**Лабораторна робота №5. Знайомство з мовою PHP**

Мета: познайомитись з процесом розробки простих програм на мові PHP

Тривалість: 2 години

**Завдання:**

Виконайте варіант завдання, який співпадає з вашим номером у списку

**Варіант 1**

Засобами PHP створити веб-сторінку "Перевірка людяності", яка показує текстове задання та кілька малюнків як варіанти відповіді. Після натискання на малюнок скрипт PHP перевіряє правильність відповіді та інформує про результати перевірки.

**Зовнішній вигляд веб-сторінки**

Запитання

|  |
| --- |
| Натисніть, будь ласка, на зображення кота |
|  |

Оцінка правильності:

|  |
| --- |
| Натисніть, будь ласка, на зображення кота |
|  |
| Ви вибрали  Це неправильно |

**Варіант 2**

Засобами PHP створити веб-сторінку, яка перевіряє знання таблиці множення. Веб-сторінка містить текстовий напис для показу загального рахунку, кнопку «наступне завдання», текстовий напис для показу завдання, кнопку "Перевірити" радіокнопки для вибору правильного варіанту відповіді, та текстовий напис для виводу результатів перевірки. Для кожного завдання користувач має лише одна спробу вибору.

**Зовнішній вигляд веб-сторінки**

|  |
| --- |
| Загальний рахунок 90% (9 правильних відповідей з 10) |
|  |
| 3 × 4 =  11  12  16  7 |
| Попередня відповідь: 6\*2=23 Помилка, правильна відповідь «12» |

**Варіант 3**

Засобами PHP створити веб-сторінку, яка перевіряє знання таблиці множення. Веб-сторінка містить текстовий напис для показу загального рахунку, посилання «наступне завдання», текстовий напис для показу завдання, посилання для вибору правильного варіанту відповіді, та текстовий напис для виводу результатів перевірки. Для кожного завдання користувач має лише одна спробу вибору.

**Зовнішній вигляд веб-сторінки**

|  |
| --- |
| Загальний рахунок 90% (9 правильних відповідей з 10) |
| [наступне завдання](http://sites.znu.edu.ua/webdevelopment/labs/1415.ukr.html) |
| 3 × 4 = [11](http://sites.znu.edu.ua/webdevelopment/labs/1415.ukr.html) [12](http://sites.znu.edu.ua/webdevelopment/labs/1415.ukr.html) [16](http://sites.znu.edu.ua/webdevelopment/labs/1415.ukr.html) [7](http://sites.znu.edu.ua/webdevelopment/labs/1415.ukr.html) |
| Попередня відповідь: 3 × 5 = 11 Помилка, правильна відповідь «15» |

**Варіант 3**

Засобами PHP створити веб-сторінку "перетворення одиниць", який за величиною температури за шкалою Фаренгейта обчислює величину температури в градусах Цельсія і навпаки. Форма має два текстових поля:

* температура за Фаренгейтом
* температура за Цельсієм

які містять відповідні числа.

**Зовнішній вигляд веб-сторінки**

|  |  |
| --- | --- |
| Температура за Фаренгейтом |  |
| Температура за Цельсієм |  |
|  |  |

Сигналом для початку обчислення служить натискання кнопки "Обчислити".

Температура за Фаренгейтом (F) і температура за Цельсієм (C) зв'язані формулою C=5/9 \* (F-32)

**Варіант 4**

Засобами PHP створити веб-сторінку, який перевіряє знання таблиці множення. Веб-сторінка містить текстовий напис для показу загального рахунку, кнопку «наступне завдання», текстовий напис для показу завдання, текстове поле для вводу відповіді, кнопку «перевірити» та текстовий напис для виводу результатів перевірки.

**Зовнішній вигляд веб-сторінки**

|  |
| --- |
| Загальний рахунок 90% (9 правильних відповідей з 10) |
|  |
| 3 × 4 = |
| Помилка, правильна відповідь «12» |

**Варіант 5**

Засобами PHP створити веб-сторінку "Перевірка людяності", який пише назву цифри словами, вимагає натиснути клавішу з відповідною цифрою та перевіряє правильність введеного числа

**Зовнішній вигляд веб-сторінки**

|  |
| --- |
| Ви мали надрукувати число 2 і надрукували число 2. Це правильно. Ще одне завдання |
| Надрукуйте, будь ласка, число ВІСІМ |
|  |
|  |

**Варіант 6**

Засобами PHP створити веб-сторінку "перегляд зображень", який в параметрах POST отримує імена файлів, розділені двокрапкою (наприклад, “fle1.jpg:file2.gif:file34.gif”) та показує одне вибране зображення. Вибір зображень відбувається за допомогою посилань, на яких зображено іконки – зменшені копії зображень.

**Зовнішній вигляд веб-сторінки**

|  |
| --- |
|  |
|  |

**Варіант 7**

Засобами PHP створити веб-сторінку "перегляд зображень", який в параметрах POST отримує імена файлів, розділені двокрапкою (наприклад, “fle1.jpg:file2.gif:file34.gif”) та показує одне вибране зображення. Вибір зображень відбувається за допомогою двох кнопок: «Вперед» та «Назад».

