ПАВЕЛ КОЛЧИН

Практическое руководство

Как написать

Парсер HTML на PHP

и заработать на этом деньги

© Записки Предприимчивого Человека, 2017

Практическое руководство «Парсеры HTML на PHP»

Вступление 3

1. Простой парсер HTML контента сайта на PHP 5

2. Практические примеры PHP парсеров 18

    2.1. Применение cURL в PHP парсерах сайтов 18

    2.2. PHP парсер с авторизацией 23

    2.3. AJAX парсер без перезагрузки страниц 43

    2.4. Парсер страниц с асинхронной подгрузкой данных 55

    2.5. Парсер картинок с сайта 69

    2.6. PHP парсер без регулярных выражений 83

    2.7. PHP парсер CSV. Импорт из CSV или экспорт в CSV 95

    2.8. Парсер лайков из ВКонтакте 105

3. Импорт записей в WordPress из CSV 114

4. Как заработать на PHP парсерах? 119

Заключение 120

Вступление

Всем большой привет!

Более 10 лет занимаюсь веб-разработкой. За эти годы набрался практического опыта в области написания парсеров различного назначения.

Долго думал, как назвать эту электронную книгу, это же не учебник и не роман какой-нибудь. В один момент появился в голове вариант «практическое руководство»… Прочитал определение в Яндекс.Словарях. Да, это же то, что нужно. Теории в данной электронной книге очень мало. Я попытался сосредоточиться на примерах решения задач, которые часто встречаются перед программистами в их повседневной работе.

Сразу предупреждаю, что код в примерах не является идеальным, ни по алгоритмам, ни по стилю кодирования, не применяются подходы ООП. Цель немного другая, показать общие особенности разработки парсеров, как могут быть решены разные задачи из практики. А «красоту» коду можно навести уже по мере необходимости. На первом этапе важно найти общий путь решения проблемы.

Примеры кода проверялись на работоспособность перед публикацией. На текущий момент я не могу гарантировать, что все скрипты корректно работают. Это одна из особенностей парсеров – они жёстко зависят от своих источников данных.

В 2012 году я пробовал написать универсальный парсер HTML, но по факту, универсальным у него было только название. Для успешной работы с этим скриптом требовались знания регулярных выражений и навыки программирования. Я подумал, подумал и решил, что более универсальным решением будет показать пример парсера на PHP и рассказать, как он работает. Тогда программисты, которые ранее не писали парсеров, смогут решить свои задачи. А заказчики смогут понять возможности PHP в области парсинга сайтов и что реально можно требоватьот разработчиков.

Так появилась серия статей по парсерам на моём блоге http://seorubl.ru/. На основе этих публикаций я собрал книжку в формате PDF.

В 1-й части показан базовый пример парсера на PHP и разобраны основные этапы его работы. Уже после изучения 1-й главы вы сможете написать свой собственный парсер текстового контента.

Во 2-й части я разобрал самые разные примеры парсеров с авторизацией и без ней, с применением библиотеки cURL, для сбора картинок. Работа с сайтами на основе технологии AJAX. Парсинг информации из социальной сети ВКонтакте и т.п.

В 3-й части показан пример имопрта данных, полученных в результате парсинга, на сайт.

В 4-й части я рассказываю о том, как можно заработать на полученных с помощью данного руководства знаниях и навыках.

К каждому варианту парсера приведён листинг исходного кода. Скрипт обычно состоит из одного файла, в котором совмешены PHP, HTML, JS, CSS. Я это сделал умышленно, чтобы можно было легко скопировать код в один файл и быстро запустить парсер. Но весь код в примерах сгруппирован так, что его очень легко разделить по разным типам файлам или даже переделать под архитектуру MVC.

В тексте книги встречаются ссылки, которые ведут на оригинальные статьи моего сайта Записки ПЧ.

Читайте, изучайте и зарабатывайте.

2. Простой парсер HTML контента сайта на PHP

Парсер — это программа, которая анализирует входные текстовые данные, извлекает нужную информацию и на основе полученных данных выдаёт результат в заданном формате.

Общий алгоритм PHP парсинга предполагает, что ваш скрипт делает запрос по заданному адресу, получает ответ от сервера в виде HTML страницы, либо в каком-то другом текстовом формате, например CSV, JSON, XML. Далее полученная информация анализируется, из неё извлекаются (парсятся) нужные данные, на основе которых формируется результат. Полученные данные можно вывести на экран, либо записать в файл или БД.

Пример простого PHP парсера HTML контента

Предположим, нам нужно спарсить цену товара на сайте gearbest.com. Скрипт считывает заданную страницу, потом посредством регулярных выражений анализирует её текстовый контент и выделяет нужные нам куски HTML кода. Далее полученный результат выводится на экран.

```
<?php
Error_Reporting(E_ALL & ~E_NOTICE);
mb_internal_encoding("UTF-8");
set_time_limit(0);    // Попытка установить своё
время выполнения скрипта

/* --- 1 --- Инициализируем переменные для запроса */
        $time_start = time();
        $error = array();
        $error_page = array();
        $action = 0;
        $gearbest_url = "";
        $charset = "UTF-8";    // Исходная кодировка
страницы
```

```
        $uni_name = date("d-m-Y-H-i-s", time());

/* --- 1.1 --- Переопределяем переменные на основе GET
или POST параметров */
        if(isset($_REQUEST['gearbest_url']))
            $gearbest_url =
trim($_REQUEST['gearbest_url']);
        if(isset($_REQUEST['action']))
            $action = $_REQUEST['action'];

/* --- 1.2 --- Загрузка страницы при помощи cURL */
    function curl_get_contents($page_url, $base_url,
$pause_time, $retry) {
        /*
        $page_url - адрес страницы-источника
        $base_url - адрес страницы для поля REFERER
        $pause_time - пауза между попытками парсинга
        $retry - 0 - не повторять запрос, 1 - повторить
запрос при неудаче
        */
        $error_page = array();
        $ch = curl_init();
        curl_setopt($ch, CURLOPT_USERAGENT,
"Mozilla/5.0 (Windows NT 6.1; WOW64; rv:38.0)
Gecko/20100101 Firefox/38.0");
        //curl_setopt($ch, CURLOPT_COOKIEJAR,
str_replace("\\", "/", getcwd()).'/gearbest.txt');
        //curl_setopt($ch, CURLOPT_COOKIEFILE,
str_replace("\\", "/", getcwd()).'/gearbest.txt');
        curl_setopt($ch, CURLOPT_FOLLOWLOCATION, 1); //
```

```
Автоматом идём по редиректам
        curl_setopt ($ch, CURLOPT_SSL_VERIFYPEER, 0);
// Не проверять SSL сертификат
        curl_setopt ($ch, CURLOPT_SSL_VERIFYHOST, 0);
// Не проверять Host SSL сертификата
        curl_setopt($ch, CURLOPT_URL, $page_url); //
Куда отправляем
        curl_setopt($ch, CURLOPT_REFERER, $base_url);
// Откуда пришли
        curl_setopt($ch, CURLOPT_HEADER, 0);
        curl_setopt($ch, CURLOPT_RETURNTRANSFER, 1); //
Возвращаем, но не выводим на экран результат
        $response['html'] = curl_exec($ch);
        $info = curl_getinfo($ch);
        if($info['http_code'] != 200 &&
$info['http_code'] != 404) {
            $error_page[] = array(1, $page_url,
$info['http_code']);
            if($retry) {
                sleep($pause_time);
                $response['html'] = curl_exec($ch);
                $info = curl_getinfo($ch);
                if($info['http_code'] != 200 &&
$info['http_code'] != 404)
                    $error_page[] = array(2, $page_url,
$info['http_code']);
            }
        }
        $response['code'] = $info['http_code'];
        $response['errors'] = $error_page;
```

```
        curl_close($ch);
        return $response;
    }
/* --- 1.3 --- Парсинг цены */
    function get_gearbest_price($gearbest_url) {
        /*
        $gearbest_url - адрес страницы товара на
Gearbest
        */
        $res_arr = array();
        $res_arr['price_list'] = array();
        $base_url = "https://www.gearbest.com";
        $response_arr = curl_get_contents($gearbest_url,
$base_url, 5, 1);
        $page = $response_arr['html'];
        $page_code = $response_arr['code'];
        $res_arr['error_page'] =
$response_arr['errors'];
    //file_put_contents('page.txt', $page);
    //$page = file_get_contents('page.txt');
        if(!empty($page) && $page_code == 200) {
            if($charset != "UTF-8") {
                $page = iconv("WINDOWS-1251", "UTF-
8//IGNORE", $page);
            }
            $buffer = array();
            $regexp = "/(id=\"unit_price\"\s*data-
orgp=\"(.*)\")|(class=\"my_shop_price
new_shop_price\"\s*data-
orgp=\"(.*?)\"\s*id=\"unit_price\")/Us";
```

```
            preg_match($regexp, $page, $buffer);
            $res_arr['price_list']['price'] =
$buffer[2] ? $buffer[2] : $buffer[4];
            $regexp = "/bizhong=\"(.*)\">/Us";
            $buffer = array();
            preg_match($regexp, $page, $buffer);
            $res_arr['price_list']['currency'] =
$buffer[1];
            $res_arr['error'] = '';
        } else {
            $res_arr['price'] = 0;
            $res_arr['currency'] = 'nodata';
            $res_arr['error'][] = 'Ошибка загрузки
страницы';
        }
        return $res_arr;
    }
/* --- 1.4 --- Вывод данных в HTML */
/* --- 1.4.1 --- Вывод полученых цен */
    function price_list_html($price_list) {
        echo '<p>Цена: ' . $price_list['price'] . '
<span class=\"currency\">' . $price_list['currency'] .
'</span></p>';
    }
/* --- 1.4.2 --- Вывод ошибок */
    function error_list_html($error) {
        if (!empty($error)) {
            echo "<p>Во время обработки запроса
произошли следующие ошибки:</p>\n";
            echo "<ul>\n";
```

```
            foreach($error as $error_row) {
                echo "<li>" . $error_row . "</li>\n";
            }
            echo "</ul>\n";
            echo "<p>Статус: <span
class=\"red\">FAIL</span></p>\n";
        } else {
            echo "<p>Статус: <span
class=\"green\">OK</span></p>\n";
        }
    }
/* --- 1.4.3 --- Вывод ошибок загрузки страниц */
    function error_page_list_html($error_page) {
        if (!empty($error_page)) {
            echo "<ul>\n";
            foreach($error_page as $error_row) {
                echo "<li>[" . $error_row[0] . "] " .
$error_row[1] . " - " . $error_row[2] . "</li>\n";
            }
            echo "</ul>\n";
        }
    }
/* --- 1.4.4 --- Вывод работы скрипта */
    function run_time_html($time_start) {
        if(!empty($time_start))
            echo "<!--p>Время работы: " . (time() -
$time_start) . "</p-->\n";
    }
/* --- 2 --- Получение контента из каталога Gearbest */
        if($action) { // если ошибок нет и данные формы
```

```
поиска получены
            if(!empty($gearbest_url)) {
                $gearbest_url = trim($gearbest_url);
                $din_url =  $gearbest_url;
                $res_arr = get_gearbest_price($din_url);
                $price_list = $res_arr['price_list'];
                $error_page = $res_arr['error_page'];
                $error = $res_arr['error'];
            } else {
                $error[] = "Не задан адрес страницы с
товаром";
            }
        }
/* --- 3 --- Вывод результатов работы парсера */
    ?>
    <!doctype html>
    <html>
    <head>
        <title>Парсер цены товара на
Gearbest.com</title>
        <meta http-equiv="Content-Type"
content="text/html; charset=utf-8">
        <!--meta name="robots"
content="noindex,nofollow"-->
    </head>
    <body>
    <style>
    .wrapper {
        max-width: 600px;
        margin: 0 auto;
```

```
}
h1 {
    text-align: center;
}
.action_form {
    max-width: 560px;
    margin: 0 auto;
}
.action_form input {
    width: 100%;
}
input[type="text"] {
    font-size: 1em;
    min-height: 36px;
    box-sizing: border-box;
}
input[type="submit"] {
    padding: 8px 12px;
    margin: 12px auto;
    font-size: 1.2em;
    font-weight: 400;
    line-height: 1.2em;
    text-decoration: none;
    display: inline-block;
    cursor: pointer;
    border: 2px solid #007700;
    border-radius: 2px;
    background-color: transparent;
    color: #007700;
}
```

```
input[type="submit"]:hover {
    background-color: #009900;
    color: #fff;
}
.result {
    border: 1px dotted #000;
    width: 100%;
    height: auto;
    overflow-y: auto;
    margin: 0px auto;
    padding: 10px;
}
.copyright {
    text-align: center;
}
.copyright a {
    color: #000;
}
.copyright a:hover {
    text-decoration: none;
}
.red {
    color: #770000;
}
.green {
    color: #007700;
}
</style>
<div class="wrapper">
    <h1>Парсер цены товара на Gearbest.com</h1>
```

```
        <form class="action_form" action="" method="post">
            <input type="hidden" name="action" value="1" />
            <input type="text" name="gearbest_url" value="<?php if(!empty($gearbest_url)) echo $gearbest_url; ?>" placeholder="URL страницы товара" />
            <input type="submit" name="submit" value="Старт" />
        </form>
        <div class="result">
    <?php
            if($action && !empty($price_list)) {
                price_list_html($price_list);
            }
    ?>
        </div>
        <div class="errors_block">
    <?php
        error_page_list_html($error_page);
        error_list_html($error);
        run_time_html($time_start);
    ?>
        </div>
        <div class="copyright">&copy Идея и реализация - <a href="http://seorubl.ru/" target="_blank" title="Записки Предприимчивого Человека" rel="generator">ПЧ</a> // 16.04.2017 г.</div>
    </div>
    </body>
```

```
</html>
```

index.php — основной файл PHP скрипта парсера. Код парсера актуален на момент публикации. Со временем HTML код сайта источника может меняться и регулярные выражения уже не будут к нему подходить.

Существуют разные способы установки скрипта. Я работал с ним из-под XAMPP. Но можно парсер запускать прямо с виртуального хостинга. Просто заливаете файл *index.php* к себе на сайт в какую-либо папку и обращаетесь к нему через адресную строку браузера. Предположим, что вы закинули скрипт в папку my-*parser* в корневой директории вашего хостинга. Тогда в адресной строке нужно набрать URL: *http://вашдомен.ru/my-parser/*.

Скриншот главной страницы парсера цены с сайта gearbest.com:

1. На главной странице парсера мы должны ввести адрес страницы товара. После нажатия на кнопку «Старт» страница перезагружается, отправляются данные формы на сервер и PHP скрипт делает запрос по заданному адресу с помощью библиотеки cURL.

За это действие отвечает функция *curl_get_contents()*, которая является аналогом стандартной PHP функции *file_get_contents()*, но с расширенным на основе cURL функционалом.

cURL — это расширение для PHP, которое обеспечивает поддержку библиотеки функций libcurl. Данный набор функций позволяет формировать

POST и PUT запросы, скачивать файлы. Поддерживаются различные протоколы http, https, ftp и пр. Можно использовать прокси-серверы, cookies и аутентификацию пользователей. В общем, отличный инструмент для имитации действий пользователя в браузере.

cURL штука для разработки HTML парсеров очень полезная, и в одной из следующих статей я более подробно расскажу о приёмах работы с ней для целей парсинга.

2. Далее в функции *get_gearbest_price()* с помощью встроенной в PHP функции *preg_match()* и регулярных выражений проводится анализ ответной HTML страницы, полученной от стороннего сервера.

```
<!--<p class="from">By<a href="" class="brand-name">CUBOT</a><span class="badge"></span></p>-->
▶ <p class="shortTitle"></p>
▶ <div class="g_review clearfix"></div>
▶ <p class="warehouse_options"></p>
▶ <div class="wh-ship-tip"></div>
▶ <div class="price_op clearfix"></div>
▼ <div class="goods_price">
    <!--邮件专享价-->
  ▶ <span id="js_emailOnly" class="emailOnly" style="display: none;"></span>
    <span class="bizhong">USD</span>
  ▶ <span id="unit_price" class="my_shop_price new_shop_price" data-orgp="149.99"></span>
    <span class="freeShip p15"></span>
  </div>
  <!--flash goods限数量-->
  <!--end of flash goods限数量-->
  <!--add new phone save price-->
▶ <div class="g_operatNum"></div>
▶ <p class="g_cateList clearfix"></p>
  <!--***隐藏属性开始***-->
▶ <p id="select_available_stock_id" style="display:none;"></p>
▶ <p style="display:none"></p>
```

Обратите внимание, что скрипт видит страницу в текстовом формате и анализировать предстоит именно её HTML код.

3. На следующем этапе формируется и выводится на экран результат, полученный на основе данных после парсинга. В нашем случае это валюта и цена товара.

Таким образом, для успешной разработки парсеров на PHP программисту нужно уметь работать с библиотекой функций cURL и регулярными выражениями PHP.

Как парсить зашифрованные данные

В некоторых случаях сервера отдают HTML страницы в сжатом или защифрованном виде, например Accept-Encoding: gzip. При этом смена поддерживаемых форматов сжатия в запросе может не влиять на формат ответа.

В таких случаях нужно расшифровать ответ, например, стандартной PHP функцией *gzdecode()*. И дальше можно будет работать по старой схеме.

Данные, заширфованные по алгоритму base64 можно расшифровать функцией *base64_encode()*.

В следующей части практисческого руководства я покажу примеры реализации более сложных парсеров с авторизацией, сбор фото, работа с AJAX, парсинг ВК и т.п.

2. Практические примеры PHP парсеров

2.1. Применение cURL в PHP парсерах сайтов

В этой статье разберём одну из ключевых деталей любого парсера сайтов. Это составление запросов к серверу источнику информации и получение ответа от него.

Один из самых простых вариантов воспользоваться PHP функцией *file_get_contents()*, но часто этого бывает мало, так как многие сайты для того, чтобы отдать нужный контент требуют отправить запрос с определёнными заголовками или куками. Также нам бывает, нужна информация о заголовках в ответе сервера.

Для более сложных случаев удобнее составлять запросы с помощью функции из модуля cURL. *cURL* — это модуль для PHP, обеспечивающий поддержку библиотеки функций libcurl. Эта библиотека позволяет формировать POST или PUT запросы к веб-серверам, скачивать файлы. Набор функций поддерживает различные протоколы обмена http, https, ftp и пр. Расширение cURL позволяет работать с поддержкой прокси-серверов, cookies и проводить аутентификацию пользователей. Получается хороший инструмент для имитации действий пользователя в браузере.

Для разработки парсеров я написал аналог функции *file_get_contents()* с помощью cURL.

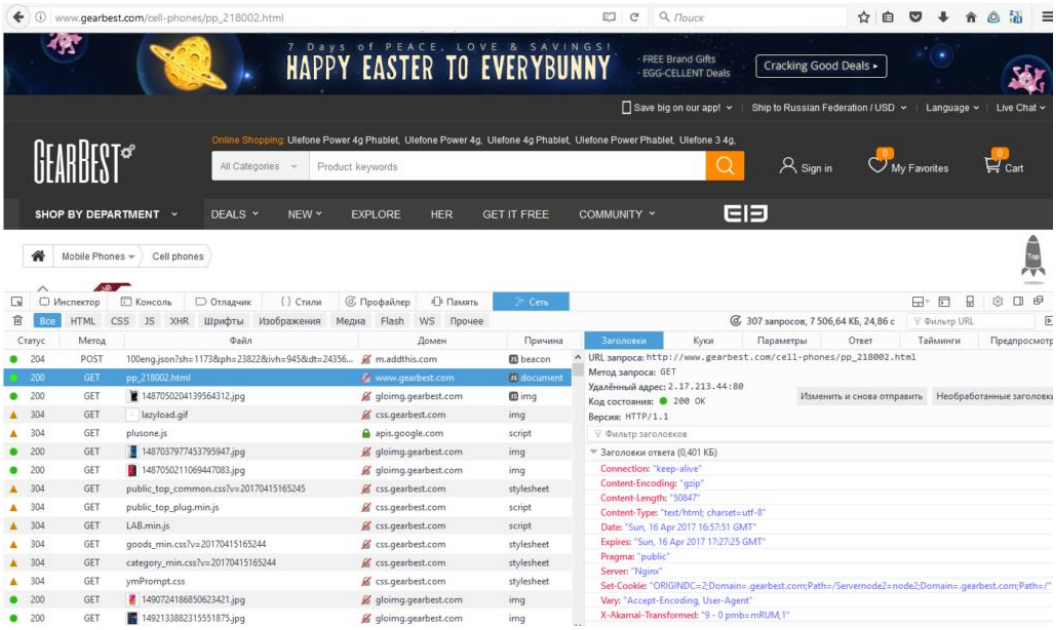
```
/* --- 1.2 --- Загрузка страницы при помощи cURL */
    function curl_get_contents($page_url, $base_url,
$pause_time, $retry) {
        /*
        $page_url - адрес страницы-источника
        $base_url - адрес страницы для поля REFERER
        $pause_time - пауза между попытками парсинга
        $retry - 0 - не повторять запрос, 1 -
```

```
повторить запрос при неудаче
        */
        $error_page = array();
        $ch = curl_init();
        curl_setopt($ch, CURLOPT_USERAGENT,
"Mozilla/5.0 (Windows NT 6.1; WOW64; rv:38.0)
Gecko/20100101 Firefox/38.0");
        curl_setopt($ch, CURLOPT_COOKIEJAR,
str_replace("\\", "/", getcwd()).'/gearbest.txt');
        curl_setopt($ch, CURLOPT_COOKIEFILE,
str_replace("\\", "/", getcwd()).'/gearbest.txt');
        curl_setopt($ch, CURLOPT_FOLLOWLOCATION, 1);
// Автоматом идём по редиректам
        curl_setopt ($ch, CURLOPT_SSL_VERIFYPEER, 0);
// Не проверять SSL сертификат
        curl_setopt ($ch, CURLOPT_SSL_VERIFYHOST, 0);
// Не проверять Host SSL сертификата
        curl_setopt($ch, CURLOPT_URL, $page_url); //
Куда отправляем
        curl_setopt($ch, CURLOPT_REFERER, $base_url);
// Откуда пришли
        curl_setopt($ch, CURLOPT_HEADER, 0);
        curl_setopt($ch, CURLOPT_RETURNTRANSFER, 1);
// Возвращаем, но не выводим на экран результат
        $response['html'] = curl_exec($ch);
        $info = curl_getinfo($ch);
        if($info['http_code'] != 200 &&
$info['http_code'] != 404) {
            $error_page[] = array(1, $page_url,
$info['http_code']);
```

```
            if($retry) {
                sleep($pause_time);
                $response['html'] = curl_exec($ch);
                $info= curl_getinfo($ch);
                if($info['http_code'] != 200 &&
$info['http_code'] != 404)
                    $error_page[] = array(2,
$page_url, $info['http_code']);
            }
        }
        $response['code'] = $info['http_code'];
        $response['errors'] = $error_page;
        curl_close($ch);
        return $response;

    }
```

Разберём детали работы функции curl_get_contents($page_url, $base_url, $pause_time, $retry). Основная её задача отвравить запрос по заданному URL ($page_url) и получить ответ от сервера.

