A linguagem XML

Índice

1. Introdução ao XML

1.1.1. O que é XML ?

XML, ou *eXtended Markup Language*, é um padrão para a formatação de dados, ou seja, uma maneira de organizar informações. Os documentos XML podem ser facilmente compreendidos por programadores facilitando o desenvolvimento de aplicativos compatíveis. Todas as informações contidas no XML estão dentro de tags.

Uma tag é representada da seguinte maneira:

<NOME_DA_TAG>Dados</NOME_DA_TAG>

Note que a tag SEMPRE abre e fecha, o comando para fechar é o comando para abrir com uma barra "/" na frente. As tags sempre estão representadas por sinais "<" e ">".

Cada tipo de documento possui tipos diferentes de Tags, pois elas são definidas pelo programador, ou seja, você pode inventar suas próprias tags. Em alguns tipos padronizados de documentos como o HTML as Tags já são definidas (<TABLE><B><A><I><U><IMG>, etc..).

Um parâmetro é um atributo da TAG, ele serve para fornecer alguma informação extra a tag. O Formato dos parâmetros é o seguinte:

<NOME_DA_TAG nome_do_parametro="parametros">Dados</NOME_DA_TAG>

Aqui observamos que os parâmetros são incluídos dentro da definição da tag. Um parâmetro possui um nome e um valor.

Caso um documento XML não seja bem formado, os analisadores sintáticos serão incapazes de interpretá-lo corretamente e rejeitarão o documento. Para ser considerado um XML bem formatado podemos considerar os seguintes tópicos:

- o documento XML possui tags de fechamento em todos os elementos;

- Atributos envoltos por aspas.

1.1.2. A história do XML

O XML foi desenvolvido pelo *World Web Consortium* para trazer para a Web uma forma simples de vencer as limitações inerentes do HTML e permitir novos tipos de aplicações para a Internet. O XML é um padrão de armazenamento de dados em um formato de texto simples, o que significa que ele também pode ser aberto em qualquer computador.

1.1.3. O que o XML faz?

O XML é uma metalinguagem que define as regras para criar as linguagens de "markup" para codificar exemplos de documentos particulares ou tipos de mensagens. A especificação formal para qualquer linguagem "markup" definida utilizando XML ou SGML chama-se Definição do Documento Tipo ou DTD.

1.1.4. Qual é a diferença entre XML e HTML?

Para contrastar com o XML, o HTML é uma linguagem "markup" específica que contem um conjunto de elementos e funções fixas. O HTML tem um repertório limitado tal como: cabeçalhos, listas e ligações, algumas tarefas para codificar informação formatada como atributos de textos e layout, e muito poucas para codificar conteúdos de tipo de informações. Esta decisão de Tim Berners-Lee, o inventor da Web, foi a escolha correta porque faz com que se compreenda e implemente facilmente o HTML, conseguindo a sua rápida adaptação. A idéia de nomear as informações em pleno texto com o conteúdo "entre tags" é altamente intuitiva.

Além disso, enquanto o HTML pode ser descrito utilizando um DTD, a maioria do HTML na Web é inválida. Conjuntamente, as limitações fundamentais do HTML e a utilização típica sem validação tornam isto difícil para os motores de busca e processos automáticos para explorar a informação da Web devido à falta de codificação semântica.

O XML pode resolver estes problemas com o HTML e dar à Web uma capacidade muito mais forte para o comércio eletrônico. O XML torna isto possível para codificar informação com uma estrutura significativa e com semântica de anotações muito acessíveis que tem leitura tanto por humanos como através de computadores. Enquanto o XML 1.0 não traz novos modelos capazes, além dos que estão disponíveis em SGML para além de uma década, a sintaxe simples XML torna isto muito mais fácil e a participação de não-especialistas no desenho de novas linguagens "markup"

1.1.5. Benefícios da utilização do XML

O XML tem por objetivo trazer flexibilidade e poder às aplicações Web. Dentre os benefícios para desenvolvedores e usuários temos:

Buscas mais eficientes

Os dados em XML podem ser unicamente "etiquetados", o que permite que, por exemplo, uma busca por livros seja feita em função do nome do autor. Atualmente, uma busca com o nome do autor poderia levar a qualquer site que tivesse referência a tal nome, não importando se fosse o autor do livro ou simplesmente um livro sobre o autor. Sem o XML é necessário para a aplicação de procura saber como é esquematizado e construído cada banco de dados que armazena os dados de interesse, o que é impossível. O XML permitiria definir livros por autor, título, assunto, etc., o que facilitaria enormemente a busca.

Desenvolvimento de aplicações flexíveis para a Web

O desenvolvimento de aplicações Web em três camadas, ou three-tier, é altamente factível com o XML. Os dados XML podem ser distribuídos para as aplicações, objetos ou servidores intermediários para processamento. Esses mesmos dados também podem ser distribuídos para o desktop (PC e similares) para ser visualizado em um navegador.

Integração de dados de fontes diferentes

Atualmente é praticamente impossível a procura em múltiplos bancos de dados devido a incompatibilidade. O XML permite que tais dados possam ser facilmente combinados. Essa combinação seria feita via software em um servidor intermediário, estando os bancos de dados na extremidade da rede. Os dados poderiam ser distribuídos para outros servidores ou clientes para que fizessem o processamento e ainda, agregarem a distribuição.

Computação e manipulação locais

Os dados XML recebidos por um cliente são analisados e podem ser editados e manipulados de acordo com o interesse do usuário. Ao contrário de somente visualizar os dados, os usuários podem manipulá-los de várias formas. Os recursos disponíveis do *Document Object Model* (DOM) permitem que os dados sejam manipulados via scripts ou outra linguagem de programação.

A separação da interface visual dos dados propriamente ditos permite a criação de aplicações mais poderosas, simples e flexíveis.

Múltiplas formas de visualizar os dados

Os dados recebidos por um usuário podem ser visualizados de diferentes formas uma vez que o XML define somente os dados e não o visual. A interpretação visual poderia ser dada de várias maneiras diferentes, de acordo com as aplicações. Os recursos de CSS permitem essas formas particulares de visualização.

