Desenvolvimento Web II

Aula 06 - AJAX: Interação com o Servidor

Apresentação

Até agora, neste curso, você aprendeu como desenvolver aplicações Web utilizando JavaScript e AJAX, no cliente; e Servlets e JSP, no servidor. Nesta aula, iremos nos aprofundar nas formas de comunicação entre o cliente AJAX e o servidor, mais especificamente no formato como esses dados são enviados e recebidos.

Bons estudos!

Vídeo 01 - Apresentação

Objetivos

- Aplicar o formato XML na comunicação.
- Utilizar o formato JSON (JavaScript Object Notation) na comunicação.
- Diferenciar as vantagens e desvantagens de cada um dos modelos.

Comunicação AJAX entre cliente e servidor

Tanto na disciplina de Web I como nas aulas anteriores, você foi apresentado a um modo de desenvolver aplicações Web onde o código dos dados fica separado do código da interface e já foi mostrado que existem uma série de vantagens em se desenvolver dessa forma. Devido à simplicidade dos dados trafegados e também ao intuito de simplificar seu primeiro contato com AJAX, nas aulas anteriores, os dados trafegados via AJAX foram formatados de modo bastante simples, utilizando apenas ponto e vírgula para separar o nome de cada livro, está lembrado?

> Na requisição enviada ao servidor para obter a lista de livros da categoria selecionada: http://localhost:8080/AulasAjax1/Controlador/Ajax
>
> A cabana;A Ilha sob o mar;Querido John;Fallen;A ultima música;Caminhos da lei;Os homens que não amavam as mulheres;A ilusão da alma;Amante desperto;O castelo dos Pirineus;

Apesar de funcionar, o formato de dados utilizado pode não se adequar para casos em que os dados trafegados sejam mais complexos. Por exemplo, pense em como você faria para enviar os dados de uma lista de livros contendo os seguintes campos para cada livro:

- Nome do livro
- Nome (s) do (s) autor (es)
- Editora
- Edição
- ISBN
- Preço

- Sinopse
- Assunto

Apesar de ser possível usar um formato apenas com pontos e vírgulas, esse formato, provavelmente, seria bastante confuso e complexo. Desse modo, usamos formatos de dados mais elaborados, como você verá ao longo desta aula.

Além de serem facilmente visualizados e editados por meio de um editor de texto comum (como, por exemplo, o programa bloco de notas do Windows), também veremos que os padrões apresentados no decorrer desta aula ajudam a promover a interoperabilidade entre plataformas e linguagens distintas. Para tanto, precisamos saber que interoperabilidade é a capacidade de um programa se comunicar com outros, mesmo com aqueles construídos em linguagens diferentes (C, C++, Python etc.) ou para rodar em plataformas distintas (Windows ou Linux, desktop ou dispositivo móvel etc.).

Utilizando AJAX com XML

A primeira alternativa de formato de dados que você irá aprender para tratar respostas enviadas pelo servidor será o formato XML. XML (eXtensible Markup Language, ou seja, linguagem de marcação extensível) é uma excelente linguagem para representar dados. Segue abaixo uma das possíveis maneiras de representar em XML o modelo sugerido na introdução. Você já pode criar um novo projeto chamado Aula06 para essa aula e criar o arquivo livros.xml na pasta WebContent com esse conteúdo a seguir:

```xml
<?xml version="1.0" encoding="ISO-8859-1"?>
<livros>
   <livro nome="A Cabana" isbn="8599296361">

      <editora>SEXTANTE FICÇAO</editora>
      <edicao>1ª Edição - 2008 </edicao>

      <autores>
         <autor>YOUNG, WILLIAM P.</autor>
      </autores>

      <preco>12,45</preco>

      <sinopse>A filha mais nova de Mackenzie Allen Philip foi raptada
            durante as férias em família e há evidências de que ela foi
            brutalmente assassinada e abandonada numa cabana. Quatro anos mais
            tarde, Mack recebe uma nota suspeita, aparentemente vinda de Deus,
            convidando-o para voltar àquela cabana e passar o fim de semana.
            Ignorando alertas de que poderia ser uma cilada, ele segue numa tarde
            de inverno e volta ao cenário de seu pior pesadelo. O que encontra lá
            muda sua vida para sempre.</sinopse>
   </livro>

   <livro nome="A Ilha  Sob o Mar" isbn="8528614441">
      <editora>BERTRAND BRASIL</editora>
      <edicao>1ª Edição - 2010</edicao>

      <autores>
         <autor>ALLENDE, ISABEL</autor>
         <autor>SEGUNDO AUTOR</autor>
      </autores>

      <preco>12,45</preco>

      <sinopse>O romance narra a vida de Zarité, a escrava que foi vendida
            aos nove anos de idade para o francês Toulouse Valmorain, dono de uma
            das maiores plantações de cana-de-açúcar nas Antilhas. Como escrava
            doméstica, ela não padeceu as dores e as humilhações de seus iguais,
            mas conheceu as misérias de seus patrões - os brancos. </sinopse>
   </livro>
</livros>
```