С помощью функции curl_setopt() мы задаём нужные нам заголовки запроса. Например, через *CURLOPT_USERAGENT* настраивается User-Agent, можно менять тип браузера, из-под которого якобы делается запрос.



Подробно изучить параметры и заголовки запросов можно с помощью панели для веб-разрабочиков прямо в браузере по вкладке «Сеть». Очень полезный инструмент для начинающих и опытных разработчиков парсеров.

Параметры заголовков cURL для парсера

Кроме User-Agent для имитации действий реального пользователя полезно задать *CURLOPT_REFERER* — это заголовок Referer, на некоторых сайтах веб-мастера проверяют его наличие, чтобы отфильтровать совсем тупых ботов.

CURLOPT_FOLLOWLOCATION — позволяет переходить по редиректам в автоматическом режиме, то же полезная настройка, чтобы не обрабатывать по отдельности разные 301-редиректы.

CURLOPT_URL — задаёт URL целевой страницы — источника данных.

Настройки *curl_setopt ($ch, CURLOPT_SSL_VERIFYPEER, 0);* — не проверять SSL сертификата и *curl_setopt ($ch, CURLOPT_SSL_VERIFYHOST, 0);* — не проверять Host SSL сертификата, удобно использовать при работе по протоколу HTTPS, когда вам не нужно проверять валидность SSL сертификат у сервера.

CURLOPT_COOKIEJAR и *CURLOPT_COOKIEFILE* — задают пути к файлам для записи и считывания параметров *COOKIE*, нужны, например, для

авторизации на сайте или когда интернет-ресурс хранит какие-либо настройки в файлах куков.

Самые интересные на мой субъективный взгляд настройки для cURL я описал. Детальное описание всех применяемых настроек лучше почитать в более официальных справочниках PHP.

Дальше в функции *curl_get_contents()* идёт отправка запроса и получение ответа в переменную. По заголовкам ответа проверяем, удалось ли достучаться до заданной страницы. Если код ответа 200 или 400, то возвращаем HTML ответ от сервера. При другом коде ответа сервера после некоторой паузы делаем повторный запрос. Это бывает полезно, когда хостинг сайта работает нестабильно и при частых запросах выдаёт 500-ю ошибку.

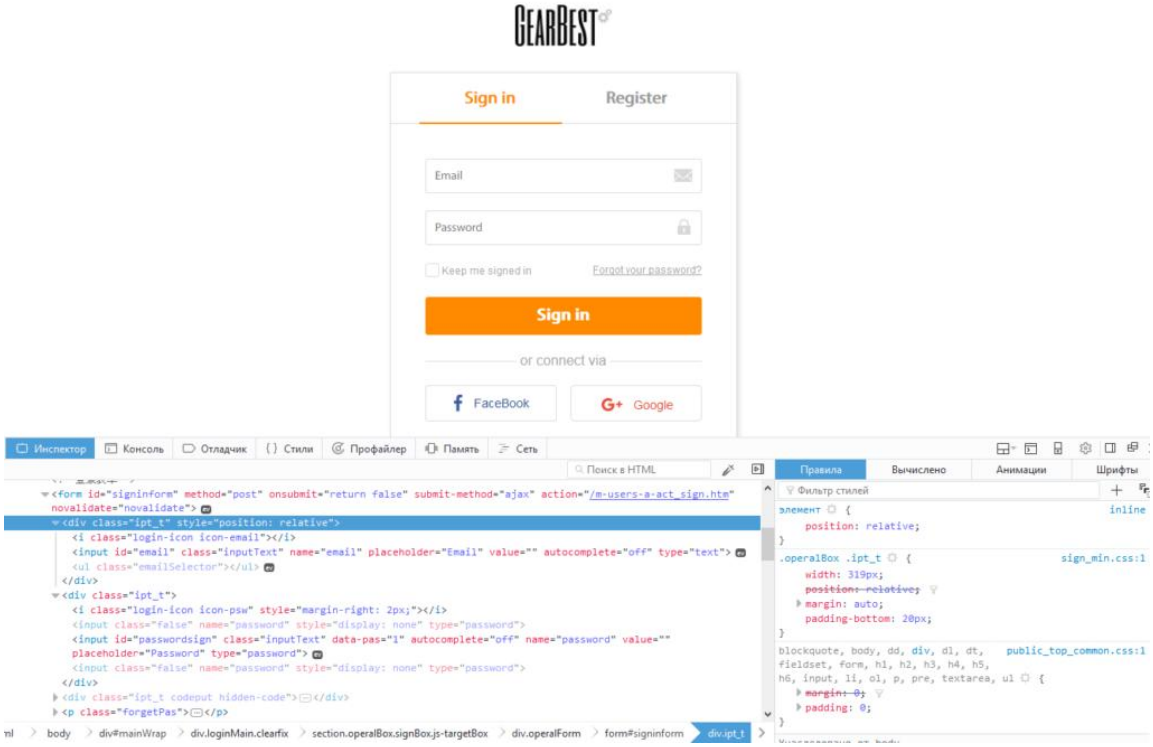
В следующих статьях по теме парсеров рассмотрим варианты скриптов с авторизацией и варианты работы с применением технологии AJAX, не переключайтесь.

2.2. PHP парсер с авторизацией

Часто возникают задачи, требующие получения данных со страниц сайта, которые показываются только залогиненным пользователям. В этом случае нам нужно написать парсер с автоматической авторизацией.

Информация о том, авторизован пользователь или нет, часто хранится в Cookies. Составить правильный запрос и обработать файлы Cookies нам поможет библиотека cURL.

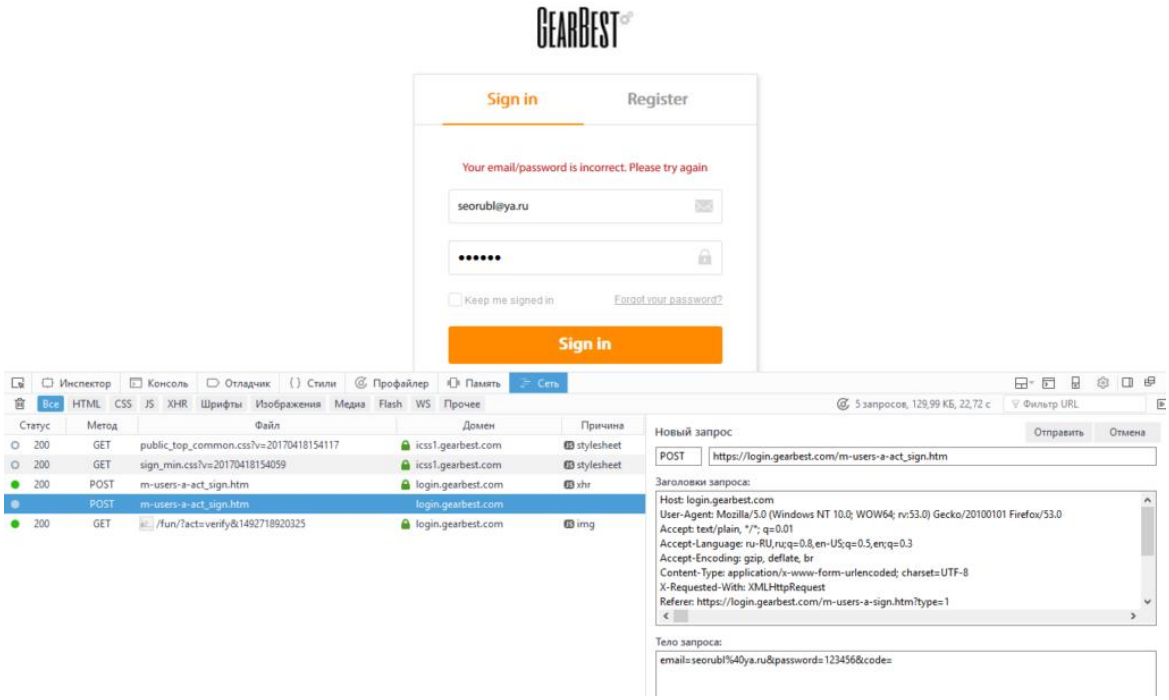
Предварительно нужно понять, как работает механизм авторизации на сайте, какие передаются переменные после отправки данных формы входа. Эти данные можно проанализировать с помощью инструментов разработчика в браузере.



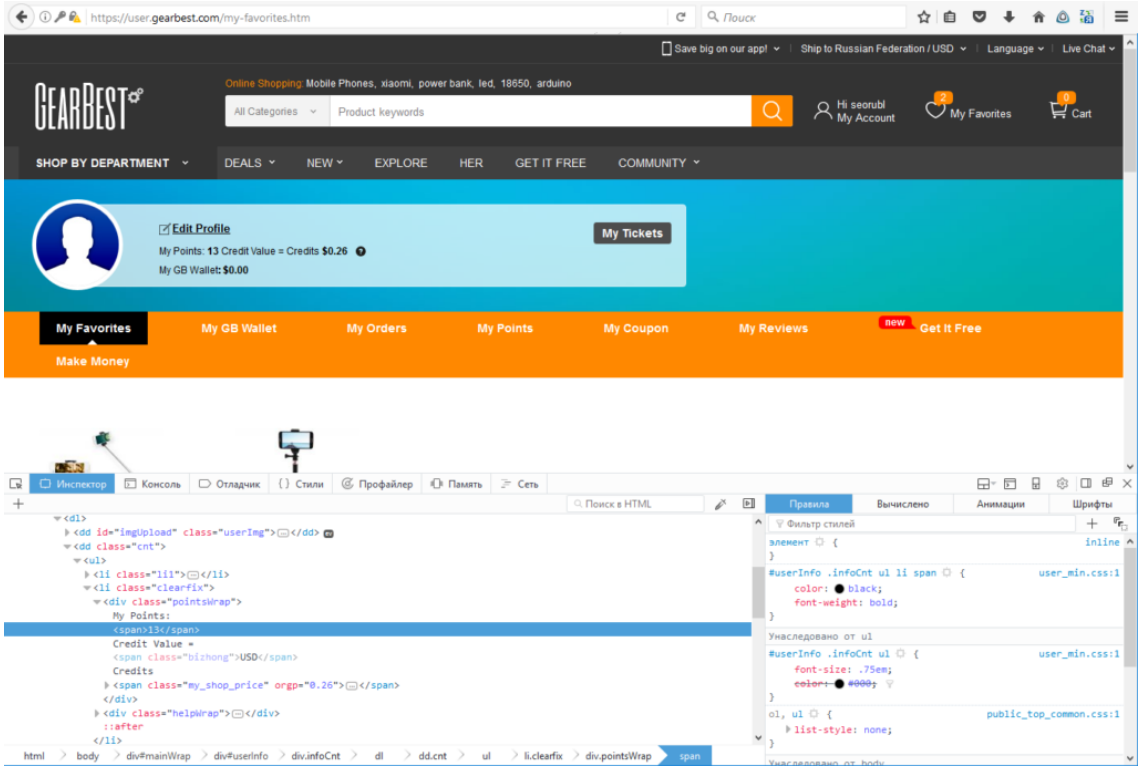
В качестве примера разберём парсер баланса баллов в интернет-магазине Gearbest. В html коде формы входа видим название полей ввода логина и пароля, а также адрес страницы-обработчика: /m-users-a-act_sign.htm.

Далее попытаемся авторизоваться и отследить, какие запросы отправляются на сервер. Введя правильные логин и пароль, мне не удалось успеть увидеть нужные запросы, так как происходит несколько редиректов. А сам запрос авторизации отправляется по AJAX.

Поэтому решил посмотреть, что происходит при вводе неверных данных. В этом случае удалось отследить нужные запросы и структуру переменных.



На следующем шаге смотрим код страницы личного кабинета и прикидываем, как спарсить данные баланса бонусных баллов.



На основе полученной информации можно написать PHP парсер баланса баллов с авторизацией. Я решил дописать ряд функций для парсера из предыдущего примера. Для составления POST запросов используем функции библиотеки cURL.

Полученный в результате код парсера с авторизацией:

```
<?php
Error_Reporting(E_ALL & ~E_NOTICE);
mb_internal_encoding("UTF-8");
set_time_limit(0);  // Попытка установить своё
время выполнения скрипта

/* --- 1 --- Инициализируем переменные для запроса */
        $time_start = time();
        $error = array();
        $error_page = array();
        $action = 0;
```

```
        $gearbest_url = "";
        $charset = "UTF-8"; // Исходная кодировка
страницы
        $uni_name = date("d-m-Y-H-i-s", time());

    /* --- 1.1 --- Переопределяем переменные на основе
GET или POST параметров */
        if(isset($_REQUEST['gearbest_url']))
            $gearbest_url =
trim($_REQUEST['gearbest_url']);
        if(isset($_REQUEST['action']))
            $action = $_REQUEST['action'];

/* --- 1.2 --- Запросы при помощи cURL */
/* --- 1.2.1 --- Загрузка страницы при помощи cURL */
    function curl_get_contents($page_url, $base_url,
$pause_time, $retry) {
        /*
        $page_url - адрес страницы-источника
        $base_url - адрес страницы для поля REFERER
        $pause_time - пауза между попытками парсинга
        $retry - 0 - не повторять запрос, 1 - повторить
запрос при неудаче
        */
        $error_page = array();
        $ch = curl_init();
        curl_setopt($ch, CURLOPT_USERAGENT,
"Mozilla/5.0 (Windows NT 6.1; WOW64; rv:38.0)
Gecko/20100101 Firefox/38.0");
        curl_setopt($ch, CURLOPT_COOKIEJAR,
```

```
str_replace("\\", "/", getcwd()).'/gearbest.txt');
        curl_setopt($ch, CURLOPT_COOKIEFILE,
str_replace("\\", "/", getcwd()).'/gearbest.txt');
        curl_setopt($ch, CURLOPT_FOLLOWLOCATION, 1); //
Автоматом идём по редиректам
        curl_setopt ($ch, CURLOPT_SSL_VERIFYPEER, 0);
// Не проверять SSL сертификат
        curl_setopt ($ch, CURLOPT_SSL_VERIFYHOST, 0);
// Не проверять Host SSL сертификата
        curl_setopt($ch, CURLOPT_URL, $page_url); //
Куда отправляем
        curl_setopt($ch, CURLOPT_REFERER, $base_url);
// Откуда пришли
        curl_setopt($ch, CURLOPT_HEADER, 0);
        curl_setopt($ch, CURLOPT_RETURNTRANSFER, 1); //
Возвращаем, но не выводим на экран результат
        $response['html'] = curl_exec($ch);
        $info= curl_getinfo($ch);
        if($info['http_code'] != 200 &&
$info['http_code'] != 404) {
            $error_page[] = array(1, $page_url,
$info['http_code']);
            if($retry) {
                sleep($pause_time);
                $response['html'] = curl_exec($ch);
                $info= curl_getinfo($ch);
                if($info['http_code'] != 200 &&
$info['http_code'] != 404)
                    $error_page[] = array(2, $page_url,
$info['http_code']);
```

```
            }
        }
        $response['code'] = $info['http_code'];
        $response['errors'] = $error_page;
        curl_close($ch);
        return $response;
    }
    /* --- 1.2.2 --- Авторизация при помощи cURL */
    function curl_login($page_url, $base_url,
$user_data) {
        /*
        $page_url - адрес страницы-источника
        $base_url - адрес страницы для поля REFERER
        $user_data - данные для авторизации
        */
        $error_page = array();
        $ch = curl_init();
        curl_setopt($ch, CURLOPT_USERAGENT,
"Mozilla/5.0 (Windows NT 6.1; WOW64; rv:38.0)
Gecko/20100101 Firefox/38.0");
        curl_setopt($ch, CURLOPT_COOKIEJAR,
str_replace("\\", "/", getcwd()).'/gearbest.txt');
        curl_setopt($ch, CURLOPT_COOKIEFILE,
str_replace("\\", "/", getcwd()).'/gearbest.txt');
        curl_setopt($ch, CURLOPT_FOLLOWLOCATION, 1); //
Автоматом идём по редиректам
        curl_setopt ($ch, CURLOPT_SSL_VERIFYPEER, 0);
// Не проверять SSL сертификат
        curl_setopt ($ch, CURLOPT_SSL_VERIFYHOST, 0);
// Не проверять Host SSL сертификата
```

```
        curl_setopt($ch, CURLOPT_URL, $page_url); //
Куда отправляем
        curl_setopt($ch, CURLOPT_REFERER, $base_url);
// Откуда пришли
        curl_setopt($ch, CURLOPT_POSTFIELDS,
http_build_query($user_data));
        curl_setopt($ch, CURLOPT_HEADER, 0);
        curl_setopt($ch, CURLOPT_RETURNTRANSFER, 1); //
Возвращаем, но не выводим на экран результат
        $response['html'] = curl_exec($ch);
        $info = curl_getinfo($ch);
        if($info['http_code'] != 200 &&
$info['http_code'] != 404) {
            $error_page[] = array(1, $page_url,
$info['http_code']);
        }
        $response['code'] = $info['http_code'];
        $response['errors'] = $error_page;
        curl_close($ch);
        return $response;
    }

/* --- 1.3 --- Функции для Gearbest.com */
/* --- 1.3.1 --- Парсинг цены */
    function get_gearbest_price($gearbest_url) {
        /*
        $gearbest_url - адрес страницы товара на
Gearbest
        */
        $res_arr = array();
```

```
        $res_arr['price_list'] = array();
        $base_url = "https://www.gearbest.com";
        $response_arr = curl_get_contents($gearbest_url,
$base_url, 5, 1);
        $page = $response_arr['html'];
        $page_code = $response_arr['code'];
        $res_arr['error_page'] =
$response_arr['errors'];
    //file_put_contents('page.txt', $page);
    //$page = file_get_contents('page.txt');
        if(!empty($page) && $page_code == 200) {
            if($charset != "UTF-8") {
                $page = iconv("WINDOWS-1251", "UTF-
8//IGNORE", $page);
            }
            $buffer = array();
            $regexp = "/(id=\"unit_price\"\s*data-
orgp=\"(.*)\")|(class=\"my_shop_price
new_shop_price\"\s*data-
orgp=\"(.*?)\"\s*id=\"unit_price\")/Us";
            preg_match($regexp, $page, $buffer);
            $res_arr['price_list']['price'] =
$buffer[2] ? $buffer[2] : $buffer[4];
            $regexp = "/bizhong=\"(.*)\">/Us";
            $buffer = array();
            preg_match($regexp, $page, $buffer);
            $res_arr['price_list']['currency'] =
$buffer[1];
            $res_arr['error'] = array();
        } else {
```

```
            $res_arr['price'] = 0;
            $res_arr['currency'] = 'nodata';
            $res_arr['error'][] = 'Ошибка загрузки
страницы';
        }
        return $res_arr;
    }
    /* --- 1.3.2 --- Авторизация на сайте */
    function get_gearbest_login($user_data) {
        /*
        $gearbest_url - адрес страницы авторизации на
Gearbest
        */
        $res_arr = array();
        $base_url = "https://www.gearbest.com";
        $gearbest_login_url =
'https://login.gearbest.com/m-users-a-act_sign.htm';
        $response_arr = curl_login($gearbest_login_url,
$base_url, $user_data);
        $page = $response_arr['html'];
        $page_code = $response_arr['code'];
        $res_arr['error_page'] =
$response_arr['errors'];
    //file_put_contents('page.txt', $page);
    //$page = file_get_contents('page.txt');
        if(!empty($page) && $page_code == 200) {
            if($charset != "UTF-8") {
                $page = iconv("WINDOWS-1251", "UTF-
8//IGNORE", $page);
            }
```

```
            $res_arr['error'] = array();
            $res_arr['login_status'] = $page;
        } else {
            $res_arr['error'][] = 'Ошибка загрузки
страницы';
        }
        return $res_arr;
    }
    /* --- 1.3.3 --- Парсинг баллов */
    function get_gearbest_point($gearbest_url) {
        /*
        $gearbest_url - адрес страницы личного кабинета
на Gearbest
        */
        $res_arr = array();
        $res_arr['points'] = 0;
        $base_url = "https://www.gearbest.com";
        $response_arr = curl_get_contents($gearbest_url,
$base_url, 5, 1);
        $page = $response_arr['html'];
        $page_code = $response_arr['code'];
        $res_arr['error_page'] =
$response_arr['errors'];
    //file_put_contents('page.txt', $page);
    //$page = file_get_contents('page.txt');
        if(!empty($page) && $page_code == 200) {
            if($charset != "UTF-8") {
                $page = iconv("WINDOWS-1251", "UTF-
8//IGNORE", $page);
            }
```

```
            $buffer = array();
            $regexp = "/<div class=\"pointsWrap\">\s+My
Points:\s<span>(\d+)<\/span>/Us";
            preg_match($regexp, $page, $buffer);
            $res_arr['points'] = $buffer[1];
            $res_arr['error'] = array();
        } else {
            $res_arr['points'] = 0;
            $res_arr['error'][] = 'Ошибка получения
баланса баллов';
        }
        return $res_arr;
    }
/* --- 1.4 --- Вывод данных в HTML */
/* --- 1.4.1 --- Вывод полученых цен */
    function price_list_html($price_list) {
        echo '<p>Цена: ' . $price_list['price'] . '
<span class=\"currency\">' . $price_list['currency'] .
'</span></p>';
    }
/* --- 1.4.2 --- Вывод статуса авторизации */
    function login_status_html($login_status) {
        echo '<p>Статус авторизации: ' . $login_status .
'</p>';
    }
/* --- 1.4.3 --- Вывод баланса баллов */
    function points_balans_html($points) {
        echo '<p>Ваша баллы: ' . $points . '</p>';
    }
/* --- 1.4.4 --- Вывод ошибок */
```

```
    function error_list_html($error) {
        if (!empty($error)) {
            echo "<p>Во время обработки запроса
произошли следующие ошибки:</p>\n";
            echo "<ul>\n";
            foreach($error as $error_row) {
                echo "<li>" . $error_row . "</li>\n";
            }
            echo "</ul>\n";
            echo "<p>Статус: <span
class=\"red\">FAIL</span></p>\n";
        } else {
            echo "<p>Статус: <span
class=\"green\">OK</span></p>\n";
        }
    }
/* --- 1.4.5 --- Вывод ошибок загрузки страниц */
    function error_page_list_html($error_page) {
        if (!empty($error_page)) {
            echo "<ul>\n";
            foreach($error_page as $error_row) {
                echo "<li>[" . $error_row[0] . "] " .
$error_row[1] . " - " . $error_row[2] . "</li>\n";
            }
            echo "</ul>\n";
        }
    }
    /* --- 1.4.7 --- Вывод времени работы скрипта */
    function run_time_html($time_start) {
        if(!empty($time_start))
```

```
            echo "<!--p>Время работы: ". (time() -
$time_start) . "</p-->\n";
    }
/* --- 2 --- Получение контента из сайта Gearbest */
        if($action == 1) { // Показать цену товара
            if(!empty($gearbest_url)) {
                $gearbest_url = trim($gearbest_url);
                $din_url =  $gearbest_url;
                $res_arr = get_gearbest_price($din_url);
                $price_list = $res_arr['price_list'];
                $error_page = $res_arr['error_page'];
                $error = array_merge($error,
$res_arr['error']);
            } else {
                $error[] = "Не задан адрес страницы с
товаром";
            }
        }
        if($action == 2) {  // Показать баланс баллов
            $user_data =
array('email'=>'yourmail@domen.ru',
'password'=>'password'); // Данные для авторизации
            $res_arr = get_gearbest_login($user_data);
            $info_arr['login'] =
$res_arr['login_status'];
            $error = array_merge($error,
$res_arr['error']);
            $gearbest_lk_url =
'https://user.gearbest.com/my-favorites.htm';
            $res_arr =
```

```
get_gearbest_point($gearbest_lk_url);
            $info_arr['points'] = $res_arr['points'];
            $error = array_merge($error,
$res_arr['error']);
        }
/* --- 3 --- Вывод результатов работы парсера */
    ?>
    <!doctype html>
    <html>
    <head>
        <title>Парсер информации на
Gearbest.com</title>
        <meta http-equiv="Content-Type"
content="text/html; charset=utf-8">
        <!--meta name="robots"
content="noindex,nofollow"-->
        <script
src="//ajax.googleapis.com/ajax/libs/jquery/3.1.0/jquer
y.min.js" type="text/javascript"></script>
    </head>
    <body>
    <style>
    .wrapper {
        max-width: 600px;
        margin: 0 auto;
    }
    h1 {
        text-align: center;
    }
    .action_form {
```

```
    max-width: 560px;
    margin: 0 auto;
}
.action_form input {
    width: 100%;
}
input[type="text"] {
    font-size: 1em;
    min-height: 36px;
    box-sizing: border-box;
}
input[type="submit"],
input[type="button"] {
    padding: 8px 12px;
    margin: 12px auto;
    font-size: 1.2em;
    font-weight: 400;
    line-height: 1.2em;
    text-decoration: none;
    display: inline-block;
    cursor: pointer;
    border: 2px solid #007700;
    border-radius: 2px;
    background-color: transparent;
    color: #007700;
}
input[type="submit"]:hover,
input[type="button"]:hover {
    background-color: #009900;
    color: #fff;
```

```
    }
    .result {
        border: 1px dotted #000;
        width: 100%;
        height: auto;
        overflow-y: auto;
        margin: 0px auto;
        padding: 10px;
    }
    .copyright {
        text-align: center;
    }
    .copyright a {
        color: #000;
    }
    .copyright a:hover {
        text-decoration: none;
    }
    .red {
        color: #770000;
    }
    .green {
        color: #007700;
    }
    </style>
    <div class="wrapper">
        <h1>Парсер цены товара на Gearbest.com</h1>
        <form class="action_form" action=""
method="post">
            <input type="hidden" id="action"
```

```
name="action" value="0" />
            <input type="text" name="gearbest_url" value
="<?php if(!empty($gearbest_url)) echo $gearbest_url;
?>" placeholder="URL страницы товара" />
            <input type="button" id="btn_action_price"
name="btn_action_price" value="Цена" />
            <input type="button" id="btn_action_points"
name="btn_action_points" value="Баллы" />
        </form>
        <div class="result">
    <?php
            if($action == 1 && !empty($price_list)) {
                price_list_html($price_list);
            }
    ?>
    <?php
            if($action == 2 && !empty($info_arr)) {
                login_status_html($info_arr['login']);
                points_balans_html($info_arr['points'])
;
            }
    ?>
        </div>
        <div class="errors_block">
    <?php
        error_page_list_html($error_page);
        error_list_html($error);
        run_time_html($time_start);
    ?>
        </div>
```

```
        <div class="copyright">&copy Идея и реализация -
<a href="http://seorubl.ru/" target="_blank"
title="Записки Предприимчивого Человека"
rel="generator">ПЧ</a> // 21.04.2017 г.</div>
    </div>
    <script type="text/javascript">
        jQuery(document).ready(function($) {
            $('#btn_action_price').click(function() {
                $('#action').val('1');
                $('.action_form').trigger("submit");
            });
            $('#btn_action_points').click(function() {
                $('#action').val('2');
                $('.action_form').trigger("submit");
            });
        });
    </script>
    </body>
    </html>
```

