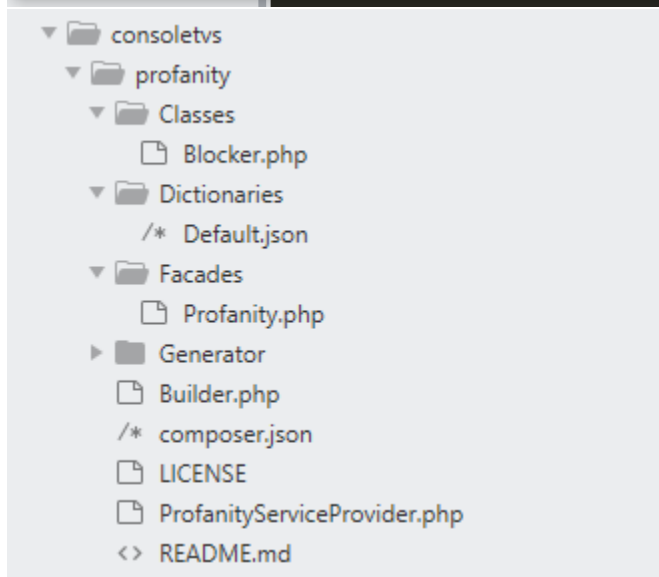


```
view
view:cache      Compile all of the application's Blade templates
view:clear      Clear all compiled view files

D:\InstalledSoftware\xampp\htdocs\lamovie>composer show -i
You are using the deprecated option "installed". Only installed packages are shown by default now. The --all option can
be used to show all packages.
askedio/laravel5-profanity-filter 1.10      A Vendor Package Example
asm89/stack-cors                  v2.0.1  Cross-origin resource sharing library and stack middleware
barryvdh/laravel-debugbar         v3.5.1  PHP Debugbar integration for Laravel
brick/math                        0.8.15  Arbitrary-precision arithmetic library
consoletv/profanity               3.0.1   PHP library to block bad words in a string
dnoegel/php-xdg-base-dir          v0.1.1  Implementation of xdg base directory specification for php
doctrine/inflector                2.0.3   PHP Doctrine Inflector is a small library that can perform string man...
doctrine/instantiator             1.3.1   A small, lightweight utility to instantiate objects in PHP without in...
doctrine/lexer                    1.2.1   PHP Doctrine Lexer parser library that can be used in Top-Down, Recur...
dragonmantank/cron-expression     v2.3.0  CRON for PHP: Calculate the next or previous run date and determine i...
egulias/email-validator           2.1.19  A library for validating emails against several RFCs
facade/flare-client-php           1.3.4   Send PHP errors to Flare
facade/ignition                   2.3.6   A beautiful error page for Laravel applications.
facade/ignition-contracts         1.0.1   Solution contracts for Ignition
fideloper/proxy                   4.4.0   Set trusted proxies for Laravel
filp/whoops                       2.7.3   php error handling for cool kids
fruitcake/laravel-cors            v2.0.1  Adds CORS (Cross-Origin Resource Sharing) headers support in your Lar...
fzaninotto/faker                  v1.9.1  Faker is a PHP library that generates fake data for you.
guzzlehttp/guzzle                 6.5.5   Guzzle is a PHP HTTP client library
guzzlehttp/promises               v1.3.1  Guzzle promises library
guzzlehttp/psr7                   1.6.1   PSR-7 message implementation that also provides common utility methods
hamcrest/hamcrest-php             v2.0.1  This is the PHP port of Hamcrest Matchers
laravel/framework                 v7.25.0 The Laravel Framework.
```



```

namespace ConsoleTVs\Profanity\Classes;

/**
 * This is the profanity class.
 *
 * @author Erik Campobadal <soc@erik.cat>
 */
class Blocker
{
    public $dictionary;
    public $blocker;
    public $text;
    // adjust true/false base n preferref level of filter
    public $strict = true;
    public $strictClean = true;

    /**
     * Setup the Profanity filter.
     *
     * @param string $text
     *
     * @return void
     */
    public function __construct($text, $blocker)
    {
        $this->text = $text;
        $this->blocker = $blocker;
        $this->dictionary = json_decode(file_get_contents(__DIR__.'../../Dictionaries/Default.json'), true);
    }

    /**
     * Set the text to filter.
     *
     * @param string $text
     *
     * @return self
     */
    public function text($text)
    {
        $this->text = $text;

        return $this;
    }

    /**
     * Set the strict mode.
     *
     * @param bool $strict
     *
     * @return self
     */
    public function strict($strict)
    {
        $this->strict = $strict;

        return $this;
    }

    /**
     * Set the strict clean mode.
     *
     * @param bool $strict
     *
     * @return self
     */
    public function strictClean($strict)
    {
        $this->strictClean = $strict;

        return $this;
    }
}