Atualizações granulares dos documentos

Os dados podem ser atualizados de forma granular, evitando que uma pequena modificação no conjunto de dados implique na busca do documento inteiro novamente. Dessa forma, somente os elementos modificados seriam enviados pelo servidor para o cliente. Atualmente, uma modificação em um item de dados acarreta na necessidade de atualização da página inteira.

O XML também permite que novos dados sejam adicionados aos já existentes, sem a necessidade de reconstrução da página.

Fácil distribuição na Web

Assim como o HTML, o XML, por ser um formato baseado em texto aberto, pode ser distribuído via HTTP sem necessidade de modificações nas redes existentes.

Escalabilidade

Devido ao fato dos documentos XML separarem completamente os dados da forma com a qual são visualizados, autores de aplicações de visualização de dados podem torná-las muito poderosas e interativas, permitindo ao usuário visualizar os dados da forma que lhe agrade. Dessa forma, a interatividade, em termos, não dependeria tanto da comunicação cliente servidor, mas sim seria feita "*offline*", reduzindo o tráfego do link com o servidor.

Fácil Compressão

A compressão de documentos XML é fácil devido à natureza repetitiva das tags usadas para definir a estrutura dos dados. A necessidade de compressão é dependente da aplicação e da quantidade de dados a serem movidos entre clientes e servidores.

1.1.6. Definição Conceitual

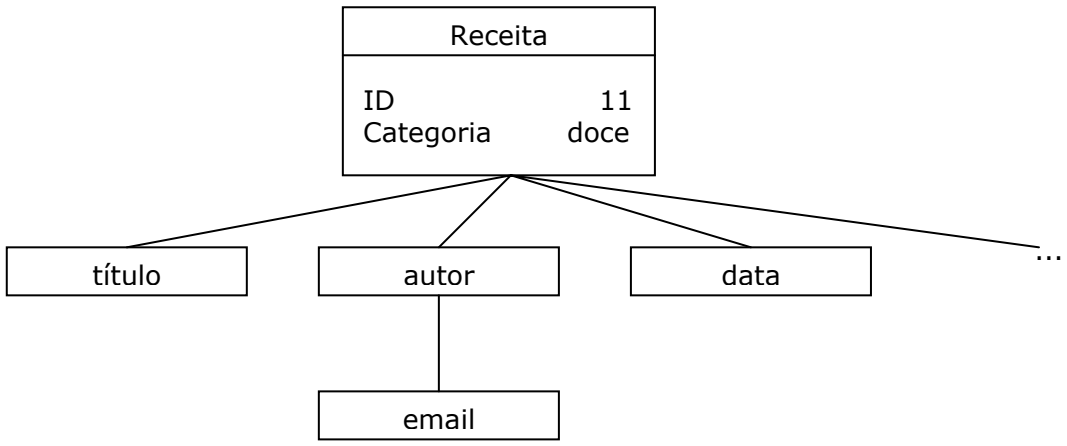
Um documento XML é uma árvore rotulada onde um nó externo consiste de:

· dados de caracteres (uma seqüência de texto)
· instruções de processamento (anotações para os processadores), tipicamente no cabeçalho do documento
· um comentário (nunca com semântica acompanhando)
· uma declaração de entidade (simples macros)
· nós DTD (*Document Type Declaration*)

Um nó interno é um elemento, o qual é rotulado com:
um nome ou

.um conjunto de atributos, cada qual consistindo de um nome e um valor.

Normalmente, comentários, declarações de entidades e informações DTD não são explicitamente representadas na árvore.



1.1.7. Uma visão prática das tags

Um documento XML é um texto (em formato *Unicode*) com tags de marcação (markup tags) e outras informações.
As markup tags denotam a seguinte estrutura:

```
...<nome_tag atributo="valor" ...>...</nome_tag>...
```

Atenção: os documentos XML são sensíveis à letras maiúsculas e minúsculas.

Um documento XML é bem formatado quando segue algumas regras básicas. Tais regras são mais simples do que nos documentos HTML e permitem que os dados sejam lidos e expostos sem nenhuma descrição externa ou conhecimento do sentido dos dados XML.

Documentos bem estruturados:
· tem casamentos das tags de início e fim
· as tags de elemento tem que ser apropriadamente posicionadas

Os elementos não podem se sobrepor. Um exemplo de sobreposição é o seguinte:

```
<titulo>Descrição dos diversos modelos de carros
<sub>da marca Ford
</titulo>Alexandre Manso
</sub>
```

E, corrigindo o erro:

```
<titulo>Descrição dos diversos modelos de carros
<sub>da marca Ford</sub>
<autor> Alexandre Manso</autor>
</titulo>
```

Um comentário que será ignorado por todos os processadores.

```
<!-- comentário -->
```

2. Integrando Flash e XML

```
<?xml version="1.0" encoding="iso-8859-1"?>
<Estudando>
    <XML id="Flash no Atributo id"/>
    <XML>Flash dentro da TAG XML</XML>
</Estudando>
```

Acima temos um pequeno exemplo do XML. Na 1ª linha temos a versão do XML e o *enconding* que habilita os acentos, dessa forma pode-se usar acentos nas informações das Tags.

Perceba que a linha 3 é diferente da linha 4. Por quê? Podemos declarar apenas atributos na tag XML ao invés de <Abrir> Escrever e </Fechar>. Muito mais simples escrever <Abrir escrever="e fechar"/>, mas isso vai de acordo com a aplicação.

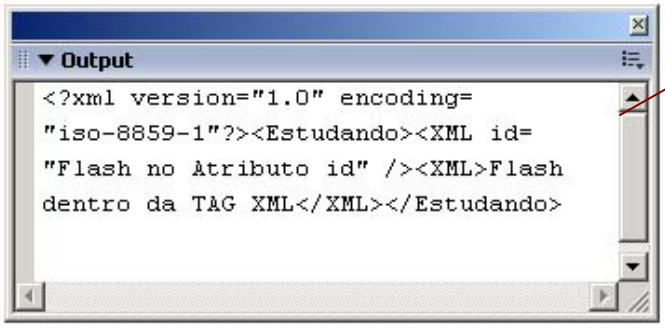
Escreva o XML acima no bloco de notas e salve o arquivo com o nome de *iniciando_xml.xml*.

