

## Appendice

# XML e PHP

*Tra tutti i linguaggi di programmazione esistenti PHP è tra quelli che meglio si adattano all'elaborazione di documenti XML.*

PHP è un linguaggio di programmazione che consente di arricchire le pagine web di codice script che sarà eseguito sul server, consentendo quindi la generazione dinamica del codice HTML.

Il motore di esecuzione di PHP è open source e si presenta tradizionalmente come un modulo da affiancare a un web server, normalmente Apache.

Normalmente una pagina PHP (estensione .php) è costituita da codice PHP inserito in codice HTML, e per questo motivo PHP viene definito un linguaggio *HTML embedded*.

Il codice PHP viene eseguito dal motore presente sul server prima che la pagina web venga inviata al browser; quest'ultimo riceverà quindi esclusivamente il codice HTML generato dinamicamente dall'esecuzione degli script sul server e non avrà alcun accesso al codice dello script PHP.

In questo contesto PHP si candida quindi come un'eccellente alternativa a linguaggi come Perl e Python, ma anche rispetto a linguaggi più blasonati come Java (nella sua incarnazione all'interno delle Java Server Pages) o linguaggi lato server di Microsoft come VbScript (nelle Active Server Pages) o C# nelle nuove pagine ASPX della piattaforma Microsoft .Net.

L'integrazione tra PHP e XML è uno dei fattori determinanti del successo di questo interessante linguaggio.

## Gestire XML con PHP

Ci sono in generale due modi per gestire flussi di dati XML: il primo è SAX (*Simple API for XML*) e il secondo è DOM (*Document Object Model*).

Indipendentemente dal linguaggio di programmazione utilizzato, questi due metodi consentono approcci interessanti nella gestione dei documenti XML.

In particolare SAX è in grado di elaborare documenti di grandi dimensioni scandendone il contenuto e rispondendo puntualmente al verificarsi di eventi sui dati contenuti nel documento XML. L'aspetto negativo di SAX è che i contenuti elaborati vengono immediatamente eliminati dalla memoria una volta che sono stati analizzati; di conseguenza se esiste la necessità di rileggere informazioni che sono già state elaborate, è necessario ripetere l'analisi del documento XML dall'inizio.

D'altra parte DOM è un metodo che legge l'intera struttura del documento XML e la duplica fedelmente utilizzando le strutture a oggetti messe a disposizione dal linguaggio di programmazione.

In questo modo è possibile navigare in modo completo all'interno della gerarchia di informazioni potendo saltare tranquillamente da un punto all'altro del contenuto informativo del documento XML. Anche in questo caso esiste però un rovescio della medaglia: il documento XML potrebbe infatti essere troppo grande per essere contenuto integralmente nella memoria dell'elaboratore. Di conseguenza potrebbe non essere possibile creare la struttura gerarchica che rispecchi il contenuto del documento XML e quindi questo metodo potrebbe non essere utilizzabile.

PHP5, oltre a supportare completamente le due modalità SAX e DOM, fornisce una metodologia alternativa per la gestione dei documenti XML: si tratta di SimpleXML.

SimpleXML ha un approccio simile a quello di DOM; genera infatti una struttura gerarchica completa del documento XML. In realtà in questo caso la struttura è organizzata e ottimizzata in modo da poter gestire le informazioni contenute nella gerarchia utilizzando primitive proprie del linguaggio e ottimizzate per il recupero delle informazioni.

## **XML in PHP con SAX**

Come detto in precedenza, SAX è in grado di elaborare documenti di grandi dimensioni scandendone il contenuto e rispondendo puntualmente al verificarsi di eventi sui dati contenuti nel documento XML.