Nas aulas anteriores, utilizamos o atributo responseText do objeto XMLHttpRequest para obter os dados enviados pelo servidor. No entanto, quando a resposta é um documento XML no qual desejamos recuperar determinados elementos, devemos utilizar o atributo responseXML, ao invés do responseText (ver códigos a seguir). O valor desse atributo é um objeto do tipo XML DOM, cujos detalhes você verá um pouco mais adiante.

```
var ajaxReq = new XMLHttpRequest();
...
var text = ajaxReq.responseText;
```

Trecho de código do Exemplo 2 da Aula 5

```
var ajaxReq = new XMLHttpRequest();
...
var xmlDoc = ajaxReq.responseXML;;
```

Trecho de código do Exemplo 2 modificado para recuperar o objeto XML DOM.

XML DOM

DOM é uma especificação utilizada por diversas linguagens de marcação (como, por exemplo, XML e HTML) para permitir que um documento seja navegado ou editado de maneira padronizada e simples. Você verá a seguir os métodos mais relevantes da API do XML DOM para nosso propósito, que é o de recuperar informações contidas em uma resposta em XML enviada pelo servidor.

> "DOM (Document Object Model – Modelo de Objetos de Documentos) é uma especificação da W3C, independente de plataforma e linguagem, onde pode-se, dinamicamente, alterar e editar a estrutura, o conteúdo e o estilo de um documento eletrônico, permitindo que o documento seja mais tarde processado e os resultados desse processamento sejam incorporados de volta no próprio documento. A API DOM oferece uma maneira padrão de se acessar os elementos de um documento, além de se poder trabalhar com cada um desses elementos separadamente, e por esses motivos criar páginas altamente dinâmicas."
>
> Fonte: <http://pt.wikipedia.org/wiki/Modelo_de_Objeto_de_Documentos>. Acesso em: 9 nov. 2010.

Vídeo 02 - JSON e Xxml

Tanto o XML quanto o JSON (que ainda será visto nesta aula) possui uma organização hierárquica, ou seja, os elementos (nós) que formam o documento são definidos sempre um dentro do outro: um documento XML tem sempre um nó principal que é nó raiz, que, por sua vez, pode conter de 0 a N nós filhos.