Для того чтобы парсер смог авторизоваться, надо ввести свои данные в переменную **$user_data**.

Из значимых доработок парсера стоит отметить написание функции *curl_login()*, которая как раз формирует запрос на авторизацию с отправкой POST параметров на сервер. Также для всех функций были добавлены парметры работы с Cookies. Это нужно для того, чтобы информация о том, что парсер залогинен, передавалась от запроса к запросу.

Пример файла Cookies, сгенерированного cURL:

```
# Netscape HTTP Cookie File
# http://curl.haxx.se/docs/http-cookies.html
# This file was generated by libcurl! Edit at your own
```

```
risk.

.gearbest.com   TRUE    /   FALSE   0   G_SESSIONID
etn15bafqhfiecpjkk3ppdlnv5
.gearbest.com   TRUE    /   FALSE   1495394177
ip_country_code ru
.gearbest.com   TRUE    /   FALSE   1492802177
client_ru_np    deleted
.gearbest.com   TRUE    /   FALSE   1493406977  WEBF-
email  seorubl%40ya.ru
.gearbest.com   TRUE    /   FALSE   1493406977  WEBF-
user_id    12478626
gearbest.com   TRUE    /   FALSE   1523906177  WEBF-
an_num    1
login.gearbest.com  FALSE   /   FALSE   1495394177
avatar
https%3A%2F%2Ficss1.gearbest.com%2Fimagecache%2FGB2%2Fi
mages%2Fdomeimg%2Fscrollimg.gif
.gearbest.com   TRUE    /   FALSE   1492802177  WEBF-
firstname  deleted
.gearbest.com   TRUE    /   FALSE   1492802177  WEBF-
lastname   deleted
.gearbest.com   TRUE    /   FALSE   1493406977
usertype    0
.gearbest.com   TRUE    /   FALSE   0   ORIGINDC    2
.gearbest.com   TRUE    /   FALSE   0   Servernode2
node2
```

После авторизации с помощью функции get_gearbest_login(), мы сразу парсим страницу с листом желаний личного кабинета с помощью функции get_gearbest_point().

Далее выводим статус попытки авторизации и бонусный баланс.

Возможные варианты ответов от сервера gearbest.com при попытке авторизации:

- Successfully sign;
- The email does not exist, please register;
- Your email/password is incorrect. Please try again;
- needCode.

Кстати, если вначале авторизоваться и спарсить баланс баллов, то дальше при парсинге цен на товары, статус авторизованного пользователя будет сохраняться.

Теперь мы знаем, как можно автоматически с помощью PHP и cURL залогиниться на сайте и получить данные со страниц для авторизованных пользователей. В следующих статьях из цикла про разработку парсеров мы рассмотрим примеры применения технологии AJAX для парсинга сайтов.

2.3. AJAX парсер без перезагрузки страниц

Предположим, что нам нужно спарсить несколько страниц интернет-магазина, например, для отслеживания цен на какой-то список товаров. Это можно организвать либо через цикл в скрипте PHP, либо можно сделать несколько AJAX запросов к PHP парсеру без перезагрузки основной страницы. Второй вариант сегодня подробно рассмотрим в данной статье.

Вначале хотел дописать парсер из предыдущей статьи, но потом решил отдельный проект сделать, чтобы не нагромождать всё в одну кучу.

AJAX запросы будем отправлять с помощью JS библиотеки jQuery. Чтобы не таскать за собой лишние файлы подключим её с серверов Google.

В проекте будет 2 файла: index.html с HTML и JS кодом, и ajax-parser.php для формирования и отправки запросов на загрузку страниц с помощью расширения cURL. Специально, что в этом проекте, что в предыдущих примерах, не делю код на разные файлы. Считаю, так удобнее в процессе первого знакомства воспринимать общие алгоритмы работы парсеров.

Код index.html:

```
<!doctype html>
<html>
<head>
        <title>Парсер информации на
Gearbest.com</title>
        <meta http-equiv="Content-Type"
content="text/html; charset=utf-8">
        <!--meta name="robots"
content="noindex,nofollow"-->
        <script
src="//ajax.googleapis.com/ajax/libs/jquery/3.1.0/jquer
y.min.js" type="text/javascript"></script>
```

```
</head>
<body>
<style>
.wrapper {
    max-width: 600px;
    margin: 0 auto;
}
h1 {
    text-align: center;
}
.action_form {
    max-width: 560px;
    margin: 0 auto;
}
.action_form input,
.action_form textarea {
    width: 100%;
}
.action_form textarea {
    min-height: 100px;
}
input[type="text"] {
    font-size: 1em;
    min-height: 36px;
    box-sizing: border-box;
}
input[type="submit"],
input[type="button"] {
    padding: 8px 12px;
    margin: 12px auto;
```

```
        font-size: 1.2em;
        font-weight: 400;
        line-height: 1.2em;
        text-decoration: none;
        display: inline-block;
        cursor: pointer;
        border: 2px solid #007700;
        border-radius: 2px;
        background-color: transparent;
        color: #007700;
    }
    input[type="submit"]:hover,
    input[type="button"]:hover {
        background-color: #009900;
        color: #fff;
    }
    .result {
        border: 1px dotted #000;
        width: 100%;
        height: auto;
        overflow-y: auto;
        margin: 0px auto;
        padding: 10px;
    }
    .copyright {
        text-align: center;
    }
    .copyright a {
        color: #000;
    }
```

```
    .copyright a:hover {
        text-decoration: none;
    }
    .red {
        color: #770000;
    }
    .green {
        color: #007700;
    }
    </style>
    <div class="wrapper">
        <h1>Парсер цены товара на Gearbest.com</h1>
        <form id="action_form" class="action_form"
action="ajax-parser.php" method="post">
            <textarea type="text" id="gearbest_urls"
name="gearbest_urls"></textarea>
            <input type="button" id="btn_action_price"
name="btn_action_price" value="Цены" />
        </form>
        <div id="result" class="result"></div>
        <div id="errors_block"
class="errors_block"></div>
        <div class="copyright">&copy Идея и реализация -
<a href="http://seorubl.ru/" target="_blank"
title="Записки Предприимчивого Человека"
rel="generator">ПЧ</a> // 22.04.2017 г.</div>
    </div>
    <script type="text/javascript">
        jQuery(document).ready(function($) {
            $('#btn_action_price').click(function() {
```

```
                var gearbest_urls =
$("#gearbest_urls").val().split('\n');
                var action_url =
$("#action_form").attr('action');
                if(gearbest_urls) {
                    $('#result').html('');
                    $('#errors_block').html('');
                    gearbest_urls.forEach(function(gear
best_url, i, arr) {
                        var send_data = {gearbest_url:
gearbest_url};
                        $.ajax({
                            type: "POST",
                            url:  action_url,
                            data: send_data,
                            success: function(msg){
                                msg = JSON.parse(msg);
                                // Показываем результат
парсинга
                                $('#result').html($('#r
esult').html() + "<p>[" + (i+1) + "] " + "<span>" +
msg.price_list.price + "</span> <span>" +
msg.price_list.currency + "</span> - <span>" +
gearbest_url + "</span></p>").hide().fadeIn('slow');
                                msg.errors.forEach(func
tion(error, j, errors_arr) {
                                    $('#errors_block').
html($('#errors_block').html() + "<p>[" + (i+1) + "] "
+ msg.errors + "</p>");
                                });
```

```
                                }
                            });
                        });
                    } else {
                        $('#errors_block').html($('#errors_
block').html() + "<p>Введите адреса
товаров</p>").hide().fadeIn('slow');
                    }
                    return true;
                });
            });
    </script>
    </body>
    </html>
```

Код ajax-parser.php

```
<?php
Error_Reporting(E_ALL & ~E_NOTICE);
mb_internal_encoding("UTF-8");
set_time_limit(0);  // Попытка установить своё
время выполнения скрипта

    /* --- 1 --- Инициализируем переменные для запроса
*/
        $time_start = time();
        $gearbest_url = "";
        $charset = "UTF-8"; // Исходная кодировка
страницы
        $uni_name = date("d-m-Y-H-i-s", time());
    /* --- 1.1 --- Переопределяем переменные на основе
GET или POST параметров */
```

```
        if(isset($_REQUEST['gearbest_url'])) {
            $gearbest_url =
trim($_REQUEST['gearbest_url']);
        }
    /* --- 1.2 --- Запросы при помощи cURL */
    /* --- 1.2.1 --- Загрузка страницы при помощи cURL
*/
    function curl_get_contents($page_url, $base_url,
$pause_time, $retry) {
        /*
        $page_url - адрес страницы-источника
        $base_url - адрес страницы для поля REFERER
        $pause_time - пауза между попытками парсинга
        $retry - 0 - не повторять запрос, 1 -
повторить запрос при неудаче
        */
        $error_page = array();
        $ch = curl_init();
        curl_setopt($ch, CURLOPT_USERAGENT,
"Mozilla/5.0 (Windows NT 6.1; WOW64; rv:38.0)
Gecko/20100101 Firefox/38.0");
        curl_setopt($ch, CURLOPT_COOKIEJAR,
str_replace("\\", "/", getcwd()).'/gearbest.txt');
        curl_setopt($ch, CURLOPT_COOKIEFILE,
str_replace("\\", "/", getcwd()).'/gearbest.txt');
        curl_setopt($ch, CURLOPT_FOLLOWLOCATION, 1);
// Автоматом идём по редиректам
        curl_setopt ($ch, CURLOPT_SSL_VERIFYPEER, 0);
// Не проверять SSL сертификат
        curl_setopt ($ch, CURLOPT_SSL_VERIFYHOST, 0);
```

```
// Не проверять Host SSL сертификата
        curl_setopt($ch, CURLOPT_URL, $page_url); //
Куда отправляем
        curl_setopt($ch, CURLOPT_REFERER, $base_url);
// Откуда пришли
        curl_setopt($ch, CURLOPT_HEADER, 0);
        curl_setopt($ch, CURLOPT_RETURNTRANSFER, 1);
// Возвращаем, но не выводим на экран результат
        $response['html'] = curl_exec($ch);
        $info= curl_getinfo($ch);
        if($info['http_code'] != 200 &&
$info['http_code'] != 404) {
            $error_page[] = array(1, $page_url,
$info['http_code']);
            if($retry) {
                sleep($pause_time);
                $response['html'] = curl_exec($ch);
                $info= curl_getinfo($ch);
                if($info['http_code'] != 200 &&
$info['http_code'] != 404)
                    $error_page[] = array(2,
$page_url, $info['http_code']);
            }
        }
        $response['code'] = $info['http_code'];
        $response['errors'] = $error_page;
        curl_close($ch);
        return $response;
    }
    /* --- 1.3 --- Функции для Gearbest.com */
```


По материалам сайта Записки ПЧ http://seorubl.ru/

```
    /* --- 1.3.1 --- Парсинг цены */
    function get_gearbest_price($gearbest_url) {
        /*
        $gearbest_url - адрес страницы товара на
Gearbest
        */
        $res_arr = array();
        $res_arr['price_list'] = array();
        $res_arr['errors'] = array();
        $base_url = "https://www.gearbest.com";
        $response_arr =
curl_get_contents($gearbest_url, $base_url, 5, 1);
        $page = $response_arr['html'];
        $page_code = $response_arr['code'];
        $res_arr['errors'] =
array_merge($res_arr['errors'],
$response_arr['errors']);
    //file_put_contents('page.txt', $page);
    //$page = file_get_contents('page.txt');
        if(!empty($page) && $page_code == 200) {
            if($charset != "UTF-8") {
                $page = iconv("WINDOWS-1251", "UTF-
8//IGNORE", $page);
            }
            $buffer = array();
            $regexp = "/(id=\"unit_price\"\s*data-
orgp=\"(.*)\")|(class=\"my_shop_price
new_shop_price\"\s*data-
orgp=\"(.*?)\"\s*id=\"unit_price\")/Us";
            preg_match($regexp, $page, $buffer);
```

```
            $res_arr['price_list']['price'] =
$buffer[2] ? $buffer[2] : $buffer[4];
            $regexp = "/bizhong=\"(.*)\">/Us";
            $buffer = array();
            preg_match($regexp, $page, $buffer);
            $res_arr['price_list']['currency'] =
$buffer[1];
        } else {
            $res_arr['price_list']['price'] = 0;
            $res_arr['price_list']['currency'] =
'nodata';
            $res_arr['errors'][] = 'Ошибка загрузки
страницы';
        }
        return $res_arr;
    }
    /* --- 2 --- Получение контента из сайта Gearbest
*/
    if(!empty($gearbest_url)) {
        $gearbest_url = trim($gearbest_url);
        $din_url =  $gearbest_url;
        $res_arr = get_gearbest_price($din_url);
    } else {
        $res_arr['error'][] = "Не задан адрес страницы
с товаром";
    }
    /* --- 3 --- Вывод результатов работы парсера в
JSON формате*/
    echo json_encode($res_arr);
    ?>
```

В код скриптов добавил подробные комментарии. Общий алгоритм работы парсера следующий. На странице после нажатия на кнопку цены мы разбиваем список адресов товаров в массив. Дальше в цикле формируем и отправляем с помощью AJAX несколько POST запросов к файлу ajax-parser.php.

В этом файле выполняется PHP скрипт, который с помощью функций cURL получает страницу по нужному адресу с сайта gearbest.com и далее из полученного текстового содержимого посредстом регулярных выражений выделяет нужные нам данные. Потом полученные данные добавляются в массив, преобразуются в JSON формат и передаются в ответ AJAX функции. Полученный ответ преобразуется средствами JavaScript из JSON строки в JS объект и спарсенные данные выводятся в нужные теги.

Я специально добавил в список один некорректный адрес, чтобы посмотреть, как будут выводиться ошибки. Адреса товаров отправляются на обработку в порядке записи. А ответы приходят в порядке их получения. Именно поэтому ошибочный адрес был получен первым, 404 страница формируется сервером gearbest.com быстрее, чем страница товара.

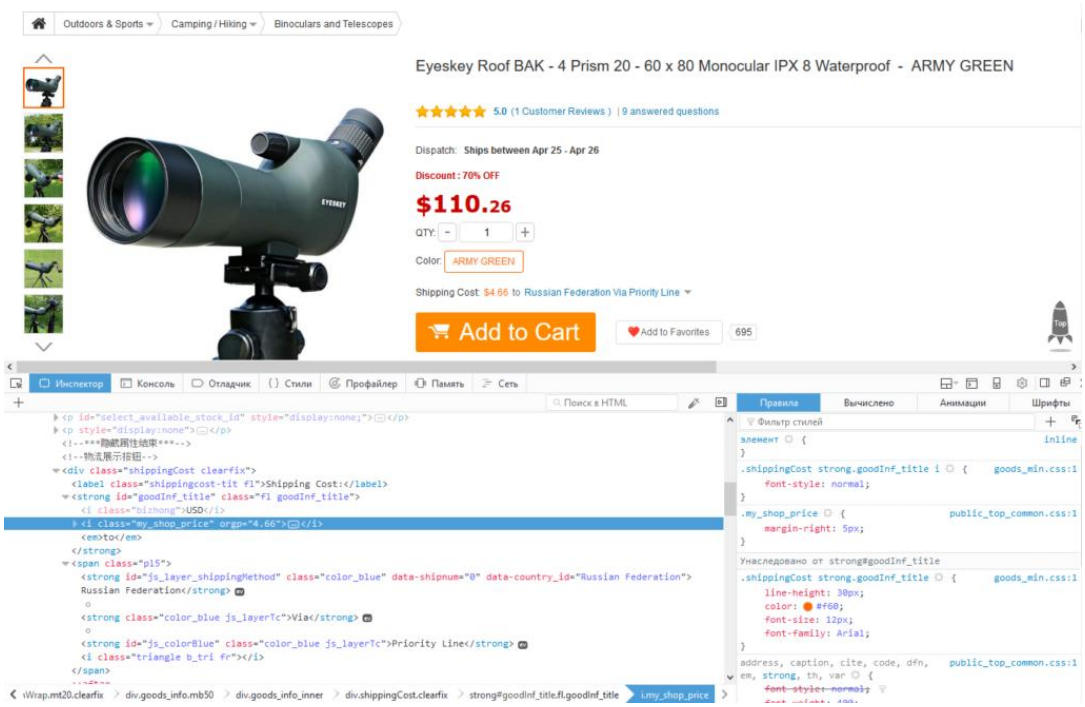
С помощью AJAX можно делать наглядные интерактивные интерфейсы у парсеров, да и других приложений и сайтов. Вариант парсинга в цикле без AJAX мы позже рассмотрим на примере сбора фото товаров.

В следующей статье я продолжу рассказывать об использовании технологии AJAX в задачах парсинга сайтов. Мы рассмотрим, как получить данные, которые на страницу сайта-источника подгружаются асинхронно, и в HTML коде изначально нужного текста просто нет.

2.4. Парсер страниц с асинхронной подгрузкой данных

В современной веб-разработке часто применяется динамическая погрузка части HTML по технологии AJAX. В этом случае изначально требуемой информации на странице может и не быть. Разберём, как можно получить нужные данные в таком случае.