**Зовнішній вигляд веб-сторінки**

|  |  |
| --- | --- |
|  |  |
|  | |

**Варіант 8**

СЗасобами PHP створити веб-сторінку «Транслітерація», який перетворює український текст за правилом транслітерації: Друкуючи у верхньому текстовому полі український текст, користувач отримує в нижньому полі перетворений текст. ( [Правила транслітерації](http://ru.wikipedia.org/wiki/%D0%A2%D1%80%D0%B0%D0%BD%D1%81%D0%BB%D0%B8%D1%82%D0%B5%D1%80%D0%B0%D1%86%D0%B8%D1%8F) )

**Зовнішній вигляд веб-сторінки**

|  |
| --- |
| Український текст: |
| Транслітерація: |
|  |

**Варіант 9**

Засобами PHP створити веб-сторінку "розрахунок прямокутника", який за шириною та довжиною прямокутника обчислює його периметр, площу та довжину діагоналі. Веб-сторінка має два текстових поля:

* Довжина
* Ширина

які містять відповідні числа. Обчислені значення периметра, площі та довжини діагоналі розміщуються як звичайний текст

**Зовнішній вигляд веб-сторінки**

|  |  |
| --- | --- |
| Довжина |  |
| Ширина |  |
|  |  |
| Периметр | 10 |
| Площа | 6 |
| Довжина діагоналі | 3.6555 |

**Варіант 10**

Засобами PHP створити "секретну веб-сторінку", який показує деякий текст тільки після того, як відвідувач сайта набрав ім'я та пароль:

|  |  |
| --- | --- |
| Вітаю тебе, о GreatUser | [Вийти](http://sites.znu.edu.ua/webdevelopment/labs/1415.ukr.html) |
| Цей текст бачать лише обрані, яким вдалося пригадати пароль для входу на сайт. | |

Якщо ім'я та пароль не надані або помилкові, то веб-сторінка показує два текстових поля:

|  |  |
| --- | --- |
| Ім'я |  |
| Пароль |  |
|  |  |

**Теоретичні відомості**

[А. Мазуркевич, Д.Еловой. PHP, настольная книга программиста](http://ebooks.znu.edu.ua/files/comp.books/2007/gd/php/a__mazurkevich_d__elovoiy_php_nastol_naia_kniga_programmista.rar)  
[Михаил Фленов. PHP глазами хакера](http://ebooks.znu.edu.ua/files/comp.books/2007/gd/php/flenov_php_glazami_hakera.djvu)  
[Самоучитель PHP5](http://ebooks.znu.edu.ua/files/comp.books/2007/gd/php/kolisnichenkosamouchitelphp5.djvu)  
[А. Кухарчик, PHP. Обучение на примерах](http://ebooks.znu.edu.ua/files/comp.books/2007/gd/php/kuharchik__php__obuchenie_na_primerah.pdf)  
[Ларри Ульман, Основы программирования на PHP. Самоучитель](http://ebooks.znu.edu.ua/files/comp.books/2007/gd/php/larri_ul_man_osnovy_programmirovaniia_na_rnr_samouchitel_.pdf)  
[Энди Гутманс, Стиг Баккен, Дерик Ретанс. PHP5.Профессиональное программирование](http://ebooks.znu.edu.ua/files/comp.books/2007/gd/php/php5__professional_noe_programmirovanie.djvu)  
[Справка по PHP5](http://ebooks.znu.edu.ua/files/comp.books/2007/gd/php/php_manual_en.chm)  
[Профессиональное PHP программирование](http://ebooks.znu.edu.ua/files/comp.books/2007/gd/php/professional_noe_rnr_programmirovanie_2_e_izdanie.djvu)

[Уроки PHP](http://www.poiu.ru/lessons/index.php.htm)

[Уроки PHP. По материалам форума forum.php.su](http://www.php.su/lessons/)

**Вказівки**

**1 варіант запуску програм на php**

Для виконання роботи вам знадобиться завантажити [архів](http://sites.znu.edu.ua/webdevelopment/files/Portable_WebServers_Denwer3.rar?v=1382617092), який містить веб-сервер Apache зі встановленим PHP та сервер MySQL з менеджером phpMyAdmin.

Сценарії PHP потрібно розміщати в директорії WebServers\home\localhost\www

**В комп'ютерних класах ЗНУ потрібно в налаштуваннях огладача інтернету вимкнути проксі-сервер**.

**2 варіант запуску програм на php**

Чтобы запустить php сервер

1) скачать http://windows.php.net/downloads/releases/php-5.4.21-nts-Win32-VC9-x86.zip

2) разархивировать в папку php-5.4.21-nts-Win32-VC9-x86

3) создать папку public\_html

4) в папке public\_html создать файл start\_php.bat:

@rem Диск, где находится папка php-5.4.21-nts-Win32-VC9-x86

E:

@rem Переход в папку, где находится файл php.exe

cd \\kit-server\stud\web\_prog\php-5.4.21-nts-Win32-VC9-x86

@rem Команда запуска встроенного веб-сервера php

@rem Параметр -t указывает на местоположение bat-файла

php -S localhost:8000 -t \\kit-server\stud\web\_prog\php\public\_html

5)Создать в папке public\_html файл test.php (выводится надпись test1)

<?php

echo 'test1';

?>

6)Запустить php - 2 раза клацнуть мышкой на bat-файле start\_php.bat

7) в Mozilla Firefox ввести в адресную строку http://localhost:8000/test.php

# Лабораторна робота №6. Робота з сервісами в PHP

Мета: Освоїти функції PHP для роботи з XML та HTTP

### Завдання:

1) Завантажити засобами PHP [стрічку новин ЗНУ](http://sites.znu.edu.ua/cms/index.php?action=news/rss&site_id=27&lang=ukr&rows=10&template=&date=desc&category_id=0)

2) За допомогою бібліотеки [SimpleXML](http://php.net/manual/en/book.simplexml.php) відобразити перелік останніх новин, в якому для кожної новини будуть показані  
1) заголовок  
2) опис  
3) посилання на сторінку з повним текстом новини

### Вказівки

Для завантаження файла з іншого сервера використовуйте функції PHP CURL.

Для завантаження файла з іншого сервера використовуйте функції PHP Simple XML

# Лабораторна робота №6. Робота з сервісами в PHP

Мета: Освоїти функції PHP для роботи з XML та HTTP

### Завдання:

1) Завантажити засобами PHP [стрічку новин ЗНУ](http://sites.znu.edu.ua/cms/index.php?action=news/rss&site_id=27&lang=ukr&rows=10&template=&date=desc&category_id=0)

2) За допомогою бібліотеки [SimpleXML](http://php.net/manual/en/book.simplexml.php) відобразити перелік останніх новин, в якому для кожної новини будуть показані  
1) заголовок  
2) опис  
3) посилання на сторінку з повним текстом новини

### Вказівки

Для завантаження файла з іншого сервера використовуйте функції PHP CURL.