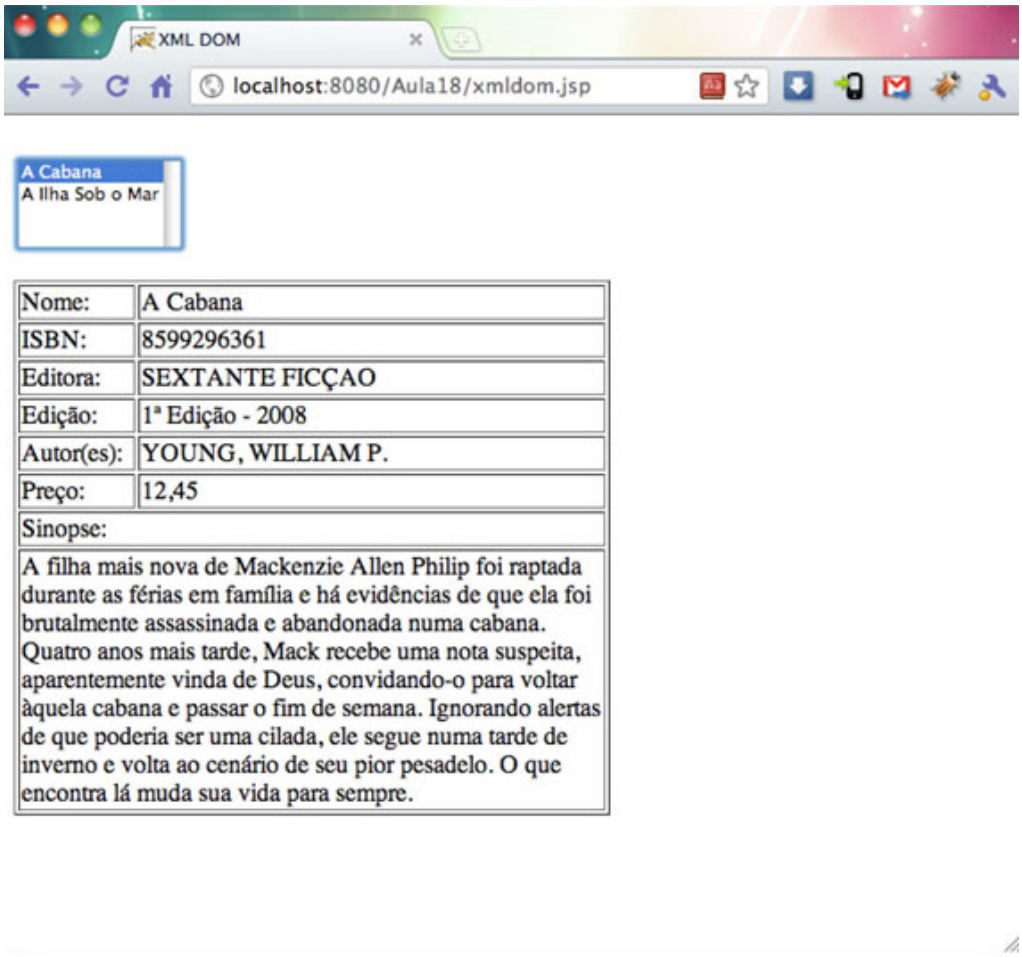
```xml
<?xml version="1.0" encoding="ISO-8859-1"?>
<noRaiz>
  <noFilho id="1">
    <noNeto id="1">
    </noNeto>
  </noFilho>
  <noFilho id="2">
  </noFilho>
</noRaiz>

```

Exemplo 1

O exemplo a seguir é composto por uma lista de nomes de livros e uma tabela que exibe os detalhes do livro selecionado. Logo, quando a página é carregada, dispara-se uma requisição AJAX que recupera uma lista de livros em XML do servidor (utilizando o modelo de dados apresentado na seção a seguir “Utilizando XML”).

A partir dessa lista de livros retornada pelo servidor, a lista de nomes é carregada com o nome de cada livro. Quando o usuário clica em algum dos livros da lista contendo os nomes dos livros, uma tabela exibindo os detalhes do livro selecionado é exibida (ou atualizada).

