Введение в XML

Язык XML привлек к себе уже достаточно много внимания со стороны разработчиков и пользователей Интернет. Сегодня количество приверженцев этой новой технологии возрастает также стремительно, как и число сообщений об очередных взятых ею преградах на пути к всеобщему признанию. Несмотря на то, что XML очень молод (международная организация W3C утвердила спецификацию "Extensible Markup Language(XML) 1.0" чуть меньше года назад - в начале февраля 1998 г) и отдельные компоненты этого языка находятся еще в стадии доработки, уже сегодня появляются новые языки, созданные на основе XML, возникают многочисленные Web-сервера, использующие эту технологию для организации хранящейся на них информации. Мир Интернет вокруг нас в очередной раз преображается, и мы можем стать участниками этого процесса уже сегодня

Для чего нужен новый язык разметки?

XML (*Extensible Markup Language*) - это язык разметки, описывающий целый класс объектов данных, называемых XML- документами. Этот язык используется в качестве средства для описания грамматики других языков и контроля за правильностью составления документов. Т.е. сам по себе XML не содержит никаких тэгов, предназначенных для разметки, он просто определяет порядок их создания. Таким образом, если, например, мы считаем, что для обозначения элемента *rose* в документе необходимо использовать тэг <*flower*>;, то XML позволяет свободно использовать определяемый нами тэг и мы можем включать в документ фрагменты, подобные следующему:

```
<flower>rose</flower>
```

Набор тэгов может быть легко расширен. Если, предположим, мы хотим также указать, что описание цветка должно по смыслу идти внутри описания оранжереи, в которой он цветет, то просто задаем новые тэги и выбираем порядок их следования:

```
<conservatory>
<flower>rose</flower>
</conservatory>
```

Как выглядит XML-документ?

XML- документ может выглядеть так:

```
<?xml version="1.0"?>
    <list_of_items>
        <item id="1">
            <first/>Первый
        </item>
        <item id="2">
            Второй
            <sub_item>подпункт 1</sub_item>
        </item>
        <item id="3">Третий</item>
        <item id="4">
            <last/>
                Последний
        </item>
    </list_of_items>
```

Требования к XML

<u>В общем случае XML- документы должны удовлетворять следующим требованиям:</u>

- В заголовке документа помещается объявление XML, в котором указывается язык разметки документа, номер его версии и дополнительная информация
- Каждый открывающий тэг, определяющий некоторую область данных в документе обязательно должен иметь своего закрывающего "напарника", т.е., в отличие от HTML, нельзя опускать закрывающие тэги
- В XML учитывается регистр символов
- Все значения атрибутов, используемых в определении тэгов, должны быть заключены в кавычки
- Вложенность тэгов в XML строго контролируется, поэтому необходимо следить за порядком следования открывающих и закрывающих тэгов
- Вся информация, располагающаяся между начальным и конечными тэгами, рассматривается в XML как данные и поэтому учитываются все символы форматирования (т.е. пробелы, переводы строк, табуляции не игнорируются, как в HTML)

Если XML- документ не нарушает приведенные правила, то он называется *формально-правильным* и все анализаторы, предназначенные для разбора XML- документов, смогут работать с ним корректно.

Элементы данных

Элемент - это структурная единица XML- документа. Заключая слово rose в в тэги <flower> </flower> , мы определяем непустой элемент, называемый <flower>, содержимым которого является *rose*. В общем случае в качестве содержимого элементов могут выступать как просто какой-то текст, так и другие, вложенные, элементы документа, секции CDATA, инструкции по обработке, комментарии, - т.е. практически любые части XML- документа.

Любой непустой элемент должен состоять из начального, конечного тэгов и данных, между ними заключенных. Например, следующие фрагменты будут являться элементами:

```
<flower>rose</flower>
<city>Novosibirsk</city>
```

,а эти - нет:

```
<rose>
<flower>
 rose
```

Атрибуты и комментарии

Комментарии

Комментариями является любая область данных, заключенная между последовательностями символов <!-- и --> Комментарии пропускаются анализатором и поэтому при разборе структуры документа в качестве значащей информации не рассматриваются.