Для завантаження файла з іншого сервера використовуйте функції PHP Simple XML

**Обробка XML в PHP**

В PHP существуют 2 модуля работы с XML, реализующие 2 разных стандарта обработки XML-данных: SAX (Simple API for XML) и DOM (Document Object Model). Необходимо вкратце пояснить, что представляют собой эти стандарты.

**Стандарт SAX** не является стандартом W3C, официальным сайтом для него является http://www.saxproject.org. Этот стандарт описывает метод парсинга XML документов для получения данных из них. Т.е. этот метод обработки XML-документов позволит вам только прочитать данные из XML-документа и не более того. Создавать и изменять XML-документы с его помощью вы не сможете.

SAX основан на т.н. событийном программировании. Его особенностью является то, что вы предоставляете парсеру XML набор собственных функций, которые будут заниматься обработкой различных типов XML-данных (элементов (тегов), текста и т.п.), а парсер затем будет сам вызывать ваши функции в процессе обработки XML-документа, передавая им найденные данные. Важной особенностью здесь является порядок вызова ваших функций, который вы должны будете учесть при их написании: ваши функции будут вызываться в той же последовательности, в которой соответствующие даные располагаются в XML-документе. В следующем разделе мы рассмотрим пример чтения данных из XML-документа с использованием SAX.

Другим стандартом для обработки XML-данных является DOM - стандарт W3C, спецификацию которого вы можете найти на http://www.w3.org/TR/REC-DOM-Level-1 . В отличие от SAX этот метод позволяет вам производить любые операции с XML-данными в очень удобной форме - представляя XML-документ как дерево объектов. Далее мы рассмотрим пример того, как можно работать с XML через DOM из PHP. Исходные тексты всех приведенных здесь примеров можно скачать здесь в виде .zip архива.

**Чтение XML-данных в PHP с использованием SAX**

Для того, чтобы показать, как использовать имеющиеся в PHP модули для работы с XML-данными возьмем реальный пример - колонка новостей, хранящаяся в XML файле.

Файл news.xml, в котором хранятся новости:

<?xml version="1.0"?>

<newsLine>

<news date="1.1.2002">

<title>title 1</title>

<text>news text 1</text>

</news>

<news date="5.1.2002">

<title>title 2</title>

<text>news text 2</text>

</news>

<news date="10.1.2002">

<title>title 3</title>

<text>news text 3</text>

</news>

</newsLine>

Файл test.php, выполняющий парсинг news.xml для получения новостей:

<?php

$news = array(); // В этом массиве будут храниться новости,полученные из XML файла

$currentNews = null; // Текущая новость. Используется в процессе импорта данных

$index = null; // Текущий индекс в массиве новостей.

// Используется в процессе импорта данных

// Функции, описанные ниже, являются обработчиками различных типов

// XML-данных и будут вызываться парсером в процессе разбора.

// Функция для обработки начальных тегов XML

// На входе:

// - указатель на SAX парсер

// - имя XML тега

// - массив аттрибутов

function saxStartElement($parser,$name,$attrs)

{

global $currentNews,$index;

switch($name)

{

case 'newsLine':

// Тег newsLine содержит все новости. Мы должны подготовить

// массив $news для приема новостей из XML файла.

$news = array();

break;

case 'news':

// Каждая новость находится в теге news. Подготавливаем массив

// $currentNews для приема этой новости

$currentNews = array();

// Если у новости есть дата - сохраняем ее в массиве

if (in\_array('date',array\_keys($attrs))) $currentNews['date'] = $attrs['date'];

break;

default:

// Все остальные теги, которые могут встретиться в XML файле

// находятся внутри тега <news>, поэтому мы просто запоминаем

// их название с тем, чтобы знать, какие именно данные мы

// обрабатываем.

$index = $name;

break;

};

}

// Функция для обработки конечных тегов XML

// На входе:

// - указатель на SAX парсер

// - имя XML тега

function saxEndElement($parser,$name)

{

global $news,$currentNews,$index;

if ((is\_array($currentNews)) && ($name=='news'))

// Если в данный момент у нас есть массив $currentNews (т.е.

// мы обрабатываем содержимое новости) и имя закрывающего

// тега - "news", то это значит, что данные для этой новости

// кончились и мы можем поместить готовую новость в массив

// новостей.

{

$news[] = $currentNews;

// Уничтожаем массив текущей новости, чтобы показать, что

// в данный момент мы не занимаемся получением данных для

// новости.

$currentNews = null;

};

// В любом случае закрытие тега означает, что символьные

// данные, получаемые парсером не нужно помещать куда-либо.

$index = null;

}

// Функция для обработки символьных данных

// На входе:

// - указатель на SAX парсер

// - символьные данные XML

function saxCharacterData($parser,$data)

{

global $currentNews,$index;

// Мы принимаем только данные для новостей, помещенные в

// какой-нибудь тег. Все остальные символьные данные

// (как правило это пустое пространство, использованное

// для форматирования) мы опускаем за ненадобностью.

if ((is\_array($currentNews)) && ($index))

$currentNews[$index] = $data;

}

// Создаем SAX парсер, который будет использоваться для обработки XML-данных.

$parser = xml\_parser\_create();

// Регистрируем функции для обработки различных типов XML-данных:

// - начальный и конечный тэги XML

xml\_set\_element\_handler($parser,'saxStartElement','saxEndElement');

// - символьные данные

xml\_set\_character\_data\_handler($parser,'saxCharacterData');

// Также существуют аналогичные функции для регистрации

// обработчиков других типов XML-данных.

// Убираем case folding, в этом случае имена тэгов будут

// передаваться обработчикам в оригинальном виде. Если case

// folding включен, то все имена тегов будут переведены

// в верхний регистр.

xml\_parser\_set\_option($parser,XML\_OPTION\_CASE\_FOLDING,false);

// Получаем содержимое XML-файла с новостями.

$xml = join('',file('news.xml'));

// Производим парсинг (разбор) полученного XML-файла.