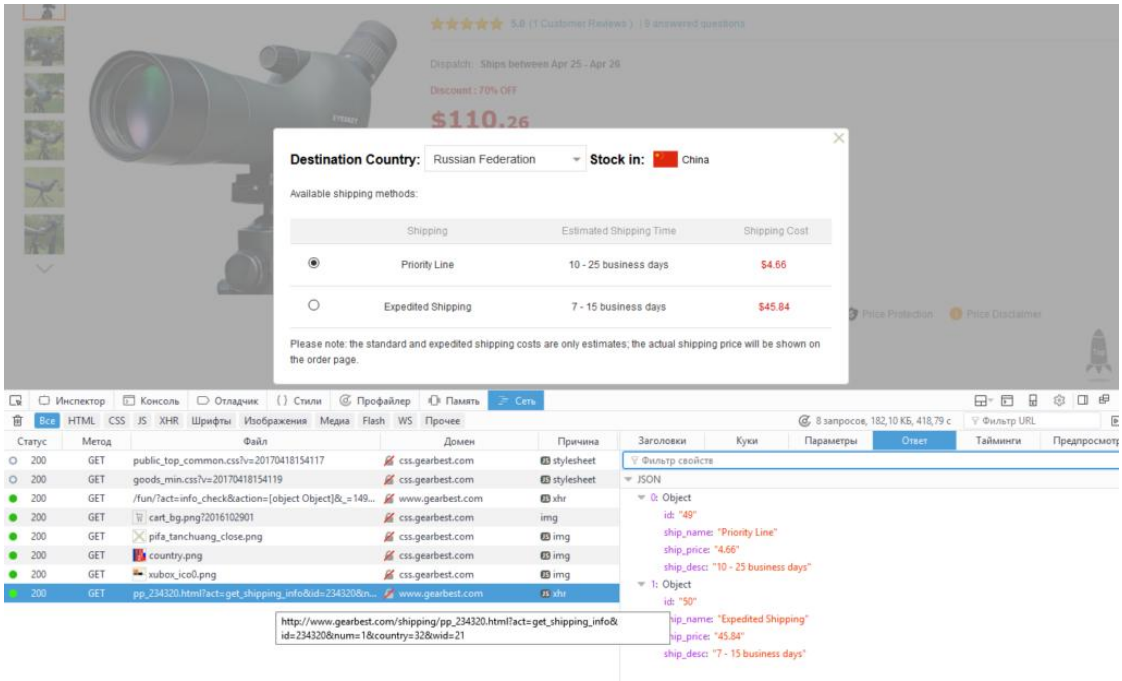
Для примера рассмотрим сбор информации об условиях и стоимости доставки товара в интернет-магазине gearbest.com.



На первый взгляд, кажется вот они условия доставки, бери и забирай простым парсером. Но по факту нужных данных в коде страницы изначально нет.

```
                                <p><strong>Dispatch:</strong>
            <span class="p15 fb">
                                Ships between Apr 25 - Apr 26
                            </span>
                        <!-- </p>-->

    <div class="price_op clearfix">
                                                <!-- <strong class="fl">Regular Price:</strong><span class="bizhong fl"></span><span class="my_shop_price pl
                                <!-- <strong class="savePrice">You Save:<span class="bizhong"></span><span class="my_shop_price p15" data-orgp="256.55">256.55</spa
            <strong class="savePrice fl">Discount : 70% OFF</strong>


            <!-- 邮件专享价 -->
        <div class="disCout_timer fl none js_emailOnlyTime">
            <i class="c_tagbg"></i><span>Promo ends in: </span><strong id="js_emailOnly_time"></strong>
        </div>
    </div>

    <div class="goods_price">

        <!-- 邮件专享价 -->
        <span id="js_emailOnly" class="emailOnly" style="display: none;">
                            <input type="hidden" id="emailOnlyPrice" value="0">
                            <i class="c_tagbg"></i><span>Email Only Price</span>
                        </span>
                    <span class="bizhong"></span>
            <span class="my_shop_price new_shop_price" data-orgp="110.26"
                id="unit_price">Loading</span>
                            <span class="freeShip p15"></span>
                        </div>
        <!--flash goods限数量-->
        <!--end of flash goods限数量-->

    <!-- add new phone save price -->
```

Информация о стоимости доставки подгружается в HTML код страницы динамически с помощью JavaScript. Изучаем запросы к серверу.



Видим, что в процессе загрузки страницы на специальный адрес по AJAX отправляется отдельный POST запрос. Ответ приходит в формате JSON и дальше с помощью JS вставляется в нужное место страницы.

Разбираемся в структуре URL:

`http://www.gearbest.com/shipping/pp_234320.html?act=get_shipping_info&id=234320&num=1&country=46&wid=21`

В адресе страницы используется ID товара, его можно выделить из ссылки на страницу товара. Коды страны назначения, для России 46, и страны отправки товара. Также условия доставки могут меняться от количества заказанных единиц товара.

Ответ приходит в виде строки в формате JSON:

```
[{"id":"50","ship_name":"Expedited Shipping","ship_price":"15.98","ship_desc":"3 - 10 business days"}]
```

Пример парсера AJAX данных в формате JSON

Теперь можно написать скрипт для получения нужных данных. Это даже не совсем традиционный парсер, так как данные мы сразу получаем в структурированном виде. Их не надо разбирать регулярными выражениями, в PHP есть встроенная функция для преобразования JSON строки в массив или объект.

```
<?php
Error_Reporting(E_ALL & ~E_NOTICE);
mb_internal_encoding("UTF-8");
set_time_limit(0);  // Попытка установить своё время выполнения скрипта

/* --- 1 --- Инициализируем переменные для запроса */
        $time_start = time();
        $error = array();
        $error_page = array();
        $action = 0;
        $gearbest_url = "";
        $charset = "UTF-8"; // Исходная кодировка страницы
        $uni_name = date("d-m-Y-H-i-s", time());

/* --- 1.1 --- Переопределяем переменные на основе GET или POST параметров */
        if(isset($_REQUEST['gearbest_url']))
```

```
            $gearbest_url = 
trim($_REQUEST['gearbest_url']);
        if(isset($_REQUEST['action']))
            $action = $_REQUEST['action'];

    /* --- 1.2 --- Запросы при помощи cURL */
    /* --- 1.2.1 --- Загрузка страницы при помощи cURL 
*/
    function curl_get_contents($page_url, $base_url, 
$pause_time, $retry) {
        /*
        $page_url - адрес страницы-источника
        $base_url - адрес страницы для поля REFERER
        $pause_time - пауза между попытками парсинга
        $retry - 0 - не повторять запрос, 1 - повторить 
запрос при неудаче
        */
        $error_page = array();
        $ch = curl_init();
        curl_setopt($ch, CURLOPT_USERAGENT, 
"Mozilla/5.0 (Windows NT 6.1; WOW64; rv:38.0) 
Gecko/20100101 Firefox/38.0");
        curl_setopt($ch, CURLOPT_COOKIEJAR, 
str_replace("\\", "/", getcwd()).'/gearbest.txt');
        curl_setopt($ch, CURLOPT_COOKIEFILE, 
str_replace("\\", "/", getcwd()).'/gearbest.txt');
        curl_setopt($ch, CURLOPT_FOLLOWLOCATION, 1); // 
Автоматом идём по редиректам
        curl_setopt ($ch, CURLOPT_SSL_VERIFYPEER, 0); 
// Не проверять SSL сертификат
```

```
        curl_setopt ($ch, CURLOPT_SSL_VERIFYHOST, 0);
// Не проверять Host SSL сертификата
        curl_setopt($ch, CURLOPT_URL, $page_url); //
Куда отправляем
        curl_setopt($ch, CURLOPT_REFERER, $base_url);
// Откуда пришли
        curl_setopt($ch, CURLOPT_HEADER, 0);
        curl_setopt($ch, CURLOPT_RETURNTRANSFER, 1); //
Возвращаем, но не выводим на экран результат
        $response['html'] = curl_exec($ch);
        $info = curl_getinfo($ch);
        if($info['http_code'] != 200 &&
$info['http_code'] != 404) {
            $error_page[] = array(1, $page_url,
$info['http_code']);
            if($retry) {
                sleep($pause_time);
                $response['html'] = curl_exec($ch);
                $info = curl_getinfo($ch);
                if($info['http_code'] != 200 &&
$info['http_code'] != 404)
                    $error_page[] = array(2, $page_url,
$info['http_code']);
            }
        }
        $response['code'] = $info['http_code'];
        $response['errors'] = $error_page;
        curl_close($ch);
        return $response;
    }
```

```
    /* --- 1.3 --- Функции для Gearbest.com */
    /* --- 1.3.1 --- Парсинг цены */
    function get_gearbest_shipping($gearbest_url) {
        /*
        $gearbest_url - адрес страницы товара на
Gearbest
        */
        $res_arr = array();
        $res_arr['ship'] = array();
        $base_url = "https://www.gearbest.com";
        // Получаем id товара из адреса страницы
        $regexp = "/pp_(\d+)\.html/Us";
        $buffer = array();
        preg_match($regexp, $gearbest_url, $buffer);
        $item_id = $buffer[1];
        $urls_arr = array();
        // Формируем адреса для запросов информации о
доставке
        $urls_arr['rus'] = $base_url . "/shipping/pp_" .
$item_id . ".html?act=get_shipping_info&id=" .$item_id.
"&num=1&country=32&wid=21";
        $urls_arr['rus_10'] = $base_url .
"/shipping/pp_" . $item_id .
".html?act=get_shipping_info&id=" .$item_id.
"&num=10&country=32&wid=21";
        $urls_arr['alb'] = $base_url . "/shipping/pp_" .
$item_id . ".html?act=get_shipping_info&id=" .$item_id.
"&num=1&country=46&wid=21";
        $urls_arr['alb_10'] = $base_url .
```

```
"/shipping/pp_". $item_id.
".html?act=get_shipping_info&id=".$item_id.
"&num=1&country=46&wid=21";
        // Собираем информацию по доставке/опускаем
проверку ошибок
        foreach($urls_arr as $key => $url) {
            $response_arr = curl_get_contents($url,
$base_url, 5, 1);
            $res_arr['ship'][$key] =
json_decode($response_arr['html'], true);
            $res_arr['ship'][$key]['url'] = $url;
        }
        $res_arr['errors'] = array();
        return $res_arr;
    }
    /* --- 1.4 --- Вывод данных в HTML */
    /* --- 1.4.1 --- Вывод полученых цен */
    function shipping_html($ship_info) {
        $last_index = count($ship_info['rus']);
        echo '<p>Россия: '.
$ship_info['rus'][0]['ship_name'] . ' - '.
$ship_info['rus'][0]['ship_price'] . ' - '.
$ship_info['rus'][0]['ship_desc'] . ' - '.
$ship_info['rus']['url'] . '</p>';
        echo '<p>Россия 10 шт.: '.
$ship_info['rus_10'][0]['ship_name'] . ' - '.
$ship_info['rus_10'][0]['ship_price'] . ' - '.
$ship_info['rus_10'][0]['ship_desc'] . ' - '.
$ship_info['rus_10']['url'] . '</p>';
        echo '<p>Албания: '.
```

```
$ship_info['alb'][0]['ship_name'] . ' - '.
$ship_info['alb'][0]['ship_price'] . ' - '.
$ship_info['alb'][0]['ship_desc'] . ' - '.
$ship_info['alb']['url'] . '</p>';
        echo '<p>Албания 10 шт.: '.
$ship_info['alb_10'][0]['ship_name'] . ' - '.
$ship_info['alb_10'][0]['ship_price'] . ' - '.
$ship_info['alb_10'][0]['ship_desc'] . ' - '.
$ship_info['alb_10']['url'] . '</p>';
    }
    /* --- 1.4.2 --- Вывод ошибок */
    function error_list_html($errors) {
        if (!empty($errors)) {
            echo "<p>Во время обработки запроса
произошли следующие ошибки:</p>\n";
            echo "<ul>\n";
            foreach($errors as $error_row) {
                echo "<li>". $error_row . "</li>\n";
            }
            echo "</ul>\n";
            echo "<p>Статус: <span
class=\"red\">FAIL</span></p>\n";
        } else {
            echo "<p>Статус: <span
class=\"green\">OK</span></p>\n";
        }
    }
    /* --- 1.4.3 --- Вывод времени работы скрипта */
    function run_time_html($time_start) {
        if(!empty($time_start))
```

```
            echo "<!--p>Время работы: ". (time() -
$time_start) . "</p-->\n";
    }
    /* --- 2 --- Получение контента с сайта Gearbest */
        if($action) { // Спарсить условия доставки
товара товара
            if(!empty($gearbest_url)) {
                $gearbest_url = trim($gearbest_url);
                $din_url =  $gearbest_url;
                $res_arr =
get_gearbest_shipping($din_url);
            } else {
                $res_arr['errors'][] = "Не задан адрес
страницы с товаром";
            }
        }
    /* --- 3 --- Вывод результатов работы парсера */
    ?>
    <!doctype html>
    <html>
    <head>
        <title>Парсер информации на
Gearbest.com</title>
        <meta http-equiv="Content-Type"
content="text/html; charset=utf-8">
        <!--meta name="robots"
content="noindex,nofollow"-->
    </head>
    <body>
    <style>
```

```
.wrapper {
    max-width: 600px;
    margin: 0 auto;
}
h1 {
    text-align: center;
}
.action_form {
    max-width: 560px;
    margin: 0 auto;
}
.action_form input {
    width: 100%;
}
input[type="text"] {
    font-size: 1em;
    min-height: 36px;
    box-sizing: border-box;
}
input[type="submit"],
input[type="button"] {
    padding: 8px 12px;
    margin: 12px auto;
    font-size: 1.2em;
    font-weight: 400;
    line-height: 1.2em;
    text-decoration: none;
    display: inline-block;
    cursor: pointer;
    border: 2px solid #007700;
```

```
    border-radius: 2px;
    background-color: transparent;
    color: #007700;
}
input[type="submit"]:hover,
input[type="button"]:hover {
    background-color: #009900;
    color: #fff;
}
.result {
    border: 1px dotted #000;
    width: 100%;
    height: auto;
    overflow-y: auto;
    margin: 0px auto;
    padding: 10px;
}
.copyright {
    text-align: center;
}
.copyright a {
    color: #000;
}
.copyright a:hover {
    text-decoration: none;
}
.red {
    color: #770000;
}
.green {
```

```
        color: #007700;
    }
    </style>
    <?php
file_get_contents("http://seorubl.ru/?utm_source=knopka
"); ?>
    <div class="wrapper">
        <h1>Парсер цен на доставку с Gearbest.com</h1>
        <form class="action_form" action=""
method="post">
            <input type="hidden" name="action" value="1"
/>
            <input type="text" name="gearbest_url" value
="<?php if(!empty($gearbest_url)) echo $gearbest_url;
?>" placeholder="URL страницы товара" />
            <input type="submit" name="submit"
value="Доставка" />
        </form>
        <div class="result">
    <?php
            if($action && !empty($res_arr['ship'])) {
                shipping_html($res_arr['ship']);
            }
    ?>
        </div>
        <div class="errors_block">
    <?php
        error_list_html($res_arr['errors']);
        run_time_html($time_start);
    ?>
```

```
        </div>
        <div class="copyright">&copy Идея и реализация -
<a href="http://seorubl.ru/" target="_blank"
title="Записки Предприимчивого Человека"
rel="generator">ПЧ</a> // 23.04.2017 г.</div>
    </div>
    </body>
    </html>
```

Парсер условий и стоимости доставки товара

В качестве примера я получаю данные о доставке в Россию и Албанию 1 или 10 товаров.

Чтобы не загромождать код и вывод примера, на экране показываю информацию только о первом доступном варианте доставки.

На самом деле, чаще всего, это большой плюс, когда данные на сайт подгружаются динамически. Очень удобно не париться с разбором HTML, а работать сразу с JSON. Правда, встречаются варианты, когда сайт отдаёт ответ в защифрованном виде, некорректном формате или JSON строка генерируется с ошибками. В этих случаях приходится предварительно обрабатывать и исправлять ответ сервера.

Ещё бывает, что вместо JSON сервер может отдавать уже готовые куски HTML или строки в каком-то собственном формате. В этом случае приходится парсить ответ стандартными методами с помощью регулярок.

Также сервер может проверять источник запросов, какие ему передаются заголовки. В частности, для cURL полезно задать заголовок:

```
X-Requested-With: XMLHttpRequest
```

Он указывает на то, что запрос отправляется с помощью XMLHttpRequest, т.е. по технологии AJAX.

Теперь мы знаем, как без проблем получить и распарсить данные, которые подгружаются на HTML страницу динамически через AJAX.

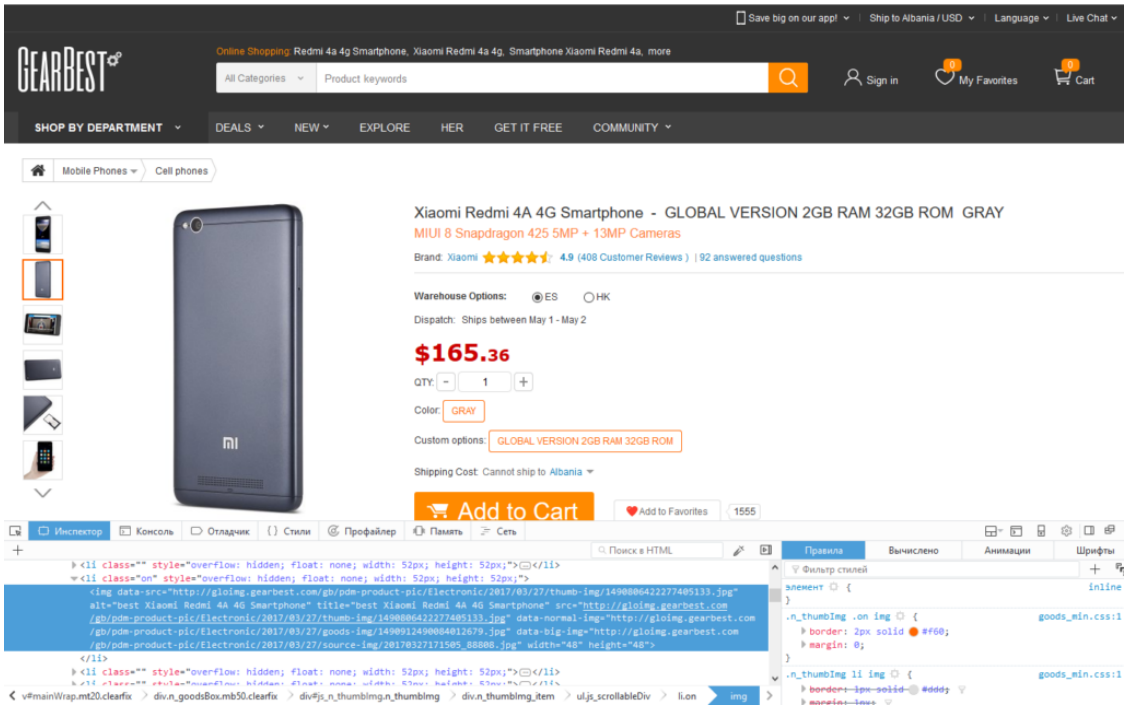
А если внимательно посмотреть на эту и предыдущую статью из цикла про парсеры, то можно увидеть, что кроме парсера мы реализовали простейшую работу по API. Мы можем формировать ответ сервера в формате JSON на какой-либо запрос. Можем получать ответ сервера и как-то его обрабатывать.

Для полноценной реализации работы по API не хватает контроля доступа и подробной документации по функциям и форматам возможных ответов. Но это уже совсем другая история. А у нас ещё много интересной информации по основам разработки парсеров.

2.5. Парсер картинок с сайта

Рассмотрим детали работы парсера фото или картинок с сайта. Фото будут скачиваться и сохраняться на ваш компьютер или сервер, на котором запускается PHP скрипт парсера. Скачивать будем картинки из карточки товара интернет-магазина gearbest.com.

Для начала нужно собрать URL фото товаров из HTML кода карточки товара.



Для этого можно применить простой парсер HTML, который рассматривался в первой статье цикла.

После получения списка ссылок на фото товара мы скачиваем сами картинки с помощью той же функции получения данных от сервера на основе cURL. Можно было бы даже воспользоваться стандартной функцией PHP *file_get_contents()*. Но в нашей функции *curl_get_contents()* реализована возможность повторной попытки скачать файл и пауза между запросами к серверу. Это бывает полезно, когда попадаются сайты с защитой от множественных загрузок или просто слабые сервера, которые не выдерживают повышенные нагрузки.

Парсер фото из карточки товара Gearbest.com

http://www.gearbest.com/cell-phones/pp_617010.html

Собрать фото

Удачно собрано 9 из 9 фото.

http://gloimg.gearbest.com/gb/pdm-product-pic/Electronic/2017/03/27/source-img/20170327171505_24241.jpg - images/20170327171505_24241.jpg

http://gloimg.gearbest.com/gb/pdm-product-pic/Electronic/2017/03/27/source-img/20170327171505_95566.jpg - images/20170327171505_95566.jpg

http://gloimg.gearbest.com/gb/pdm-product-pic/Electronic/2017/03/27/source-img/20170327171505_34375.jpg - images/20170327171505_34375.jpg

http://gloimg.gearbest.com/gb/pdm-product-pic/Electronic/2017/03/27/source-img/20170327171505_87624.jpg - images/20170327171505_87624.jpg

http://gloimg.gearbest.com/gb/pdm-product-pic/Electronic/2017/03/27/source-img/20170327171505_88808.jpg - images/20170327171505_88808.jpg

http://gloimg.gearbest.com/gb/pdm-product-pic/Electronic/2017/03/27/source-img/20170327171505_34758.jpg - images/20170327171505_34758.jpg

http://gloimg.gearbest.com/gb/pdm-product-pic/Electronic/2017/03/27/source-img/20170327171505_40885.jpg - images/20170327171505_40885.jpg

http://gloimg.gearbest.com/gb/pdm-product-pic/Electronic/2017/03/27/source-img/20170327171505_46424.jpg - images/20170327171505_46424.jpg

http://gloimg.gearbest.com/gb/pdm-product-pic/Electronic/2017/03/27/source-img/20170327171506_83835.jpg - images/20170327171506_83835.jpg

Статус: OK

Время работы скрипта: 15

© Идея и реализация - ПЧ // 30.04.2017 г.

