirs dumb terminals. Although the JavaScript language reduced the waste of resources in the client side, these 2nd generation of WEB applications still lacks of performance in the user interaction with the interface. The technique that became known by its AJAX acronym promises to offer to the users of 3rd generation of WEB applications the same experience obtained by using the popular windows-based applications.*

Resumo.. *O modelo de processamento centralizado das primeiras aplicações Internet, aqui chamadas de 1ª geração, representaram um retrocesso computacional, uma vez que remontam ao modelo usado pelos ultrapassados mainframes e seus terminais burros. Embora a linguagem JavaScript tenha minimizado o desperdício de recursos do lado cliente, essas aplicações WEB de 2ª geração continuam carecendo de melhorias de performance na interação do usuário com a interface. A técnica que se popularizou pelo acrônimo AJAX promete oferecer ao usuário das aplicações WEB de 3ª geração a mesma experiência no manejo da interface da obtida nas já populares aplicações de janela.*

1. Introdução

Recentemente, a maioria dos novos sistemas aplicativos está sendo desenvolvida com interface WEB. As vantagens desse novo modelo de desenvolvimento são muitas, entre as principais podemos citar: (i) menor curva de aprendizado no entendimento da interface pelo usuário final, (ii) maior facilidade na distribuição do aplicativo, sem necessidade de instaladores, e, (iii) acesso remoto transparente. Entretanto, muitos dos recursos que apaixonaram os usuários finais com o advento das aplicações de janelas gráficas sumiram nas novas aplicações WEB. Um dos exemplos que podemos citar é o interessante recurso de autocompletar em caixas de texto, onde o usuário digita parte de uma palavra e o sistema sugere várias opções que possivelmente completariam o texto parcialmente digitado. Esse e outros recursos das janelas gráficas resultavam, em comparação com as aplicações WEB clássicas, um menor tempo na interação do usuário com o aplicativo para realizar uma mesma tarefa.

Essa limitação dos primeiros aplicativos WEB devia-se a questões de arquitetura. Nas aplicações WEB tradicionais, neste artigo denominadas de 1ª geração[1], boa parte das interações do usuário com a interface demandam que uma requisição seja submetida ao servidor e este, por sua vez, produza uma nova interface para ser exibida ao usuário. Durante esse vai-e-volta (*round-trip*) de comunicação entre o lado cliente (navegador WEB) e o servidor, o usuário fica impossibilitado de usar a aplicação. Dependendo do canal de comunicação, essa interação pode produzir uma demora além do razoável.

Para resolver essa limitação das aplicações WEB tradicionais, surge uma combinação de tecnologias denominada AJAX [Garret, 2005]. AJAX é um acrônimo para *Asynchronous JavaScript + XML*. Essa tecnologia promete revolucionar a nova geração de aplicações WEB, como já podemos observar em vários sistemas disponíveis na Internet, como o Goggle(c) Gmail, onde podemos encontrar recursos como autocompletar com uma experiência bastante semelhante à observada nos aplicativos de janelas.

2. Interação entre Cliente e Servidor na WEB

Nas aplicações WEB clássicas, o servidor, em várias das interações do usuário com a interface, recebia demanda de processamento para a geração de uma nova página. Essa ida-e-volta entre cliente e servidor aumenta o tempo de resposta do sistema, obrigando o usuário esperar o recebimento da nova página para prosseguir com o uso dela, e sobrecarrega o servidor, que precisa processar toda a página incondicionalmente, mesmo que apenas parte dela tenha sido modificada.