// В процессе разбора парсер будет вызывать описанные нами

// функции и в результате мы получим массив $news,

// содержащий новости из XML-файла.

if (!xml\_parse($parser,$xml,true))

// Парсер возвращает значение FALSE, если произошла

// какая-либо ошибка. В этом случае мы также прекращаем

// выполнение скрипта и возвращаем ошибку.

die(sprintf('Ошибка XML: %s в строке %d',

xml\_error\_string(xml\_get\_error\_code($parser)),

xml\_get\_current\_line\_number($parser)));

// Уничтожаем парсер, освобождая занятые им ресурсы

xml\_parser\_free($parser);

?>

В результате работы файла test.php мы получим массив $news, содержащий все новости из файла news.xml. Если, к примеру сделать print\_r($news), чтобы посмотреть содержимое массива, то мы увидим примерно следующее:

Array

(

[0] => Array

(

[date] => 1.1.2002

[title] => title 1

[text] => news text 1

)

[1] => Array

(

[date] => 5.1.2002

[title] => title 2

[text] => news text 2

)

[2] => Array

(

[date] => 10.1.2002

[title] => title 3

[text] => news text 3

)

)

Для того, чтобы потом вывести новости на странице, нам будет необходимо всего-лишь обработать элементы массива, например так:

<html>

<head>

<title>Новости</title>

</head>

<body>

<table width="100%" border="1">

<?php

foreach($news as $n)

{

?>

<tr>

<td width="90%"><b><?php echo $n['title']; ?></b></td>

<td><?php echo $n['date']; ?></td>

</tr>

<tr>

<td colspan="2"><?php echo $n['text']; ?><br><br></td>

</tr>

<?php

};

?>

</table>

</body>

</html>

Возможно, вы уже заметили очевидные неудобства этого метода обработки XML-документов. В первую очередь это очевидная сложность в написании обработчиков для различных типов XML-данных. Ведь вам необходимо самостоятельно заботиться о том, чтобы передать другим обработчикам информацию о том, что делать с той или иной информацией, которую они получили. Несложно догадаться, что в случаях более сложной структуры XML документа эти проверки станут еще более сложными и легко могут стать источником ошибок. Кроме того очевидно, что набор обработчиков различен для каждой отдельно взятой струтруры XML-документа и при изменении этой структуры код обработчиков необходимо переписывать.

Все эти недостатки делают область применения SAX достаточно ограниченной, тем более в условиях того, что существует другой, гораздо более мощный и удобный стандарт обработки XML-документов - DOM. О нем и пойдет речь в следующем разделе.

**Обработка XML-данных в PHP с использованием DOM**

Стандарт DOM обладает двумя основными достоинствами:

* Он не привязан к какой-то конкретной платформе или языку программирования. Используя DOM можно работать с XML-данными совершенно одинаково на C++, Java, JavaScript и т.д.
* Он позволяет выполять все операции по обработке XML-данных. Т.е. вы можете не только читать их, но и модифицировать содержимое XML-документа, вставляя туда новые теги, удаляя и изменяя их.

К сожалению в отношении PHP первое высказывание не относится т.к. единственная имеющаяся реализация интерфейса DOM для PHP не является таковой в полной мере, поскольку имеет совершенно другой интерфейс, что, конечно же не очень хорошо. Кроме того имеющийся интерфейс (по крайней мере пока) не поддерживает многие вещи, подчас являющиеся концептуальными для стандарта DOM. Но даже в таком виде этот интерфейс намного удобнее в использовании чем SAX и, я думаю, вы сами убедитесь в этом, посмотрев пару примеров, приведенных ниже:

Для начала попробуем реализовать ту же функциональность, что и в примере, рассмотренном в предыдущем разделе (чтение колонки новостей из XML файла) но с использованием интерфейса DOM:

<?php

// Массив, в котором будут сохранены новости из XML файла

$news = array();

// Читаем содержимое XML файла в строковую переменную

$xml = join('',file('news.xml'));

// Создаем дерево DOM XML на основе имеющегося у нас XML файла.

// Заметьте, что это заняло у нас всего лишь одну строчку кода!

$xml = xmldoc($xml);

// Теперь в переменной $xml у нас находится дерево объектов,

// представляющее собой наш XML файл. Для того, чтобы начать

// обход этого дерева - необходимо получить объект родительской node (тега).

// Для нашего XML файла это будет node <newsList>

$root = $xml->root();

// Получаем массив потомков родительской node (в нашем случае это

// массив <news>)

$nodes = $root->children();

// Начинаем обработку каждой node в массиве

foreach($nodes as $node)

{

// Если текущая node - это одна из node <news>, то продолжаем ее обработку,

// чтобы получить информацию об этой новости

if ($node->name=='news')

{

// Создаем новый массив, куда будем собирать информацию о текущей новости

$currentNews = array();

// Получаем дату этой новости (она хранится в аттрибуте 'date')

$currentNews['date'] = $node->get\_attribute('date');

// Остальная информация хранится во вложенных node <title> и <text>,

// поэтому мы должны взять список вложенных nodes и обработать его.

$content = $node->children();

foreach($content as $contentNode)

{

// Проверяем: если node, которую мы сейчас обрабатываем, является тегом

// (т.е. имеет тип XML\_ELEMENT\_NODE) и ее имя - одно из тех, которые мы

// ищем - добавляем информацию из этой node в массив с информацией о новости

if (($contentNode->type==XML\_ELEMENT\_NODE) &&

(in\_array($contentNode->name,array('title','text'))))

$currentNews[$contentNode->name] = $contentNode->content;

};

// Сохраняем текущую новость в массиве новостей

$news[] = $currentNews;

};

};

?>

Как видите - код получился гораздо более простой и понятный. Кроме того он намного меньше зависит от структуры исходного XML файла и легче поддается модификации. Результат работы, естественно, тот же самый, что и для варианта с работй через SAX.