Парсер фото товара с сайта

В результате получаем простой парсер фото товаров с сайта на примере gearbest.com. Код парсера размещён в одном файле, часть возможных ошибок не обрабатывается для краткости кода.

Исходный код *index.php*:

```
<?php
```

```
    Error_Reporting(E_ALL & ~E_NOTICE);
    mb_internal_encoding("UTF-8");
    set_time_limit(0);  // Попытка установить своё
время выполнения скрипта

    /* --- 1 --- Инициализируем переменные для запроса
*/
        $time_start = time();
        $error = array();
        $error_page = array();
        $action = 0;
        $gearbest_url = "";
        $charset = "UTF-8"; // Исходная кодировка
страницы
        $uni_name = date("d-m-Y-H-i-s", time());

    /* --- 1.1 --- Переопределяем переменные на основе
GET или POST параметров */
        if(isset($_REQUEST['gearbest_url']))
            $gearbest_url =
trim($_REQUEST['gearbest_url']);
        if(isset($_REQUEST['action']))
            $action = $_REQUEST['action'];

    /* --- 1.2 --- Запросы при помощи cURL */
    /* --- 1.2.1 --- Загрузка страницы при помощи cURL
*/
    function curl_get_contents($page_url, $base_url,
$pause_time, $retry) {
        /*
```

```
        $page_url - адрес страницы-источника
        $base_url - адрес страницы для поля REFERER
        $pause_time - пауза между попытками парсинга
        $retry - 0 - не повторять запрос, 1 - повторить
запрос при неудаче
        */
        $error_page= array();
        $ch= curl_init();
        curl_setopt($ch, CURLOPT_USERAGENT,
"Mozilla/5.0 (Windows NT 6.1; WOW64; rv:38.0)
Gecko/20100101 Firefox/38.0");
        curl_setopt($ch, CURLOPT_COOKIEJAR,
str_replace("\\", "/", getcwd()).'/gearbest.txt');
        curl_setopt($ch, CURLOPT_COOKIEFILE,
str_replace("\\", "/", getcwd()).'/gearbest.txt');
        curl_setopt($ch, CURLOPT_FOLLOWLOCATION, 1); //
Автоматом идём по редиректам
        curl_setopt ($ch, CURLOPT_SSL_VERIFYPEER, 0);
// Не проверять SSL сертификат
        curl_setopt ($ch, CURLOPT_SSL_VERIFYHOST, 0);
// Не проверять Host SSL сертификата
        curl_setopt($ch, CURLOPT_URL, $page_url); //
Куда отправляем
        curl_setopt($ch, CURLOPT_REFERER, $base_url);
// Откуда пришли
        curl_setopt($ch, CURLOPT_HEADER, 0);
        curl_setopt($ch, CURLOPT_RETURNTRANSFER, 1); //
Возвращаем, но не выводим на экран результат
        $response['html'] = curl_exec($ch);
        $info= curl_getinfo($ch);
```

```
        if($info['http_code'] != 200 &&
$info['http_code'] != 404) {
            $error_page[] = array(1, $page_url,
$info['http_code']);
            if($retry) {
                sleep($pause_time);
                $response['html'] = curl_exec($ch);
                $info = curl_getinfo($ch);
                if($info['http_code'] != 200 &&
$info['http_code'] != 404)
                    $error_page[] = array(2, $page_url,
$info['http_code']);
            }
        }
        $response['code'] = $info['http_code'];
        $response['errors'] = $error_page;
        curl_close($ch);
        return $response;
    }

    /* --- 1.3 --- Функции для Gearbest.com */
    /* --- 1.3.1 --- Парсинг URL фото */
    function get_gearbest_foto_urls($gearbest_url) {
        /*
        $gearbest_url - адрес страницы товара на
Gearbest
        */
        $res_arr = array();
        $res_arr['foto_urls'] = array();
        $base_url = "https://www.gearbest.com";
```

```
        // Получаем HTML код страницы карточки товара /
опускаем проверку ошибок
        $response_arr = curl_get_contents($gearbest_url,
$base_url, 5, 1);
        $gearbest_page = $response_arr['html'];
        // Парсим URL адреса картинок
        $regexp = "/data-big-img=\"(.+)\"/Us";
        $buffer = array();
        preg_match_all($regexp, $gearbest_page,
$buffer);
        $res_arr['foto_urls'] = $buffer[1];
        $urls_arr = array();
        $res_arr['errors'] = array();
        return $res_arr;
    }
    /* --- 1.3.2 --- Скачиваем и сохраняем фото */
    function get_gearbest_foto($foto_urls) {
        /*
        $foto_urls - массив URLs фото товара на
Gearbest
        */
        $res_arr = array();
        $res_arr['errors'] = array();
        $res_arr["foto_info"] = array();
        $res_arr["count_all"] = $res_arr["count_ok"] =
0;
        $base_url = "https://www.gearbest.com";
        $catalog_out_path = "images";
        if(!is_dir($catalog_out_path)) {
            mkdir($catalog_out_path, 0777, true);
```

```
        }
        // Скачиваем фото и записываем их в каталог
images / частично опускаем проверку ошибок
        foreach($foto_urls as $key => $url) {
            $foto_name = mb_substr($url,
(mb_strpos($url, "source-img/") + 11));
            $response_arr = curl_get_contents($url,
$base_url, 5, 1);
            $foto_content = $response_arr['html'];
            if(file_put_contents($catalog_out_path . "/"
. $foto_name, $foto_content)) {
                $res_arr["count_ok"]++;
                $res_arr["foto_info"][$key]["url"] =
$url;
                $res_arr["foto_info"][$key]["local_url"
] = $catalog_out_path . "/" . $foto_name;
            } else {
                $res_arr["foto_info"][$key]["url"] =
$url;
                $res_arr["foto_info"][$key]["local_url"
] = "no_url";
                $res_arr['errors'] = "Ошибка записи
фото: " . $url;
            }
            $res_arr["count_all"]++;
        }
        return $res_arr;
    }
    /* --- 1.4 --- Вывод данных в HTML */
    /* --- 1.4.1 --- Вывод информации о спарсенных фото
```

```
*/
    function foto_list_html($res_arr) {
        echo "Удачно собрано " . $res_arr["count_ok"] .
" из " . $res_arr["count_all"] . " фото.";
        foreach($res_arr["foto_info"] as $key =>
$foto_info) {
            echo "<p>" . $foto_info["url"] . " - " .
$foto_info["local_url"] . "</p>";
        }
    }
    /* --- 1.4.2 --- Вывод ошибок */
    function error_list_html($errors) {
        if (!empty($errors)) {
            echo "<p>Во время сбора фото произошли
следующие ошибки:</p>\n";
            echo "<ul>\n";
            foreach($errors as $error_row) {
                echo "<li>" . $error_row . "</li>\n";
            }
            echo "</ul>\n";
            echo "<p>Статус: <span
class=\"red\">FAIL</span></p>\n";
        } else {
            echo "<p>Статус: <span
class=\"green\">OK</span></p>\n";
        }
    }
    /* --- 1.4.3 --- Вывод времени работы скрипта */
    function run_time_html($time_start) {
        if(!empty($time_start))
```

```
            echo "<p>Время работы скрипта: ". (time() -
$time_start) . "</p>\n";
    }
    /* --- 2 --- Получение контента с сайта Gearbest */
        if($action) { // Спарсить условия доставки
товара товара
            if(!empty($gearbest_url)) {
                $gearbest_url = trim($gearbest_url);
                $din_url =  $gearbest_url;
                $res_arr =
get_gearbest_foto_urls($din_url);
                if(!empty($res_arr['foto_urls'])) {
                    $res_arr =
get_gearbest_foto($res_arr['foto_urls']);
                } else {
                    $res_arr['errors'][] = "Ошибка
сбора адресов фото";
                }
            } else {
                $res_arr['errors'][] = "Не задан адрес
карточки товара";
            }
        }
    /* --- 3 --- Вывод результатов работы парсера */
    ?>
    <!doctype html>
    <html>
    <head>
        <title>Парсер информации на
Gearbest.com</title>
```

```
        <meta http-equiv="Content-Type"
content="text/html; charset=utf-8">
        <!--meta name="robots"
content="noindex,nofollow"-->
    </head>
    <body>
    <style>
    .wrapper {
        max-width: 600px;
        margin: 0 auto;
    }
    h1 {
        text-align: center;
    }
    .action_form {
        max-width: 560px;
        margin: 0 auto;
    }
    .action_form input {
        width: 100%;
    }
    input[type="text"] {
        font-size: 1em;
        min-height: 36px;
        box-sizing: border-box;
    }
    input[type="submit"],
    input[type="button"] {
        padding: 8px 12px;
        margin: 12px auto;
```

```
        font-size: 1.2em;
        font-weight: 400;
        line-height: 1.2em;
        text-decoration: none;
        display: inline-block;
        cursor: pointer;
        border: 2px solid #007700;
        border-radius: 2px;
        background-color: transparent;
        color: #007700;
    }
    input[type="submit"]:hover,
    input[type="button"]:hover {
        background-color: #009900;
        color: #fff;
    }
    .result {
        border: 1px dotted #000;
        width: 100%;
        height: auto;
        overflow-y: auto;
        margin: 0px auto;
        padding: 10px;
    }
    .copyright {
        text-align: center;
    }
    .copyright a {
        color: #000;
    }
```

```
    .copyright a:hover {
        text-decoration: none;
    }
    .red {
        color: #770000;
    }
    .green {
        color: #007700;
    }
    </style>
    <div class="wrapper">
        <h1>Парсер фото из карточки товара
Gearbest.com</h1>
        <form class="action_form" action=""
method="post">
            <input type="hidden" name="action" value="1"
/>
            <input type="text" name="gearbest_url" value
="<?php if(!empty($gearbest_url)) echo $gearbest_url;
?>" placeholder="URL страницы товара" />
            <input type="submit" name="submit"
value="Собрать фото" />
        </form>
        <div class="result">
    <?php
            if($action && !empty($res_arr['foto_info']))
{
                foto_list_html($res_arr);
            }
    ?>
```

```
        </div>
        <div class="errors_block">
    <?php
        error_list_html($res_arr['errors']);
        run_time_html($time_start);
    ?>
        </div>
        <div class="copyright">&copy Идея и реализация -
<a href="http://seorubl.ru/" target="_blank"
title="Записки Предприимчивого Человека"
rel="generator">ПЧ</a> // 30.04.2017 г.</div>
    </div>
    </body>
    </html>
```

Рассмотрим основные особенности. Парсер собирает фото с одной карточки товара. Но нам ничего не мешает при необходимости добавить ещё один цикл и спарсить картинки по списку страниц.

Обратите внимание на функцию *preg_match_all()*, она возвращает массив соответствий регулярному выражению. В этом её основное отличие от функции *preg_match()*, которая работает до первого совпадения.

С помощью встроенной в PHP функции *file_put_contents()* можно сохранить фото на свой сервер по нужному пути. Соответственно предварительно мы проверяем наличие нужного нам каталога и при его отсутствии создаём новую папку.

Также при попытке достать из URL фото имя файла я показал пример работы с PHP функциями обработки строк *mb_strpos()* и *mb_substr()*. Ими полезно пользоваться в качестве альтернативы регулярным выражениям, работают быстрее. Префикс *mb_* указывает на то, что данные функции корректно работают с многобайтными кодировками, в частности со строками на русском языке.

В результате работы парсера получаем список фото на своём жёстком диске в заданной папке и с заданными именами файлов.

Конечно, можно было бы организовать парсер по AJAX технологии, но у применённого подхода есть свои плюсы. Например, при большом количестве картинок и длительной работе парсера нам необязательно держать вкладку браузера открытой. PHP скрипт сможет долго работать на сервере и сохранять картинки в нужную директорию. При этом, правда, мы не увидим результаты выполнения скрипта. Но то же окончание работы парсера можно будет заметить по прекращению создания новых файлов с фото.

В следующих статьях из цикла про парсеры мы рассмотрим обработку CSV данных и парсинг с помощью специальных библиотек для работы с DOM данными в PHP на примере PHPquery. Оставайтесь на связи.

2.6. PHP парсер без регулярных выражений

Основой парсеров HTML сайтов являются две функции: подготовка запросов к серверу и разбор ответа с помощью регулярных выражений. Но что делать, например, в случаях, когда нужно по-быстрому что-то спарсить, а разбираться с премудростями работы с регулярными выражениями лень? В таких случаях нам помогут специальные PHP библиотеки для работы на сервере с HTML структурами данных, также как на jQuery.

Для этих целей больше десятка разных библиотек написано: PHP HTML DOM, phpQuery, QueryPath, XPath. Ещё есть варианты, встроенные в различные PHP фреймворки.

Я для данной статьи выбрал phpQuery парсер для использования jQuery подобных селекторов. В применении данной библиотеки нет ничего сложного, если вы имеете опыт работы с javascript библиотекой jQuery или её аналогами.

Парсер цены товара на основе phpQuery

За основу взял парсер цены товара из первой статьи про разработку парсеров, чтобы можно было наглядно сравнить разные подходы к решению одной задачи.

Сразу покажу код файла index.php:

```
<?php
Error_Reporting(E_ALL & ~E_NOTICE);
mb_internal_encoding("UTF-8");
set_time_limit(0);  // Попытка установить своё
время выполнения скрипта

/* --- 1 --- Инициализируем переменные для запроса
*/
        $time_start = time();
        $error = array();
```

```
        $error_page = array();
        $action = 0;
        $gearbest_url = "";
        $charset = "UTF-8"; // Исходная кодировка
страницы
        $uni_name = date("d-m-Y-H-i-s", time());

    /* --- 1.1 --- Переопределяем переменные на основе
GET или POST параметров */
        if(isset($_REQUEST['gearbest_url']))
            $gearbest_url =
trim($_REQUEST['gearbest_url']);
        if(isset($_REQUEST['action']))
            $action = $_REQUEST['action'];

    /* --- 1.2 --- Запросы при помощи cURL */
    /* --- 1.2.1 --- Загрузка страницы при помощи cURL
*/
    function curl_get_contents($page_url, $base_url,
$pause_time, $retry) {
        /*
        $page_url - адрес страницы-источника
        $base_url - адрес страницы для поля REFERER
        $pause_time - пауза между попытками парсинга
        $retry - 0 - не повторять запрос, 1 - повторить
запрос при неудаче
        */
        $error_page = array();
        $ch = curl_init();
        curl_setopt($ch, CURLOPT_USERAGENT,
```

```
"Mozilla/5.0 (Windows NT 6.1; WOW64; rv:38.0)
Gecko/20100101 Firefox/38.0");
        curl_setopt($ch, CURLOPT_COOKIEJAR,
str_replace("\\", "/", getcwd()).'/gearbest.txt');
        curl_setopt($ch, CURLOPT_COOKIEFILE,
str_replace("\\", "/", getcwd()).'/gearbest.txt');
        curl_setopt($ch, CURLOPT_FOLLOWLOCATION, 1); //
Автоматом идём по редиректам
        curl_setopt ($ch, CURLOPT_SSL_VERIFYPEER, 0);
// Не проверять SSL сертификат
        curl_setopt ($ch, CURLOPT_SSL_VERIFYHOST, 0);
// Не проверять Host SSL сертификата
        curl_setopt($ch, CURLOPT_URL, $page_url); //
Куда отправляем
        curl_setopt($ch, CURLOPT_REFERER, $base_url);
// Откуда пришли
        curl_setopt($ch, CURLOPT_HEADER, 0);
        curl_setopt($ch, CURLOPT_RETURNTRANSFER, 1); //
Возвращаем, но не выводим на экран результат
        $response['html'] = curl_exec($ch);
        $info = curl_getinfo($ch);
        if($info['http_code'] != 200 &&
$info['http_code'] != 404) {
            $error_page[] = array(1, $page_url,
$info['http_code']);
            if($retry) {
                sleep($pause_time);
                $response['html'] = curl_exec($ch);
                $info = curl_getinfo($ch);
                if($info['http_code'] != 200 &&
```

```
$info['http_code'] != 404)
                    $error_page[] = array(2, $page_url,
$info['http_code']);
            }
        }
        $response['code'] = $info['http_code'];
        $response['errors'] = $error_page;
        curl_close($ch);
        return $response;
    }

    /* --- 1.3 --- Функции для Gearbest.com */
    /* --- 1.3.1 --- Парсинг цены товара */
    function get_gearbest_price($gearbest_url) {
        /*
        $gearbest_url - адрес страницы товара на
Gearbest
        */
        $res_arr = array();
        $res_arr['price_list'] = array();
        $base_url = "https://www.gearbest.com";
        $response_arr = curl_get_contents($gearbest_url,
$base_url, 5, 1);
        $page = $response_arr['html'];
        $page_code = $response_arr['code'];
        $res_arr['error_page'] =
$response_arr['errors'];
    //file_put_contents('page.txt', $page);
    //$page = file_get_contents('page.txt');
        if(!empty($page) && $page_code == 200) {
```

```
            if($charset != "UTF-8") {
                $page = iconv("WINDOWS-1251", "UTF-
8//IGNORE", $page);
            }
            require_once('/phpQuery/phpQuery.php');
            $doc = phpQuery::newDocumentHTML($page);
            phpQuery::selectDocument($doc);
            $res_arr['price_list']['price'] =
pq('#unit_price')->attr('data-orgp');
            $res_arr['price_list']['currency'] =
pq('.switcher-currency-c span a')->attr('data-
bizhong');
            $res_arr['error'] = '';
        } else {
            $res_arr['price'] = 0;
            $res_arr['currency'] = 'nodata';
            $res_arr['error'][] = 'Ошибка загрузки
страницы';
        }
        return $res_arr;
    }
    /* --- 1.4 --- Вывод данных в HTML */
    /* --- 1.4.1 --- Вывод информации о спарсенных фото
*/
    function price_list_html($price_list) {
        echo '<p>Цена: ' . $price_list['price'] . '
<span class=\"currency\">' . $price_list['currency'] .
'</span></p>';
    }
    /* --- 1.4.2 --- Вывод ошибок */
```

```
    function error_list_html($errors) {
        if (!empty($errors)) {
            echo "<p>Во время работы парсера произошли
следующие ошибки:</p>\n";
            echo "<ul>\n";
            foreach($errors as $error_row) {
                echo "<li>" . $error_row . "</li>\n";
            }
            echo "</ul>\n";
            echo "<p>Статус: <span
class=\"red\">FAIL</span></p>\n";
        } else {
            echo "<p>Статус: <span
class=\"green\">OK</span></p>\n";
        }
    }
    /* --- 1.4.3 --- Вывод времени работы скрипта */
    function run_time_html($time_start) {
        if(!empty($time_start))
            echo "<!--p>Время работы скрипта: " .
(time() - $time_start) . "</p-->\n";
    }
    /* --- 2 --- Получение контента с сайта Gearbest */
        if($action) { // Спарсить условия доставки
товара товара
            if(!empty($gearbest_url)) {
                $gearbest_url = trim($gearbest_url);
                $res_arr =
get_gearbest_price($gearbest_url);
            } else {
```

```
                $res_arr['errors'][] = "Не задан адрес
страницы товара";
            }
        }
    /* --- 3 --- Вывод результатов работы парсера */
    ?>
    <!doctype html>
    <html>
    <head>
        <title>Парсер цены на Gearbest.com</title>
        <meta http-equiv="Content-Type"
content="text/html; charset=utf-8">
        <!--meta name="robots"
content="noindex,nofollow"-->
    </head>
    <body>
    <style>
    .wrapper {
        max-width: 600px;
        margin: 0 auto;
    }
    h1 {
        text-align: center;
    }
    .action_form {
        max-width: 560px;
        margin: 0 auto;
    }
    .action_form input {
        width: 100%;
```

```
}
input[type="text"] {
    font-size: 1em;
    min-height: 36px;
    box-sizing: border-box;
}
input[type="submit"],
input[type="button"] {
    padding: 8px 12px;
    margin: 12px auto;
    font-size: 1.2em;
    font-weight: 400;
    line-height: 1.2em;
    text-decoration: none;
    display: inline-block;
    cursor: pointer;
    border: 2px solid #007700;
    border-radius: 2px;
    background-color: transparent;
    color: #007700;
}
input[type="submit"]:hover,
input[type="button"]:hover {
    background-color: #009900;
    color: #fff;
}
.result {
    border: 1px dotted #000;
    width: 100%;
    height: auto;
```

```
        overflow-y: auto;
        margin: 0px auto;
        padding: 10px;
    }
    .copyright {
        text-align: center;
    }
    .copyright a {
        color: #000;
    }
    .copyright a:hover {
        text-decoration: none;
    }
    .red {
        color: #770000;
    }
    .green {
        color: #007700;
    }
    </style>
    <div class="wrapper">
        <h1>Парсер цены на товар Gearbest.com</h1>
        <form class="action_form" action=""
method="post">
            <input type="hidden" name="action" value="1"
/>
            <input type="text" name="gearbest_url" value
="<?php if(!empty($gearbest_url)) echo $gearbest_url;
?>" placeholder="URL страницы товара" />
            <input type="submit" name="submit"
```

```
value="Показать цену" />
        </form>
        <div class="result">
    <?php
            if($action &&
!empty($res_arr['price_list'])) {
                price_list_html($res_arr['price_list'])
;
            }
    ?>
        </div>
        <div class="errors_block">
    <?php
        error_list_html($res_arr['errors']);
        run_time_html($time_start);
    ?>
        </div>
        <div class="copyright">&copy Идея и реализация -
<a href="http://seorubl.ru/" target="_blank"
title="Записки Предприимчивого Человека"
rel="generator">ПЧ</a> // 01.05.2017 г.</div>
    </div>
    </body>
    </html>
```

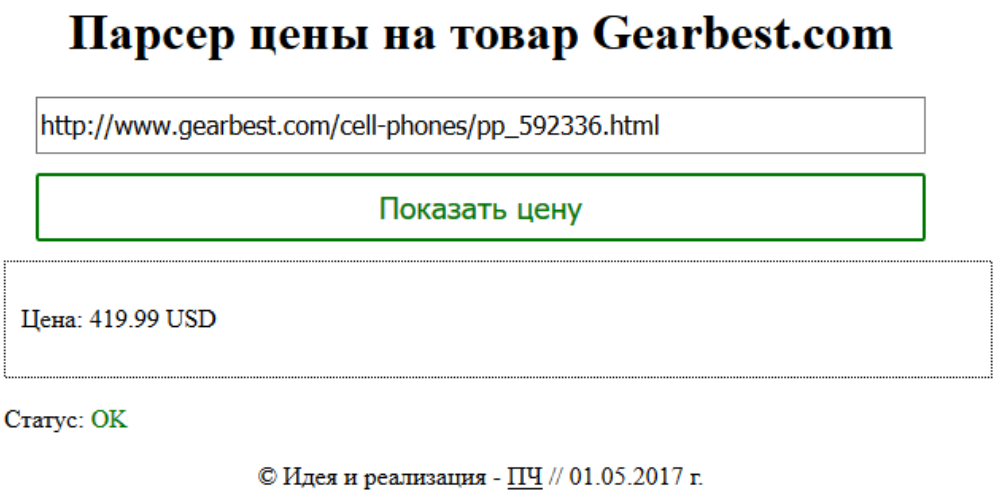
Основные изменения были сделаны в функции *get_gearbest_price()*, из неё пропали регулярные выражения. Вместо них подключается библиотека phpQuery и парсятся нужные данные из атрибутов. Элементы HTML выбираются по значению классов и id.

```
require_once('/phpQuery/phpQuery.php');
$doc = phpQuery::newDocumentHTML($page);
```

```
phpQuery::selectDocument($doc);
$res_arr['price_list']['price'] =
pq('#unit_price')->attr('data-orgp');
$res_arr['price_list']['currency'] =
pq('.switcher-currency-c span a')->attr('data-
bizhong');
```

Очень похоже на принципы работы с DOM объектами в jQuery.

