



Table of Contents

- Installation
- The dump() Function
- DebugBundle and Twig Integration
- Using the VarDumper Component in your PHPUnit Test Suite
- Dump Examples and Output
- Learn More

[Home \(/\)](#) / [Documentation \(/doc/current/index.html\)](#) / [Components \(/doc/current/components/index.html\)](#)

You are browsing the **Symfony 4** documentation, which changes significantly from Symfony 3.x. If your app doesn't use Symfony 4 yet, browse the **Symfony 3.4** documentation (/doc/3.4/components/var_dumper.html).

The VarDumper Component

4.0 version

[edit this page](#)
(https://github.com/symfony/symfony-docs/edit/4.0/components/var_dumper.rst)

The VarDumper component provides mechanisms for extracting the state out of any PHP variables. Built on top, it provides a better `dump()` function that you can use instead of `var_dump` (<http://php.net/manual/en/function.var-dump.php>) .

Installation ¶

You can install the component in 2 different ways:

- Install it via Composer (using [components.html](#)) (`symfony/var-dumper` on Packagist (<https://packagist.org/packages/symfony/var-dumper>));
- Use the official Git repository (<https://github.com/symfony/var-dumper> (<https://github.com/symfony/var-dumper>)).

Note

If using it inside a Symfony application, make sure that the DebugBundle has been installed (or run `composer require debug` to install it).

The dump() Function ¶

The VarDumper component creates a global `dump()` function that you can use instead of e.g. `var_dump` (<http://php.net/manual/en/function.var-dump.php>) . By using it, you'll gain:

- Per object and resource types specialized view to e.g. filter out Doctrine internals while dumping a single proxy entity, or get more insight on opened files with `stream_get_meta_data` (<http://php.net/manual/en/function.stream-get-meta-data.php>) ;
- Configurable output formats: HTML or colored command line output;
- Ability to dump internal references, either soft ones (objects or resources) or hard ones (=> on arrays or objects properties). Repeated occurrences of the same object/array/resource won't appear again and again anymore.

Moreover, you'll be able to inspect the reference structure of your data;

- Ability to operate in the context of an output buffering handler.

For example:

```
1 require __DIR__.'/vendor/autoload.php';
2
3 // create a variable, which could be anything!
4 $someVar = ...;
5
6 dump($someVar);
7
8 // dump() returns the passed value, so you can dump an object and keep using it
9 dump($someObject)->someMethod();
```

By default, the output format and destination are selected based on your current PHP SAPI:

- On the command line (CLI SAPI), the output is written on `STDOUT`. This can be surprising to some because this bypasses PHP's output buffering mechanism;
- On other SAPIs, dumps are written as HTML in the regular output.

Note

If you want to catch the dump output as a string, please read the *advanced documentation* ([var_dumper/advanced.html](#)) which contains examples of it. You'll also learn how to change the format or redirect the output to wherever you want.

Tip

In order to have the `dump()` function always available when running any PHP code, you can install it globally on your computer:

1. Run `composer global require symfony/var-dumper`;
2. Add `auto_prepend_file = ${HOME}/.composer/vendor/autoload.php` to your `php.ini` file;
3. From time to time, run `composer global update symfony/var-dumper` to have the latest bug fixes.

DebugBundle and Twig Integration ¶

The DebugBundle allows greater integration of this component into Symfony applications.

Since generating (even debug) output in the controller or in the model of your application may just break it by e.g. sending HTTP headers or corrupting your view, the bundle configures the `dump()` function so that variables are dumped in the web debug toolbar.

But if the toolbar cannot be displayed because you e.g. called `die` / `exit` or a fatal error occurred, then dumps are written on the regular output.

In a Twig template, two constructs are available for dumping a variable. Choosing between both is mostly a matter of personal taste, still:

- `{% dump foo.bar %}` is the way to go when the original template output shall not be modified: variables are not dumped inline, but in the web debug toolbar;
- on the contrary, `{{ dump(foo.bar) }}` dumps inline and thus may or not be suited to your use case (e.g. you shouldn't use it in an HTML attribute or a `<script>` tag).

This behavior can be changed by configuring the `debug.dump_destination` option. Read more about this and other options in *the DebugBundle configuration reference* ([../reference/configuration/debug.html](#)).