Однако и это не самый простой вариант обработки XML-документов, особенно для больших документов со сложной структурой. Основной проблемой в этом случае становится процесс доступа к данным, т.к. необходимо "вручную" пройти весь путь по XML-дереву до необходимых данных. К счастью из этой ситуации есть простой и элегантный выход и имя ему XPath. Это язык, также разработанный W3C и специально предназначенный для получения данных из деревьев DOM XML. Спецификация этого языка доступна здесь, а здесь вы можете взять ее перевод на русский язык. К сожалению ввиду того, что имеющийся у нас XML-документ имеет очень простую структуру - я не могу продемонстрировать большинство интересных конструкций XPath на нем, но после прочтения спецификации вы поймете, что это очень мощный и удобный инструмент. Например выражение **//category[@id=5]/product[position()=3]/name** вернет вам имя продукта, расположенного под номером 3 в категории, имеющей id=5, причем положение самой node <category> может быть произвольным внутри дерева XML.

Посмотрим, как может выглядеть код предыдущего примера, если использовать XPath для получения всех node <news>:

<?php

// Сначала все то же самое...

$news = array();

$xml = join('',file('news.xml'));

$xml = xmldoc($xml);

// А теперь, вместо того, чтобы добираться до необходимых данных

// "вручную" мы воспользуемся XPath.

// Инициализируем

$xml->xpath\_init();

// Создаем новый контекст

$ctx = xpath\_new\_context($xml);

// Вычисляем XPath-выражение, результатом которого являются все node <news>

$nodes = xpath\_eval($ctx,'//news');

// Теперь нам остается только обработать полученные nodes.

foreach($nodes->nodeset as $node)

{

// Сама обработка осталась такой же, правда за ненадобностью исчезла

// проверка имени обрабатываемой node.

$currentNews = array();

$currentNews['date'] = $node->get\_attribute('date');

$content = $node->children();

foreach($content as $contentNode)

{

if (($contentNode->type==XML\_ELEMENT\_NODE) &&

(in\_array($contentNode->name,array('title','text'))))

$currentNews[$contentNode->name] = $contentNode->content;

};

$news[] = $currentNews;

};

?>

Как видите - с использованием XPath обработка XML стала еще проще. Но это еще не все. Как я уже говорил раньше - интерфейс DOM позволяет вам модифицировать дерево XML. Воспользуемся этим, чтобы добавить еще одну новость в XML-файл:

<?php

// Данные для новости, которая будет добавлена в XML файл

$newsToAdd = array(

'date' => '17.1.2002',

'title' => 'title 4',

'text' => 'news text 4'

);

// Загружаем имеющийся XML файл и создаем DOM-дерево

$xml = join('',file('news.xml'));

$xml = xmldoc($xml);

// Получаем reference на корневую node (<newsLine>)

// поскольку мы должны добавить новость к ней.

$root = $xml->root();

// Создаем новую node для новости

$newsNode = $root->new\_child('news',null);

// Сохраняем дату новости в аттибуте 'date'

$newsNode->setattr('date',$newsToAdd['date']);

// Создаем вложенные nodes <title> и <text>

$newsNode->new\_child('title',$newsToAdd['title']);

$newsNode->new\_child('text',$newsToAdd['text']);

// Добавление новости завершено. Получаем текст XML файла

// и сохраняем измененный файл на диске.

$text = $xml->dumpmem();

$fp = fopen('news.xml','w');

fwrite($fp,$text);

fclose($fp);

?>

Этот пример не нуждается в каких-то особых комментариях - структура создаваемого XML достаточно четко видна из того, что и к какой node добавляется. Запустив этот пример и посмотрев на файл news.xml после окончания его работы вы увидете, что в нем появилась та новость, которую мы добавляли.

**Обработка с помощью SimpleXML**

PHP версии 5 представляет SimpleXML, новый интерфейс прикладного программирования (API) для чтения и письма в XML. Расширения SimpleXML, такие как

$doc->rss->channel->item->title

выбирают элементы из документа. Пока вы имеете хорошее представление о структуре вашего документа, писать эти выражения легко. Однако, если вы не знаете точно, где интересующие вас элементы появляются (как например, в случае с Docbook, HTML и другими подобными текстовыми документами), SimpleXML может пользоваться выражениями XPath для поиска этих элементов.

**Начало работы с SimpleXML**

Представим себе, что вы хотите создать страницу PHP, которая конвертирует канал RSS в HTML код. RSS является основным форматом XML для публикации содержания, взятого из нескольких источников. Корневой элемент этого документа – rss, который содержит единственный элемент channel. Элемент channel содержит метаданные о содержимом, включая его заголовок, язык и URL. Он также содержит разнообразные текстовые элементы, вложенные в элементы item. Каждый элемент item имеет элемент link, содержащий URL или title, или description (обычно оба), в которых находится читаемый текст. Области имен не используются. Конечно, ещё много чего можно сказать об RSS, но для целей этой статьи информации достаточно. Листинг 1 показывает типичный пример с парой информационных сообщений.

Листинг 1. Канал RSS

<?xml version="1.0" encoding="UTF-8"?>

<rss version="0.92">

<channel>

<title>Mokka mit Schlag</title>

<link>http://www.elharo.com/blog</link>

<language>en</language>

<item>

<title>Penn Station: Gone but not Forgotten</title>

<description>

The old Penn Station in New York was torn down before I was born.

Looking at these pictures, that feels like a mistake. The current site is

functional, but no more; really just some office towers and underground

corridors of no particular interest or beauty. The new Madison Square...

</description>

<link>http://www.elharo.com/blog/new-york/2006/07/31/penn-station</link>

</item>

<item>

<title>Personal for Elliotte Harold</title>

<description>Some people use very obnoxious spam filters that require you

to type some random string in your subject such as E37T to get through.

Needless to say neither I nor most other people bother to communicate with

these paranoids. They are grossly overreacting to the spam problem.

Personally I won't...</description>

<link>http://www.elharo.com/blog/tech/2006/07/28/personal-for-elliotte-harold/</link>

</item>

</channel>

</rss>

Давайте сделаем страницу PHP, которая форматирует каждый RSS канал как HTML. Листинг 2 показывает структуру будущей страницы.