Vediamo un esempio di documento XML che può essere elaborato:

```
<?xml version="1.0"?>
<?xml-stylesheet type="text/xsl" href="8.2.xsl"?>
<clienti>
  <cliente>
    <dati_anagrafici>
      <ragione_sociale>
        ABC S.p.a.</ragione_sociale>
      <partita_iva>123456789011</partita_iva>
    </dati_anagrafici>
    <dati_geografici>
      <indirizzo>via Torino, 14</indirizzo>
      <citta>Milano</citta>
    </dati_geografici>
    <contatti>
      <telefono>02123456879</telefono>
      <url>http://www.abc.it</url>
      <email>info@abc.it</email>
    </contatti>
  </cliente>
  <cliente>
    <dati_anagrafici>
      <ragione_sociale>
        DEF S.p.a.</ragione_sociale>
      <partita_iva>123456789012</partita_iva>
    </dati_anagrafici>
    <dati_geografici>
      <indirizzo>via Milano, 14</indirizzo>
      <citta>Torino</citta>
    </dati_geografici>
    <contatti>
      <telefono>011123456879</telefono>
      <url>http://www.def.it</url>
      <email>info@def.it</email>
    </contatti>
  </cliente>
  <cliente>
    <dati_anagrafici>
      <ragione_sociale>
```

```

        GHI S.p.a.</ragione_sociale>
        <partita_iva>123456789013</partita_iva>
    </dati_anagrafici>
    <dati_geografici>
        <indirizzo>via Genova, 14</indirizzo>
        <citta>Roma</citta>
    </dati_geografici>
    <contatti>
        <telefono>06123456879</telefono>
        <url>http://www.ghi.it</url>
        <email>info@ghi.it</email>
    </contatti>
</cliente>
</clienti>

```

Con PHP è possibile usare le funzioni del parser SAX per elaborare il contenuto del documento e scatenare eventi specifici quando il parser incontra l'apertura o la chiusura di un determinato elemento XML.

Vediamo un esempio di file PHP in grado di elaborare correttamente questo documento XML:

```

<!DOCTYPE html PUBLIC
    "-//W3C//DTD XHTML 1.0 Transitional//EN"
    "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
    <head>
        <title>Esempio di utilizzo di SAX</title>
        <meta http-equiv="Content-Type"
            content="text/html; charset=iso-8859-1" />
    </head>
    <body>
        <?php
function start_element
($parser, $element_name, $element_attrs) {
    switch ($element_name) {
        case 'CLIENTI':
            echo '<h1>Elenco Clienti</h1><ul>';
            break;

```

```
        case 'RAGIONE_SOCIALE':
            echo '<li>';
            break;
        case 'PARTITA_IVA':
            echo '(';
            break;
    }
}
function end_element($parser, $element_name) {
    switch ($element_name) {
        case 'CLIENTI':
            echo '</ul>';
            break;
        case 'RAGIONE_SOCIALE':
            echo '</li>';
            break;
        case 'PARTITA_IVA':
            echo ')';
            break;
    }
}
function character_data($parser, $data) {
    echo htmlentities($data);
}
$parser = xml_parser_create();
xml_set_element_handler
    ($parser, 'start_element', 'end_element');
xml_set_character_data_handler
    ($parser, 'character_data');
$fp = fopen('data.xml', 'r')
    or die ("Cannot open keyword-data.xml!");
while ($data = fread($fp, 4096)) {
    xml_parse($parser, $data, feof($fp))
        or die(sprintf('XML ERROR: %s at line %d',
            xml_error_string(xml_get_error_code($parser)),
            xml_get_current_line_number($parser)));
}
xml_parser_free($parser);
?>
```

```
</body>
</html>
```

Questo codice PHP può sembrare complicato, ma in realtà è riutilizzabile in ogni situazione in cui sia necessario elaborare documenti XML con SAX.

In particolare la funzione

```
xml_set_element_handler
    ($parser, 'start_element', 'end_element');
```

avvia la scansione del documento e individua il nome di due funzioni che saranno utilizzate rispettivamente all'apertura e alla chiusura di un determinato elemento XML.

Nel caso specifico saranno utilizzate le funzioni `start_element()` ed `end_element()`.