Атрибуты

Если при определении элементов необходимо задать какие-либо параметры, уточняющие его характеристики, то имеется возможность использовать атрибуты эдлемента. Атрибут - это пара "название" = "значение", которую надо задавать при определении элемента в начальном тэге. Пример:

```
<color RGB="true">#ff08ff</color>
<color RGB="false">white</color>
```

или

```
<author id=0>Ivan Petrov</author>
```

Примером использования атрибутов в HTML является описание элемента <font>:

```
<font color=¦white¦ name=¦Arial¦>Black</font>
```

Специальные символы

Для того, чтобы включить в документ символ, используемый для определения каких-либо конструкций языка (например, символ угловой скобки) и не вызвать при этом ошибок в процессе разбора такого документа, нужно использовать его специальный символьный либо числовой идентификатор. Например, < , > " или $(десятичная форма записи), &#x1a (шестнадцатеричная) и т.д. Строковые обозначения спецсиволов могут определяться в XML документе при помощи компонентов (entity), о чем мы еще поговорим немного позже.

Директивы анализатора и CDATA

Директивы анализатора

- Инструкции, предназначенные для анализаторов языка, описываются в XML документе при помощи специальных тэгов - <? и ?>;. Программа клиента использует эти инструкции для управления процессом разбора документа. Наиболее часто инструкции используются при определении типа документа (например, <? Xml version=¦1.0¦?>) или создании пространства имен[11].

CDATA

- Чтобы задать область документа, которую при разборе анализатор будет рассматривать как простой текст, игнорируя любые инструкции и специальные символы, но, в отличии от комментариев, иметь возможность использовать их в приложении, необходимо использовать тэги <![CDATA] и]]>. Внутри этого блока можно помещать любую информацию, которая может понадобится программе-клиенту для выполнения каких-либо действий (в область CDATA, можно помещать, например, инструкции JavaScript). Естественно, надо следить за тем, чтобы в области, ограниченной этими тэгами не было последовательности символов]].

Работа с XML в PHP через SimpleXML : Введение.

Расширение SimpleXML предоставляет очень простой и легкий в использовании набор инструментов для преобразования XML в объект, с которым можно затем работать через его свойства и с помощью итераторов.

Базовое использование SimpleXML

SimpleXML пользоваться очень просто! Попробуйте получить какую-нибудь строку или число из базового XML документа.

```php
<?php
include 'example.php';
$movies = new SimpleXMLElement($xmlstr);
echo $movies->movie[0]->plot;
?>
```

Результат выполнения данного примера:

```
Таким образом, это язык. Это все равно язык программирования. Или
это скриптовый язык? Все раскрывается в этом документальном фильме,
похожем на фильм ужасов.
```

Недопустимые символы в названии тегов для php

В PHP получить доступ к элементу в XML документе, содержащим в названии недопустимые символы (например, дефис), можно путем заключения данного имени элемента в фигурные скобки и апострофы.

```
<?php
include 'example.php';
$movies = new SimpleXMLElement($xmlst
r);
echo $movies->movie->{'great-lines'}-
>line;
?>
```

Результат выполнения данного примера:

```
PHP решает все мои проблемы в web
```

Доступ к не уникальным элементам в SimpleXML

В том случае, если существует несколько экземпляров дочерних элементов в одном родительском элементе, то нужно применять стандартные методы итерации.

```php
<?php
include 'example.php';
$movies = new SimpleXMLElement($xmlstr);
/* Для каждого узла <character>, мы отдельно выведем имя <name>. */
foreach ($movies->movie->characters->character as $character) {
   echo $character->name, ' играет ', $character->actor, PHP_EOL;
}
?>
```

Результат выполнения данного примера:

```
Ms. Coder играет Onlivia Actora
Mr. Coder играет El ActÓr
```

Использование атрибутов

- До сих пор мы только читали названия и значения элементов. SimpleXML может так же получать доступ к атрибутам элемента. Получить доступ к атрибуту элемента можно так же, как к элементам массива.

```php
<?php
include 'example.php';
$movies = new SimpleXMLElement($xmlstr);
/* Доступ к узлу <rating> первого фильма.
 * Так же выведем шкалу оценок. */
foreach ($movies->movie[0]->rating as $rating) {
    switch((string) $rating['type']) { // Получение атриб
екcу
    case 'thumbs':
        echo $rating, ' thumbs up';
        break;
    case 'stars':
        echo $rating, ' stars';
        break;
    }
}
?>
```

Результат выполнения данного примера:

```
7 thumbs up5 stars
```

Сравнение двух элементов

Два элемента SimpleXMLElements считаются различными, даже если они указывают на один и тот же объект начиная с PHP 5.2.0.

```
<?php
include 'example.php';
$movies1 = new SimpleXMLElement($xmlstr);
$movies2 = new SimpleXMLElement($xmlstr);
var_dump($movies1 == $movies2); // false начиная с PHP 5.
2.0
?>
```