Agora no Flash crie um arquivo, salve na mesma pasta em que salvou o documento XML, com o nome de *iniciando_xml.fla* e em seguida abra o painel de Ações (pressione F9 ou vá em Window->Development Panels->Actions).

Não se preocupe com os códigos. A partir dessa fase inicial será mais fácil entender todos os comandos ActionScript voltados para XML.

Vamos fazer com que o Flash leia o arquivo XML e nos mostre na janela de saída:

```
/*Lê o documento com os acentos,
se o System.useCodepage for igual a false, nenhum acento será
exibido.*/
System.useCodepage = true;
var meuXML:XML = new XML(); /*Declara um novo objeto XML*/
meuXML.load("iniciando_xml.xml");/*Carrega o arquivo XML*/
meuXML.ignoreWhite = true;/*Ignora os espaços em branco do XML*/
/*Quando o XML for carregado, executa a função*/
meuXML.onLoad = function(){
   trace(this);/*mostra o conteúdo de meuXML na janela de saída*/
}
```



O script acima mostra o conteúdo do nosso arquivo XML na janela de saída.

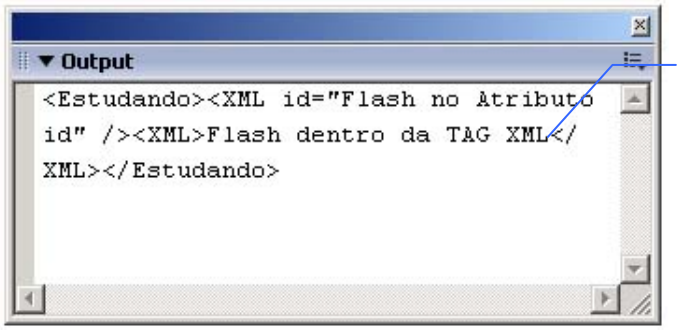
Agora vamos mostrar a 2ª linha na Janela de saída, ou seja, do 1º nó filho do XML e seu conteúdo.

```
System.useCodepage = true;
```

Este chidNodes se refere ao 1º nó filho do XML.

```
var meuXML:XML = new XML();
meuXML.load("iniciando_xml.xml");
meuXML.ignoreWhite = true;
meuXML.onLoad = function(){
   trace(this.childNodes);
}
```

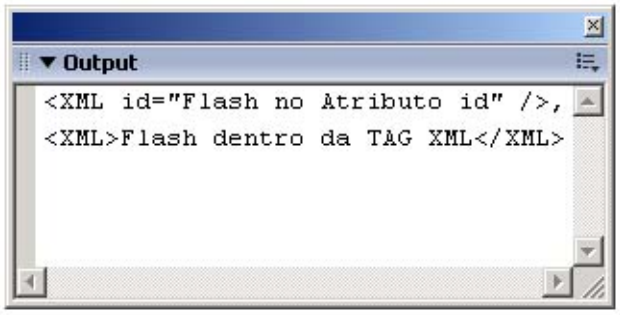
Na janela de saída aparecerá o seguinte:



Note que foi exibida apenas a tag: <XML>Flash dentro da TAG XML</XML>, ou seja, o primeiro filho.

Vamos exibir apenas os nós filhos da tag <Estudando>.

```
System.useCodepage = true;
var meuXML:XML = new XML();
meuXML.load("iniciando_xml.xml");
meuXML.ignoreWhite = true;
meuXML.onLoad = function(){
   trace(this.childNodes[0].childNodes);
}
```

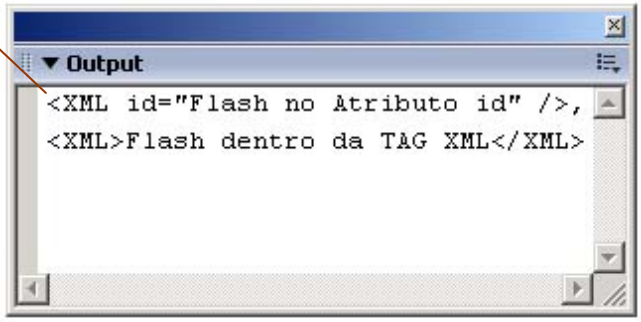


Note que depois de childNodes vem a identificação '[0]' simbolizando um array (ou vetor) que indica qual o nó que deve ser lido. Por exemplo, o primeiro filho está na posição 0 (zero), enquanto que o segundo está na posição 1. Ainda podemos utilizar o firstChild que tem o mesmo valor que childNodes[0], afinal ambos fazem referência ao 1º nó do XML.

Então poderíamos utilizar:

```
System.useCodepage = true;
var meuXML:XML = new XML();
meuXML.load("iniciando_xml.xml");
meuXML.ignoreWhite = true;
meuXML.onLoad = function(){
   trace(this.firstChild.childNodes);
}
```

Note que o resultado é identico ao anterior. Substituímos childNodes[0] por firstChild pois ambos referem-se ao primeiro nó do arquivo XML (a posição 0 é a primeira assim como first significa primeiro)



Também é possível perceber que existem mais um childNodes. Tente interpretar o código da seguinte forma:

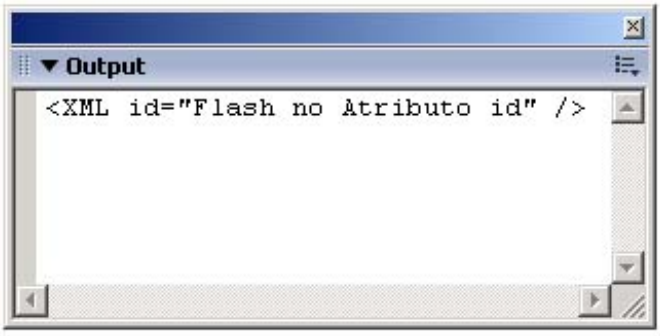
this = este é o XML, dentro dele temos o nó filho *Estudando* e os demais nós filhos.

this.childNodes[0] ou this.firstChild = este é o 1º nó filho do XML (*Estudando*).

this.childNodes[0].childNodes = estes são os nós filhos da tag Estudando.