Se analizziamo il codice di queste funzioni, possiamo vedere che quando il parser XML incontra l'apertura dell'elemento `<clienti>` produce in output il codice HTML `<h1>Elenco Clienti</h1><ul>`, che ha come effetto la scrittura di un titolo e l'apertura di una lista non ordinata. Si noti a tal proposito che tutti i nomi degli elementi che devono essere passati al parser SAX appaiono in maiuscolo.

Quando invece il parser incontra la fine dell'elemento XML, ecco che il codice HTML che viene prodotto è `</ul>`, che corrisponde alla chiusura della lista non ordinata.

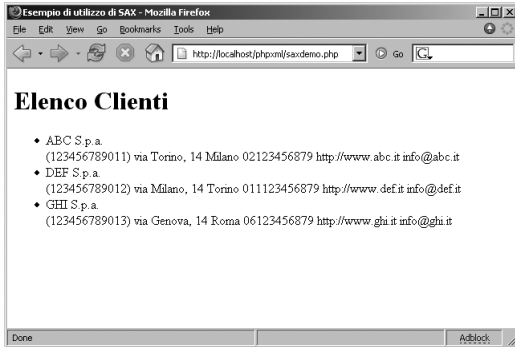
Per ciascun elemento interno a `<clienti>` vengono poi individuati dei comportamenti speciali da applicare all'elemento `<ragione_sociale>` e `<partita_iva>`; il primo viene utilizzato come elemento della lista non ordinata e il secondo viene racchiuso tra parentesi tonde.

L'esecuzione del codice PHP produce l'output visibile nella Figura A.1.

## XML in PHP con DOM

L'approccio della modalità DOM nell'analisi e nell'elaborazione dei documenti XML è molto diverso da quanto visto finora.

Utilizzando come base di partenza lo stesso documento XML dell'esempio precedente, possiamo produrre uno script PHP che generi in memoria una copia dell'intera struttura del documento XML:



**Figura A.1** Esempio di utilizzo del parser SAX in uno script PHP.

```

<!DOCTYPE html PUBLIC
    "-//W3C//DTD XHTML 1.0 Transitional//EN"
    "http://www.w3.org/TR/xhtml1/DTD/xhtml1-
transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
  <head>
    <title>Esempio di utilizzo di DOM</title>
    <meta
      http-equiv="Content-Type"
      content="text/html; charset=iso-8859-1" />
  </head>
  <body>
    <h1>Elenco Clienti</h1>
  </body>
</html>
<?php
$doc = new DOMDocument();
$doc->preserveWhiteSpace = false;
$doc->load("data.xml");
$root = $doc->documentElement;
$keywords = $root->
  getElementsByTagName('ragione_sociale');
echo '<ul>';
foreach ($keywords as $kw) {
  echo '<li>' .
    htmlentities($kw->nodeValue) . '</li>';
}

```

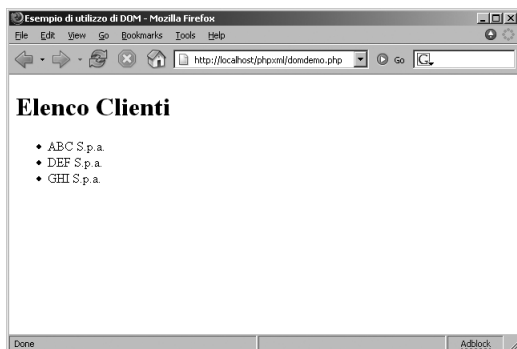
```
echo '</ul>';  
?>  
    </body>  
</html>
```

In questo caso il frammento di codice

```
$doc = new DOMDocument();  
$doc->load("data.xml");
```

ha proprio il significato di creare un oggetto di tipo `DOMDocument` e di caricare al suo interno tutto il contenuto del file `data.xml`.

Il risultato dell'esecuzione di questo script è visibile nella Figura A.2.



