Parsare (analizzare) un documento XML con PHP 5 e SimpleXML

1. Introduzione a SimpleXML

Come è noto, parsare un documento **XML** significa manipolarlo attraverso un linguaggio utilizzato come interfaccia, in generale le azioni di *parsing* riguardano l'accesso in lettura, la modifica e l'interrogazione di un file **XML**. In **PHP**, fino alla versione **4**, per il *parsing* di documenti **XML** venivano sfruttate in particolar modo le note librerie DOM e **SAX**, ma grazie ha **PHP 5** abbiamo ora a disposizione un nuovo strumento basato su un'interfaccia ad oggetti, **SimpleXML**, che ha l'indubbio vantaggio di rendere più facile il compito dello sviluppatore.

Rispetto a **SAX**, **SimpleXML** conserva del tutto la struttura propria del file **XML** anche per quanto riguarda la rappresentazione interna; quindi, non sarà più necessario gestire gli eventi prodotti dal documento tracciandone manualmente le gerarchie.

Rispetto a **DOM**, vi è innanzitutto un vantaggio indiscutibile nella creazione degli script per il *parsing*, **SimpleXML** necessita infatti di una sinatssi molto più semplice e listati di conseguenza molto più "leggeri".

SimpleXML, mette inoltre a disposizione la possibilità di eseguire azioni di ricerca all'interno dei file **XML**, per la precisione query basate sul metodo **XPath** dell'oggetto **SimpleXML**, gestibili attraverso l'introduzione di alcune semplici funzione native.

Nel corso di questa breve serie di articoli su **SimpleXML**, utilizzeremo un semplice documento **XML** generato dinamicamente da parsare secondo le varie modalità previste dalla libreria in oggetto, ne riportiamo di seguito il listato completo mettendolo a disposizione del lettore per eventuali test:

```
<?php
$doc_xml = <<<XML
<?xml version='1.0' standalone='yes'?>
<news>
  <articolo>
    <titolo>Guida a XML</titolo>
    <info>
      <informazioni>
        <autore>Eliox</autore>
        <argomento>PHP e SimpleXML</argomento>
      </informazioni>
      <informazioni>
        <autore>Luke</autore>
        <argomento>XML in PHP e ASP</argomento>
      </informazioni>
    </info>
    <descrizione>Guida completa al parsing XML.</descrizione>
    <data type="it">21 08 06</data>
    <data type="uk">06 08 21</data>
  </articolo>
</news>
XML;
?>
```

Chiameremo il file contenente il codice appena presentato "test.php" e lo includeremo in tutti i gli esempi dei prossimi articoli.

2. Accesso agli elementi di un file XML con SimpleXML e PHP

La funzione **simplexml_load_string()** interpreta una stringa, contenuta in un documento **XML**, all'interno di un oggetto della classe **SimpleXMLElement** le cui proprietà contengono i dati rilevati all'interno della stringa.

Ne consegue che la funzione appena descritta potrà essere utilizzata per l'accesso in lettura di elementi e contenuti all'interno di un documento **XML**; nel prossimo esempio la nostra azione di *parser* sul file "test.php" sarà indirizzata verso l'elemento "<descrizione>" presente nel listato:

```
<?php
include 'test.php';
$xml = simplexml_load_string($doc_xml);
echo $xml->articolo[0]->descrizione;
?>
```

Dall'esecuzione del codice proposto, otterremo la stampa a video del dato contenuto all'interno del tag in esame, quindi visualizzeremo la stringa:

```
Guida completa al parsing XML.
```

Questo perchè la funzione individua l'elemento indicato attraverso l'istanza dell'oggetto della classe **SimpleXMLElement** ("$xml->articolo[0]->descrizione"); i dati relativi al tag divengono quindi parte delle proprietà dell'oggetto e di conseguenza divengono accessibili.

La funzione **simplexml_load_string()** permette di *parsare* sia elementi singoli che multipli all'interno delle gerarchie di tag **XML**. Questo ci consente di isolare tutte le istanze appartenenti ad un determinato elemento ricorrendo ad un semplice ciclo di iterazione:

```
<?php
include 'test.php';
$xml = simplexml_load_string($doc_xml);
foreach ($xml->articolo as $articolo)
{
  echo $articolo->descrizione, '<br />';
}
?>
```

Quindi, in presenza di più contenuti delimitati dall'elemento "<descrizione>", verrà creato un array contenente tutti i dati rilevati all'interno del ciclo **foreach** che per ciascun nodo "<articolo>" stamperà un diverso dato appartenente a "<descrizione>".

3. Accesso e confronto tra elementi con SimpleXML

L'utilizzo della funzione **simplexml_load_string()** ci permette non soltanto di accedere in lettura agli elementi di un documento **XML** e alle sue proprietà, ma consente inoltre di accedere agli attributi degli elementi.