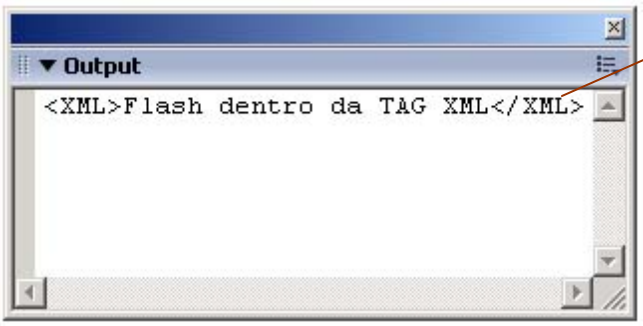
Agora vamos mostrar somente o 1º nó filho da tag *Estudando*:

```
System.useCodepage = true;
var meuXML:XML = new XML();
meuXML.load("iniciando_xml.xml");
meuXML.ignoreWhite = true;
meuXML.onLoad = function(){
   trace(this.childNodes[0].childNodes[0]);
}
```



Novamente utilizamos o conceito de array para localizarmos um nó no XML. O segundo childNodes refere-se aos nós filhos dentro da tag Estudando.

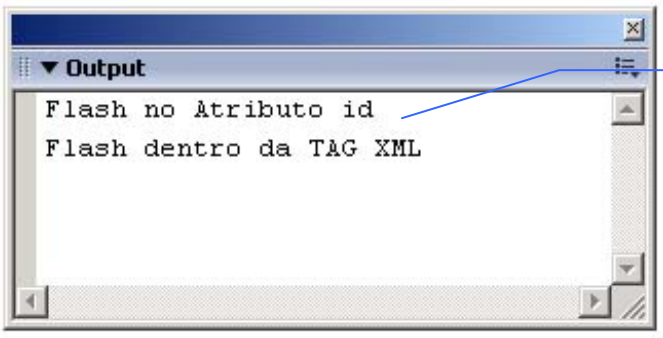
Perceba que agora o 2º childNodes tem uma array também: o [0] se refere ao 1º nó filho da tag Estudando, se você colocar [1] ao invés de [0] verá o 2º nó filho da tag Estudando.
Novo Resultado:



Segundo nó da tag <Estudando>

atributo id (primeiro nó da tag <Estudando>) e logo abaixo vamos mostrar o conteúdo do 2º nó filho da tag <Estudando>:

```
System.useCodepage = true;
var meuXML:XML = new XML();
meuXML.load("iniciando_xml.xml");
meuXML.ignoreWhite = true;
meuXML.onLoad = function(){
   trace(this.childNodes[0].childNodes[0].attributes.id);
   trace(this.childNodes[0].childNodes[1].childNodes[0].nodeValue);
}
```



Mostrou apenas os valores contidos nos nós utilizando o nome do atributo e a propriedade *nodeValue*. Note que não aparece a estrutura das tags (<XML>...)

this.childNodes[0].childNodes[0].nodeValue = indica o valor do 1º nó filho da tag Estudando que é o 1º nó do XML.

this.childNodes[0].childNodes[0].attributes.id = indica um atributo de nome "id", assim o Flash lê o que consta dentro de id.

- Para que o Flash leia todo o XML usamos apenas o this.

- Para que o Flash leia a linha 2 e o seu conteúdo usamos this.firstChild ou this.childNodes.

- Para ler a linha 3 usamos this.firstChild.firstChild ou this.childNodes[0].childNodes[0]

- Para ler o atributo "id" da linha 3 usamos this.firstChild.firstChild.attributes.id ou this.childNodes[0].childNodes[0].attributes.id

- Para ler toda a linha 4 usamos this.firstChild.childNodes[1] ou this.childNodes[0].childNodes[1]

- Para **ler o conteúdo** que está entre as tags <XML> na linha 4 usamos this.firstChild.childNodes[1].childNodes[0].nodeValue ou this.childNodes[0].childNodes[1].childNodes[0]..nodeValue

2.1.1. Classe XML

Os métodos e as propriedades do objeto XML do Flash são usados para carregar, analisar, enviar, montar e manipular árvores de documento XML. Uma instância do XML Object representa um documento XML válido ou não.

```
<CURSO nome="Flash e ActionScript">
     <PROFESSOR>
          <NOME>Barbara Franco</NOME>
          <EMAIL>barbaradefranco
```

```
                <DOMINIO>
                      yahoo.com.br
                </DOMINIO>
           </EMAIL>
      </PROFESSOR>
</CURSO>
```

Para o documento XML apresentado acima, o objeto XML do Flash irá entender <CURSO> como sendo um elemento XML, 'nome' como atributo, <NOME> como elemento, 'Barbara Franco' como nó de texto, barbaradefranco@yahoo.com.br sendo filho de <EMAIL>. e a tag <DOMINIO> é filho de <EMAIL>.

Para instanciar um objeto XML, utiliza-se o operador new e o método construtor do objeto conforme na figura abaixo.

```
objXML = new XML();
//instancia de um objeto XML vazio ou
// instancia um objeto XML conforme o parâmetro passado.
strXML = "<PAI><FILHO>Este é um nó filho</FILHO></PAI>";
objXML = new XML(strXML);
```

Não se esqueça que os comandos em **ActionScript** devem ser implementados na painel Actions.

2.1.2. Propriedades do objeto XML

A seguir, serão apresentadas as principais propriedades do objeto XML do Flash.

contentType

Indica o tipo de MIME (Multipurpose Internet Mail Extensions) transmitido para o servidor quando o método XML.send ou XML.sendAndLoad é chamado. O padrão é application/x-www-form-urlencoded.

docTypeDecl

Define e retorna informações sobre a declaração DOCTYPE do documento XML. O analisador XML do ActionScript não é um analisador de validação, a declaração DOCTYPE é lida pelo analisador e armazenada na propriedade docTypeDecl, mas nenhuma validação DTD é executada. Esta propriedade não é armazenada no objXML como um nó, mas apenas como uma seqüência de caracteres.