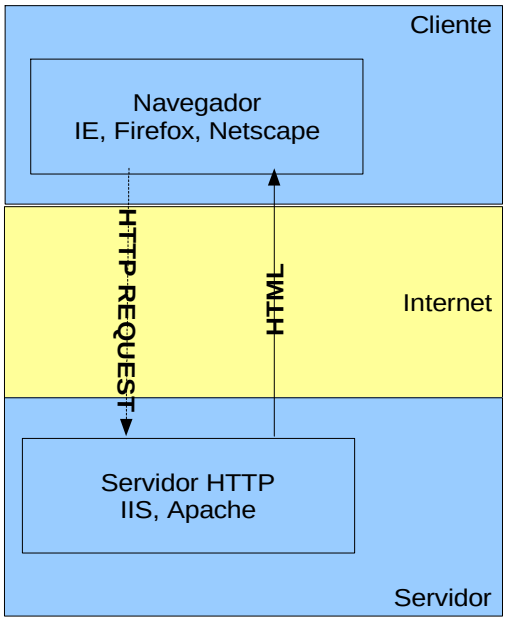

Ilustração 1: WEB Clássica

[1] Termo cunhado pelo autor para referir-se à primeira geração de aplicações WEB, como marco cronológico da evolução da arquitetura desse tipo de aplicativo.

O fluxo de comunicação citado anteriormente lembra muito o modo centralizado de computação que existia na época dos computadores de grande porte, os *mainframes*, o que pode ser considerado um retrocesso computacional, uma vez que os terminais ligados aos computadores de grande porte, chamados não à toa de terminais burros, tinham pouco ou nenhum poder computacional, enquanto os computadores clientes da era Internet tem, com raras exceções, grande capacidade computacional, sendo então subutilizados nesse tipo de arquitetura WEB clássica.

Era esperado que essa situação de desperdício não tardasse a mudar: o uso da linguagem JavaScript possibilitou aos desenvolvedores criar funcionalidades básicas que rodam na máquina cliente através dos navegadores WEB com suporte à linguagem. Aplicações como validações de dados, mensagens e mudanças de estado nos objetos da interface, entre outras, começaram a ser amplamente utilizadas nas aplicações WEB de 2ª geração[2].

Mesmo com as melhorias introduzidas na 2ª geração de aplicações WEB, sempre que o aplicativo rodando no navegador cliente necessitava de dados armazenados no servidor, era necessária uma troca de dados do tipo ida-e-volta entre ambos, gerando uma página inteiramente nova, mesmo que os dados requeridos tenham alterado uma pequena parte da interface. No meio tempo em que o aplicativo demandava esses dados e o servidor enviasse em resposta a nova página, cabia ao usuário esperar e esperar. A ilustração 2 representa essa dinâmica no uso da aplicação.

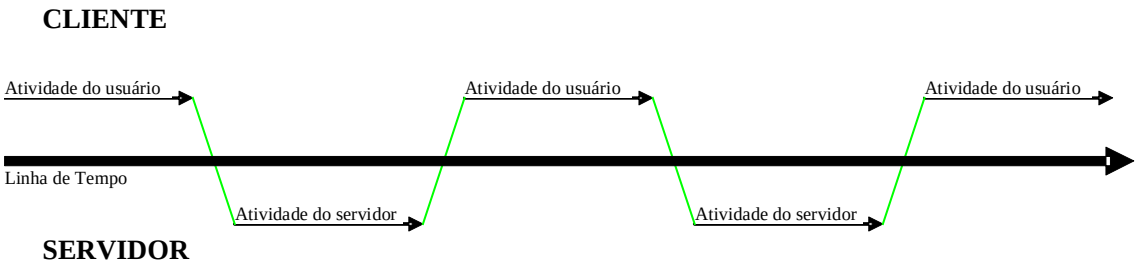

Ilustração 2: Dinâmica das aplicações WEB clássicas

3. AJAX

AJAX não é uma tecnologia em si, mas sim uma técnica para aplicação de várias tecnologias que, combinadas, provém aos aplicativos o mesmo dinamismo possível em aplicações de janela. A seguir são listadas as tecnologias envolvidas e suas aplicações:

- XHTML e CSS: para apresentação e formatação de dados;
- DOM (*Document Object Model*): para manipulação dinâmica dos objetos do navegador;

[2] Termo cunhado pelo autor para referir-se à geração de aplicações WEB que introduziram o uso de JavaScript, como marco cronológico da evolução da arquitetura desse tipo de aplicativo.

