УДК 004.021

Компоненты онлайн обработки больших данных XML(YML) Web-порталов

Козич Полина Александровна
Приамурский государственный университет имени Шолом-Алейхема
Студент

Аннотация
Одним из распространенных способ передачи данных в веб-технологиях является передача в формате xml. Данный формат используют различные протоколы передачи данных, на нем реализованы некоторые хранилища в виде баз данных. Для работы с ним существует много библиотек на всех языках программирования. В данной статье будет рассмотрена проблема обработки больших данных, данного формата. Как решить проблему, если файлы весят мегабайты, а то и гигабайты. Будут рассмотрены существующие решения и выбрано наилучшее для языка программирования PHP.
Ключевые слова: XML, SimpleXML, XMLReader, PHP

Components of online processing of large XML (YML) data web-portals

Kozich Polina Alexandrovna
Sholom-Aleichem Priamursky State University
Student

Abstract
One of the most common ways to transfer data in web technologies is to transfer in xml format. This format uses different data transfer protocols, it implements some storage in the form of databases. To work with it, there are many libraries in all programming languages. In this article we will consider the problem of processing large data of this format. How to solve the problem if the files weigh megabytes or even gigabytes. The existing solutions will be reviewed and the best one for the PHP programming language will be selected.
Keyword: XML, SimpleXML, XMLReader, PHP

Одним из распространенных способ передачи данных в веб-технологиях является передача в формате xml. Данный формат используют различные протоколы передачи данных, на нем реализованы некоторые хранилища в виде баз данных. Для работы с ним существует много библиотек на всех языках программирования.

Например, для языка программирования PHP существует несколько известных решений: SimpleXML, использование DOM, XML-анализатор (SAX-парсер), XMLReader.

Таким образом, цель данного исследования: рассмотреть существующие решения проблемы обработки больших данных формата xml и выбрать наилучшее для языка программирования PHP.

Данными формата xml в своих работах рассматривали А.А. Горбунов и М.Б. Хорошко в своей статье рассмотрели использование расширения simplexml на основе языка программирования php [1]. Д.С. Яковлев рассмотрел вопрос использования XML - документов в качестве баз данных для веб-приложений [2]. А.А. Горбунов рассмотрел подсистему разработанной Amazon для взаимодействия со своей торговой площадкой [3]. В.А. Малышев, И.С. Голубь, В.И. Глаголев рассмотрели обработку и импорт больших XML-данных партнерских сайтов [10]. Также иностранные научные работники занимаются исследованием данной проблемы [4].

Основные проблемы, которые могут возникнуть при работе с XML файлами больших объемов. Это дефицит оперативной памяти и слишком долгое выполнение алгоритма обработки xml. В большинстве случаев работа веб-сервера ограничена в использовании этих двух ресурсов.

Рассмотрим имеющиеся решения, их особенности и примеры использования:

SimpleXML – позволяет работать с XML как с объектами базы данных. Прост в использовании, удобный API. К недостаткам стоит отнести медленную работу и необходимость загружать весь XML в память [5, 7, 8]. Пример использования на рисунке 1.

```
$xml = simplexml_load_file("price.xml");
echo "<table border='1'>\n";

foreach ($xml->xpath('/DocumentElement/price') as $producs) { ?>
    <tr>
        <td><?php echo $producs->name; ?></td>
        <td><?php echo $producs->company; ?></td>
        <td><?php echo $producs->city; ?></td>
        <td><?php echo $producs->amount ?></td>
    </tr>
<?
}
echo "</table>\n";
```

Рисунок 1 – пример использования SimpleXML

DOM – расширение, которое позволяет работать с xml как с DOM(Document Object Model), представляя структуру документа в виде дерева. По такому же принципу можно работать и с HTML. Похож на simpleXML тем, что удобно работать, но также требует собирать весь файл в память. Пример использования на рисунке 2:

```php
$doc = new DOMDocument();
$doc->load( 'books.xml' );

$books = $doc->getElementsByTagName( "book" );
foreach( $books as $book )
{
$authors = $book->getElementsByTagName( "author" );
$author = $authors->item(0)->nodeValue;

$publishers = $book->getElementsByTagName( "publisher" );
$publisher = $publishers->item(0)->nodeValue;

$titles = $book->getElementsByTagName( "title" );
$title = $titles->item(0)->nodeValue;

echo "$title - $author - $publisher\n";
```