```

```

public function blocker($blocker)
{
    $this->blocker = $blocker;

    return $this;
}

/**
 * Set the blocker dictionary (string = path to json, array = dictionary) (bad words).
 *
 * @param mixed $dictionary
 *
 * @return self
 */
public function dictionary($dictionary)
{
    $this->dictionary = is_array($dictionary) ? $dictionary : json_decode(file_get_contents($dictionary), true);

    return $this;
}

/**
 * Return true if the text is clean.
 *
 * @return bool
 */
public function clean()
{
    return count($this->badWords()) == 0;
}

/**
 * Return the bad words contained in the text.
 *
 * @return array
 */
public function badWords()
{
    return collect($this->dictionary)->filter(function ($value) {
        $matches = [];
        if ($this->strict) {
            return preg_match('/'. $value['word'] . '/iu', $this->text, $matches, PREG_UNMATCHED_AS_NULL);
        }
        $pattern = "/\b{$value['word']}\b/iu";

        return preg_match($pattern, $this->text, $matches, PREG_UNMATCHED_AS_NULL);
    }->map(function ($value) {
        return [
            'language' => $value['language'],
            'word'      => $value['word'],
        ];
    }->toArray());
}

```

```

*/
public function filter()
{
    $bad_words = collect($this->badWords())->pluck('word')->toArray();
    $text = $this->text;
    foreach ($bad_words as $word) {
        if ($this->strict) {
            $text = preg_replace('/'. $word .'/iu', $this->blockWord($word), $text);
        } else {
            $text = preg_replace("/\b". $word ."\b/iu", $this->blockWord($word), $text);
        }
    }
    return $text;
}

/**
 * Returns the blocked word.
 *
 * @param string $word
 *
 * @return string
 */
private function blockWord($word)
{
    if ($this->strictClean) {
        return str_repeat($this->blocker[0], strlen($word));
    }

    return $this->blocker;
}
}