Exemplo:
objXML.docTypeDecl="<!DOCTYPE CURSO SYSTEM \"curso.dtd\">"

Vale esclarecer que no ActionScript, para adicionar aspas duplas(") numa string utiliza-se a codificação barra-aspas (\"). Se o conteúdo de objXML.docTypeDecl fosse exibido ao usuário, a saída seria <!DOCTYPE CURSO SYSTEM "curso.dtd">.

xmlDecl

Define e retorna informações sobre uma declaração de um documento XML. Depois de o documento XML ser analisado em um objeto XML do Flash, essa propriedade é definida como o texto da declaração XML do documento e não de um

objeto do nó XML. Ainda que um determinado documento XML possua uma declaração, esta será desconsiderada pelo objeto XML do Flash. Esta declaração será tratada como uma string que pode ser recuperada, mas não tem efeito sobre a apresentação do documento e não será considerada num processo de navegação pelo documento.

nodeType

Retorna um valor inteiro informando se o nó especificado é um elemento XML (nodeType = 1) ou um nó de texto (nodeType = 3).

nodeName

Retorna o nome da marca de um elemento XML. Se o nó for um nó de texto (nodeType= 3) o valor de retorno será null.

nodeValue

Retorna o texto do nó especificado se o nó for um nó de texto. Se o nó especificado for um elemento XML será retornado null.

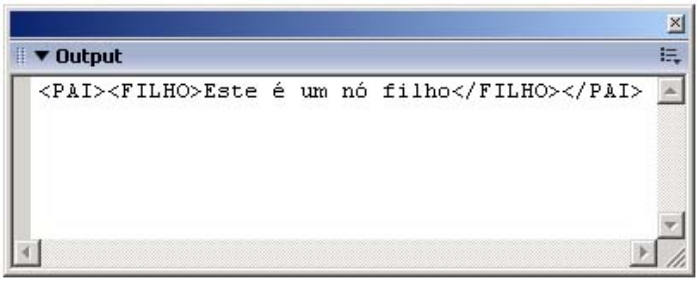
firstChild

Avalia o objeto XML especificado e faz referência ao primeiro filho na lista de filhos do nó pai. Essa propriedade é null se o nó não tiver filhos e indefinida se o nó for um nó de texto. O código abaixo demonstra o conceito de firstChild.

```
filho = objXML.firstChild;
trace(filho);
```

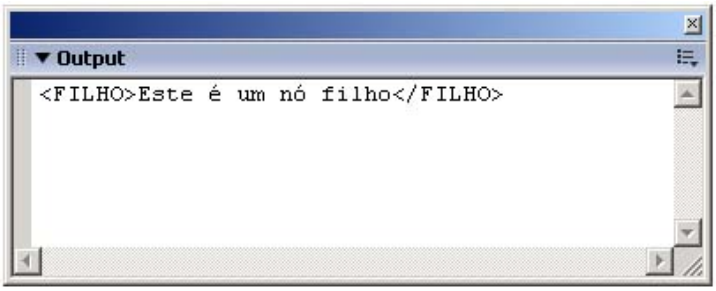
Não descarte o código anterior pois iremos utilizar o código XML

Note que o primeiro nó ou firstChild é <PAI>, pois este se encontra no primeiro nível, como no código XML criado anteriormente.



Se adicionarmos firstChild após filho, vamos encontrar a tag <FILHO>.

```
trace(filho.firstChild);
```



lastChild

Faz referência ao último filho na lista do nó especificado.

parentNode

Faz referência ao nó pai do nó especificado. Exemplo:

```
objPai = objFilho.firstChild.parentNode;
trace(objPai.nodeName);
```

A propriedade nodeName mostra apenas o nome da tag e não toda a sua estrutura. Neste caso, quando utilizarmos a função na tag <PAI><FILHO>...</FILHO><PAI>, vamos visualizar somente o nome da tag que é PAI.

Neste caso, vamos visualizar na tela de Output a tag <PAI>. Note que foi utlizado a propriedade firstChild foi utilizada duas vezes: a primeira chamada do firstChild nos leva ao primeiro nível do XML <PAI>; e na segunda chamada, na taga <FILHO>. O parentNode vai nos dizer qual é o nível mais alto de <FILHO>.

nextSibling

Avalia o objeto XML e faz referência ao próximo irmão na lista de filhos do nó pai. Esse método retorna null se o nó não tiver um nó irmão próximo.

previousSibling

Faz referência ao irmão anterior na lista de filhos do nó pai.

status

Retorna automaticamente um valor numérico (inteiro) que indica se um documento XML foi analisado com êxito em um objeto XML. Abaixo estão listados os possíveis valores de retorno com suas respectivas descrições:

`status = 0`: a análise foi concluída com êxito;
`status = -2`: uma seção CDATA não foi terminada adequadamente;
`status = -3`: a declaração XML não foi terminada adequadamente;
`status = -4`: a declaração DOCTYPE não foi terminada adequadamente;
`status = -5`: um comentário não foi terminado adequadamente;

`status = -6`: um elemento XML foi mal formado;
`status = -7`: memória insuficiente.
`status = -8`: um valor de atributo não foi terminado adequadamente;
`status = -9`: uma marca de início não correspondeu a uma marca de fim;
`status = -10`: foi encontrada uma marca de fim sem uma marca de início correspondente.

```
//A marca de fim </PROFESSOR> está ausente
objXML = new XML("<PROFESSOR><NOME>Barbara</NOME>");
if(objXML.status == 0)
trace("Documento bem formado");
else
trace("Falha no documento. Erro: " + objXML.status);
```

A figura acima ilustra a utilização da propriedade *status* do objeto XML do Flash em que a tag final </PROFESSOR> não é encontrada. Logo, o documento não está bem formado e o Flash acusa o erro, no caso, status=9.