Установка значений

Данные в SimpleXML могут быть не постоянными. Объект позволяет манипулировать всеми элементами.

```php
<?php
include 'example.php';
$movies = new SimpleXMLElement($xmlstr);
$movies->movie[0]->characters->character[0]->name = 'Miss Coder';
echo $movies->asXML();
?>
```

```
<characters>
 <character>
  <name>Miss Coder</name>
  <actor>Onlivia Actora</actor>
 </character>
 <character>
  <name>Mr. Coder</name>
```

Добавление элементов и атрибутов

Начиная с PHP 5.1.3, SimpleXML имеет возможность легко добавлять дочерние элементы и атрибуты.

```php
<?php
include 'example.php';
$movies = new SimpleXMLElement($xmlstr);
$character = $movies->movie[0]->characters->addChild('character
');
$character->addChild('name', 'Mr. Parser');
$character->addChild('actor', 'John Doe');
$rating = $movies->movie[0]->addChild('rating', 'PG');
$rating->addAttribute('type', 'mpaa');
echo $movies->asXML();
```

```
<character><name>Mr. Parser</name><actor>John Doe</actor></character></characters>

<rating type="mpaa">PG</rating></movie>
```

Что такое cURL и libcurl? Общие моменты

Библиотека libcurl предоставляет нам возможность передачи данных на сервер, и получения ответов от него. Что нам это дает? Возможность эмуляции поведения пользователя или браузера! Вы можете получать содержимое страниц для последующего парсинга, можете получать заголовки ответов сервиса и программно авторизироваться на сайтах, делать скрипты постинга сообщений (например, в твиттер или на форумах) или грабберы информации. Все ограничивается лишь вашей фантазией!

Установка cURL на Denwer

- Скачиваем готовый пакет расширений «PHP5: дополнительные модули».
- Соответственно, устанавливаем его. Ничего сложного, согласитесь :)
- Открываем в блокноте файл: `X:/webservers/usr/local/php5/php.ini //где X - ваш диск, куда установлен вебсервер`
- и убираем точку с запятой в начале сроки:
- `;extension=php_curl.dll`
- Делаем рестарт сервера Денвер.

Описание cURL и первые шаги

Для началом работы с инструментом, его нужно инициализировать. Делается это следующим образом:

- `$ch = curl_init();`

Мы использовали функцию инициализации сессии cURL. При этом, можно задать URL сразу, вот так:

- `$ch = curl_init('http://myblaze.ru');`

А можно сделать это потом, в опциях. Порядок установки опций не имеет значения. Делается это другой функцией:

- `curl_setopt (resource ch, string option, mixed value)`

Первый параметр этой функции, то есть resource ch мы уже создали чуть выше, а вот параметров option и value очень много. Я думаю, что не стоит копипастить сюда их все, а достаточно лишь дать ссылку на подробное описание функции, надеюсь никто не обидится: curl_setopt.

Приведу пример установки опций как раз на примере URL:

-
```
$url = "http://myblaze.ru";
curl_setopt($ch, CURLOPT_URL,$url);
```

Еще парочка примеров задания опций: давайте получим заголовок ответа сервера, при этом не будем получать саму страницу:

-
```
curl_setopt($ch, CURLOPT_HEADER, 1); // читать заголовок
curl_setopt($ch, CURLOPT_NOBODY, 1); // читать ТОЛЬКО заголовок без тела
```

Итак, мы инициализировали сессию, задали нужные нам параметры, теперь выполняем получившийся запрос, закрываем сессию и выводим результат:

-
```
$result = curl_exec($ch);
curl_close($ch);
echo $result;
```

Результаты

В итоге получаем наш первый полностью рабочий пример использования библиотеки libcurl:

```
$ch = curl_init();
$url = "http://myblaze.ru";
curl_setopt($ch, CURLOPT_URL,$url);
curl_setopt($ch, CURLOPT_HEADER, 1); // читать заголовок
curl_setopt($ch, CURLOPT_NOBODY, 1); // читать ТОЛЬКО заголовок без тела
$result = curl_exec($ch);
curl_close($ch);
echo $result;
```

В результате мы получаем заголовок HTTP ответа от сервера:

```
HTTP/1.1 200 OK
Server: nginx/1.2.6
Date: Sat, 09 Mar 2013 16:38:39
GMT Content-Type: text/html; charset=UTF-8
Connection: keep-alive
Keep-Alive: timeout=10
X-Pingback: http://myblaze.ru/xmlrpc.php 1
```