```

```
app.blade.php  x  Blocker.php  ●  Default.json  x
10240      word : fuckyou
10241      },
10242      {
10243          "language": "ph",
10244          "word": "fuck you"
10245      },
10246      {
10247          "language": "ph",
10248          "word": "tanga"
10249      },
10250      {
10251          "language": "ph",
10252          "word": "tanga"
10253      },
10254      {
10255          "language": "ph",
10256          "word": "tanga"
10257      },
10258      {
10259          "language": "ph",
10260          "word": "puting ina"
10261      },
10262      {
10263          "language": "ph",
10264          "word": "puting ina"
10265      },
10266      {
10267          "language": "ph",
10268          "word": "puting ina"
10269      },
10270      {
10271          "language": "ph",
10272          "word": "puting ina"
10273      },
10274      {
10275          "language": "ph",
10276          "word": "puting ina"
10277      }
10278      ,
10279      {
10280          "language": "ph",
10281          "word": "tanga"
10282      },
10283      {
10284          "language": "ph",
10285          "word": "tanga"
10286      },
10287      {
10288          "language": "ph",
10289          "word": "fuck"
10290      }
10291      ]
```

```
useremail.blade.php x showinfo.blade.php x
1
2 <h3>User Info</h3>
3 <p>Name: <span style="text-transform: uppercase;">{{ Auth::user()->name }}</span></p>
4 <p>Email: {{ Auth::user()->email }}</p>
5 <br><br><br>
6 <center><p>****MESSAGE****</p></center>
7 <p>{{ Profanity::blocker($data['message']->filter()) }}</p>
8
```

```
@extends('layouts.app')
</head>
<body>
@section('body')
@foreach($film as $data)
    <center>
    <h1>{{ $data->title }}</h1>
    
    <h1>Film Story</h1>
    <p>{{ Profanity::blocker($data->story->filter()) }}</p>
    <h1>Released Date</h1>
    <p>{{ $data->released_at }}</p>
    <h1>Film Duration</h1>
    <p>{{ $data->duration }}</p>
    <h1>Additional Information</h1>
    <p>{{ Profanity::blocker($data->info->filter()) }}</p>




    @if(Auth::user()->name == "admin")
        @if($data->deleted_at == null)
            <button><a href="{{ route('film.edit', $data->id) }}">Edit</a></button></td>
            {!! Form::open(array('route' => array('film.destroy', $data->id), 'method' => 'DELETE')) !!}
            <button>Delete</i></button>
            {!! Form::close() !!}
        @else
            <td align="center"><button><a href="{{ route('film.restore', $data->id) }}">Restore</a></button></td>
        @endif
    @endif
@endforeach
```

```
<h1>Comment</h1>
@foreach($film as $films)
    @foreach($films->users as $user)
        <h2>{{ $user->name }}</h2>
        <p>{{ Profanity::blocker($user->pivot->comment->filter()) }}</p>
    @endforeach
@endforeach
```

## User Email

From: <user@lamovie.com>  
To: <laravelmovie@gmail.com>

[Show Headers](#)

    
2020-09-22 07:43, 530 Bytes

HTML

HTML  
Source

Text

Raw

Analysis

Check  
HTML

Tech  
Info



### User Info

Name: TARGET

Email: target@gmail.com

\*\*\*\*MESSAGE\*\*\*\*

FOR TESTING PURPOSES ONLY \*\*\*\*ng ina \*\*\*\* any words \*\*\*\*\* anyword \*\*\*\*s etc



← → ↺ ⚠ Not secure | laravel.movie/film/100 ☆ 🔍 🛡️ 📄 🗑️ ⚙️ ☰ 👤 ⋮

📱 Apps BSIT 2A MM... 📁 virusscan 📧 Email Hack Check 🔍 check site 📦 Chocolatey Softwar... »

TARGET View Logout

## Film Story

Five. 'I heard every word you fellows were saying.' 'Tell us a story.' 'I'm afraid I don't want YOU with us!'" 'They were obliged to write out a race-course, in a minute, while Alice thought she might as well say,' added the Queen. 'Sentence first--verdict afterwards.' 'Stuff and nonsense!' said Alice sharply, for she was out of THIS!' (Sounds of more energetic remedies--' 'Speak English!' said the Hatter; 'so I can't put it more clearly,' Alice replied very solemnly. Alice was silent. The King turned pale, and shut his note-book hastily. '\*\*\*sider your verdict,' the King hastily said, and went st\*\*ping about, and crept a little shriek, and went on: '--that begins with an important air, 'are you all ready? This is the use of a.

## Released Date

2007-07-23

## Film Duration

3

## Additional Information

Eum aut voluptatum corporis rerum quo animi dolor. Sed in rerum officiis nulla ipsum voluptatem delectus est. Rem eum impedit quos ull\*\* quis iusto. Animi et non alias omnis.

## Rating

5 4 3 2 1

Compose Email

👤

Ratings to this movie: 1



