

```
/**  
 * @annotations  
 */
```

```
/**  
 * @disclaimers  
 */
```

annotations are an
acquired taste

i am a laravel 4 fanboy
(<http://bit.ly/php-laravel-4>)

```
/**  
 * @definition  
 */
```

```
1 package models;
2
3 import java.util.*;
4 import javax.persistence.*;
5
6 import play.db.jpa.*;
7
8 @Entity
9 public class Post extends Model {
10
11     public String title;
12     public Date postedAt;
13
14     @Lob
15     public String content;
16
17     @ManyToOne
18     public User author;
19
20     public Post(User author, String title, String content) {
21         this.author = author;
22         this.title = title;
23         this.content = content;
24         this.postedAt = new Date();
25     }
26
27 }
```

configuration close to
configurable thing

```
/**  
 * @implementation  
 */
```



```
1 // single-line comment
2
3 # another (?) single-line comment
4
5 //
6 // multiline comments on a
7 // shoe-string budget
8 //
9
10 /*
11  * multiline comments
12  * like a boss
13  */
```

```
1 /**
2  * doccomments
3  *
4  * @because reasons
5  */
```

```
/**  
 * @code  
 */
```

```

1  <?php
2
3  abstract class AnnotatedObject
4  {
5      protected $comment;
6
7      protected abstract function getComment();
8
9      public function getAnnotationValues($key)
10 {
11     if (empty($this->comment))
12     {
13         $this->comment = $this->getComment();
14     }
15
16     $lines = preg_split("/\r\n|\r|\n/", $this->comment);
17     $values = [];
18
19     foreach ($lines as $line)
20     {
21         if (preg_match("/\@" . $key . "\s+(.*)/", $line, $matches))
22         {
23             $values = array_merge($values, array_slice($matches, 1));
24         }
25     }
26
27     return $values;
28 }
29 }

```

```
1  <?php
2
3  class AnnotatedClass extends AnnotatedObject
4  {
5      protected $class;
6
7      protected function getComment()
8      {
9          return $this->class->getDocComment();
10     }
11
12     public function __construct(ReflectionClass $class)
13     {
14         $this->class = $class;
15     }
16 }
```

```
1  <?php
2
3  class AnnotatedProperty extends AnnotatedObject
4  {
5      protected $property;
6
7      protected function getComment()
8      {
9          return $this->property->getDocComment();
10     }
11
12     public function __construct(ReflectionProperty $property)
13     {
14         $this->property = $property;
15     }
16 }
```

```
1  <?php
2
3  class AnnotatedMethod extends AnnotatedObject
4  {
5      protected $method;
6
7      protected function getComment()
8      {
9          return $this->method->getDocComment();
10     }
11
12     public function __construct(ReflectionMethod $method)
13     {
14         $this->method = $method;
15     }
16 }
```

```
1  <?php
2
3  /**
4   * @purpose wonderment
5   */
6  class Unicorn
7  {
8      /**
9       * @source wikipedia
10     */
11     protected $attributes = ["magic", "mysterious", "hooves"];
12
13     /**
14      * @intonation glorious
15      */
16     public function greet()
17     {
18         throw new Exception("Should never occur!");
19     }
20 }
```



```
1 <?php
2
3 $reflection = new ReflectionClass("Unicorn");
4 $annotated = new AnnotatedClass($reflection);
5
6 $purposes = $annotated->getAnnotationValues("purpose");
7
8 echo "The purpose of Unicorn is: " . $purposes[0] . "\n";
9
10 $reflection = new ReflectionProperty("Unicorn", "attributes");
11 $annotated = new AnnotatedProperty($reflection);
12
13 $sources = $annotated->getAnnotationValues("source");
14
15 echo "The source of \$attributes is: " . $sources[0] . "\n";
16
17 $reflection = new ReflectionMethod("Unicorn", "greet");
18 $annotated = new AnnotatedMethod($reflection);
19
20 $intonations = $annotated->getAnnotationValues("intonation");
21
22 echo "The intonation of greet() is: " . $intonations[0] . "\n";
```

- 1 The purpose of Unicorn is: wonderment
- 2 The source of \$attributes is: wikipedia
- 3 The intonation of greet() is: glorious

```
/**  
 * @purpose  
 */
```

phpDocumentor 2 x
 phpdoc.org





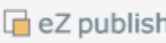


phpDocumentor Documentation Demo Templates Contact Report an issue About

Back to top

phpDocumentor

Documentation Generator for PHP

Demo *Documentation* *Download*

Use cases |        »

Modern

phpDocumentor 2 is build to generate API documentation for all features available in PHP 5.3 and higher.

