Публикация на тему

# Парсим youtube.com

*Пример парсера youtube.com с использованием Laravel. Требуется сервер Apache и PHP 7.2 и выше.*

**Автор**
Михалькевич Александр Викторович

## Публикация

**Наименование** Парсим youtube.com
**Автор** А.В.Михалькевич
**Специальность** Пример парсера youtube.com с использованием Laravel. Требуется сервер Apache и PHP 7.2 и выше.,
**Анотация**
**Anotation in English**
**Ключевые слова**
**Количество символов** 6545

## Содержание

Приложения

# Введение

# 1 Инструментарий

В примере задействованы следующие инструменты

Apache - сервер;

MySQL - сервер баз данных;

PHP 7.2 - язык программирования;

Laravel - PHP фрэймворк;

# 2 Обновление менеджера зависимостей

Обновим менеджер зависимостей Composer

```
composer self-update
```

# 3 Установка модуля DomCrawler

После обновления менеджера зависимостей, передем в папку с проектом:

```
cd /var/www/laravel
```

Подключим зависимость DomCrawler с помощью следующей команды:

```
 composer require symfony/dom-crawler
```

# 4 Интерфейс

В папке /app создадим папку /Parser. В ней файл ParseContract.php. Рассмотрим содержимое файла.

```
namespace App\Parser;

Interface ParseContract
{
    public function getParse();
    //public function text($obj, $val = null);
    //public function html($obj, $val = null);
}
```

# 5 Имплемент Youtube.php

Создаем еще один файл в этой же папке.

Файл Youtube.php

```
namespace App\Parser;

use Symfony\Component\DomCrawler\Crawler;
//use App\ProductUser;
use App\Googlenew;
use Auth;

class Youtube implements ParseContract
{
    use ParseTrait;
    public $crawler;

    public function __construct()
    {
        set_time_limit(0);
        header('Content-Type: text/html; charset=utf-8');
    }

    public function getParse()
    {
        $ff = 'https://www.youtube.com/results?search_query=test';
        $file = file_get_contents($ff);
        $this->crawler = new Crawler($file);
                $body=$this->crawler->filter('body')->html();
                return $body;
    }
}
```

Получив, таким образом страницу парсинга, можем пройтись по необходимым атрибутам:

```
    public function getParse($country="Belarus")
    {
        $ff = 'https://www.youtube.com/results?search_query='.$country;
        $file = file_get_contents($ff);
        $this->crawler = new Crawler($file);
        $this->crawler->filter(".contains-addto")->each(function (Crawler
$node, $i) {
            $pic = $node->filter('img')->attr('src');
            $this->body .= $pic.'
```

---

```
';
        });
        return $this->body;
    }
```

# 6 HTML код

```
<input type="text" placeholder="Url from Youtube" id="youtube_parse_url">
<input type="button" id="{{$category->id}}" value="Parse" class="parse">
<div id="empty"></div>
```

# 7 jQuery скрипт

Подключаем скрипт parse.js

```
$('.parse').click(function(){
var url = $('#yotube_parse_url').val();
var id = $(this).attr('data-id');
console.log(url, id);
});
```

# 8 Контроллер AjaxParseController и маршрут для него

Сперва создадим контроллер AjaxParseController

```
php artisan make:controller AjaxParseController
```

Перенесем данный контроллер в папку Admin

В файле routes/web.php создадим следующий маршрут

```
 Route::group(['middleware'=>'admin'], function(){
  Route::get('ajax/parse/catalog', 'Admin\AjaxParseController@getCatalog');
 });
```

Рассмотрим контроллер. Т.к. сейчас он находится в папке Admin, то меняем namespace.

```
namespace App\Http\Controllers\Admin;
use App\Http\Controllers\Controller;
use App\Parser\Aliexpress;
class AjaxParseController extends Controller{
 public function getCatalog(){
  $url = $_GET['url'];
  $id = (int)$_GET['id']
  $parse = new Aliexpress();
  $parse->getParse($url, $id);
 }
}
```

# 9 Ajax

Снова возвращаемся к файлу parse.js. После определения переменных, запускаем ajax,

который будет обращаться к серверу и передавать в него нужные значения.

```
$('.parse').click(function(){
var url = $('#parse_aliexpress').val();
var id = $(this).attr('data-id');
 $.ajax({
  type: 'get',
  data: 'id='+id+'&url='+url,
  url: '/ajax/parse/catalog',
  beforeSend: function(){
   $('#empty').html('G');
  },
  success: function(data){
   $('#empty').html(data);
  },
  error: function(msg){
   console.log(msg);
  }
 });
});
```

## 10 Замена пробелов символом +

Пробелы в названиях Yutube заменяет символами +.

```
$str = str_replace(" ", "+", $country);
```

## 11 Перебор элементов

Для перебора элементов воспользуемся методом each

```
$this->crawler->filter('.list-item')->each(function(Crawler $node, $i){
   $name = $node->text();
   $pic = $node->filter('img')->attr('src');
   echo $name;
   echo ' - '.$pic;
});
```

## 12 Добавление данных в базу

Подключаем модель к классу парсинга, и добавляем данные в базу.

```
use App\Product;
...
$this->crawler->filter('.list-item')->each(function(Crawler $node, $i){
   $name = $node->text();
   $pic = $node->filter('img')->attr('src');
```

```
   $obj = new Product;
   $obj->name = $name;
   $obj->picture = $pic;
   $obj->save();
});
...
```

## 13 Проверка элементов на пустоту

Прежде чем обращаться к элементу парсинга, его нужно проверить на пустоту. Делаем это следующим образом:

```
 $risk_price = $node->filter('.price')->count();
 if($risk_price == 0){
  $price = 0;
 }else{
  $price = $node->filter('.price')->text();
 }
d
```

## 14 Сохранение изображений на сервере

Для сохранения изображения на сервере, воспользуемся стандартной функцией copy

```
$arr_pic = explode('/', $pic);
$newfile = $_SERVER['DOCUMENT_ROOT'].'/uploads/thumb/'.end($arr_pic);
if (!copy($pic, $newfile)) {
    echo "не удалось скопировать $pic...\n";
}
```

## Заключение

## Список использованных источников

## Приложения