Структура заголовка HTTP запроса

Для примера обратимся к странице ya.ru и в Opera Dragonfly просмотрим сформированный запрос браузера и полученный от сервера ответ. Вот и они:
Запрос:
GET / HTTP/1.1 — Пытаемся получить страницу по адресу /, то есть главную, находящуюся в корне папки. Используем протокол версии 1.1.
User-Agent: Opera/9.80 (Windows NT 6.1; WOW64) Presto/2.12.388 Version/12.14 — Представляемся серверу, мы — браузер Опера.
Host: ya.ru — Доменное имя запрашиваемого ресурса.
Accept: text/html, application/xml;q=0.9, application/xhtml+xml, image/png, image/webp, image/jpeg, image/gif, image/x-xbitmap, */*;q=0.1 — Список допустимых форматов ресурса.
Accept-Language: ru-RU,ru;q=0.9,en;q=0.8 — Список поддерживаемых языков.
Accept-Encoding: gzip, deflate — Поддерживаемые способы кодирования.
Cookie: yandexuid=XXXXX — Куки, при необходимости.
Connection: Keep-Alive — Просим не разрывать соединение и оставаться на связи.

Структура заголовка HTTP запроса

Ответ:
HTTP/1.1 200 Ok — Получаем ответ с кодом 200, значит все ОК.
Server: nginx — Сервер представился — это nginx.
Date: Sun, 10 Mar 2013 14:10:50 GMT — Текущие дата и время на сервере.
Content-Type: text/html; charset=UTF-8 — Тип контента и кодировка.
Connection: close — Сервер не хочет поддерживать с нами постоянного соединения, поэтому сразу же его закрывает. Для следующего запроса будет установлено новое соединение.
Cache-Control: no-cache,no-store,max-age=0,must-revalidate — Управление кэшированием. В данном случае оно отключено.
Expires: Sun, 10 Mar 2013 14:10:50 GMT — Дата предполагаемого истечения срока действия сессии. В нашем случае оно совпадает с временем открытия, так как сервер тут же его закрыл, сразу после обработки.
Last-Modified: Sun, 10 Mar 2013 14:10:50 GMT — Время последней модификации.
Content-Encoding: gzip — Способ кодирования информации.
Полный список всех параметров, которые можно встретить в заголовке HTTP запроса можно посмотреть на википедии.
Теперь вы примерно представляете как общаются между собой ваш браузер и web-сервер. Это очень полезно знать и понимать, ведь мы будем пытаться эмулировать действия браузера с помощью библиотеки libcurl.

Пример работы с библиотекой

Раз уж cURL так хорош для парсеров, то рассмотрим функцию получения кода страницы по ее адресу. При этом на выходе получим массив с заголовком, содержимым страницы и даже коды ошибок, если что-то пойдет не так.

```
function get_web_page( $url )
{
  $uagent = "Opera/9.80 (Windows NT 6.1; WOW64) Presto/2.12.388 Version/12.14";

  $ch = curl_init( $url );

  curl_setopt($ch, CURLOPT_RETURNTRANSFER, 1);   // возвращает веб-страницу
  curl_setopt($ch, CURLOPT_HEADER, 0);           // не возвращает заголовки
  curl_setopt($ch, CURLOPT_FOLLOWLOCATION, 1);   // переходит по редиректам
  curl_setopt($ch, CURLOPT_ENCODING, "");        // обрабатывает все кодировки
  curl_setopt($ch, CURLOPT_USERAGENT, $uagent);  // useragent
  curl_setopt($ch, CURLOPT_CONNECTTIMEOUT, 120); // таймаут соединения
  curl_setopt($ch, CURLOPT_TIMEOUT, 120);        // таймаут ответа
  curl_setopt($ch, CURLOPT_MAXREDIRS, 10);       // останавливаться после 10-ого редиректа

  $content = curl_exec( $ch );
  $err     = curl_errno( $ch );
  $errmsg  = curl_error( $ch );
  $header  = curl_getinfo( $ch );
  curl_close( $ch );

  $header['errno']   = $err;
  $header['errmsg']  = $errmsg;
  $header['content'] = $content;
  return $header;
}
```

Пример использования функции

Входные параметры:
url — адрес страницы или сайта.
Значения выходных параметров (массив с тремя элементами):
header[‘errno’] — если что-то пошло не так, то тут будет код ошибки.
header[‘errmsg’] — здесь при этом будет текст ошибки.
header[‘content’] — собственно сама страница\файл\картинка и т.д.

Используем функцию, например, так:

```
$result = get_web_page( "http://ya.ru" );
if (($result['errno'] != 0 )||($result['http_code'] != 200))
    {
 echo $result['errmsg'];
 }
else
 {
 $page = $result['content'];
 echo $page;
 }
```