2.1.3. Coleções

O objeto XML do Flash disponibiliza duas coleções que facilitam a navegação pelo objeto. Estas coleções são: attributes e childNodes.

attributes (leitura e gravação)

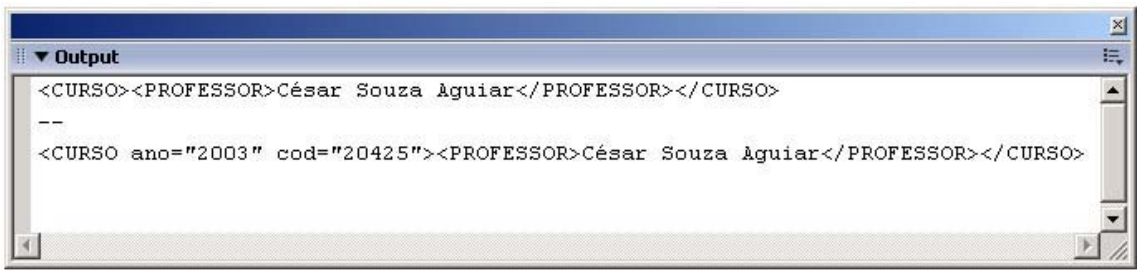
Retorna um vetor associativo que contém todos os atributos do objeto XML especificado. Todos os atributos de um determinado elemento XML podem ser listados a partir da leitura do array ou vetor.

A coleção attributes possibilita a adição de novos atributos a um determinado elemento XML. Para adicionar um atributo, basta especificar (objXML.fisrtChild, por exemplo) em qual elemento os atributos serão adicionados.

Em seguida, "deve-se informar a coleção attributes antecedida de "." e o nome do atributo que se deseja adicionar, atribuindo-lhe determinado valor. A figura abaixo exemplifica a adição de atributos.

```
objXML = new XML("<CURSO><PROFESSOR>César Souza Aguiar" +
"</PROFESSOR></CURSO>");
trace(objXML.toString());
objXML.firstChild.attributes.cod = "20425";
objXML.firstChild.attributes.ano = "2003";
trace("--");
trace(objXML.toString());
```

Estamos adicionando atributos na tag <CURSO>



childNodes (somente leitura)

Retorna um vetor dos filhos do objeto XML especificado. Cada elemento no vetor é uma referência a um objeto XML que representa um nó filho. Da mesma forma que a coleção attributes, todos os filhos de um determinado elemento XML podem ser listados a partir do vetor. No entanto, a coleção childNodes não permite a adição ou remoção de elementos.

2.1.4. Métodos

A seguir, são apresentados alguns métodos do objeto XML do Flash para navegação, análise e validação de documentos XML.

appendChild

Anexa um nó no fim da lista filha do objeto especificado. O nó a ser adicionado deve ser um outro objeto XML. Este método retorna nada.

```
filho = new XML("<NO>Novo No</NO>");
objXML.appendChild(filho);
```

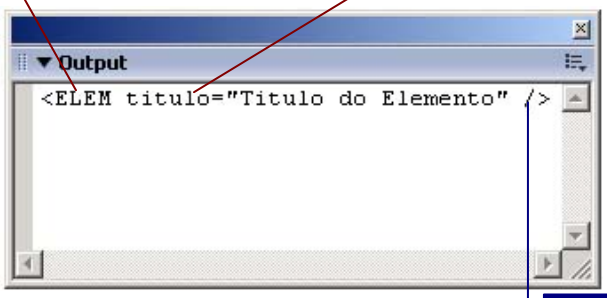
Utilize para esse exemplo o código anterior.

createElement

Cria e retorna um novo elemento XML.

```
// objeto XML criado anteriormente
//cria um novo elemento chamado ELEM
E = objXML.createElement("ELEM");
//adiciona o atributo titulo ao elemento criado
E.attributes.titulo = "Titulo do Elemento";
```

Nome informado no parâmetro do método createElement



Atributo da tag <ELEM>

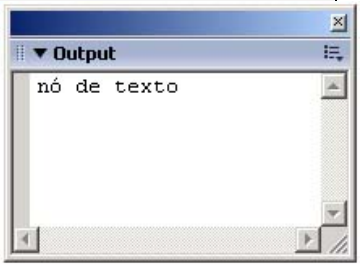
Note a barra no final da tag: com o seu uso não é necessário criar a tag final, ou melhor </ELEM>

Depois de criado, sendo o elemento um elemento XML válido, o mesmo pode ser adicionado ao objXML usando o método appendChild ou insertBefore.

createTextNode

Cria um novo nó de texto XML. Este método retorna uma referência ao nó de texto criado.

```
noTexto = new XML();
texto = noTexto.createTextNode("nó de texto")
trace(texto.nodeValue);
```



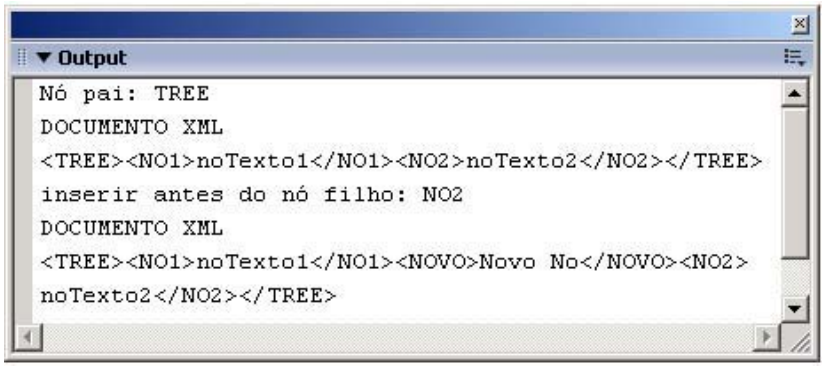
hasChildNodes

Retorna true se o nó especificado possuir nós filhos; caso contrário retorna false.