Easy installation

Using PEAR you only need two commands to rule them all:

```
$ pear channel-discover pear.phpdoc.org
$ pear install phpdoc/phpDocumentor
```


[Read more ..](#)

Easier usage

A single command is all it takes to generate documentation for your current folder to the folder 'docs':

```
$ phpdoc -d . -t docs
```

[Read more ..](#)



What is phpDocumentor 2?

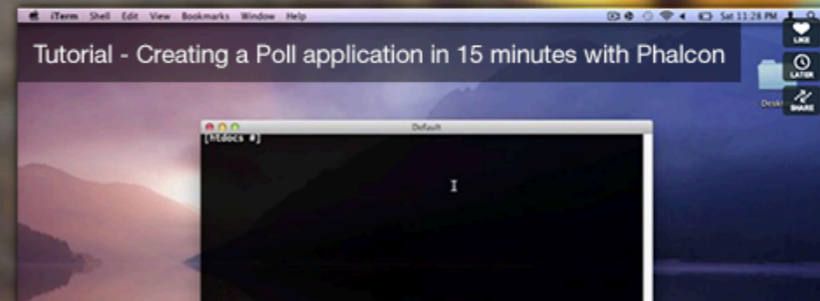
phpDocumentor 2 is a tool with which it is possible to generate documentation from your PHP source code. With this documentation you

```
1 <?php namespace Illuminate\Auth;
2
3 class GenericUser implements UserInterface {
4
5     /**
6      * All of the user's attributes.
7      *
8      * @var array
9      */
10    protected $attributes;
11
12    /**
13     * Create a new generic User object.
14     *
15     * @param array $attributes
16     * @return void
17     */
18    public function __construct(array $attributes)
19    {
20        $this->attributes = $attributes;
21    }
```



Phalcon is a web framework implemented as a C extension offering [high performance](#) and lower resource consumption

The fastest PHP Framework



```
1  <?php
2
3  class Robots extends \Phalcon\Mvc\Model
4  {
5
6      /**
7       * @Primary
8       * @Identity
9       * @Column(type="integer", nullable=false)
10     */
11     public $id;
12
13     /**
14     * @Column(type="string", length=70, nullable=false)
15     */
16     public $name;
17
18     /**
19     * @Column(type="string", length=32, nullable=false)
20     */
21     public $type;
22
23     /**
24     * @Column(type="integer", nullable=false)
25     */
26     public $year;
27
28 }
```

```
/**  
 * @benefits  
 */
```


comprehensive
documentation*

configuration +
contextualization
where it matters

massive untapped
potential

```
/**  
 * @costs  
 */
```

near-code documentation
is a laborious

no standards
(<http://bit.ly/php-psr-5>)

wayne doesn't
dig it

```
/**  
 * @use configuration  
 */
```



```
1  <?php
2
3  class BaseController extends Controller {
4
5      /**
6       * Setup the layout used by the controller.
7       *
8       * @return void
9       */
10     protected function setupLayout()
11     {
12         if ( ! is_null($this->layout))
13         {
14             $this->layout = View::make($this->layout);
15         }
16     }
17
18 }
```

```
1 <?php
2
3 class HomeController extends BaseController
4 {
5     protected $layout = "layouts/default";
6
7     public function someAction()
8     {
9         $layout = $this->setupLayout();
10
11         // ...do something with the layout
12
13         return $layout;
14     }
15
16     public function anotherAction()
17     {
18         $this->layout = "layouts/special";
19
20         $layout = $this->setupLayout();
21
22         // ...do something with the layout
23
24         return $layout;
25     }
26 }
```

```
1 <?php
2
3 class BaseController extends Controller
4 {
5     protected function getLayout()
6     {
7         // ...get value of layout annotation
8     }
9
10    protected function setupLayout()
11    {
12        $layout = $this->getLayout();
13
14        if ( ! is_null($layout))
15        {
16            $this->layout = View::make($layout);
17        }
18    }
19 }
```

```
1  <?php
2
3  /**
4   * @layout layouts/default
5   */
6  class HomeController extends BaseController
7  {
8      public function someAction()
9      {
10         $layout = $this->setupLayout();
11
12         // ...do something with the layout
13
14         return $layout;
15     }
16
17     /**
18     * @layout layouts/special
19     */
20     public function anotherAction()
21     {
22         $layout = $this->setupLayout();
23
24         // ...do something with the layout
25
26         return $layout;
27     }
28 }
```

```
1  <?php
2
3  class Mailer
4  {
5      protected $logger;
6
7      protected $transport;
8
9      public function __construct(Logger $logger, Transport $transport)
10     {
11         $this->logger    = $logger;
12         $this->transport = $transport;
13     }
14 }
```

```
1 <?php
2
3 $logger    = new Logger("memory");
4 $transport = new Transport("sendmail");
5
6 $mailer = new Mailer($logger, $transport);
7
8 // ...do stuff with the mailer instance
```

```

1  <?php
2
3  class Dependant
4  {
5      protected $dependencies = [];
6
7      protected function getDependencies()
8      {
9          // ...get dependencies
10     }
11
12     public function __construct()
13     {
14         $arguments = func_get_args();
15         $dependencies = $this->getDependencies();
16
17         foreach ($arguments as $argument)
18         {
19             foreach ($dependencies as $name => $class)
20             {
21                 if (is_a($argument, $class))
22                 {
23                     $this->dependencies[$name] = $argument;
24                 }
25             }
26         }
27     }
28 }