Tip

If the dumped contents are complex, consider using the local search box to look for specific variables or values. First, click anywhere on the dumped contents and then press `Ctrl. + F` or `Cmd. + F` to make the local search box appear. All the common shortcuts to navigate the search results are supported (`Ctrl. + G` or `Cmd. + G`, `F3`, etc.) When finished, press `Esc.` to hide the box again.

Using the VarDumper Component in your PHPUnit Test Suite ¶

The VarDumper component provides a trait

(<http://api.symfony.com/4.0/Symfony/Component/VarDumper/Test/VarDumperTestTrait.html>) that can help writing some of your tests for PHPUnit.

This will provide you with two new assertions:

`assertDumpEquals()`

(http://api.symfony.com/4.0/Symfony/Component/VarDumper/Test/VarDumperTestTrait.html#method_assertDumpEquals)

verifies that the dump of the variable given as the second argument matches the expected dump provided as a string in the first argument.

`assertDumpMatchesFormat()`

(http://api.symfony.com/4.0/Symfony/Component/VarDumper/Test/VarDumperTestTrait.html#method_assertDumpMatchesFormat)

is like the previous method but accepts placeholders in the expected dump, based on the `assertStringMatchesFormat()` method provided by PHPUnit.

Example:

```
1  use PHPUnit\Framework\TestCase;
2
3  class ExampleTest extends TestCase
4  {
5      use \Symfony\Component\VarDumper\Test\VarDumperTestTrait;
6
7      public function testWithDumpEquals()
8      {
9          $testedVar = array(123, 'foo');
10
11          $expectedDump = <<<EOTXT
12 array:2 [
13     0 => 123
14     1 => "foo"
15 ]
16 EOTXT;
17
18         $this->assertDumpEquals($expectedDump, $testedVar);
19     }
20 }
```

Dump Examples and Output ¶

For simple variables, reading the output should be straightforward. Here are some examples showing first a variable defined in PHP, then its dump representation:

```

1 $var = array(
2     'a simple string' => "in an array of 5 elements",
3     'a float' => 1.0,
4     'an integer' => 1,
5     'a boolean' => true,
6     'an empty array' => array(),
7 );
8 dump($var);

```

```

array:5 [▼
    "a simple string" => "in an array of 5 elements"
    "a float" => 1.0
    "an integer" => 1
    "a boolean" => true
    "an empty array" => []
]

```

Note

The gray arrow is a toggle button for hiding/showing children of nested structures.

```

1 $var = "This is a multi-line string.\n";
2 $var .= "Hovering a string shows its length.\n";
3 $var .= "The length of UTF-8 strings is counted in terms of UTF-8 characters.\n";
4 $var .= "Non-UTF-8 strings length are counted in octet size.\n";
5 $var .= "Because of this `xÉ9` octet (\\xE9),\n";
6 $var .= "this string is not UTF-8 valid, thus the `b` prefix.\n";
7 dump($var);

```

```

b"""
This is a multi-line string.
Hovering a string shows its length.
The length of UTF-8 strings is counted in terms of UTF-8 characters.
Non-UTF-8 strings length are counted in octet size.
Because of this `é` octet (\\xE9),
this string is not UTF-8 valid, thus the `b` prefix.
"""

```

```

1 class PropertyExample
2 {
3     public $publicProperty = 'The `+` prefix denotes public properties.';
4     protected $protectedProperty = '`#` protected ones and `-` private ones.';
5     private $privateProperty = 'Hovering a property shows a reminder.';
6 }
7
8 $var = new PropertyExample();
9 dump($var);

```

```

PropertyExample [#14 ▼
    +publicProperty: "The `+` prefix denotes public properties,"
    #protectedProperty: "`#` protected ones and `-` private ones."
    -privateProperty: "Hovering a property shows a reminder."
}