Листинг 2. Статическая структура для PHP-кода

<?php // Загрузите и проанализируйте XML-document ?>

<html xml:lang="en" lang="en">

<head>

<title><?php // Заголовок будет читаться из RSS ?></title>

</head>

<body>

<h1><?php // Заголовок снова будет читаться из RSS ?></h1>

<?php

// Здесь мы поместим цикл, чтобы включить заголвок каждого элемента и описание

?>

</body>

</html>

**Анализ XML-документа**

Первый шаг – анализ XML-документа и его сохранение в переменной. Это требует написания всего лишь одной строки кода, которая передает URL функции simplexml\_load\_file():

$rss = simplexml\_load\_file('http://partners.userland.com/nytRss/nytHomepage.xml');

Для этого примера взята страница из Userland канала New York Times по адресу http://partners.userland.com/nytRss/nytHomepage.xml. Конечно, вы можете использовать вместо этого любой другой URL для другого канала RSS.

Заметьте, что несмотря на имя simplexml\_load\_file(), этой функции будет необходимо анализировать XML документ на удаленном HTTP URL. Но это не единственная неожиданность в этой функции. Возвращаемое значение функции, которое здесь хранится в переменной $rss, не указывает на весь документ, как вы могли бы ожидать, исходя из опыта работы с другими интерфейсами API, такими, как, например, Объектная модель документа (DOM). Скорее оно указывает на корневой элемент документа. Контент, который находится в прологе и эпилоге документа, недоступен из SimpleXML.

**Нахождение имени канала**

Имя всего канала (в отличие от заголовков отдельных текстовых фрагментов этого канала) находится в дочернем элементе title от элемента channel, порожденного корневым элементом rss. Вы можете загрузить этот заголовок, как будто бы XML-документ был бы просто последовательной формой объекта класса rss с полем channel, которое в свою очередь имело бы поле title. Используя регулярный PHP-синтаксис ссылки на объект, этот оператор находит заголовок:

$title = $rss->channel->title;

Найдя заголовок, вы должны добавить его к выходным данным HTML. Сделать это просто: повторите переменную $title:

<title><?php echo $title; ?></title>

Эта строка выводит строковое значение элемента, но не весь элемент. То есть текст записывается, а теги нет. Вы можете даже полностью пропустить промежуточную переменную $title:

<title><?php echo $rss->channel->title; ?></title>

Потому что эта страница многократно использует это значение во многих местах, более удобно хранить его как описательно озаглавленную переменную.

**Выполнение итераций через элементы**

Далее вы должны найти элементы канала. Выражение, которое выполняет это задание:

$rss->channel->item

Однако каналы обычно содержат больше, чем один элемент. Или их даже может не быть вовсе. Соответственно, этот оператор возвращает массив, который вы можете повторить с помощью цикла for-each:

foreach ($rss->channel->item as $item) {

echo "<h2>". $item->title. "</h2>";

echo "<p>". $item->description. "</p>";

}

Вы можете легко добавить ссылки, считывая значение элемента link из канала RSS. Просто выведите элемент a из PHP и используйте $item->link, чтобы возвратить URL. Листинг 3 добавляет этот элемент и заполняет структуру из Листинга 1.

Листинг 3. Простая, но полная программа считывания PHP RSS

<?php // Load and parse the XML document

$rss = simplexml\_load\_file('http://partners.userland.com/nytRss/nytHomepage.xml');

$title = $rss->channel->title;

?>

<html xml:lang="en" lang="en">

<head>

<title><?php echo $title; ?></title>

</head>

<body>

<h1><?php echo $title; ?></h1>

<?php

// Здесь мы поместим цикл, чтобы включить заголовок элемента и описание

foreach ($rss->channel->item as $item) {

echo "<h2><a href='". $item->link. "'>". $item->title. "</a></h2>";

echo "<p>". $item->description. "</p>";

}

?>

</body>

</html>

Это все, что нужно, чтобы написать простую программу чтения RSS в PHP – несколько строк HTML и несколько строк PHP. Не считая пробелов, в общей сложности 20 строк. Конечно, это не самая широкофункциональная, оптимизированная или надежная разработка. Давайте посмотрим, что мы можем сделать, чтобы исправить это.

**Обработка ошибок**

Не все каналы RSS так хорошо сформированы, как это должно быть. Спецификация XML требует, чтобы процессоры прекратили обрабатывать документы, как только обнаружена формальная ошибка, а SimpleXML соответствовал программе обработки XML. Однако это вам особенно не поможет, когда будет обнаружена ошибка. Как правило, программа записывает предупреждение в файл php-ошибок (но без детального сообщения об ошибке), и функция simplexml-load-file() выдает ошибку. Если вы не уверены, что файл, который вы анализируете хорошо выстроен, проверьте на наличие этой ошибки, перед тем как использовать данные файла, как это показано в Листинге 4.

Листинг 4. Остерегайтесь деформированного ввода данных

<?php

$rss = simplexml\_load\_file('http://www.cafeaulait.org/today.rss');

if ($rss) {

foreach ($rss->xpath('//title') as $title) {

echo "<h2>". $title. "</h2>";

}

}

else {

echo "Упс! Ввод деформирован!";

}

?>

Другая распространенная ошибка случается, когда документ хорошо сформатирован, но не содержит те элементы, которые вы ожидаете там, где вы ожидаете их найти. Что происходит, например, с таким выражением $doc->rss->channel->item->title, когда элементная группа не имеет заголовка (как это случается, по меньшей мере, с одним из ста наиболее частотных RSS-каналов)? Самый простой подход – всегда обращаться с возвращаемым функцией значением как с массивом данных и заключать его в цикл. В этом случае вы защищены от факта наличия большего или меньшего количества элементов, чем вы ожидали. Однако, если вы знаете, что вы хотите первый элемент в документе, даже если там имеется больше чем один, вы можете запросить его через индекс, начинающийся с нуля. Например, для запроса заголовка первой группы элементов, вы можете написать:

$doc->rss->channel->item[0]->title[0]

Если первая группа элементов отсутствует, или не имеет названия, она рассматривается так же, как и любой другой, находящийся за установленными рамками индекс, в массиве PHP. То есть результат равен нулю, который превращается в пустую строку, когда вы попытаетесь вставить его в выходной код HTML.

