Cosa debbo rispondere?

From: "XY" <XY@hotmail.com>

To: <fabio@CS.UniBO.IT>

Subject: Semantic Search Engine

Date: Thu, 16 May 2002 18:05:05 +0200

Professor Vitali, mi chiamo XY e sono uno studente dell'universita' di Roma La Sapienza, facoltà di Ingegneria Informatica. Sto realizzando uno studio approfondito sul WEB SEMANTICO. Attraverso un motore di ricerca sono vennuto a conoscenza del progetto relativo al "semantic search engine". Qualora ve ne fosse la possibilità, con il suo consenso e con quello dei suoi studenti, volevo sapere se fosse possibile visionare una realizzazione del sopra citato "engine". La ringrazio per l'attenzione.

Cordialità

XY

Document Object Model

Fabio Vitali

I parser XML

I parser XML si dividono in due categorie per quel che riguarda i modelli di elaborazione

- Parser SAX (Simple API for XML): è un modello ad eventi, attraverso il quale è possibile associare funzioni callback ai vari elementi significativi del documento XML. SAX 2.0 non è uno standard, ma una proposta originariamente per JAVA poi adottata su molte architetture (http://www.saxproject.org/)
- Parser DOM (Documento Object Model): è un modello gerarchico, che fornisce accesso all'intero documento DOPO il completamento del parsing. DOM è una serie di raccomandazioni W3C implementate in varia maniera su tutte le architetture.

Elaborazione SAX

Dato il documento

```
<?xml version="1.0"?>
<book>
<chapter>
<title>Capitolo 1</title>
<p>Una frase</p>
...
</chapter>
</book>
```

Il parser SAX registrerà gli eventi:

- `startDocument()`
- `startElement(book)`
- `startElement(chapter)`
- `startElement(title)`
- `characters(Capitolo 1)`
- `endElement(title)`
- `startElement(p)`
- `character(Una frase)`
- `endElement(p)`
- `...`
- `endElement(chapter)`
- `endElement(book)`
- `endDocument()`

Vantaggi e svantaggi dei parser SAX

Vantaggi

- SAX richiede pochissima memoria d'uso: il documento non è mai tutto in memoria contemporaneamente. E' ideale per documenti MOLTO grandi.
- SAX è velocissimo: non ha praticamente overhead di parsing.
- SAX è ideale per cercare velocemente un'informazione specifica all'interno di un documento XML, poiché lo si può fermare appena l'ha trovata.

Svantaggi

- SAX non può essere usato per cercare e ricercare informazioni su un documento XML (navigare sulla struttura del documento)
- SAX non può essere usato per modificare o aggiornare la struttura del documento XML (aggiungere o togliere nodi).
- SAX non è implementato dai browser, ma solo come libreria di linguaggi server-side (o in Java anche come applet)

Elaborazione DOM

Il Document Object Model è un interfaccia di programmazione (API) per documenti sia HTML sia XML.

Definisce la struttura logica dei documenti ed il modo in cui si accede e si manipola un documento.

Utilizzando DOM i programmatori possono costruire documenti, navigare attraverso la loro struttura, e aggiungere, modificare o cancellare elementi.