insertBefore

Insere um nó XML antes do nó filho especificado. Se o nó filho for indefinido ou null, o nó será inserido usando o método appendChild. Se o nó filho especificado não for um nó da referida instância do objeto XML, será retornado um erro.

```
objXML = new XML("<TREE><NO1>noTexto1</NO1>" +
"<NO2>noTexto2</NO2></TREE>");
objXML = objXML.firstChild; //referencia TREE
trace("Nó pai: " + objXML.nodeName);
trace("DOCUMENTO XML");
trace(objXML.toString());
novo = new XML("<NOVO>Novo No</NOVO>");
ant = objXML.firstChild.nextSibling; //referencia a NO2
trace("inserir antes do nó filho: "+ant.nodeName); //NO2
objXML.insertBefore(novo,ant);
trace("DOCUMENTO XML");
trace(objXML.toString());
```



load

Carrega um documento XML a partir de uma URL. O documento XML carregado é analisado pelo objeto XML do Flash. A verificação desta análise pode ser obtida com a propriedade status.

send

Envia o objeto XML especificado para uma URL utilizando o método POST. Opcionalmente, pode-se definir a janela para onde o documento XML será enviado/tratado. A sintaxe do método *send* é: objXML.send(strURL,window) em que strURL é um endereço URL válido e *window* é a janela destino que pode ser definida como:

`_self` – frame atual na janela atual;
`_blank` – nova janela;
`_top` – frame de alto nível na janela atual ou;
`_parent` – pai do quadro atual.

sendAndLoad

Codifica objeto XML especificado em um documento XML, envia-o para a URL especificada usando o método POST, faz o download da resposta do servidor e a carrega no objeto objXMLDest especificado nos parâmetros. A sintaxe deste método é:

objXML.sendAndLoad(strURL,objXMLDest), em que strURL é uma URL válida e objXMLDest, uma instância do objeto XML e pode ser o mesmo objeto que invocou o método, por exemplo, **objXML.sendAndLoad(strURL,objXML)**.

getBytesTotal
Retorna o tamanho do documento XML em bytes.

parseXML
Analisa o texto XML especificado no parâmetro origem e preenche o objeto XML com a árvore XML resultante. Quaisquer árvores existentes no objeto XML são descartadas. A validade da árvore resultante pode ser verificada com a propriedade status do objeto XML.

removeNode
Remove o nó especificado (e seus filhos) de seu pai.

2.1.5. Identificadores de eventos

O XML Object possui alguns identificadores de eventos que auxiliam o usuário, por exemplo, no acompanhamento do processo de download de um documento XML disponível em um servidor.

onData
É chamado quando o download de um texto XML foi concluído (ou quando ocorre um erro ao fazer o download). Esse identificador é chamado antes de o XML ser analisado e, portanto, pode ser usado para chamar uma rotina de análise personalizada. Retorna o respectivo documento XML e, caso ocorra um erro durante o processo de download, será retornado *undefined*.

Por padrão, o método onData invoca o método onLoad, mas se for redefinido pelo usuário, o método onLoad não será invocado, a menos que sua invocação seja explicitada na implementação customizada de onData. O exemplo abaixo exemplifica o método onData.

onLoad
É invocado quando um documento XML é recebido do servidor. Este método não possui implementação. Para utilizá-lo, deve-se implementar uma função que realize o processamento desejado.

Sintaxe:
objXML.onLoad(doc), onde *doc* é um valor booleano que indica se o objeto XML foi carregado com êxito por meio de uma operação XML.load ou XML.sendAndLoad.

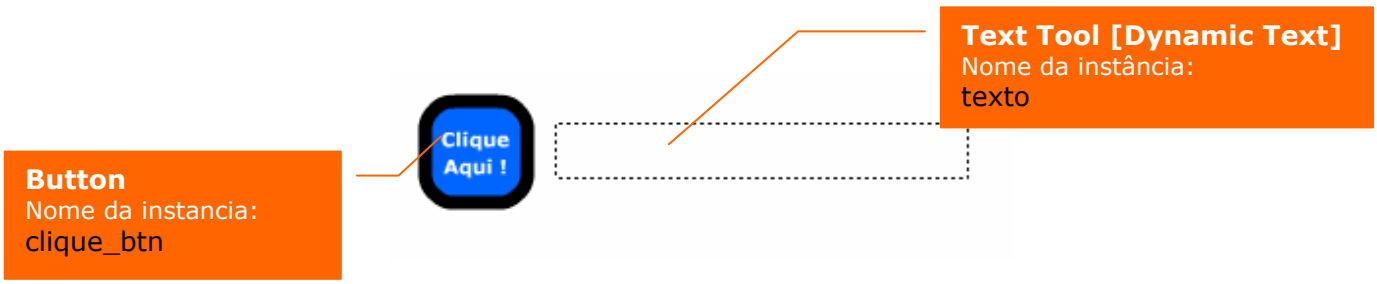
3. Conectando-se ao XML – Exercício Prático

Esse exercício tem por finalidade buscar uma informação que está em um arquivo XML. Crie uma pasta com o nome passo_a_passo_xml.

Primeiro vamos criar um arquivo XML simples. Salve-o como doc_xml.xml na pasta criada anteriormente.

```
<?xml version="1.0" encoding="iso-8859-1"?>
<texto>
      <mensagem>Olá, bem vindo ao mundo Flash e XML!</ mensagem>
</texto>
```

Abra o Flash. Vamos criar a interface da nossa aplicação. Nomeie a primeira layer do Timeline como Figuras e crie uma nova com o nome de Actions.



Salve o seu documento FLA na mesma pasta que o documento XML. Nomeie como flash_xml.fla.