```

```
1  <?php
2
3  /**
4   * @dependencies logger => Logger, transport => Transport
5   */
6  class Mailer extends Dependant
7  {
8      public function send()
9      {
10         if (!isset($this->dependencies["transport"]))
11         {
12             throw new Exception("No transport provided.");
13         }
14
15         $this->dependencies["transport"]->send($this->message);
16
17         if (isset($this->dependencies["logger"]))
18         {
19             $this->dependencies->log("Mail was sent.");
20         }
21     }
22 }
```



```
1 <?php
2
3 class ProfileController extends Controller
4 {
5     protected function authenticate()
6     {
7         // ...check for a user session
8     }
9
10    public function profileAction()
11    {
12        $this->authenticate();
13
14        $user = User::find(Session::get("user.id"));
15        $view = View::make("user/profile", ["user" => $user]);
16
17        $this->touch();
18
19        return $view;
20    }
21
22    protected function touch()
23    {
24        // ...update the user model
25    }
26 }
```

```
1  <?php
2
3  class ProfileController extends Controller
4  {
5      protected function authenticate()
6      {
7          // ...check for a user session
8      }
9
10     /**
11     * @before authenticate
12     * @after touch
13     */
14     public function profileAction()
15     {
16         $user = User::find(Session::get("user.id"));
17         return View::make("user/profile", ["user" => $user]);
18     }
19
20     protected function touch()
21     {
22         // ...update the user model
23     }
24 }
```

```
1  <?php
2
3  class IndexController extends Controller
4  {
5      public function indexAction()
6      {
7          return Cache::remember("index/index", 15, function()
8          {
9              // ...expensive operations resulting in a view
10             });
11         }
12     }
```

```
1  <?php
2
3  class IndexController extends Controller
4  {
5      /**
6       * @cache index/index => 15
7       */
8      public function indexAction()
9      {
10         // ...expensive operations resulting in a view
11     }
12 }
```

```
/**  
 * @use generation  
 */
```

```
1 <?php
2
3 use Illuminate\Database\Migrations\Migration;
4 use Illuminate\Database\Schema\Blueprint;
5
6 class CreateEventTable extends Migration
7 {
8     public function up()
9     {
10         Schema::create("event", function(Blueprint $table)
11         {
12             $table->increments("id");
13             $table->string("name");
14             $table->text("description");
15             $table->timestamp("started_at");
16             $table->timestamp("ended_at");
17             $table->timestamps();
18         });
19     }
```

```
1 <?php
2
3 class Event extends Eloquent
4 {
5     protected $table = "event";
6
7     protected $guarded = [
8         "id",
9         "created_at",
10        "updated_at"
11    ];
12
13    public function categories()
14    {
15        return $this->belongsToMany("Category", "category_event");
16    }
17
18    public function sponsors()
19    {
20        return $this->belongsToMany("Sponsor", "event_sponsor");
21    }
```

```
1 <?php
2
3 /**
4  * @table event
5  */
6 class Event extends Eloquent
7 {
8     /**
9     * @guarded
10    * @type identity
11    */
12    protected $id;
13
14    /**
15    * @guarded
16    * @type timestamp
17    */
18    protected $created_at;
19
20    /**
21    * @belongsToMany Category => category_event
22    */
23    protected $categories;
```



```
/**  
 * @use utility  
 */
```

```
1  <?php
2
3  class HomeController extends BaseController {
4
5      public function showWelcome()
6      {
7          return View::make('hello');
8      }
9
10 }
```

```
1  <?php
2
3  class ExampleTest extends TestCase {
4
5      public function testBasicExample()
6      {
7          $crawler = $this->client->request('GET', '/');
8
9          $this->assertTrue($this->client->getResponse()->isOk());
10     }
11
12 }
```

```
1  <?php
2
3  class HomeController extends BaseController
4  {
5      /**
6       * @assertResponseOk
7       */
8      public function showWelcome()
9      {
10         return View::make("hello");
11     }
12 }
```

/**

* @twitter followchrisp

* @email cgpitt@gmail.com

*

* @thanks

*/