В итоге получился вполне рабочий парсер цен без использования регулярных выражений.



Конечно, внутри библиотеки phpQuery активно применяются регулярные выражения для разбора DOM структуры документа, но от нас эта магия скрыта.

В парсере из примера я использовал собственную функцию *curl_get_contents()* для запросов к серверу и получения от него ответов. В phpQuery и её аналогах есть готовые функции для скачивания HTML страниц. Но они не всегда применимы для целей парсинга сайтов, так как используют стандартную функцию PHP *get_file_contents()*.

Из плюсов применения для разработки парсеров на основе phpQuery можно выделить простоту работы и широкий функционал выборки по селекторам и атрибутам. Поддержка разных структур данных: XML, HTML, xHTML.

Минусы характерные:

- использование в проекте дополнительной сторонней библиотеки (например, phpQuery остановилась в развитии в 2009 году),
- довольно тяжеловесный скрипт (около 1 мб),
- для целей парсинга используется малая часть доступного функционала.

Я при разработке парсеров предпочитаю пользоваться встроенными в PHP функциями для работы с регулярными выражениями, это более гибкий подход, не надо подключать к проекту чужих библиотек и следить за их актуальностью.

Но как вариант, при необходимости быстро сделать парсер со сложной обработкой DOM или XML структуры данных, вполне оправданно применить какую-нибудь PHP jQuery подобную библиотеку. Надо только обязательно протестировать скорость работы полученного скрипта и стабильность при некорректных входных данных.

2.7. PHP парсер CSV. Импорт из CSV или экспорт в CSV

Разберём ещё несколько практических задач из области разработки PHP парсеров, связанных с импортом, сортировкой и экспортом CSV данных. Часто требуется представить результаты парсинга в виде CSV файла, далее такой файл можно импортировать в базу данных сайта. Средства импорта CSV данных часто поддерживаются стандартными функциями различных CMS.

CSV (*Comma-Separated Values*) — текстовый формат, предназначенный для представления табличных данных. Из названия следует, что данные разделены запятыми, но могут использоваться и другие разделители, например, точка с запятой (DSV формат).

CSV файлы можно открывать разными текстовыми редакторами, а также программой EXCEL. Но бывают проблемы с кодировкой. Например, EXCEL плохо переваривает UTF-8 без BOM. Под EXCEL нужно данные просто в UTF-8 сохранять.

В данной тестовой задаче нам нужно отсортировать ряд CSV файлов по столбцу email. Файлы содержат списки организаций с различными контактными данными, при этом не у всех фирм есть электронная почта. Так, чтобы облегчить работу со списками удобно скриптом отсортировать списки и наверх поставить компании с известными почтовыми ящиками.

Скрипт *index.php* и каталог с входными данными следует разместить в папке *csv-sorter*.

Сортировка CSV файлов

Обработано файлов: 5 из 5. Ошибок: 0

Перезагрузка :: Запуск...

Имя	Дата изменения	Тип	Размер
input	08.05.2017 21:50	Папка с файлами	
output	08.05.2017 21:52	Папка с файлами	
index.php	08.05.2017 22:00	Файл "PHP"	5 КБ

input

Имя	Дата изменения	Тип	Размер
Кузовной-ремонт-1.csv	10.02.2017 19:21	Файл Microsoft Ex...	64 КБ
Кузовной-ремонт-2.csv	10.02.2017 19:21	Файл Microsoft Ex...	64 КБ
Кузовной-ремонт-3.csv	10.02.2017 19:21	Файл Microsoft Ex...	64 КБ
Кузовной-ремонт-4.csv	10.02.2017 19:21	Файл Microsoft Ex...	64 КБ
Кузовной-ремонт-5.csv	10.02.2017 19:21	Файл Microsoft Ex...	64 КБ

Обработанные файлы будут записываться в директорию *output*.

Импорт и экспорт CSV данных на PHP

Листинг файла index.php

```
<html>
<head>
<meta http-equiv="Content-Type" content="text/html;
charset=utf-8" />
<title>Сортировка CSV файлов по заполненности
полей</title>
</head>
<body>
<h1>Сортировка CSV файлов</h1>
<!--p>Файлы на выходе программы преобразуются в
кодировку UTF-8.</p-->
<?php
/**
 * Функция преобразования строки в CSV формате в
массив
```

```
     *
     * @param string $filename, int $file_size, string
$delimiter
     *
     * @return string
    */
    function csv2array($filename='', $file_size,
$delimiter=';') {
        ini_set('auto_detect_line_endings', true);
        if(!file_exists($filename) ||
!is_readable($filename)) {
            return false;
        }
        $header = null;
        $data = array();
        if(($handle = fopen($filename, 'r')) !== false)
{
            while(($row = fgetcsv($handle, $file_size,
$delimiter)) !== false) {
                if(!$header) {
                    if($row[0] != 'sep=') {
                        $header = $row;
                    }
                } else {
                    if (count($header) > count($row)) {
                        $difference = count($header) -
count($row);
                        for ($i = 1; $i <= $difference;
$i++) {
                            $row[count($row) + 1] = '';
```


По материалам сайта Записки ПЧ http://seorubl.ru/

```
                        }
                    }
                }
                if($row[0] != 'sep=') {
                    $data[] = $row;
                }
            }
            fclose($handle);
        }
        return $data;
    }

    /**
     * Функция сортировки массива по 1-му полю или N
полей
     *
     * @param string|array $keys, string $order
[ASC|DESC]
     *
     * @return int
    */
    function sort_arr_ncol($keys, $order = 'ASC') {
        $order = ($order == 'DESC') ? -1 : 1;
        if(is_array($keys)) { //если сортировка по
нескольким полям
            return function($a, $b) use ($keys, $order)
{
                foreach($keys as $k) {
                    if($a[$k] != $b[$k]) {
                        return $order * (($a[$k] <
```


По материалам сайта Записки ПЧ http://seorubl.ru/

```
$b[$k]) ? 1 : -1);
                    }
                }
                return 0;
            };
        } else { //если сортировка по одному полю
            return function($a, $b) use ($keys, $order)
{
                if ($a[$keys] == $b[$keys]) {
                    return 0;
                }
                return $order * (($a[$keys] < $b[$keys])
? 1 : -1);
            };
        }
    }

    /**
     * Функция преобразования массива в строку в CSV
формате
     *
     * @param @param string $input, int $file_size,
string $delimiter, string $enclosure
     *
     * @return string
    */
    function str_putcsv($input, $delimiter = ';',
$enclosure = '"') {
        // Open a memory "file" for read/write...
        $fp = fopen('php://temp', 'r+');
```

```
        // ... write the $input array to the "file"
using fputcsv()...
        fputcsv($fp, $input, $delimiter, $enclosure);
        // ... rewind the "file" so we can read what we
just wrote...
        rewind($fp);
        // ... read the entire line into a variable...
        $data = stream_get_contents($fp);
        // ... close the "file"...
        fclose($fp);
        // ... and return the $data to the caller, with
the trailing newline from fgets() removed.
        return $data; //rtrim($data, "\r");
    }

    // Параметры сортировщика
    $data_dir_name = "input";   // Каталог с исходными
файлами
    $res_dir_name = "output";   // Каталог с
отсортированными файлами
    $key_list_str = 'email';        // Название столбца
для сортировки (с нуля)
    if(!empty($_REQUEST['action']) &&
$_REQUEST['action'] = 'run') {
        if(!isset($_REQUEST['charset'])) {
            $charset =  'no';
        } else {
            $charset =  $_REQUEST['charset'];
        }
        $k = 0;
```

```
        $er = 0;
        $error_file_names = array();
        $all_file_count = 0;
        $entries =  scandir($data_dir_name);
        foreach($entries as $entry) {
            if(mb_strpos($entry, '.csv') !== false) {
// Обрабатываем только CSV файлы
                $filepath_in = $data_dir_name . "/" .
$entry;
                $file_size = filesize($filepath_in);
                $csv_data_arr = csv2array($filepath_in,
$file_size);
                $header_arr[0] =
array_shift($csv_data_arr);
                $key_list = array_keys($header_arr[0],
$key_list_str);
                usort($csv_data_arr,
sort_arr_ncol($key_list, 'ASC'));
                $csv_data_arr = array_merge($header_arr,
$csv_data_arr);
    // Формируем строку для CSV файла
                $res_csv_file = '';
                foreach($csv_data_arr as $key_row =>
$csv_data_arr_row) {
                    $res_csv_file .=
str_putcsv($csv_data_arr_row);
                }
                if($charset == 'yes') {
                    $res_csv_file = iconv("WINDOWS-
1251", "UTF-8", $res_csv_file);
```

```
                    if($res_csv_file == false) {
                        $res_csv_file = iconv("WINDOWS-
1251", "UTF-8//IGNORE", $res_csv_file);
                        $error_file_names[] = $entry;
                    }
                }
                $filepath_out = $res_dir_name . "/" .
$entry;
                file_put_contents($filepath_out,
$res_csv_file) ? $k++ : $er++;
                $all_file_count++;
            }
        }
        echo "<div id=\"stat_block\">Обработано файлов:
<span>" . $k . "</span> из <span>" . $all_file_count .
"</span>. Ошибок: <span>" . $er . "</span></div>";
        echo "<div id=\"error_files\"><p>" .
implode("</p><p>", $error_file_names) . "</p></div>";
    }
    ?>
    <div><a href="/csv-sorter/">Перезагрузка</a> :: <a
href="/csv-sorter/?action=run">Запуск</a>...</div>
    </body>
    </html>
```

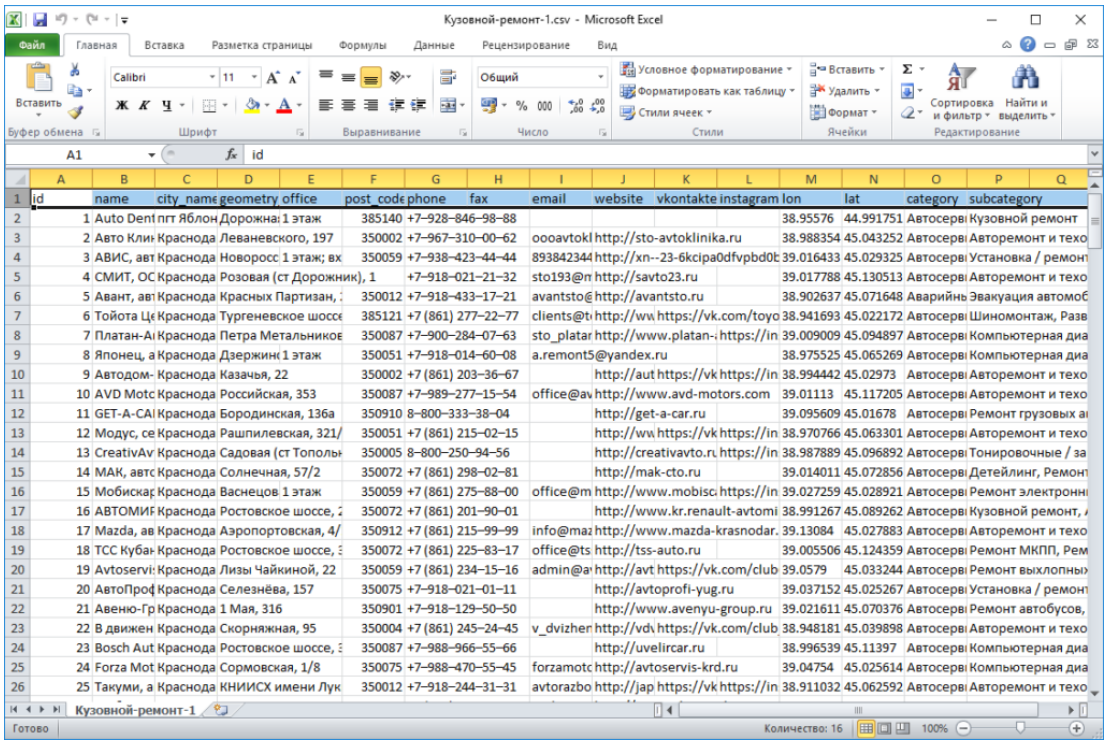
Алгоритм работы следующий. Считываются файлы из заданного каталога. Далее данные преобразуются из CSV формата в PHP массив — это, по сути, импорт CSV.

Функция *csv2array($filepath_in, $file_size)* считывает CSV данные из файла и преобразует их в массив с помощью стандартной функции PHP *fgetcsv()*.

Далее массив сортируется по заданному полю с помощью функции *sort_arr_ncol()*. Полезная штука в разных задачах для сортировки многомерных массивов по столбцам.

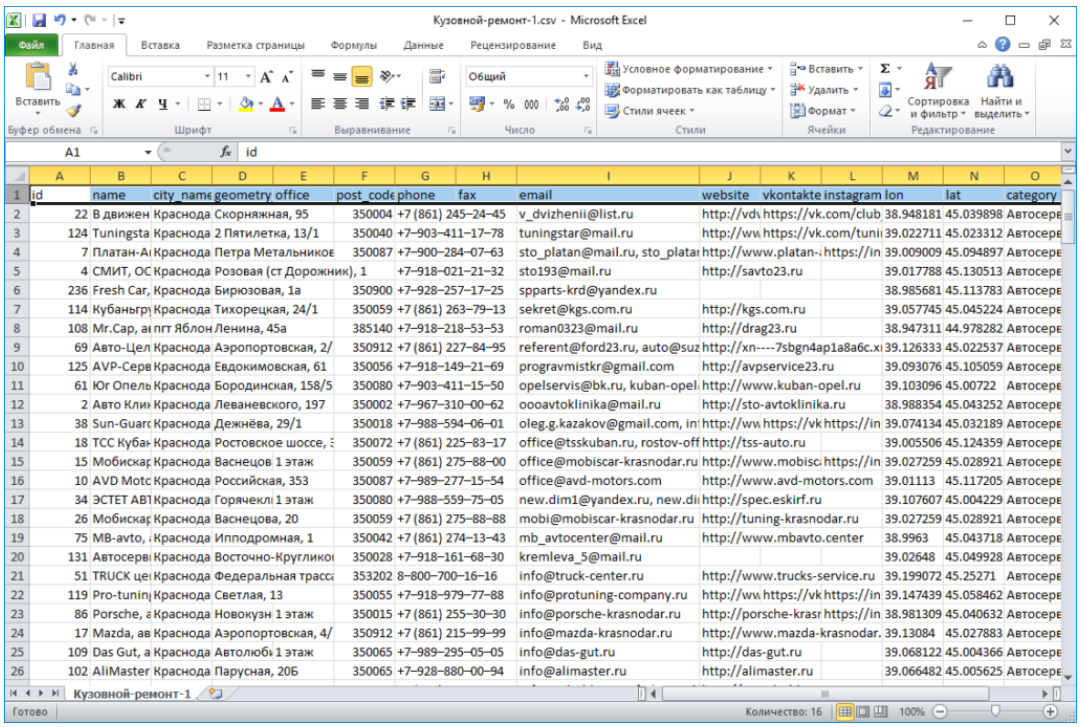
Далее обработанные данные экспортируются обратно в файл. Вначале преобразуем PHP массив в строку в формате CSV при помощи функции *str_putcsv()* и записываем результат в файл.

Список компаний до сортировки:



id	name	city_name	geometry	office	post_code	phone	fax	email	website	vkontakte	instagram	lon	lat	category	subcategory
1	Auto Dent	пгт Яблон	Дорожна	1 этаж	385140	+7–928–846–98–88						38.95576	44.991751	Автосерві	Кузовной ремонт
2	Авто Клин	Краснода	Леваневского, 197		350002	+7–967–310–00–62		oooavtokl	http://sto-avtoklinika.ru			38.988354	45.043252	Автосерві	Авторемонт и техо
3	АВИС, авт	Краснода	Новоросс	1 этаж; вх	350059	+7–938–423–44–44		893842344	http://xn--23-6kcipa0dfvpbd0b			39.016433	45.029325	Автосерві	Установка / ремонт
4	СМИТ, ОС	Краснода	Розовая (ст Дорожник), 1			+7–918–021–21–32		sto193@m	http://savto23.ru			39.017788	45.130513	Автосерві	Авторемонт и техо
5	Авант, авт	Краснода	Красных Партизан,		350012	+7–918–433–17–21		avantsto@	http://avantsto.ru			38.902637	45.071648	Аварийнь	Эвакуация автомоб
6	Тойота Це	Краснода	Тургеневское шосс		385121	+7 (861) 277–22–77		clients@to	http://ww	https://vk.com/toyo		38.941693	45.022172	Автосерві	Шиномонтаж, Разв
7	Платан-Ав	Краснода	Петра Метальникое		350087	+7–900–284–07–63		sto_platar	http://www.platan-	https://in		39.009009	45.094897	Автосерві	Компьютерная диа
8	Японец, а	Краснода	Дзержинс	1 этаж	350051	+7–918–014–60–08		a.remont5@yandex.ru				38.975525	45.065269	Автосерві	Компьютерная диа
9	Автодом-	Краснода	Казачья, 22		350002	+7 (861) 203–36–67			http://aut	https://vk	https://in	38.994442	45.02973	Автосерві	Авторемонт и техо
10	AVD Motc	Краснода	Российская, 353		350087	+7–989–277–15–54		office@av	http://www.avd-motors.com			39.01113	45.117205	Автосерві	Авторемонт и техо
11	GET-A-CAI	Краснода	Бородинская, 136а		350910	8–800–333–38–04			http://get-a-car.ru			39.095609	45.01678	Автосерві	Ремонт грузовых а
12	Модус, се	Краснода	Рашпилевская, 321/		350051	+7 (861) 215–02–15			http://ww	https://vk	https://in	38.970766	45.063301	Автосерві	Авторемонт и техо
13	CreativAv	Краснода	Садовая (ст Тополь		350005	8–800–250–94–56			http://creativavto.r	https://in		38.987889	45.096892	Автосерві	Тонировочные / за
14	МАК, авто	Краснода	Солнечная, 57/2		350072	+7 (861) 298–02–81			http://mak-cto.ru			39.014011	45.072856	Автосерві	Детейлинг, Ремонт
15	Мобискар	Краснода	Васнецов	1 этаж	350059	+7 (861) 275–88–00		office@m	http://www.mobisc	https://in		39.027259	45.028921	Автосерві	Ремонт электронні
16	АВТОМИР	Краснода	Ростовское шоссе,		350072	+7 (861) 201–90–01			http://www.kr.renault-avtomi			38.991267	45.089262	Автосерві	Кузовной ремонт,
17	Mazda, ав	Краснода	Аэропортовская, 4/		350912	+7 (861) 215–99–99		info@maz	http://www.mazda-krasnodar.			39.13084	45.027883	Автосерві	Авторемонт и техо
18	ТСС Куба	Краснода	Ростовское шоссе,		350072	+7 (861) 225–83–17		office@ts	http://tss-auto.ru			39.005506	45.124359	Автосерві	Ремонт МКПП, Рем
19	Avtoservi	Краснода	Лизы Чайкиной, 22		350059	+7 (861) 234–15–16		admin@av	http://avt	https://vk.com/club		39.0579	45.033244	Автосерві	Ремонт выхлопны
20	АвтоПроф	Краснода	Селезнёва, 157		350075	+7–918–021–01–11			http://avtoprofi-yug.ru			39.037152	45.025267	Автосерві	Установка / ремонт
21	Авеню-Гр	Краснода	1 Мая, 316		350901	+7–918–129–50–50			http://www.avenyu-group.ru			39.021611	45.070376	Автосерві	Ремонт автобусов,
22	В движен	Краснода	Скорняжная, 95		350004	+7 (861) 245–24–45		v_dvizhen	http://vd	https://vk.com/club		38.948181	45.039898	Автосерві	Авторемонт и техо
23	Bosch Aut	Краснода	Ростовское шоссе,		350087	+7–988–966–55–66			http://uvelircar.ru			38.996539	45.11397	Автосерві	Компьютерная диа
24	Forza Mot	Краснода	Сормовская, 1/8		350075	+7–988–470–55–45		forzamotc	http://avtoservis-krd.ru			39.04754	45.025614	Автосерві	Компьютерная диа
25	Такуми, а	Краснода	КНИИСХ имени Лук		350012	+7–918–244–31–31		avtorazbo	http://jap	https://vk	https://in	38.911032	45.062592	Автосерві	Авторемонт и техо

Пример файла после сортировки по столбцу email:



Кузовной-ремонт-1.csv - Microsoft Excel

id	name	city_name	geometry office	post_code	phone	fax	email	website	vkontakte	instagram	lon	lat	category
22	В движен	Краснода	Скорняжная, 95	350004	+7 (861) 245-24-45		v_dvizhenii@list.ru	http://vd	https://vk.com/club		38.948181	45.039898	Автосерв
124	Tuningsta	Краснода	2 Пятилетка, 13/1	350040	+7-903-411-17-78		tuningstar@mail.ru	http://ww	https://vk.com/tuni		39.022711	45.023312	Автосерв
7	Платан-А	Краснода	Петра Метальникове	350087	+7-900-284-07-63		sto_platan@mail.ru, sto_plata	http://www.platan-	https://in		39.009009	45.094897	Автосерв
4	СМИТ, ОС	Краснода	Розовая (ст Дорожник), 1		+7-918-021-21-32		sto193@mail.ru	http://savto23.ru			39.017788	45.130513	Автосерв
236	Fresh Car,	Краснода	Бирюзовая, 1а	350900	+7-928-257-17-25		spparts-krd@yandex.ru				38.985681	45.113783	Автосерв
114	Кубаньгр	Краснода	Тихорецкая, 24/1	350059	+7 (861) 263-79-13		sekret@kgs.com.ru	http://kgs.com.ru			39.057745	45.045224	Автосерв
108	Mr.Car, ав	пгт Яблон	Ленина, 45а	385140	+7-918-218-53-53		roman0323@mail.ru	http://drag23.ru			38.947311	44.978282	Автосерв
69	Авто-Цел	Краснода	Аэропортовская, 2/	350912	+7 (861) 227-84-95		referent@ford23.ru, auto@suz	http://xn----7sbgn4ap1a8a6c.x			39.126333	45.022537	Автосерв
125	AVP-Серв	Краснода	Евдокимовская, 61	350056	+7-918-149-21-69		progravmistkr@gmail.com	http://avpservice23.ru			39.093076	45.105059	Автосерв
61	Юг Опель	Краснода	Бородинская, 158/5	350080	+7-903-411-15-50		opelservis@bk.ru, kuban-opel	http://www.kuban-opel.ru			39.103096	45.00722	Автосерв
2	Авто Клин	Краснода	Леваневского, 197	350002	+7-967-310-00-62		oooavtoklinika@mail.ru	http://sto-avtoklinika.ru			38.988354	45.043252	Автосерв
38	Sun-Guar	Краснода	Дежнёва, 29/1	350018	+7-988-594-06-01		oleg.g.kazakov@gmail.com, in	http://ww	https://vk	https://in	39.074134	45.032189	Автосерв
18	ТСС Куба	Краснода	Ростовское шоссе,	350072	+7 (861) 225-83-17		office@tsskuban.ru, rostov-off	http://tss-auto.ru			39.005506	45.124359	Автосерв
15	Мобискар	Краснода	Васнецов 1 этаж	350059	+7 (861) 275-88-00		office@mobiscar-krasnodar.ru	http://www.mobisc	https://in		39.027259	45.028921	Автосерв
10	AVD Motc	Краснода	Российская, 353	350087	+7-989-277-15-54		office@avd-motors.com	http://www.avd-motors.com			39.01113	45.117205	Автосерв
34	ЭСТЕТ АВТ	Краснода	Горячекл 1 этаж	350080	+7-988-559-75-05		new.dim1@yandex.ru, new.di	http://spec.eskirf.ru			39.107607	45.004229	Автосерв
26	Мобискар	Краснода	Васнецова, 20	350059	+7 (861) 275-88-88		mobi@mobiscar-krasnodar.ru	http://tuning-krasnodar.ru			39.027259	45.028921	Автосерв
75	MB-avto,	Краснода	Ипподромная, 1	350042	+7 (861) 274-13-43		mb_avtocenter@mail.ru	http://www.mbavto.center			38.9963	45.043718	Автосерв
131	Автосерв	Краснода	Восточно-Круглико	350028	+7-918-161-68-30		kremleva_5@mail.ru				39.02648	45.049928	Автосерв
51	TRUCK це	Краснода	Федеральная трасс	353202	8-800-700-16-16		info@truck-center.ru	http://www.trucks-service.ru			39.199072	45.25271	Автосерв
119	Pro-tuning	Краснода	Светлая, 13	350055	+7-918-979-77-88		info@protuning-company.ru	http://ww	https://vk	https://in	39.147439	45.058462	Автосерв
86	Porsche, а	Краснода	Новокузн 1 этаж	350015	+7 (861) 255-30-30		info@porsche-krasnodar.ru	http://porsche-krasr	https://in		38.981309	45.040632	Автосерв
17	Mazda, ав	Краснода	Аэропортовская, 4/	350912	+7 (861) 215-99-99		info@mazda-krasnodar.ru	http://www.mazda-krasnodar.			39.13084	45.027883	Автосерв
109	Das Gut, а	Краснода	Автолюб 1 этаж	350065	+7-989-295-05-05		info@das-gut.ru	http://das-gut.ru			39.068122	45.004366	Автосерв
102	AliMaster	Краснода	Парусная, 20Б	350065	+7-928-880-00-94		info@alimaster.ru	http://alimaster.ru			39.066482	45.005625	Автосерв

Теперь мы с вами умеем сохранять результаты парсинга не только в JSON формате, но и в CSV. А уже с CSV форматом могут работать много разных программ и движков сайтов. Например, часто с помощью CSV файлов экспортируются и импортируются товары в интернет-магазинах.

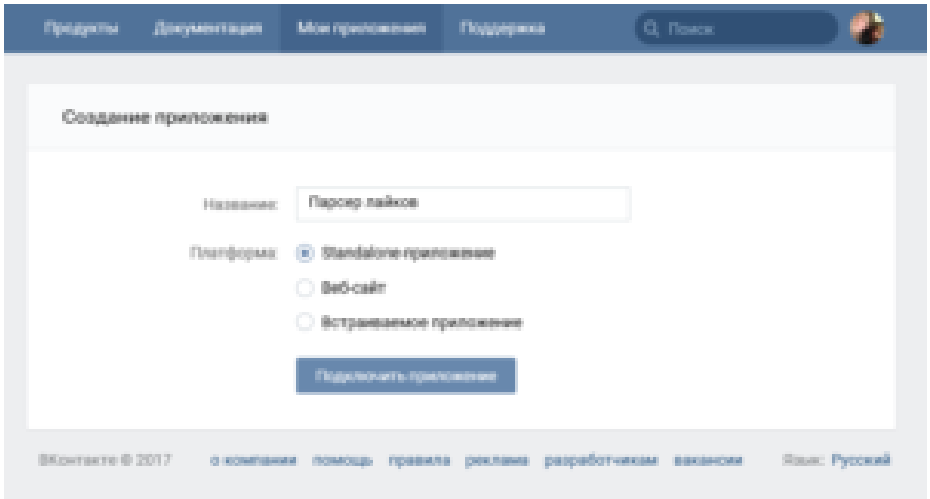
2.8. Парсер лайков из ВКонтакте

В данном уроке по PHP парсерам обойдёмся без PHP. Будем собирать ID пользователей, лайкнувших заданный товар. Для этой цели воспользуемся API социальной сети ВКонтакте.

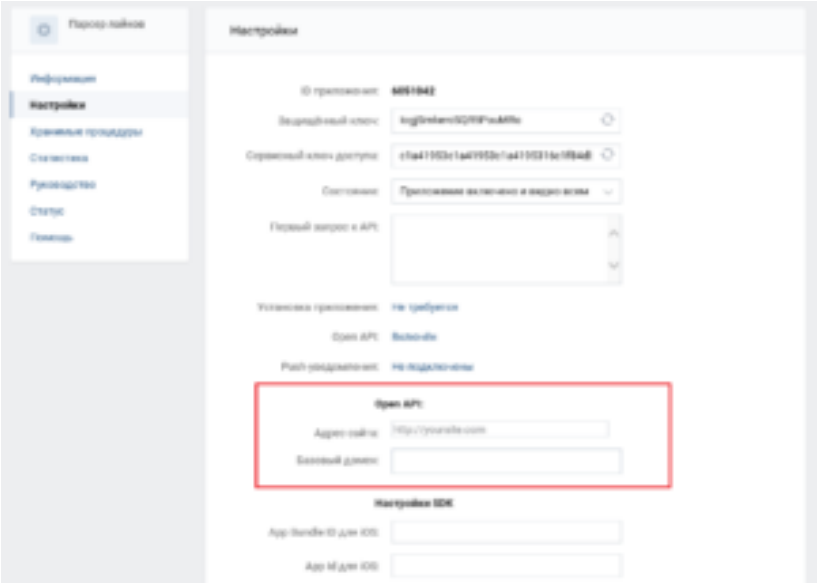
Можно, конечно, пойти стандартным путём и парсить ВК с помощью PHP и cURL, но когда у сайта есть рабочее API, то удобнее пользоваться им. В частности, в ВК можно получить через встроенное API самые разные данные, но при этом есть ограничение на частоту запросов.

Для успешного старта работы с API VK стоит изучить официальную документацию и скудные примеры. Для себя выбрал вариант работы по Open API. Кстати, запросы к API VK отправляются по технологии AJAX, а ответы приходят в формате JSON, не зря мы ранее знакомились с этой схемой данных.

Чтобы можно было работать с API социальной сети нужно в своём скрипте подключить JS библиотеку //vk.com/js/api/openapi.js и создать Standalone приложение.



Далее в настройках видим *ID приложения*, а также надо задать домен и адрес сайта с нашим парсером.



Теперь мы готовы к написанию парсера лайков. В качестве подопытного выберем маркет с товарами бренда Кухни Мария.

Парсер лайков во ВКонтакте без PHP

Код файла index.html.

```
<!doctype html>
<html>
<head>
    <title>Парсер информации из ВКонтакте</title>
    <meta http-equiv="Content-Type"
```

```
content="text/html; charset=utf-8">
        <!--meta name="robots"
content="noindex,nofollow"-->
        <script src="//vk.com/js/api/openapi.js"
type="text/javascript" defer></script>
        <script
src="//ajax.googleapis.com/ajax/libs/jquery/3.1.0/jquer
y.min.js" type="text/javascript"></script>
    </head>
    <body>
    <style>
    .wrapper {
        max-width: 600px;
        margin: 0 auto;
    }
    h1, p {
        text-align: center;
    }
    .action_form {
        max-width: 560px;
        margin: 0 auto;
    }
    .action_form input,
    .action_form textarea {
        width: 100%;
    }
    .action_form textarea {
        min-height: 100px;
    }
    input[type="text"] {
```

```
    font-size: 1em;
    min-height: 36px;
    box-sizing: border-box;
}
input[type="submit"],
input[type="button"] {
    padding: 8px 12px;
    margin: 12px auto;
    font-size: 1.2em;
    font-weight: 400;
    line-height: 1.2em;
    text-decoration: none;
    display: inline-block;
    cursor: pointer;
    border: 2px solid #007700;
    border-radius: 2px;
    background-color: transparent;
    color: #007700;
}
input[type="submit"]:hover,
input[type="button"]:hover {
    background-color: #009900;
    color: #fff;
}
.result {
    border: 1px dotted #000;
    width: 100%;
    height: auto;
    overflow-y: auto;
    margin: 0px auto;
```

```
        padding: 10px;
    }
    .copyright {
        text-align: center;
    }
    .copyright a {
        color: #000;
    }
    .copyright a:hover {
        text-decoration: none;
    }
    .red {
        color: #770000;
    }
    .green {
        color: #007700;
    }
    </style>
    <div class="wrapper">
        <h1>Парсер лайков на товары из ВК</h1>
        <p>Собираем список ID пользователей, которые
лайкнули заданный товар.</p>
        <form id="action_form" class="action_form"
action="" method="post">
            <input type="text" id="vk_url" name="vk_url"
/>
            <input type="button" id="btn_action_likes"
name="btn_action_likes" value="Собрать IDs" />
        </form>
        <div id="result" class="result"></div>
```

```
        <div id="errors_block"
class="errors_block"></div>
        <div class="copyright">&copy; Идея и реализация
- <a href="http://seorubl.ru/" target="_blank"
title="Записки Предприимчивого Человека"
rel="generator">ПЧ</a> // 28.05.2017 г.</div>
    </div>
    <script type="text/javascript">
        jQuery(document).ready(function($) {
            $('#btn_action_likes').click(function() {
                function
get_vk_market_likes(vk_owner_id, vk_post_id,
vk_post_type) {
                    // Инициализация VK API
                    VK.init({
                        apiId: ID приложения
                    });

                    $('#result').html('<p
class="green">Загрузка данных ... </p>');

                    VK.Api.call('likes.getList',
{owner_id: vk_owner_id, item_id: vk_post_id, type:
vk_post_type, count: 1000, offset: 0, v: '5.8'},
function(data) {
                        if(data.response) {
                            $('#result').html('');
                            data.response.items.forEach
(function(item, i, arr) {
                                $('#result').html($('#r
```

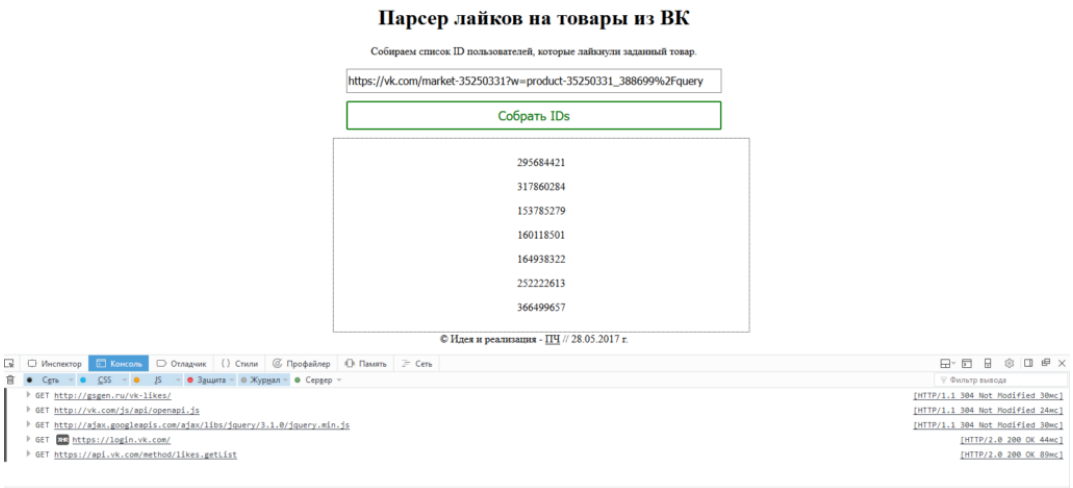
```
esult').html() + "<p>" + item +
"</p>").hide().fadeIn('slow');
                                    });
                                } else {
                                    $('#errors_block').html($('
.app_alert').html() + '<p class="red">' +
data.error.error_code + '// ' + data.error.error_msg +
'</p>');
                                }
                            });
                        }
                        var vk_url = $("#vk_url").val();
                        if(vk_url) {
                            $('#result').html('');
                            $('#errors_block').html('');
                            vk_url =
vk_url.slice(vk_url.indexOf("w=product") +
9).split('_');
                            var vk_owner_id =
parseInt(vk_url[0]);
                            var vk_post_id =
parseInt(vk_url[1]);
                            var vk_post_type = 'market';
                            get_vk_market_likes(vk_owner_id,
vk_post_id, vk_post_type);
                        } else {
                            $('#errors_block').html($('#errors_
block').html() + "<p>Введите адрес страницы с
товаром</p>").hide().fadeIn('slow');
                        }
```

```
                return true;
            });
        });
    </script>
    </body>
    </html>
```

При запуске парсера не забудьте в настройке *apiId: ID приложения* поставить ID своего приложения.

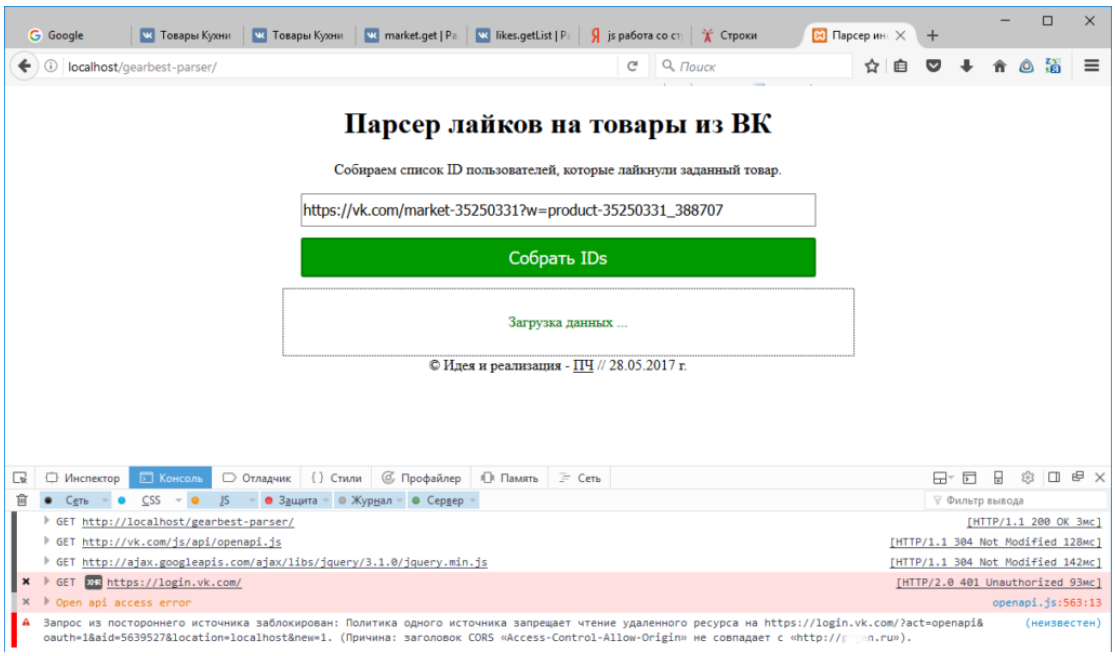
Алгоритм работы парсера лайков следующий. Мы задаём адрес карточки нужного нам товара. Скрипт из URL достаёт ID маркета и товара, далее делает запрос к API ВК с целью получить список ID пользователей, которые поставили лайки заданному проекту кухни. Для получения информации по лайкам используется метод *likes.getList*.

Затем список ID пользователей выводится на экран. Потом полученную информацию можно использовать, например, для настройки таргетированной рекламы в социальной сети.



Особенности работы с API VK

Запросы к API VK можно делать только с заданного в приложении домена. Если попробовать сделать AJAX запрос с другого сайта, то вывалится ошибка ограничения кроссдоменных запроосов.



В примере приведён очень простой вариант парсера. В более сложных случаях нам бывает нужно, например, вначале спарсить категории (альбомы) магазина, а потом товары и лайки. В таких случаях следует делать запросы последовательно, вложенными друг в друга, чтобы не было проблем с асинхронной подгрузкой данных.

Также обратите внимание на то, что число возращаемых методом API результатов ограничено 1000. Но нередко бывают записи или товары, которые собирают по несколько тысяч лайков или репостов. В этом случае надо пользоваться рекурсией и вызывать методы со смещением, задаваемым параметром *offset*.

Парсер лайков из ВК написан без применения PHP, в скрипте использовался только JS и открытое API социальной сети. После небольших доработок этот же парсер можно применять для сбора информации по лайкам для постов, фото, видео и пр.

Далее поговорим на тему автоматического постинга информации в CMS, на примере WordPress. Данные не только нужно собирать, но и потом куда-то с пользой применять.

3. Импорт записей в WordPress из CSV

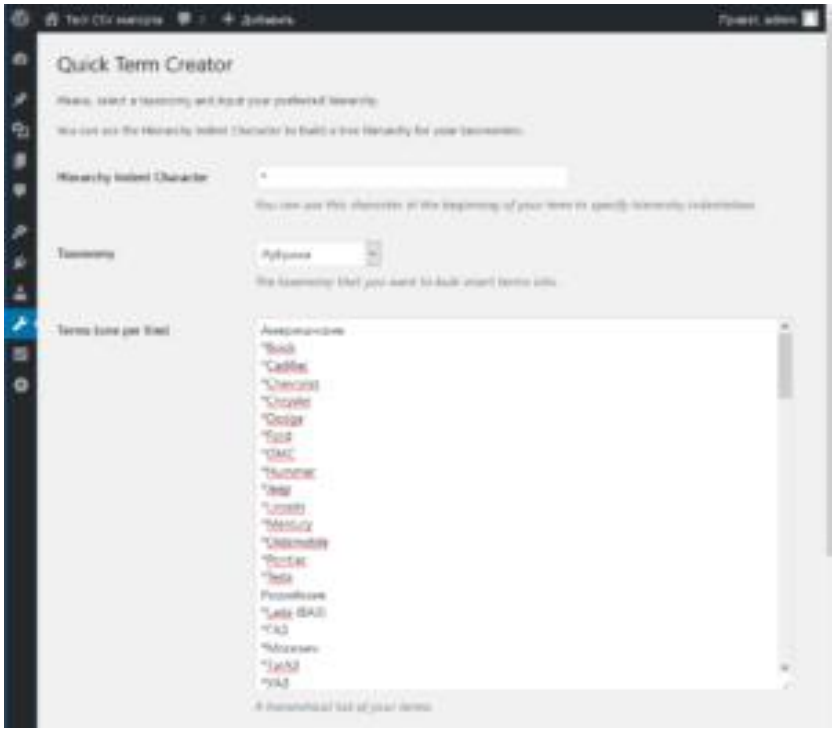
В предыдущей статье пообещал показать пример импорта записей в WordPress из файла с CSV данными. Вначале думал написать собственный скрипт для автоматической публикации записей, но потом решил покопаться в готовых плагинах. И нашёл несколько подходящих вариантов.

Для примера поставил для себя задачу создать категории по маркам автомобилей и закинуть в них по одной статье с шаблонным контентом. Исходные данные не стал парсить, а для простоты просто скопировал методом копипаст.