Figura 01 - Exemplo utilizando XML DOM



Veja a seguir o código da página da Figura 1, que por comodidade foi criado com a tecnologia JSP que você já conhece . Você já pode criar um arquivo na pasta WebContent do seu projeto Aula06 com esse conteúdo. Ele terá suas partes principais comentadas mais adiante:

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
  pageEncoding="ISO-8859-1"%>
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core"%>

<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">

<html>
   <head>
      <meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
      <title>XML DOM</title>
   </head>

   <body onload="onLoad()">

      <script type="text/javascript">

      var xmlDoc = null;
      var livro = null;

      function onLoad(){

      var ajaxReq = new XMLHttpRequest();
      var url = "/Aula06/livros.xml";

      ajaxReq.onreadystatechange = function() {
         if (ajaxReq.readyState == 4 && ajaxReq.status == 200) {
            xmlDoc = ajaxReq.responseXML;
            carregarLista();
         }
      };

      ajaxReq.open("GET", url, true);
      ajaxReq.send();

      }

      function carregarLista(){

         var livros = xmlDoc.getElementsByTagName("livro");
            for ( var i = 0; i < livros.length; i++) {

            var livro = livros.item(i);

            var nome="";
            var isbn="";

            if(livro.hasAttributes()){
               nome = livro.attributes.getNamedItem("nome").nodeValue;
               document.getElementById("nome").innerHTML = nome;
               isbn = livro.attributes.getNamedItem("isbn").nodeValue;
```

```
            document.getElementById("isbn").innerHTML = isbn;
        }

        var campoLivro = document.getElementById("livros");
        campoLivro.options[campoLivro.length] = new Option(nome, ""+i);

    }

}

function carregarLivro(indice){

    var livros = xmlDoc.getElementsByTagName("livro");
    var livro = livros.item(indice);

    var nome="";
    var isbn="";
    if(livro.hasAttributes()){
        nome = livro.attributes.getNamedItem("nome").nodeValue;
        document.getElementById("nome").innerHTML = nome;
        isbn = livro.attributes.getNamedItem("isbn").nodeValue;
        document.getElementById("isbn").innerHTML = isbn;
    }

    var nosFilhos = livro.childNodes;
    for ( var i = 0; i < nosFilhos.length; i++) {
        var noFilho = nosFilhos.item(i);

        switch (noFilho.nodeName) {
            case "editora":
            case "edicao":
            case "preco":
            case "sinopse":
                var valor = noFilho.firstChild.nodeValue;
                document.getElementById(noFilho.nodeName).innerHTML = valor
            break;
            case "autores":
                var valor = "";
                for ( var j = 0; j < noFilho.childNodes.length; j++) {
                    if(noFilho.childNodes.item(j).nodeName=="autor"){
                        if( valor != "" ) {
                            valor+=" | ";
                        }
                        valor += noFilho.childNodes.item(j).firstChild.nodeValue;
                    }
                }
                document.getElementById(noFilho.nodeName).innerHTML = valor;

                default:
        break;
        }
```

```
103
104         }
105
106         var atributos = document.getElementById("divLivroSelecionado").attributes;
107         atributos.getNamedItem("style").nodeValue="";
108       }
109
110       </script>
111       <br>
112
113       <select id="livros" size="2" onclick="carregarLivro(this.value)">
114       </select>
115
116       <br>
117
118       <div id="divLivroSelecionado" style="visibility: hidden;"><br>
119         <table border="1" width="60%">
120           <tr>
121             <td width="20%">Nome:</td>
122             <td id="nome"></td>
123           </tr>
124           <tr>
125             <td>ISBN:</td>
126             <td id="isbn"></td>
127           </tr>
128           <tr>
129             <td>Editora:</td>
130             <td id="editora"></td>
131           </tr>
132           <tr>
133             <td>Edição:</td>
134             <td id="edicao"></td>
135           </tr>
136           <tr>
137             <td>Autor(es):</td>
138             <td id="autores"></td>
139           </tr>
140           <tr>
141             <td>Preço:</td>
142             <td id="preco"></td>
143           </tr>
144           <tr>
145             <td colspan="2">Sinopse:</td>
146           </tr>
147           <tr height="100">
148             <td colspan="2" id="sinopse"></td>
149           </tr>
150         </table>
151       </div>
152     </body>
153 </html>
```

Lembre de adicionar na pasta WebContent\WEB-INF\lib o arquivo jstl-1.2.jar que pode ser obtido em http://www.java2s.com/Code/JarDownload/jstl/jstl-1.2.jar.zip (descompacte o ZIP antes de adicionar o arquivo na pasta). Sempre teste os exemplo em um navegador Web recente e moderno como o Firefox, Internet Explorer, Google Chrome, etc. Alguns exemplos com muito Javascript pode não funcionar bem no navegador embutido no Eclipse.

Código passo a passo

Nesta seção, o código do **Exemplo 1** será explicado em detalhes, principalmente quando algo novo, em relação às aulas anteriores, for introduzido.

Função onLoad:

```
1  function onLoad(){
2
3    var ajaxReq = new XMLHttpRequest();
4    var url = "/Aula06/livros.xml";
5
6    ajaxReq.onreadystatechange = function() {
7      if (ajaxReq.readyState == 4 && ajaxReq.status == 200) {
8        xmlDoc = ajaxReq.responseXML;
9        carregarLista();
10     }
11   };
12
13   ajaxReq.open("GET", url, true);
14   ajaxReq.send();
15
16 }
```

Nesse trecho de código, praticamente não tem nenhuma novidade em relação ao conteúdo visto nas aulas passadas. A única novidade introduzida está na linha 26, que, ao invés de utilizar o atributo respondeText, utiliza o responseXML e salva uma referência do mesmo na variável xmlDoc para uso posterior.

Função carregarLista:

```
function carregarLista(){

  var livros = xmlDoc.getElementsByTagName("livro");
  for ( var i = 0; i < livros.length; i++) {

    var livro = livros.item(i);

    var nome="";
    var isbn="";

    if(livro.hasAttributes()){
      nome = livro.attributes.getNamedItem("nome").nodeValue;
      document.getElementById("nome").innerHTML = nome;
      isbn = livro.attributes.getNamedItem("isbn").nodeValue;
      document.getElementById("isbn").innerHTML = isbn;
    }

    var campoLivro = document.getElementById("livros");
    campoLivro.options[campoLivro.length] = new Option(nome, ""+i);

  }

}
```

A função contida no trecho acima é responsável por carregar a lista que exibe os nomes dos livros contidos no documento XML enviado pelo servidor. Nesse trecho, foram introduzidas algumas funções da API do XML DOM, que serão explicadas a seguir:

```
var livros = xmlDoc.getElementsByTagName("livro");
```

O método getElementsByTagName() retorna um array com todos os nós do tipo informado como parâmetro. Observe que esse método é utilizado por meio da referência do objeto XML DOM.

```
for ( var i = 0; i < livros.length; i++) {
```

Como você já deve saber, o atributo ength informa a quantidade de elementos do array.

```
var livro = livros.item(i)
```

O método item retorna o objeto contido na posição informada como parâmetro. Ele também pode ser acessado da maneira convencional dos arrays, utilizando chaves (ex.: livros[i]).

```
1 if(livro.hasAttributes()){
```

O método hasAttributes() indica se o nó (elemento) em questão possui ou não atributos. Esse método é definido apenas para elementos do tipo nó. O seu uso é importante, pois, caso o nó em questão não tenha atributos, a propriedade “attributes” (explicada logo a seguir) estará nula (valor null).

```
1 nome = livro.attributes.getNamedItem("nome").nodeValue;
```

O método getNamedItem() retorna um objeto contendo informações do atributo informado, e para acessar o seu valor é necessário utilizar a propriedade nodeValue.