Распознавание и отклонение неожиданных форматов, с которыми вы не готовы работать, обычно является сферой деятельности проверяющего на правильность парсера XML. Однако SimpleXML не может проверить относительно шаблона DTD (DTD) или схемы данных. Он проверяет только на формальную правильность.

**Как работать с пространством имен**

Многие сайты сегодня переходят с RSS на Atom. Листинг 5 показывает пример документа в Atom. По многим параметрам этот документ идентичен примеру с RSS. Однако здесь больше метаданных, а корневой элемент - feed, вместо - rss. Элемент feed имеет списки вместо элементов. Элемент content заменяет элемент description. Гораздо важнее то, что документ Atom использует пространство имен, в то время как RSS нет. Таким образом, документ Atom может выводить реальное, не урезанное содержание Extensible HTML (XHTML).

Листинг 5. Документ в Atom

<?xml version="1.0"?>

<feed xmlns="http://www.w3.org/2005/Atom" xml:lang="en-US"

xml:base="http://www.cafeconleche.org/today.atom">

<updated>2006-08-04T16:00:04-04:00</updated>

<id>http://www.cafeconleche.org/</id>

<title>Cafe con Leche XML News and Resources</title>

<link rel="self" type="application/atom+xml" href="/today.atom"/>

<rights>Copyright 2006 Elliotte Rusty Harold</rights>

<entry>

<title>Steve Palmer has posted a beta of Vienna 2.1, an open source

RSS/Atom client for Mac OS X.

</title>

<content type="xhtml">

<div xmlns="http://www.w3.org/1999/xhtml"

id="August\_1\_2006\_25279" class="2006-08-01T07:01:19Z">

<p>

Steve Palmer has posted a beta of <a shape="rect"

href="http://www.opencommunity.co.uk/vienna21.php">Vienna

2.1</a>, an open source RSS/Atom client for Mac OS X. Vienna

is the first reader I've found acceptable for daily use; not

great but good enough. (Of course my standards for "good

enough" are pretty high.) 2.1 focuses on improving the user

interface with a unified layout that lets you scroll through

several articles, article filtering (e.g. read all articles

since the last refresh), manual folder reordering, a new get

info window, and an improved condensed layout.

</p>

</div>

</content>

<link href="/#August\_1\_2006\_25279"/>

<id>http://www.cafeconleche.org/#August\_1\_2006\_25279</id>

<updated>2006-08-01T07:01:19Z</updated>

</entry>

<entry>

<title>Matt Mullenweg has released Wordpress 2.0.4,

a blog engine based on PHP and MySQL.

</title>

<content type="xhtml">

<div xmlns="http://www.w3.org/1999/xhtml"

id="August\_1\_2006\_21750" class="2006-08-01T06:02:30Z">

<p>

Matt Mullenweg has released <a shape="rect"

href="http://wordpress.org/development/2006/07/wordpress-204

/">Wordpress 2.0.4</a>, a blog engine based on PHP and

MySQL. 2.0.4 plugs various security holes, mostly involving

plugins.

</p>

</div>

</content>

<link href="/#August\_1\_2006\_21750"/>

<id>http://www.cafeconleche.org/#August\_1\_2006\_21750</id>

<updated>2006-08-01T06:02:30Z</updated>

</entry>

</feed>

Хотя имена элементов изменились, основной подход к работе с SimpleXML в документах в Atom такой же, как и с RSS. Единственная разница в том, что вам необходимо указать пространство имен, т.е. универсальный идентификатор ресурса (URI), когда вы запрашиваете элемент с именем, так же как и локальное имя. Это двухступенчатый процесс: во-первых, запросите дочерние элементы в данном пространстве имен, передав пространство имен URI функции children(). Затем запросите элементы с правильным местным именем в этом пространстве имени. Представьте, что вы сначала загрузили канал Atom в переменную $feed, следующим образом:

$feed = simplexml\_load\_file('http://www.cafeconleche.org/today.atom');

Эти две строки теперь находят элемент title:

$children = $feed->children('http://www.w3.org/2005/Atom');

$title = $children->title;

Вы можете сжать этот код в единое выражение, если хотите, хотя строка становится немного длинной. Все другие элементы в пространствах имен должны быть проработаны подобным же образом. Листинг 6 показывает полную страницу PHP, которая отображает заголовки из поименованного канала Atom.

Листинг 6. Простой ридер PHP-заголовков Atom

<?php $feed = simplexml\_load\_file('http://www.cafeconleche.org/today.atom');

$children = $feed->children('http://www.w3.org/2005/Atom');

$title = $children->title;

?>

<html xml:lang="en" lang="en">

<head>

<title><?php echo $title; ?></title>

</head>

<body>

<h1><?php echo $title; ?></h1>

<?php

$entries = $children->entry;

foreach ($entries as $entry) {

$details = $entry->children('http://www.w3.org/2005/Atom');

echo "<h2>". $details->title. "</h2>";

}

?>

</body>

</html>

**Смешанный контент**

Почему я показал только заголовки в этом примере? Потому что в Atom контент любого списка может содержать полный текст фрагмента и не только сам текст, но и всю разметку. Это - повествовательная структура: слова в ряду предназначены для чтения людьми. Как и в большинстве данных подобного рода здесь смешанный контент. XML уже не упрощенный, и поэтому подход SimpleXML начинает давать сбои. Он не может корректно работать со смешанным контентом, и этот пропуск данных препятствует использованию во многих случаях.