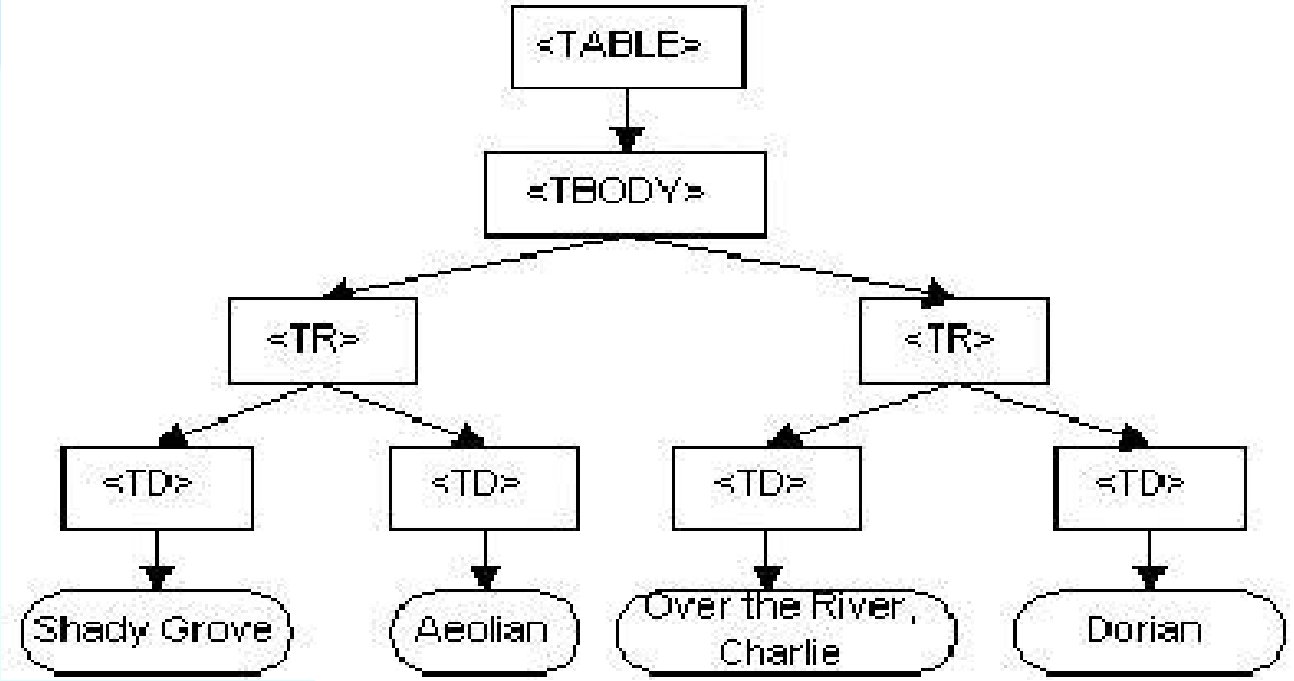
Ogni componente di un documento HTML o XML può essere letto, modificato, cancellato o aggiunto utilizzando il Document Object Model.

Elaborazione DOM (1)

```
<TABLE>
  <TBODY>
    <TR>
      <TD>Shady Grove</TD>
      <TD>Aeolian</TD>
    </TR>
    <TR>
      <TD>Over the River, Charlie</TD>
      <TD>Dorian</TD>
    </TR>
  </TBODY>
</TABLE>
```


Elaborazione DOM (2)

Il Document Object Model rappresenta la tabella della slide precedente in questo modo:



Oggetti del DOM

Le implementazioni del DOM mettono a disposizione una classe fondamentale, **DOMNode**, che fornisce metodi per accedere a tutti gli elementi di un nodo di un documento XML, inclusi il nodo radice, i nodi elemento, i nodi attributo, i nodi testo, ecc.

Spesso questa classe viene sottoclassata per specificare tipi di nodo specifici. In particolare **DOMDocument** contiene una quantità di informazioni sul documento in generale.

Usare DOM (1)

Creare un oggetto documento e parsare un file XML:

- Javascript server-side (ASP): MSXML 3.0

```
var myDoc=Server.CreateObject("msxml2.DOMDocument");
myDoc.load(xmlFile);
```

- Javascript client-side (IE): MSXML 3.0

```
var myDoc = new ActiveXObject("Msxml2.DOMDocument");
myDoc.load(xmlFile)
```

- Java (Xerces)

```
DOMParser parser = new DOMParser();
parser.parse(xmlFile);
Document myDoc = parser.getDocument();
```

- PHP

```
$myDoc = xmldocfile($xmlFile);
```


Usare DOM (2)

Navigare sul documento dopo il parsing:

- Javascript server-side (ASP): MSXML 3.0

```
var root = myDoc.documentElement
var nodes = root.childNodes
for (var x=0; x<.nodes.length; x++) { ... }
```

- Java (Xerces)

```
for(Node child = myDoc.getFirstChild();
    child != null;
    child = child.getNextSibling()) { ... }
```

- PHP

```
$root = $mydoc->root();
$nodes = $root->children();
for ($x=0; $x<sizeof($nodes); $x++) { ... }
```


Usare DOM (3)

Creare nuovo contenuto:

- Javascript server-side (ASP): MSXML 3.0

```
var doc=Server.CreateObject("msxml2.DOMDocument");
root = doc.createElement("book");
item = doc.createElement("title");
text = doc.createTextNode("Titolo 1")
item.appendChild(text);
root.appendChild(item);
doc.documentElement = root
```