Рисунок 2 – пример использования расширения DOM

Xml_parser – синтаксический анализатор XML. Особенность анализатора заключается в том, что он читает файл построчно и не требует целиком загружать его в память. Реагирует на открывающийся и закрывающийся тег. Для работы необходимо написать свои триггеры для вызова функций обработки событий открытый/закрытый тег. Пример использования на рисунках 3-4.

```php
class Simple_Parser
{
    var $parser;
    var $error_code;
    var $error_string;
    var $current_line;
    var $current_column;
    var $data = array();
    var $datas = array();

    function parse($data)
    {
        $this->parser = xml_parser_create('UTF-8');
        xml_set_object($this->parser, $this);
        xml_parser_set_option($this->parser, XML_OPTION_SKIP_WHITE, 1);
        xml_set_element_handler($this->parser, 'tag_open', 'tag_close');
        xml_set_character_data_handler($this->parser, 'cdata');
        if (!xml_parse($this->parser, $data))
        {
            $this->data = array();
            $this->error_code = xml_get_error_code($this->parser);
            $this->error_string = xml_error_string($this->error_code);
            $this->current_line = xml_get_current_line_number($this->parser);
            $this->current_column = xml_get_current_column_number($this->parser);
        }
        else
        {
            $this->data = $this->data['child'];
        }
        xml_parser_free($this->parser);
    }
```

Рисунок 3 – Пример использования xml_parser. 1 часть

```
    function tag_open($parser, $tag, $attribs)
    {
        $this->data['child'][$tag][] = array('data' => '', 'attribs' => $attribs, 'ch
ild' => array());
        $this->datas[] =& $this->data;
        $this->data =& $this->data['child'][$tag][count($this->data['child'][$tag])-1
];
    }

    function cdata($parser, $cdata)
    {
        $this->data['data'] .= $cdata;
    }

    function tag_close($parser, $tag)
    {
        $this->data =& $this->datas[count($this->datas)-1];
        array_pop($this->datas);
    }
}

$xml_parser = new Simple_Parser;
$xml_parser->parse('<foo><bar>test</bar></foo>');
```

Рисунок 4 – Пример использования xml_parser. 2 часть

XMLReader – также является синтаксическим анализатором, также считывает файл построчно. Преимуществом над xml_parsrer является то, что триггеры уже заданы в виде констант [6]. Пример можно посмотреть на рисунках 5-6.

```
<?php Class StoreXMLReader
{
  private $reader;
  private $tag;

  // if $ignoreDepth == 1 then will parse just first level, else parse 2th level too
  private function parseBlock($name, $ignoreDepth = 1) {
    if ($this->reader->name == $name && $this->reader->nodeType == XMLReader::ELEMENT) {
      $result = array();
      while (!($this->reader->name == $name && $this->reader->nodeType == XMLReader::END_ELEMENT)) {
        //echo $this->reader->name. ' - '.$this->reader->nodeType." - ".$this->reader->depth."\n";
        switch ($this->reader->nodeType) {
          case 1:
            if ($this->reader->depth > 3 && !$ignoreDepth) {
              $result[$nodeName] = (isset($result[$nodeName]) ? $result[$nodeName] : array());
              while (!($this->reader->name == $nodeName && $this->reader->nodeType == XMLReader::END_ELEMENT)) {
                $resultSubBlock = $this->parseBlock($this->reader->name, 1);

                if (!empty($resultSubBlock))
                  $result[$nodeName][] = $resultSubBlock;
                unset($resultSubBlock);
                $this->reader->read();
              }
            }
            $nodeName = $this->reader->name;
            if ($this->reader->hasAttributes) {
              $attributeCount = $this->reader->attributeCount;

              for ($i = 0; $i < $attributeCount; $i++) {
                $this->reader->moveToAttributeNo($i);
                $result['attr'][$this->reader->name] = $this->reader->value;
              }
              $this->reader->moveToElement();
            }
            break;
          case 3:
          case 4:
            $result[$nodeName] = $this->reader->value;
            $this->reader->read();
            break;
        }
        $this->reader->read();
      }
      return $result;
    }
  }
```

Рисунок 5 – Пример использования XMLReader. Часть 1.

```php
public function parse($filename) {

  if (!$filename) return array();

  $this->reader = new XMLReader();
  $this->reader->open($filename);

  // begin read XML
  while ($this->reader->read()) {

    if ($this->reader->name == 'store_categories') {
    // while not found end tag read blocks
    while (!($this->reader->name == 'store_categories' && $this->reader->nodeType == XMLReader::END_ELEMENT)) {
      $store_category = $this->parseBlock('store_category');

      /*
        Do some code
      */

      $this->reader->read();
    }

    $this->reader->read();
  }

  } // while
 } // func
}

$xmlr = new StoreXMLReader();
$r = $xmlr->parse('example.xml');
```

Рисунок 6 – Пример использования XMLReader. Часть 2.