**Figura A.2** Esempio di utilizzo del parser DOM in uno script PHP.

Una volta che l'oggetto è stato creato, è possibile navigare in esso e ricercare informazioni al suo interno utilizzando alcune primitive proprie di PHP.

Inoltre, è possibile modificare la struttura aggiungendo elementi, nodi e valori, e infine creare un documento XML a partire dalla struttura stessa presente in memoria.

## XML in PHP con SimpleXML

SimpleXML è una nuova modalità di accesso ai documenti XML ed è presente a partire dalla versione 5.0 del linguaggio PHP.

L'obiettivo di SimpleXML è fornire al programmatore un set predefinito di modalità di accesso ai documenti XML, più semplici possibile e più integrate con i costrutti standard del linguaggio.

L'obiettivo è stato brillantemente raggiunto; vediamo questo script PHP che, analizzando il documento XML di partenza ed estraendone alcuni dati, crea una tabella dati in HTML:

```
<!DOCTYPE html PUBLIC
    "-//W3C//DTD XHTML 1.0 Transitional//EN"
    "http://www.w3.org/TR/xhtml1/DTD/xhtml1-
    transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
  <head>
    <title>Esempio di utilizzo di SimpleXML</title>
    <meta
      http-equiv="Content-Type"
      content="text/html; charset=iso-8859-1" />
  </head>
  <body>
    <h1>Elenco Clienti</h1>
    <?php
      $clienti = simplexml_load_file('data.xml');
      echo '<table border="1">';
      foreach ($clienti->cliente as $c)
      {
        foreach ($c->dati_anagrafici->ragione_sociale
          as $kw)
        {
          echo '<tr><td>' .
            htmlentities($kw) . '</td></tr>';
        }
      }
      echo '</table>';
    ?>
  </body>
</html>
```

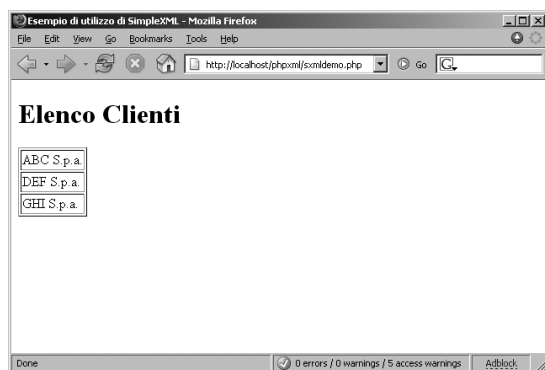
Come si può vedere, l'unica operazione che è necessario compiere per caricare in memoria un documento XML è contenuta in questa riga di codice PHP:

```
$clienti = simplexml_load_file('data.xml');
```

una volta effettuata questa operazione, tutta la gerarchia del documento XML sarà presente all'interno dell'oggetto `$clienti` e sarà quindi accessibile direttamente attraverso la classica sintassi PHP. In particolare si può accedere direttamente all'elemento `<ragione_sociale>` utilizzando il frammento di codice

```
$clienti->cliente ->dati_anagrafici->ragione_sociale
```

Il risultato dell'esecuzione dello script è visibile nella Figura A.3.



**Figura A.3** Esempio di utilizzo del parser SimpleXML in uno script PHP.

## Conclusioni

In questa appendice sono state analizzate le modalità utilizzabili con PHP per l'accesso ai documenti XML.

Si è passati dall'analisi dei classici SAX e DOM fino ad arrivare alla nuova modalità SimpleXML presente in PHP5.

Il linguaggio PHP è sempre stato dotato di una notevole potenza nell'analisi ed elaborazione dei documenti XML. Dalla versione 5 ha fatto un ulteriore passo avanti con SimpleXML, una modalità di accesso e gestione dei documenti XML che rende di fatto trasparente al programmatore PHP l'accesso a tutte quelle operazioni di gestione tipiche delle altre modalità.