- XML e XSTL: para representação e transformação dos dados transportados entre o servidor e o cliente;
- Objeto XMLHttpRequest: para o transporte assíncrono de dados entre o navegador e o servidor;
- E a linguagem JavaScript.

É importante ressaltar que essas várias tecnologias já existiam quando o termo AJAX foi primeiramente mencionado em [Garret, 2005].

Utilizando AJAX, os desenvolvedores podem criar aplicações que busquem dados no servidor através de uma requisição assíncrona, mantendo a página que está sendo exibida ao usuário e, quando os dados requisitados são recebidos, apenas uma pequena parte da página que originou a requisição é alterada.

Para exemplificar as vantagens no uso AJAX, foi desenvolvida uma pequena aplicação que tira proveito da técnica para atualizar campos de um formulário a partir de seleções dependentes. Nela, o usuário indica uma unidade da federação (UF) e o sistema preenche, sem necessitar de um *post back*, o campo cidade com a lista dos municípios pertencentes àquela UF. O mesmo mecanismo é aplicado na interação com os campos bairros, ruas e CEP. A ilustração 3 demonstra o modelo de comunicação utilizado pela aplicação.

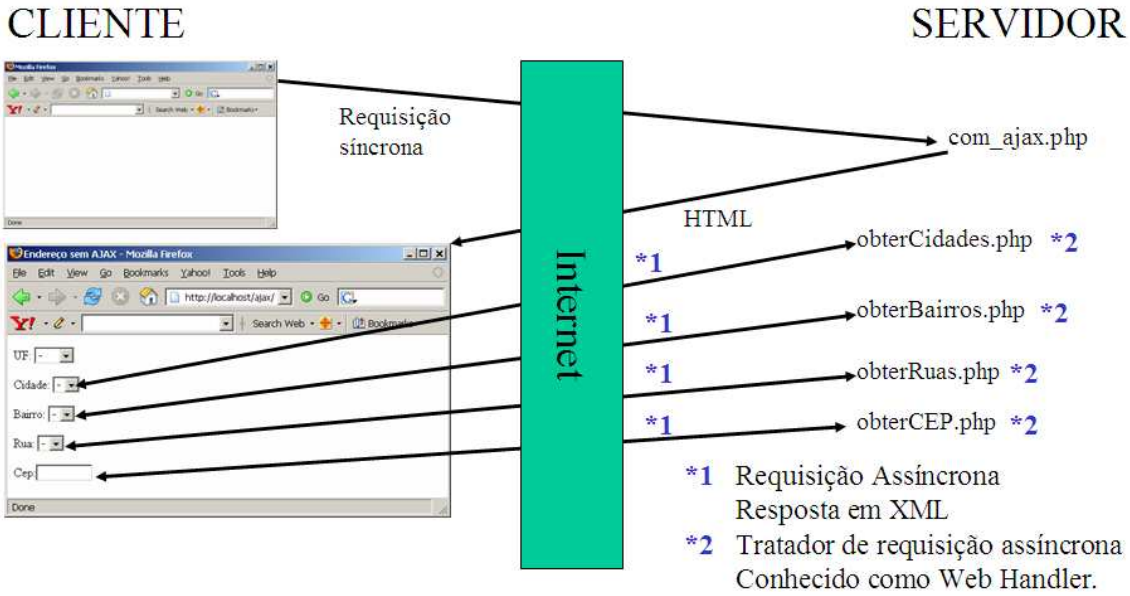

Ilustração 3 Exemplo de aplicativo utilizando AJAX

A implementação básica desse mecanismo exige a codificação de 3 trechos de código distintos, sendo dois para o cliente WEB e 1 para o servidor, são eles:

1. Emissor da Requisição Assíncrona: codificado em JavaScript para execução no navegador cliente, esse código tem o objetivo gerar a requisição assíncrona ao servidor, mas precisamente direcionada ao Tratador de Requisições Assíncronas (*Web Handler*), através do objeto XMLHttpRequest. O código 1 é o trecho do aplicativo exemplo que desempenha esse papel.

```
function carregarLista(url,funcaoProcessaResposta) {

   // Instancia objeto de invocação remota
   try
   {
      // Tenta instanciar objeto Internet Explorer
      req=new ActiveXObject("Msxml2.XMLHTTP");
   }
   catch (e)
   {
      // Tenta instaciar outro objeto IE
      try
      {
        req=new ActiveXObject("Microsoft.XMLHTTP");
      }
      catch (e2)
      {
        // Instancia objeto Mozilla Firefox
        if(typeof XMLHttpRequest != "undefined") req = new XMLHttpRequest();
      }
   }

   // Aborta execução em caso de falha
   if(!req) exit();

   req.open("GET", url, true);
   req.onreadystatechange=funcaoProcessaResposta;
   req.send(null);
   return true;
}
```

Código 1 Gera requisição assíncrona ao *Web Handler* no servidor

2. Tratador da Requisição Assíncrona: também conhecido como *Web Handler*, escrito em qualquer linguagem de programação do lado WEB servidor, esse código recebe a requisição assíncrona, identificando os argumentos passados (por um dos métodos GET ou POST), realiza o processamento e retorna o resultado – geralmente em XML - para o lado cliente que o requisitou. O código 2 é um exemplo tirado do aplicativo demonstração, que utilizou a linguagem PHP [PHP Group, 2006] para programação do lado WEB servidor.

```
<?php
  require_once("conexao.php");
   header("Expires: Mon, 26 Jul 1997 05:00:00 GMT");
   header("Last-Modified: " . gmdate("D, d M Y H:i:s") . " GMT");
   header("Cache-Control: no-cache, must-revalidate");
   header("Pragma: no-cache");
   header("Content-type: text/xml; charset=utf-8");
   echo "<?xml version=\"1.0\" encoding=\"UTF-8\"?>";
   echo "<cidades>";
   $uf=$_REQUEST["uf"];
   $sql="select CCODIGO,CNOME from CEP_CIDADE where CUF_CODIGO=$uf order by CNOME";
   $query=ibase_query($con, $sql);
   while( $row = ibase_fetch_object($query) ) {
      $CCODIGO = $row->CCODIGO;
      $CNOME = $row->CNOME;
      echo "<cidade codigo=\"$CCODIGO\">$CNOME</cidade>";
   }
   echo "</cidades>";
?>
```

Código 2 Tratador de Requisição Assíncrona ou *Web Handler*

3. Tratador de Resposta Assíncrona: codificado em JavaScript no lado cliente WEB, esse código é o responsável por receber os dados enviados como resposta pelo Tratador de Requisições Assíncronas, geralmente formatados em XML, e promover uma reação na interface atualmente exibida ao usuário. O código 3 é um exemplo de tratador de resposta retirado do aplicativo demonstração.

```
function processaResposta() {

   // Este evento é chamado 4 vezes, para cada mudança de estado no xmlhttp
   // Os estados são: 0 uninitialized, 1 loading, 2 loaded, 3 interactive, 4
complete
   if( req.readyState != 4) return;

   if (req.status == 200) {

      // responseXML contém um objeto XMLDOM
      // documentElement pega o elemento de maior nível do XML
      var xml = req.responseXML.documentElement;
      switch( xml.nodeName ) {
        case "cidades": caixa=document.forms["correio"].cidade; break;

        case "bairros": caixa=document.forms["correio"].bairro; break;

        case "ruas": caixa=document.forms["correio"].rua; break;
        case "cep":
          document.forms["correio"].CEP.value = xml.attributes[0].value;
          exit();
      }

      // Adiciona o item na lista
      for(var i=0; i < xml.childNodes.length; i++) {
        var o = new Option();
        o.text = xml.childNodes[i].firstChild.nodeValue;
        o.value = xml.childNodes[i].attributes[0].value;
        caixa.options.add(o);
      }
      caixa.options[0].text=""; /* apagando o conteúdo carregando */
   }
}
```

Código 3 Tratador de Resposta Assíncrona

Outro interessante exemplo de aplicação que utiliza AJAX pode ser encontrado em [Info online, 2006].