Вы можете сделать одну вещь, но это только частичное решение проблемы. Она сработает только потому, что элемент content содержит реальный XHTML. Вы можете скопировать этот XHTML, как непроанализированный исходный код непосредственно в конечный продукт, используя функцию asXML(), например, следующим образом

echo "<p>". $details->content->asXML(). "</p>";

В результате получится что-то вроде Листинга 7.

Листинг 7. Выходные данные XML

<content type="xhtml">

<div xmlns="http://www.w3.org/1999/xhtml"

id="August\_7\_2006\_31098" class="2006-08-07T09:38:18Z">

<p>

Nikolai Grigoriev has released <a shape="rect"

href="http://www.grigoriev.ru/svgmath">SVGMath 0.3</a>, a

presentation MathML formatter that produces SVG written in

pure Python and published under an MIT license. According to

Grigoriev, "The new version can work with multiple-namespace

documents (e.g. replace all MathML subtrees with SVG in an

XSL-FO or XHTML document); configuration is made more

flexible, and several bugs are fixed. There is also a

stylesheet to adjust the vertical position of the resulting

SVG image in XSL-FO."

</p>

</div>

</content>

Это не чистый XHTML. Элемент content извлекается из Atom документа, и вам бы, на самом деле, лучше его не иметь. Даже хуже, он проникает в неправильное пространство имени, поэтому он не может быть распознан таким, какой он есть. К счастью на практике этот дополнительный элемент не очень сильно вредит, потому что Web-браузеры просто игнорируют любые теги, которые они не распознают. Законченный документ неисправен, но это не имеет большого значения. Если это действительно вас беспокоит, сведите это на нет с помощью строковых операций, например, следующим образом:

$description = $details->content->asXML();

$tags = array('<content type="xhtml"'>", "</content>");

$notags = array("", "");

$description = str\_replace($tags, $notags, $description);

Чтобы сделать код немного более надежным, используйте регулярное выражение, а не допущение, что начальный тег именно такой, как он показан выше. Особенно вы можете просчитать множество возможных атрибутов:

// тег конца имеет фиксированную форму, поэтому его легко заменить

$description = str\_replace("</content>", "", $description);

// удалите тег начала, по возможности включая атрибуты и пробел

$description = ereg\_replace("<content[^>]\*>", "", $description);

Даже с этой модификацией ваш код может выдавать комментарии, вычислительные команды и отрезки CDATA. Так или иначе, вы срезаете это, хотя боюсь, что это больше не так просто. Смешанный контент просто переходит границы, для работы в которых был разработан SimpleXML.

**XPath**

Такие выражения как $rss->channel->item->title великолепны, только если вы точно знаете, какие элементы находятся в документе, и где точно они находятся. Однако вы далеко не всегда это знаете. Например, в XHTML элементы в заголовке (h1,h2,h3, и т.д.) могут быть дочерними от body, div, table и от нескольких других элементов. Более того, div, table, blockquote и другие элементы могут быть вложены друг в друга множество раз. Для многих менее определенных случаев использования, легче использовать выражения XPath, такие как //h1 или //h1[contains('Ben')]. SimpleXML имеет этот набор функциональных возможностей через функцию xpath().

Листинг 8 показывает страницу PHP, которая содержит все заголовки в документе RSS – как заголовок самого канала, так и заголовки отдельных групп элементов.

Листинг 8. Использование XPath для нахождения элементов заголовков

<html xml:lang="en" lang="en">

<head>

<title>XPath Example</title>

</head>

<body>

<?php

$rss = simplexml\_load\_file('http://partners.userland.com/nytRss/nytHomepage.xml');

foreach ($rss->xpath('//title') as $title) {

echo "<h2>". $title. "</h2>";

}

?>

</body>

</html>

SimpleXML поддерживает только строковые выражения XPath, задающие местонахождении файла и объединения этих выражений. Он не поддерживает выражения XPath, которые не возвращают множества узлов, таких как count(//para) илиcontains(title).

Начиная с PHP версии 5.1, SimpleXML может создавать XPath-запросы относительно документов с пространствами имен. Как обычно в XPath, строковое выражение, задающее местонахождение файла, должно использовать префиксы пространств имен, даже если искомый документ использует пространство имен по умолчанию. Функция registerXPathNamespace() ассоциирует префикс с пространством имени URI для использования в следующем запросе. Например, если вы хотите найти все элементы title в документе, вы используете код подобный представленному в Листинге 9.

Листинг 9. Использование XPath с пространствами имен

$atom = simplexml\_load\_file('http://www.cafeconleche.org/today.atom');

$atom->registerXPathNamespace('atm', 'http://www.w3.org/2005/Atom');

$titles = $atom->xpath('//atm:title');

foreach ($titles as $title) {

echo "<h2>". $title. "</h2>";

}

Еще одно последнее предупреждение: XPath в PHP работает довольно медленно. Загрузка страницы может занимать от мгновения до нескольких секунд, когда я переключился на это XPath выражение, даже на незагруженном местном сервере. Если вы используете эти приемы, вы должны использовать какого-то рода кэширование для разумной работы. Динамическое генерирование каждой страницы просто не сработает.

**Вывод**

SimpleXML является полезным дополнением к инструментарию разработчика PHP при условии, что не нужно работать со смешанным контентом. Это покрывает большое количество случаев использования. Особенно хорошо это работает с простыми данными в виде записей. При условии, что документ не слишком глубок, не слишком сложен и не имеет смешанного содержания, SimpleXML гораздо проще, чем его альтернатива DOM. Он также помогает, если вы знаете структуру документа заранее, хотя использование XPath может помочь существенно облегчить это требование. Отсутствие проверки и поддержки смешанного содержания доставляет неудобства, не всегда препятствует работе. Многие простые форматы не имеют смешанного контента, а во многих случаях использования применяются только очень предсказуемые форматы данных. Если именно это характеризует вашу работу, не откажите себе в удовольствие попробовать SimpleXML. Немного внимания ошибкам и немного усилий, направленных на настройку кэш-памяти, позволяют свести проблемы с эффективностью системы к минимуму, SimpleXML может быть надежным, устойчивым к ошибкам, средством обработки XML в рамках PHP