Nel nostro file **XML** d'esempio, "test.php", abbiamo all'interno dei nodi "<data>" due differenti "type", uno riferito alla datazione nostrana, "it", un altro relativo alla datazione anglossassone "uk"; **simplexml_load_string()** potrà in questo caso essere utilizzata per accedere ai diversi contenuti dei "type":

```
<?php
include 'test.php';
$xml = simplexml_load_string($doc_xml);
foreach ($xml->articolo[0]->data as $data)
{
  switch((string) $data['type'])
  {
    case 'it':
      echo 'Data italiana: '.$data. "<br />";
      break;
    case 'uk':
      echo 'Data inglese: '.$data. "<br />";
      break;
  }
}
?>
```

Da cui la stampa a video di:

```
Data italiana: 21 08 06
Data inglese: 06 08 21
```

Nello stesso modo, la funzione **simplexml_load_string()** potrà essere utilizzata per effettuare confronti tra elementi, o attributi, e stringhe:

```
<?php
include 'test.php';
$xml = simplexml_load_string($doc_xml);
if ((string) $xml->articolo->titolo == 'Guida a XML')
{
  echo 'Rilevato!';
}
htmlentities((string) $xml->articolo->titolo);
?>
```

Da notare innanzitutto come venga specificato il valore stringa (*string*) del termine di confronto, in secondo luogo và sottolineata l'introduzione della funzione **htmlentities()**, che converte tutti i possibili caratteri in entità **HTML** a cui viene passata come argomento la stringa per il confronto.

4. XPath e setaggio dei valori

XPath ci permette di effettuare delle interrogazioni (*query*) su di un documento **XML**, un pò come **SQL** per i database, ma attenzione, **XPath** non è un linguaggio ma più propriamente un metodo.

Nell'esempio seguente, vedremo un breve script utilizzabile per accedere a tutti i contenuti relativi al nodo "<informazioni>" e cioè "<autore>" e "<argomento>":

```
<?php
include 'test.php';
$xml = simplexml_load_string($doc_xml);
foreach ($xml->xpath('//informazioni') as $informazioni)
{
  echo $informazioni->autore, ' Autore di: ',
  $informazioni->argomento, '<br />';
}
?>
```

L'output generato in seguito al test del listato appena proposto sarà il seguente:

```
Eliox, autore di: PHP e SimpleXML
Luke, autore di: XML in PHP e ASP
```

Da notare l'utilizzo di "//" quale *wilcard* che dovrà essere invece semplicemente "/" nel caso in cui si vogliano specificare percorsi assoluti.

Naturalmente, non sarà possibile concludere il nostro discorso sul *parsing* di documenti **XML** con **SimpleXML** senza prima proporre un esempio di manipolazione dei contenuti come per esempio il settaggio di nuovi valori:

```
<?php
include 'test.php';
$xml = simplexml_load_string($doc_xml);
$xml->articolo[0]->info->informazioni[0]->autore = 'Elios';
echo $xml->asXML();
?>
```

L'output dello script mostrerà a video tutti i contenuti dei nodi indicati con una differenza, il contenuto del primo nodo "<autore>", "Eliox", verrà modificato in "Elios"; **SimpleXMLElement->asXML** consente di generare una stringa basata su un elemento **SimpleXML**.

5. Interoperabilità tra librerie in SimpleXML

SimpleXML nasce dall'esigenza di creare una libreria che semplifichi i processi di costruzione delle interfacce averso **XML**, si legga per esempio questo interessante articolo in cui una stessa azione di *parsing* viene realizzata tramite **DOM** con ben 47 righe di codice e con appena 10 righe sfruttando **SimpleXML**.

Al momento però **SimpleXML** non supporta ancora pienamente alcune fondamentali operazioni del *parsing*, come per esempio la gestione dei **namespaces**, quindi non può ancora essere utilizzata per sostituire del tutto **DOM** o **SAX**.

Fortunatamente, gli sviluppatori di **PHP 5** hanno visto bene di ovviare a questo problema introducendo il concetto di interoperabilità tra librerie, per cui sarà possibile, per esempio, utilizzare in modo combinato le potenzialità di **DOM** e **SimpleXML**; vediamo di seguito un breve esempio:

```
<?php
$xml_dom = new domDocument;
$xml_dom->loadXML('<a><b><c>String</c></b></a>');
if (!$xml_dom)
{
  echo 'Impossibile eseguire il parsing';
  exit;
}
$imp = simplexml_import_dom($xml_dom);
echo $imp->b[0]->c;
?>
```

L'esempio ci mostra in pratica come sia possibile importare e modificare un elemento **DOM** in elemento **SimpleXML**.

Nel listato appena proposto notiamo in particolar modo l'introduzione della funzione **simplexml_import_dom()** che ha il compito di estrarre un oggetto **SimpleXMLElement** da un nodo **DOM**.

simplexml_import_dom() rileva un nodo **domDocument** e lo converte in un oggetto che può essere manipolato come fosse un elemento **SimpleXML** nativo.