Clique na layer Actions e selecione o primeiro frame do Timeline. Pressione a tecla F9 para abrir o painel Actions.

```
stop();
if (Xml == null) {
      Xml = new XML();
      Xml.ignoreWhite = true;
      Xml.load("doc_xml.xml");
}

Xml.onLoad=function(){
      trace('O documento XML foi carregado com sucesso')
}

clique_btn.onRelease = function() {
      texto.text =
Xml.firstChild.childNodes[0].childNodes[0].nodeValue;
};
```

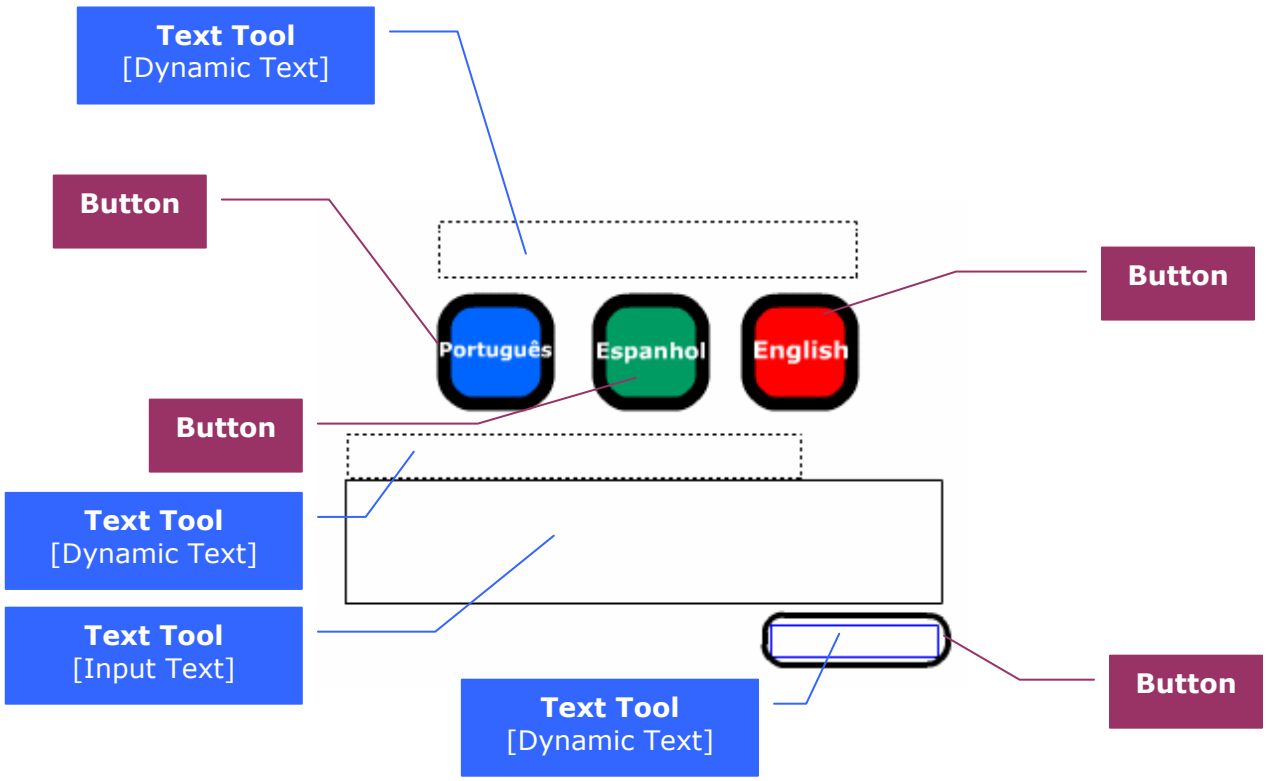
Teste a aplicação e altere as configurações do campo texto caso seja necessário.

Exercício

```
<?xml version="1.0" encoding="UTF-8"?>
<animation>
 <screens>
  <screen index="1">
   <texts>
    <text>Bem Vindo!</text>
    <text>Escreva uma mensagem</text>
   </texts>
   <buttons>
    <button>Salvar</button>
   </buttons>
  </screen>
  <screen index="2">
   <texts>
    <text>Welcome!</text>
    <text>Write a message</text>
   </texts>
   <buttons>
    <button>Save</button>
   </buttons>
  </screen>
  <screen index="3">
   <texts>
```

```
    <text>Bien Viendo!</text>
    <text>Escriba un mensaje</text>
   </texts>
   <buttons>
    <button>Excepto</button>
   </buttons>
  </screen>
 </screens>
</animation>
```

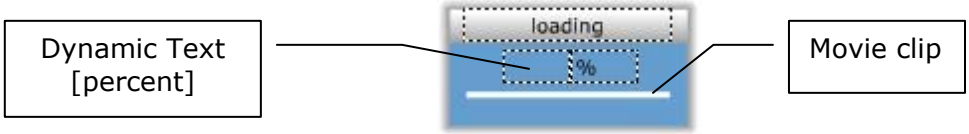
Dado o documento XML crie uma aplicação com a seguinte interface:



Observando o documento XML, para cada linguagem existem 3 textos: saudação, label e botao. O primeiro deve ser posicionado no primeiro Dynamic Text (no topo do Stage). O texto dentro da <label> deve ficar no Dynamic Text abaixo dos três botões de seleção de linguagem; e finalmente, o texto de <botao> será posicionado no Dynamic Text atrás do último botão da aplicação.
A nossa aplicação pode ser usada por qualquer pessoa, pois pode ser selecionada uma língua, como inglês, português e espanhol. A lógica é: quando você clicar no botão 'Português' os textos que irão aparecer na tela serão os que estão no primeiro nó da tag <Linguagem>.

Vamos dividir a nossa aplicação: crie outra cena (Scene). Na primeira, adicione somente o código para carregar o XML. E na segunda cena, adicione a interface e os códigos referentes.

Na primeira cena, vamos criar um pre-loader. Um pre-loader informa quantos kbytes da nossa aplicação já foram carregados.
Crie um Movie Clip com formato de barra (observe a figura abaixo).



Selecione o mesmo e abra o Painel de Actions. Insira o código abaixo:

```
onClipEvent (load) {
      total_kb = _root.getBytesTotal();
      _root.CountDown = 0;
}
onClipEvent (enterFrame) {
      p = Math.round((_root.getBytesLoaded()/total_kb)*100);
      _root.percent = p;
      scale = ((_root.getBytesLoaded()/total_kb)*100);
      this._xscale = scale;
      if (p == 100) {
            _root.nextFrame();
      }
}
```

Utilize os seus conhecimentos sobre programação orientada a objetos.

Ao finalizar seu exercício, envie para o seu tutor os arquivos que foram gerados pelo Flash (FLA e SWF). Qualquer duvida, entre no Fórum de FAQ ou envie um e-mail para seu tutor.