```
1 var campoLivro = document.getElementById("livros")
```

A linha 53 utiliza HTML DOM para recuperar uma referência da lista HTML.

As demais linhas desse método são apenas repetições dos itens já explicados até o momento.

Função carregarLivro:

```
function carregarLivro(indice){

  var livros = xmlDoc.getElementsByTagName("livro");
  var livro = livros.item(indice);

  var nome="";
  var isbn="";
  if(livro.hasAttributes()){
    nome = livro.attributes.getNamedItem("nome").nodeValue;
    document.getElementById("nome").innerHTML = nome;
    isbn = livro.attributes.getNamedItem("isbn").nodeValue;
    document.getElementById("isbn").innerHTML = isbn;
  }

  var nosFilhos = livro.childNodes;
  for ( var i = 0; i < nosFilhos.length; i++) {
    var noFilho = nosFilhos.item(i);

    switch (noFilho.nodeName) {
    case "editora":
    case "edicao":
    case "preco":
    case "sinopse":
      var valor = noFilho.firstChild.nodeValue;
      document.getElementById(noFilho.nodeName).innerHTML = valor
      break;
    case "autores":
      var valor = "";
      for ( var j = 0; j < noFilho.childNodes.length; j++) {
        if(noFilho.childNodes.item(j).nodeName=="autor"){
          if( valor != "" ) {
            valor+=" | ";
          }
          valor += noFilho.childNodes.item(j).firstChild.nodeValue;
        }
      }
      document.getElementById(noFilho.nodeName).innerHTML = valor;

    default:
      break;
    }

  }

  var atributos = document.getElementById("divLivroSelecionado").attributes;
  atributos.getNamedItem("style").nodeValue="";
}
```

Essa função atualiza a tabela contendo os dados do livro selecionado. Ela recebe como parâmetro o índice do nó (tag livro) a ser carregado. As funções do XML DOM introduzidas nesse trecho são:

```
1 var nosFilhos = livro.childNodes
```

O atributo childNodes retorna todos os nós filhos do nó em questão. Como no exemplo do código acima, o nó em questão é do tipo “livro”, então, os nós filhos retornados são: editora, edicao, autores, preco e sinopse.

```
1 var valor = noFilho.firstChild.nodeValue;
```

Quando o nó em questão contém apenas texto (como no caso dos nós editora, edicao, preco, sinopse e autor), para acessar o seu valor, é necessário utilizar os atributos firstChild.nodeValue<, pois o texto contido dentro do nó também é considerado com um nó (do tipo texto).

Atividade 01

1. Altere o exemplo final da Aula 05 para que quando o usuário selecionar um livro apareça uma tabela contendo as informações detalhadas do livro selecionado (ao invés de exibir apenas um campo texto com informações). A resposta enviada pelo servidor deve estar em XML, como o seguinte formato:

```
1  < ?xml version="1.0" encoding="ISO-8859-1"?>
2  <livro>
3      <nome>nome</nome>
4      <isbn>nome</isbn>
5      <editora>editora</editora>
6      <edicao>edicao</edicao>
7      <autores>
8          <autor>autor1</autor>
9          <autor>autor2</autor>
10         <autor>autor3</autor>
11         <autor>autorN</autor>
12     </autores>
13     <preco>preco</preco>
14     <sinopse></sinopse>
15 </livro>
```

JSON

JSON (Acrônimo para *JavaScript Object Notation*, ou seja, notação de objetos JavaScript) é uma subparte da linguagem/especificação JavaScript para representação de dados. Como veremos logo a seguir, JSON é extremamente simples e fácil de utilizar. Por ser parte da própria linguagem, JavaScript não requer nenhum recurso adicional.

JSON e XML ompartilham uma série de características em comum: são legíveis e editáveis em um editor de texto comum (são textuais) e também são independentes de linguagem e plataforma. No entanto, como veremos a seguir, JSON é mais leve e mais prático de se editar manualmente.

Vídeo 03 - JSON

JSON define os seguintes elementos:

1. **Objetos/Containers JSON**

   São definidos/delimitados por chaves ({}).

   Exemplo:

   var objetoJSON = { ... }

2. **Atributos de objetos**

   São definidos dentro do objeto JSON o qual pertencem, são separados por vírgulas e possuem o seguinte formato: nome do atributo seguido por dois pontos (:) e o valor do atributo.

   Exemplo:

```
var objetoJSON = {
atributo1: "valor",
atributo2: "valor",
atributo3: "valor"
}
```

3. **Valores de atributos**

   Podem ser objetos JSON (item 1), array, texto, número ou null.

   1. Valores tipo texto:

      São definidos/delimitados por aspas duplas.

      Exemplo:

      ```
      var objetoJSON = {
      atributo1: "valor tipo texto",
      }
      ```

   2. Valores numéricos

      São números inteiros ou decimais.

      Exemplo:

      ```
      var objetoJSON = {
      atributo1: 10,
      atributo2: 10.5
      }
      ```

   3. Valores do tipo array

      São definidos por colchetes ([]), seus elementos são separados por vírgulas e podem conter todos os tipos de dados (inclusive do próprio tipo (array) e objetos JSON).

      Exemplo:

      ```
      var objetoJSON = {
      atributo1: "valor",
      array1: ["Marcelo",  "Maria", "Patricia"],
      array2: ["string",  10, 10.5, null ],
      ```

```
array3: ["string", 10, ["um array dentro de outro array", "segundo elemento"] ],
array4: ["Marcelo",  {nome:"marcelo"}],
atributo3: "valor"
}
```

Link interessante:

Conversor de XML para JSON: <http://www.thomasfrank.se/xml_to_json.html>

JSON/AJAX

Para utilizar JSON em conjunto com AJAX, basta utilizar o método “eval()” da API do JavaScript para interpretar/processar uma String, que, basicamente, é o texto entre parênteses contido no atributo responseText do objeto XMLHTTPRequest.

Vídeo 03 - JSON Exemplo

Exemplo:

```
1 ajaxReq.onreadystatechange = function() {
2     if (ajaxReq.readyState == 4 && ajaxReq.status == 200) {
3         jsonObj = eval(    '(' + ajaxReq.responseText + ')'   );
4     }
5 }
```

Atividade 02

1. Converta o Exemplo 1 dessa aula para utilizar JSON e faça com que o resultado visual seja o mesmo.

Conclusão

Ambos os formatos vistos neste material são amplamente utilizados em sistemas atuais. Os dois oferecem uma série de vantagens em relação a possíveis formatos proprietários/específicos. Ambos são em formatos de texto, logo, podem ser visualizados e editados em qualquer editor de texto. São formatados padrões, que não possuem dependência de plataforma ou linguagem. Apesar de os dois formatos servirem perfeitamente para utilização com AJAX, ultimamente, o JSON está ganhando mais força, pois, além de possuir uma API/sintaxe muito mais fácil e simples, é mais leve e menos cansativo de se editar manualmente.

Resumo

Nesta aula, você aprendeu duas maneiras diferentes de enviar e receber dados do servidor a partir do AJAX. Esses modelos foram o XML e JSON. Ambos modelos são bem populares e permitem o transporte de objetos complexos de maneira bem definida. Cada uma dessas abordagens possui vantagens e desvantagens, por isso, resta a você decidir em que situações é melhor utilizar uma ou outra abordagem.

Autoavaliação

1. Faça um resumo sobre o formato XML e como ele pode ser usado via AJAX.
2. Faça um resumo sobre o formato JSON e como ele pode ser usado via AJAX.
3. Baseado na analise dos exemplos e atividades desta aula, apresente sua opinião sobre qual dos dois formatos é melhor, XML ou JSON. Justifique sua resposta.

Referências

CONVERSOR de XML para JSON. Disponível em: <http://www.thomasfrank.se/xml_to_json.html>. Acesso em: 31 jul. 2012.

ESPECIFICAÇÃO formal do JSON (RFC 4627). Disponível em: <http://tools.ietf.org/html/rfc4627>. Acesso em: 31 jul. 2012.

EXEMPLO de processamento de XML em AJAX. Disponível em: <http://www.captain.at/howto-ajax-process-xml.php>. Acesso em: 31 jul. 2012.

JSON.org. Disponível em: <http://www.json.org/>. Acesso em: 31 jul. 2012.