Поводя итоги обзора, стоит отметить, что сводится либо к удобству использования, либо к экономии потребляемых ресурсов. Идеальным результатом был бы инструмент, обладающий возможностью работать с xml как с объектами или DOM и не требующий выгружать весь xml файл, своего рода гибрид.

Данная мысль уже приходила в головы разработчиков, что вылилось в попытке совместить эти два типа инструмента. Примером может послужить библиотека с открытым исходным кодом SimpleXMLReader. При желании ее реализацию можно посмотреть на ресурсе GitHub[9]

Пример использования библиотеки представлен на рисунках 7-8.

```php
<?php

header ("Content-type: text/html, charset=utf-8;");
require_once dirname(__FILE__). "/../library/SimpleXMLReader.php";

class ExampleXmlReader1 extends SimpleXMLReader
{

    public function __construct()
    {
        // by node name
        $this->registerCallback("Цена", array($this, "callbackPrice"));
        // by xpath
        $this->registerCallback("/Данные/Остатки/Остаток", array($this, "callbackRest"));
    }

    protected function callbackPrice($reader)
    {
        $xml = $reader->expandSimpleXml();
        $attributes = $xml->attributes();
        $ref = (string) $attributes->{"Номенклатура"};
        if ($ref) {
            $price = floatval((string)$xml);
            $xpath = $this->currentXpath();
            echo "$xpath: $ref = $price;\n";
        }
        return true;
    }

    protected function callbackRest($reader)
    {
        $xml = $reader->expandSimpleXml();
        $attributes = $xml->attributes();
        $ref = (string) $attributes->{"Номенклатура"};
        if ($ref) {
            $rest = floatval((string) $xml);
            $xpath = $this->currentXpath();
            echo "$xpath: $ref = $rest;\n";
        }
        return true;
    }

}
```

Рисунок 7 - Пример использования SimpleXMLReader. Часть 1.

```php
echo "<pre>";
$file = dirname(__FILE__) . "/example1.xml";
$reader = new ExampleXmlReader1;
$reader->open($file);
$reader->parse();
$reader->close();
```

Рисунок 8 - Пример использования SimpleXMLReader. Часть 2.

Как видим на основе существующих инструментов и библиотек можно создавать новые инструменты, используя лучшие стороны каждого. К

недостаткам данной библиотеки стоит отнести отсутствие документации и «молодой возраст».

Библиографический список

1. Горбунов А.А., Хорошко М.Б. Использование расширения simplexml на основе языка программирования php // Информационные и измерительные системы и технологии. Новочеркасск: ООО "Лик", 2016. С. 34-36.
2. Яковлев Д.С. Использование xml-документов в качестве баз данных для вэб-приложений // Вестник магистратуры. 2015. № 4-1 (43). С. 8-10.
3. Горбунов А.А. Подсистема информационного взаимодействия с amazon // Информационные и измерительные системы и технологии. Новочеркасск: Лик, 2016. С. 63-66.
4. Wang F., Li J., Homayounfar H. A space efficient XML DOM parser // Data & Knowledge Engineering. 2007. Т.60. №1. С. 185-207
5. SimpleXML // PHP URL: http://php.net/manual/ru/book.simplexml.php (дата обращения: 25.01.2019).
6. XMLReader // PHP URL: http://php.net/manual/ru/book.xmlreader.php (дата обращения: 13.01.2019).
7. Глаголев В.А Разработка банка данных метеорологических параметров для анализа пожарной опасности территории // Региональные проблемы. 2007. №8. С. 152-155.
8. Глаголев В.А. Создание баз данных для оценки и прогноза пожарной опасности растительности по природно-антропогенным условиям // Региональные проблемы. 2014. Т. 17. № 2. С. 78-83.
9. GitHub // SimpleXMLReader URL: https://github.com/dkrnl/SimpleXMLReader (дата обращения: 13.01.2019).
10. Малышев В.А., Голубь И.С., Глаголев В.А.Обработка и импорт больших xml-данных партнерских сайтов // Постулат. 2018. № 5-1 (31). С. 15.