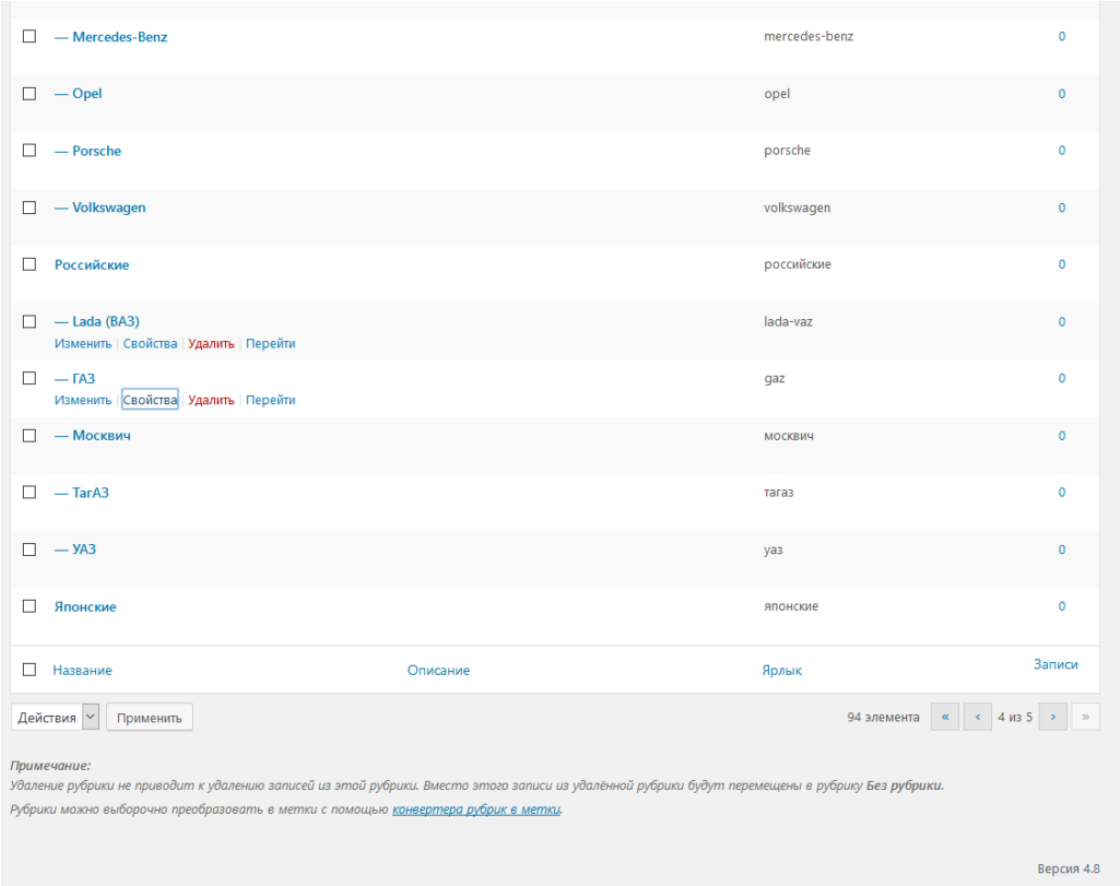
Создание большого количества категорий в WordPress

Поискал плагины для импорта записей и создания категорий. Много попадалось скриптов, которые давно не обновлялись, часть уже не работает с новыми версиями WordPress. В итоге решил действовать в 2 этапа: вначале создать категории, а потом залить в них записи.

Для автоматического создания рубрик воспользовался плагином Quick Term Creator. Нужно сформировать список категорий и подкатегорий с помощью символа иерархии «*».



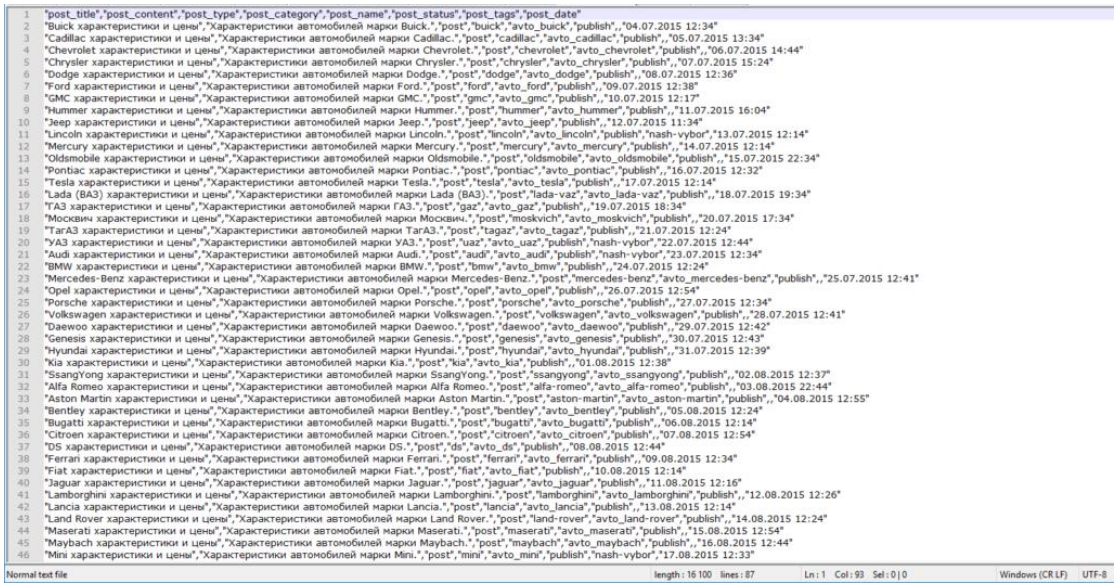
Проблема с этим плагином в том, что он не приспособлен для работы с категориями на русском языке — нет автоматической транслитерации для генерации алиасов из названий.



Исправил неправильные адреса с кириллицы на транслит.

Импорт записей из CSV файла

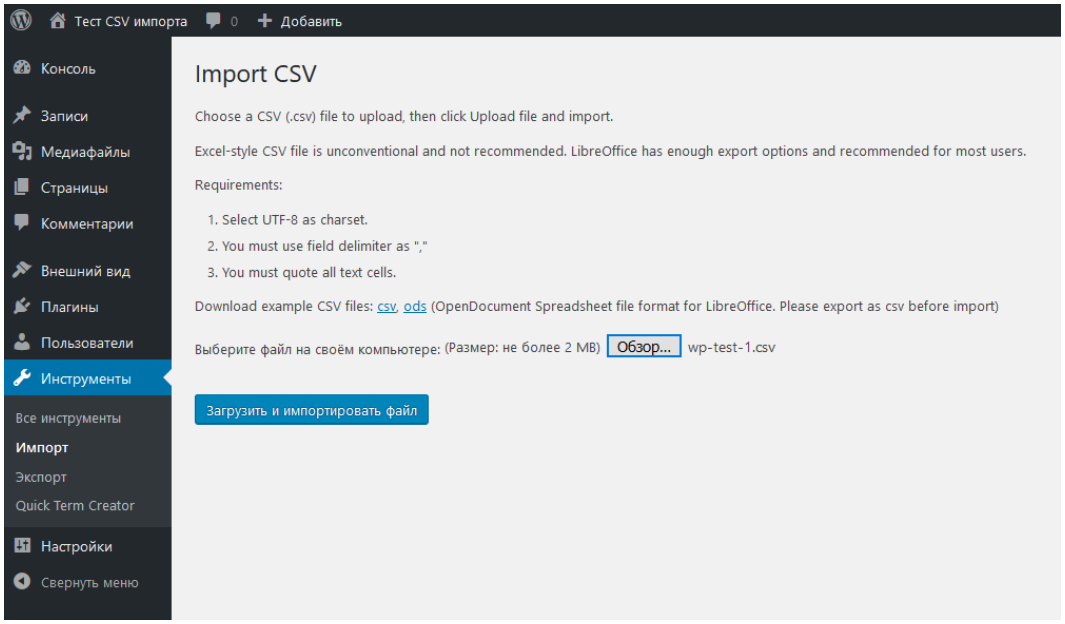
В общем случае данные можно было спарсить и результат экспортировать в CSV формат. В этом случае я просто вручную сформировал CSV файл с помощью Excel.



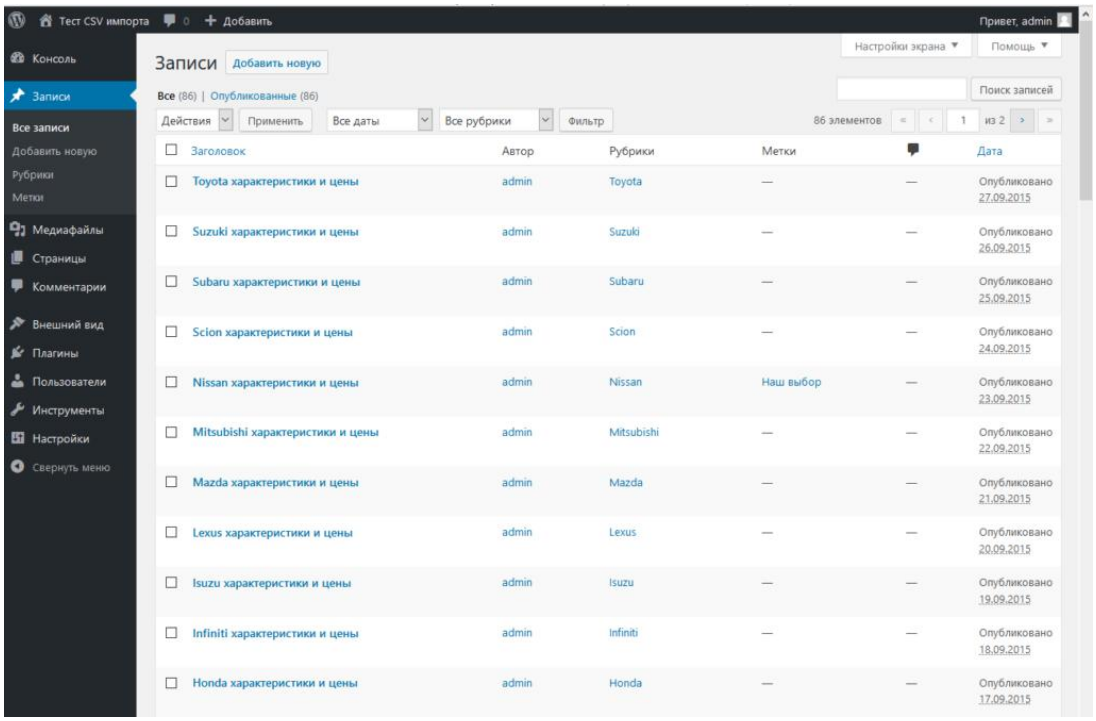
```
"post_title","post_content","post_type","post_category","post_name","post_status","post_tags","post_date"
"Buick характеристики и цены","Характеристики автомобилей марки Buick.","post","buick","avto_buick","publish",,"04.07.2015 12:34"
"Cadillac характеристики и цены","Характеристики автомобилей марки Cadillac.","post","cadillac","avto_cadillac","publish",,"05.07.2015 13:34"
"Chevrolet характеристики и цены","Характеристики автомобилей марки Chevrolet.","post","chevrolet","avto_chevrolet","publish",,"06.07.2015 14:44"
"Chrysler характеристики и цены","Характеристики автомобилей марки Chrysler.","post","chrysler","avto_chrysler","publish",,"07.07.2015 15:24"
"Dodge характеристики и цены","Характеристики автомобилей марки Dodge.","post","dodge","avto_dodge","publish",,"08.07.2015 12:36"
"Ford характеристики и цены","Характеристики автомобилей марки Ford.","post","ford","avto_ford","publish",,"09.07.2015 12:38"
"GMC характеристики и цены","Характеристики автомобилей марки GMC.","post","gmc","avto_gmc","publish",,"10.07.2015 12:17"
"Hummer характеристики и цены","Характеристики автомобилей марки Hummer.","post","hummer","avto_hummer","publish",,"11.07.2015 16:04"
"Jeep характеристики и цены","Характеристики автомобилей марки Jeep.","post","jeep","avto_jeep","publish",,"12.07.2015 11:34"
"Lincoln характеристики и цены","Характеристики автомобилей марки Lincoln.","post","lincoln","avto_lincoln","publish","nash-vybor","13.07.2015 12:14"
"Mercury характеристики и цены","Характеристики автомобилей марки Mercury.","post","mercury","avto_mercury","publish",,"14.07.2015 12:14"
"Oldsmobile характеристики и цены","Характеристики автомобилей марки Oldsmobile.","post","oldsmobile","avto_oldsmobile","publish","15.07.2015 22:34"
"Pontiac характеристики и цены","Характеристики автомобилей марки Pontiac.","post","pontiac","avto_pontiac","publish",,"16.07.2015 12:32"
"Tesla характеристики и цены","Характеристики автомобилей марки Tesla.","post","tesla","avto_tesla","publish",,"17.07.2015 12:14"
"Lada (ВАЗ) характеристики и цены","Характеристики автомобилей марки Lada (ВАЗ).","post","lada-vaz","avto_lada-vaz","publish",,"18.07.2015 19:34"
"ГАЗ характеристики и цены","Характеристики автомобилей марки ГАЗ.","post","gaz","avto_gaz","publish",,"19.07.2015 18:34"
"Москвич характеристики и цены","Характеристики автомобилей марки Москвич.","post","moskvich","avto_moskvich","publish",,"20.07.2015 17:34"
"ТагАЗ характеристики и цены","Характеристики автомобилей марки ТагАЗ.","post","tagaz","avto_tagaz","publish",,"21.07.2015 12:24"
"УАЗ характеристики и цены","Характеристики автомобилей марки УАЗ.","post","uaz","avto_uaz","publish","nash-vybor","22.07.2015 12:44"
"Audi характеристики и цены","Характеристики автомобилей марки Audi.","post","audi","avto_audi","publish","nash-vybor","23.07.2015 12:34"
"BMW характеристики и цены","Характеристики автомобилей марки BMW.","post","bmw","avto_bmw","publish",,"24.07.2015 12:24"
"Mercedes-Benz характеристики и цены","Характеристики автомобилей марки Mercedes-Benz.","post","mercedes-benz","avto_mercedes-benz","publish",,"25.07.2015 12:41"
"Opel характеристики и цены","Характеристики автомобилей марки Opel.","post","opel","avto_opel","publish",,"26.07.2015 12:54"
"Porsche характеристики и цены","Характеристики автомобилей марки Porsche.","post","porsche","avto_porsche","publish",,"27.07.2015 12:34"
"Volkswagen характеристики и цены","Характеристики автомобилей марки Volkswagen.","post","volkswagen","avto_volkswagen","publish",,"28.07.2015 12:41"
"Daewoo характеристики и цены","Характеристики автомобилей марки Daewoo.","post","daewoo","avto_daewoo","publish",,"29.07.2015 12:42"
"Genesis характеристики и цены","Характеристики автомобилей марки Genesis.","post","genesis","avto_genesis","publish",,"30.07.2015 12:43"
"Hyundai характеристики и цены","Характеристики автомобилей марки Hyundai.","post","hyundai","avto_hyundai","publish",,"31.07.2015 12:39"
"Kia характеристики и цены","Характеристики автомобилей марки Kia.","post","kia","avto_kia","publish",,"01.08.2015 12:38"
"SsangYong характеристики и цены","Характеристики автомобилей марки SsangYong.","post","ssangyong","avto_ssangyong","publish",,"02.08.2015 12:37"
"Alfa Romeo характеристики и цены","Характеристики автомобилей марки Alfa Romeo.","post","alfa-romeo","avto_alfa-romeo","publish",,"03.08.2015 22:44"
"Aston Martin характеристики и цены","Характеристики автомобилей марки Aston Martin.","post","aston-martin","avto_aston-martin","publish",,"04.08.2015 12:55"
"Bentley характеристики и цены","Характеристики автомобилей марки Bentley.","post","bentley","avto_bentley","publish",,"05.08.2015 12:24"
"Bugatti характеристики и цены","Характеристики автомобилей марки Bugatti.","post","bugatti","avto_bugatti","publish",,"06.08.2015 12:14"
"Citroen характеристики и цены","Характеристики автомобилей марки Citroen.","post","citroen","avto_citroen","publish",,"07.08.2015 12:54"
"DS характеристики и цены","Характеристики автомобилей марки DS.","post","ds","avto_ds","publish",,"08.08.2015 12:44"
"Ferrari характеристики и цены","Характеристики автомобилей марки Ferrari.","post","ferrari","avto_ferrari","publish",,"09.08.2015 12:34"
"Fiat характеристики и цены","Характеристики автомобилей марки Fiat.","post","fiat","avto_fiat","publish",,"10.08.2015 12:14"
"Jaguar характеристики и цены","Характеристики автомобилей марки Jaguar.","post","jaguar","avto_jaguar","publish",,"11.08.2015 12:16"
"Lamborghini характеристики и цены","Характеристики автомобилей марки Lamborghini.","post","lamborghini","avto_lamborghini","publish",,"12.08.2015 12:26"
"Lancia характеристики и цены","Характеристики автомобилей марки Lancia.","post","lancia","avto_lancia","publish",,"13.08.2015 12:14"
"Land Rover характеристики и цены","Характеристики автомобилей марки Land Rover.","post","land-rover","avto_land-rover","publish",,"14.08.2015 12:24"
"Maserati характеристики и цены","Характеристики автомобилей марки Maserati.","post","maserati","avto_maserati","publish",,"15.08.2015 12:54"
"Maybach характеристики и цены","Характеристики автомобилей марки Maybach.","post","maybach","avto_maybach","publish",,"16.08.2015 12:44"
"Mini характеристики и цены","Характеристики автомобилей марки Mini.","post","mini","avto_mini","publish","nash-vybor","17.08.2015 12:33"
```

Normal text file | length : 16 100 lines : 87 | Ln : 1 Col : 93 Sel : 0 | 0 | Windows (CR LF) | UTF-8

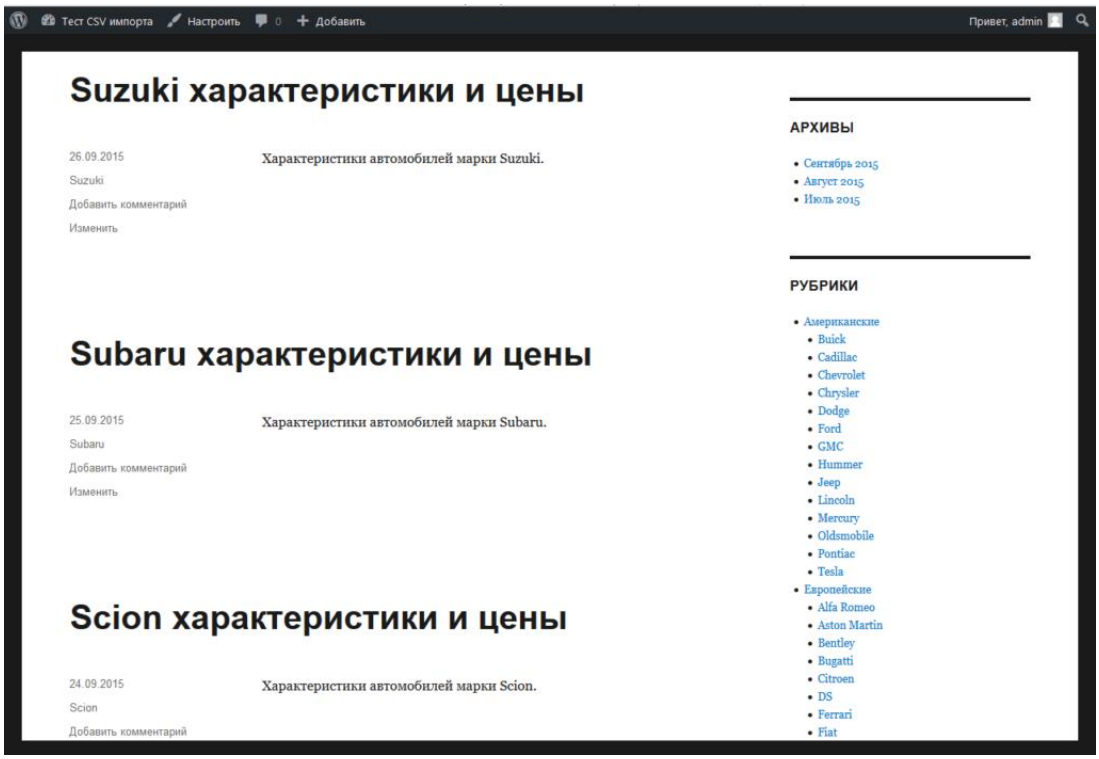
Далее воспользовался плагином Import CSV.



В течение нескольких минут записи импортировались в WordPress.



В итоге получился заполненный сайт с категориями и тестовыми записями.



Если не учитывать время, затраченное на изучение разных плагинов, то на создание подобного сайта потребуется 20-30 минут со сбором данных и установкой движка.

Это была заключительная статья по разработке парсеров. Теперь мы умеем парсить сайты с авторизацией и без авторизации. Работать с Cookies и API разных сайтов, например vk.com. Преобразовывать полученные данные в JSON или CSV формат. И теперь мы знаем, как залить спарсенные данные на наш сайт. Пора начать зарабатывать на своих новых знаниях и навыках.

4. Как заработать на PHP парсерах?

Спрос на различные парсеры на рынке точно есть, но есть и конкуренция.

Написание парсеров на заказ – удобная услуга для площадки Kwork.

Можно разработать несколько типовых парсеров под популярные площадки и продавать их модификации под конкретные требования заказчиков. Например, парсеры крупных оптовых магазинов, у которых нет API и готовых решений для интеграции с ритейлерами.

Другой вариант – это продажа не самого парсера, а спарсенных и сгруппированных данных.

Также парсеры часто входят в состав различных калькуляторов. Можно написать типовые отраслевые калькуляторы и продавать установку и настройку полученных скриптов.

Часто просят написать парсер бесплатно, либо за отзыв. Для таких случаев подготовил универсальный ответ любителям халявы.

PHP парсер HTML сайта бесплатно

Собственно ответ на вопрос, где взять PHP парсер сайтов бесплатно, простой — напишите его сами. Базовый алгоритм работы парсеров я выше разобрал в деталях.

Можно поискать уже готовые, написанные ранее кем-то решения вашей задачи. Но сделать парсер, который на автомате подходил бы ко всем вариантам, наверное, невозможно. Под каждый тип задач нужно разрабатывать свой конкретный продукт.

Итоговая стоимость услуг разработки определяется после получения конкретного технического задания. Часто цена парсера выходит довольно высокой, а связано это с тем, что разработка получается уникальной, заточенной под конкретного заказчика. Разные бывают варианты входных данных, требуется индивидуальный формат результата на выходе.

В этом случае формально тоже можно получить парсер для себя бесплатно. Например, цена разработки парсера получилась 9 000 рублей. Вы ищете 9 человек с похожей проблемой и собираете с них по 1000 рублей, заказываете разработку парсера. Потом делаете 10 копий, 1 себе и 9 отдаёте вашим знакомым.

Спасибо за внимание. Рад быть для вас полезным.

Записки Предприимчивого Человека, 2017