4. Desvantagens

Numa análise antecipada, poderíamos concluir que são muitas as vantagens oferecidas pela técnica AJAX e que qualquer próxima aplicação WEB a ser desenvolvida deve aplicar essa tecnologia. Embora essa idéia seja verdadeira, existem alguns obstáculos que precisam ser transpostos antes da tecnologia atingir seu estágio de amadurecimento.

Um dos problemas recorrentes desde o início das tecnologias de programação WEB, a falta de suporte do navegador para esse ou aquele padrão, ou mais grave, implementação não adequada do padrão, torna-se novamente um entrave na decisão por adotá-lo ou não, sendo nesse caso a técnica AJAX.

O problema que os desenvolvedores encontrarão está principalmente na ausência de suporte a uma ou mais tecnologias necessárias para aplicar o AJAX. Para lidar com esse problema existem três alternativas: (i) dispensar o uso da tecnologia e permitir que a aplicação rode na maioria dos navegadores em uso no mercado, (ii) obrigar que o usuário tenha um navegador com suporte AJAX, abrindo mão dos usuários com

navegadores antigos ou (iii) criar uma aplicação sensível ao navegador do usuário. Nessa última opção, o código do servidor deve detectar o tipo de navegador usado e os recursos que nele estão habilitados, e, de forma seletiva, enviar código adequado ao navegador. Caso o navegador dê suporte às tecnologias AJAX, o código adequado é enviado ao cliente, caso contrário, o servidor envia, por exemplo, páginas em HTML puro e usa o mesmo mecanismo de *postback* das aplicações de 1ª geração.

5. Conclusão

A tecnologia AJAX promete revolucionar a 3ª geração[3] de aplicações. Nesse artigo, apenas um pequeno exemplo no emprego da técnica AJAX foi citado. Cada vez mais, novos aplicativos na WEB empregam a técnica AJAX, e certamente serão os preferidos entre a comunidade usuária, devido aos recursos diferenciados que ela pode oferecer e a uma experiência similar às tradicionais aplicações de janelas. Embora todas as tecnologias envolvidas estejam disponíveis há algum tempo, certamente ainda não podemos vislumbrar todos os possíveis usos que a técnica possibilita. Para os próximos anos, podemos esperar que novos aplicativos em AJAX introduzam novidades na interação do usuário com aplicativo que hoje não nos são conhecidas.

Mesmo com todas as vantagens que a tecnologia oferece, ainda não podemos criar aplicações que só rodem em navegadores com suporte AJAX, pois estaríamos limitando o número de potenciais usuários de nossas aplicações. A alternativa mais plausível envolverá a criação de aplicativos sensíveis ao navegador do usuário.

Referências

Garret, Jesse J., (2005) “AJAX: A New Approach to Web Applications”, Adaptative Path. http://www.adaptivepath.com/publications/essays/archives/000385print.php, Fevereiro.

Info online (2006), “Saiba como usar o Ajax (Asyncronous JavaScript and XML) para criar sites dinâmicos”, Abril.

Chilá, Marco A. (2006), “Ajax - Asynchronous JavaScript and XML - Um exemplo Simples”, http://www.devaspnet.com.br

PHP Group (2006), “PHP Manual”, www.php.net

[3] Termo cunhado pelo autor para referir-se à geração de aplicações WEB que introduziram o uso da técnica AJAX, como marco cronológico da evolução da arquitetura desse tipo de aplicativo.