Usare DOM (4)

Creare nuovo contenuto:

- Java (Xerces)

```
DocumentBuilderFactory
   dbf=DocumentBuilderFactory.newInstance();
DocumentBuilder db = dbf.newDocumentBuilder();
doc = db.newDocument();
root = doc.createElement("book");
item = doc.createElement("title");
text = doc.createTextNode("Titolo 1")
item.appendChild(text);
root.appendChild(item);
doc.appendChild(root)
```


Usare DOM (5)

Creare nuovo contenuto:

- PHP

```
$doc = domxml_new_doc("1.0");
$root = $doc->create_element("book");
$item = $doc->create_element("title");
$text = $doc->create_text_node("Titolo 1");
$newnode = $item->append_child($text);
$newnode = $root->append_child($item);
$newnode = $doc->append_child($root);
```


Usare DOM (6)

Altri modi per navigare

- Si può anche usare la funzione DOM **getElementsByTagName**. Essa restituisce una lista di nodi, indipendentemente dalla loro posizione nell'albero.

```
NodeList = myDoc.getElementsByTagName("chapter");
for (var i=0; i<NodeList.length; i++) { ... }
```

- Alternativamente, posso avere un supporto per XPath, che mi permette di selezionare non solo di selezionare tag per nome, ma anche per espressioni più complesse (non fa parte di DOM, però!):
  - Javascript:

```
NodeList = myDoc.selectNodes("//chapter[@id]");
```

  - PHP:

```
$ctx=xpath_new_context($mydoc);
$foo=xpath_eval($ctx,"//chapter[@id]");
```


Altre classi di DOM

Oltre a DOMNode, le implementazioni di solito offrono un grande numero di altre classi, tra cui:

- Element
- Attribute
- TextNode
- Entity
- Comment, CDATASection, ProcessingInstruction

```
var pi =
  myDoc.createProcessingInstruction("xml",
  "version=\"1.0\"");
  mydoc.appendChild(pi)
```

- NodeList

XSLT via programma

Attivare un processore XSLT non fa parte di DOM. Tuttavia molte implementazioni lo permettono, o comunque all'interno dello stesso linguaggio esistono le funzioni per chiamare anche un motore XSLT ed ottenere un documento XML in risposta.

- MSXML fornisce all'interno della libreria il parser XSLT
- Insieme a Xerces (parser XML in Java) esiste un motore XSLT chiamato Xalan
- PHP utilizza un processore XSLT chiamato Sablotron

XSLT: MSXML in Javascript

Ottenere un albero

```
var source = new ActiveXObject("Msxml2.DOMDocument");
source.load(xmlFile);
var xsl = new ActiveXObject("Msxml2.DOMDocument");
xsl.load(xslFile);

var result = new ActiveXObject("Msxml2.DOMDocument");
source.transformNodeToObject(xsl,result);
```

Ottenere una stringa

```
var source = new ActiveXObject("Msxml2.DOMDocument");
source.load(xmlFile);
var xsl = new ActiveXObject("Msxml2.DOMDocument");
xsl.load(xslFile);

var result = source.transformNode(xsl);
```


XSLT: Sablotron in PHP

```
<?php

$xslthandler = xslt_create() or die("No XSLT");

xslt_run($xslthandler, $xslfile, $xmlfile);

$result = xslt_fetch_result($xslthandler);

xslt_free($xslthandler);

?>
```


XSLT: Xalan in Java

```
import javax.xml.transform.*;
import javax.xml.transform.stream.*;

...
TransformerFactory TF =
    TransformerFactory.newInstance();
Transformer t = TF.newTransformer(
        new StreamSource(xslFile));
transformer.transform(
    new StreamSource(xmlFile),
    new StreamResult(resultFile));
```


Riferimenti

DOM, raccomandazioni W3C

- http://www.w3.org/DOM/DOMTR

MSXML in Javascript e VBScript

- http://msdn.microsoft.com/library/en-us/xmlsdk30/htm/xmmscxmloverview.asp

XML, DOM XML e Sablotron in PHP

- http://www.php.net/manual/en/ref.xml.php
- http://www.php.net/manual/en/ref.domxml.php
- http://www.php.net/manual/en/ref.xslt.php

Xerces e Xalan

- http://xml.apache.org/xerces2-j/index.html
- http://xml.apache.org/xalan-j/index.html