```

Note

#14 is the internal object handle. It allows comparing two consecutive dumps of the same object.

```
1 class DynamicPropertyExample
2 {
3     public $declaredProperty = 'This property is declared in the class definition';
4 }
5
6 $var = new DynamicPropertyExample();
7 $var->undeclaredProperty = 'Runtime added dynamic properties have `` around their name.';
8 dump($var);
```

```
DynamicPropertyExample {#15 ▾
+declaredProperty: "This property is declared in the class definition"
+"undeclaredProperty": "Runtime added dynamic properties have `` around their name."
}
```

```
1 class ReferenceExample
2 {
3     public $info = "Circular and sibling references are displayed as `#number`. \nHovering them highlights a
4 }
5 $var = new ReferenceExample();
6 $var->aCircularReference = $var;
7 dump($var);
```

```
ReferenceExample {#16 ▾
+info: ""
Circular and sibling references are displayed as `#number`.
Hovering them highlights all instances in the same dump.
""
+"aCircularReference": ReferenceExample {#16}
}
```

```
1 $var = new \ErrorException(
2     "For some objects, properties have special values\n"
3     ."that are best represented as constants, like\n"
4     ."`severity` below. Hovering displays the value (`2`).\n",
5     0,
6     E_WARNING
7 );
8 dump($var);
```

```
ErrorException {#14 ▾
#message: ""
For some objects, properties have special values
that are best represented as constants, like
`severity` below. Hovering displays the value (`2`).
""
#code: 0
#file: ".../var-dumper-example.php"
#line: 58
#severity: E_WARNING
-trace: array:1 [▸]
}
```

```

1 $var = array();
2 $var[0] = 1;
3 $var[1] =& $var[0];
4 $var[1] += 1;
5 $var[2] = array("Hard references (circular or sibling)");
6 $var[3] =& $var[2];
7 $var[3][] = "are dumped using `&number` prefixes.";
8 dump($var);

```

```

array:4 [▼
  0 => &1 2
  1 => &1 2
  2 => &2 array:2 [▼
    0 => "Hard references (circular or sibling)"
    1 => "are dumped using `&number` prefixes."
  ]
  3 => &2 array:2 [▶]
]

```

```

1 $var = new \ArrayObject();
2 $var[] = "Some resources and special objects like the current";
3 $var[] = "one are sometimes best represented using virtual";
4 $var[] = "properties that describe their internal state.";
5 dump($var);

```

```

ArrayObject {#17 ▼
  flag::STD_PROP_LIST: false
  flag::ARRAY_AS_PROPS: false
  iteratorClass: "ArrayIterator"
  storage: array:3 [▼
    0 => "Some resources and special objects like the current"
    1 => "one are sometimes best represented using virtual"
    2 => "properties that describe their internal state."
  ]
}

```

```

1 $var = new AcmeController(
2     "When a dump goes over its maximum items limit,\n"
3     ".or when some special objects are encountered,\n"
4     ".children can be replaced by an ellipsis and\n"
5     ".optionally followed by a number that says how\n"
6     ".many have been removed; `9` in this case.\n"
7 );
8 dump($var);

```

```

AcmeController {#14 ▼
  -info: ""
    When a dump goes over its maximum items limit,
    or when some special objects are encountered,
    children can be replaced by an ellipsis and
    optionnally followed by a number that says how
    many have been removed; `9` in this case.
    ""
  #container: Symfony\Component\DependencyInjection\Container {#15 ...9}
}

```

Learn More ¶

- [Advanced Usage of the VarDumper Component \(var_dumper/advanced.html\)](#)

This work, including the code samples, is licensed under a Creative Commons BY-SA 3.0 (<https://creativecommons.org/licenses/by-sa/3.0/>) license.

News from the Symfony blog

A week of symfony #580 (5-11 February 2018) (</blog/a-week-of-symfony-580-5-11-february-2018>)
February 11, 2018

New in Symfony 4.1: Added support for immutable dates in forms (</blog/new-in-symfony-4-1-added-support-for-immutable-dates-in-forms>)
February 7, 2018

A week of symfony #579 (29 January - 4 February 2018) (</blog/a-week-of-symfony-579-29-january-4-february-2018>)
February 4, 2018

[Visit The Symfony Blog →](#)

In the news



(<https://sensiolabs.com/symfony-certification>)

Symfony 3 Certification now available in 4,000 centers around the world!

[Get certified →](#)

Upcoming training sessions

Getting Started with Symfony 3 (<https://training.sensiolabs.com/en/courses?q=SF3C1&from=02/12/2018&to=02/13/2018>)
Berlin - February 12, 2018

Web Development with Symfony 3 (<https://training.sensiolabs.com/en/courses?q=SF3C4&from=02/12/2018&to=02/15/2018>)
Berlin - February 12, 2018

Mastering Symfony 3 (<https://training.sensiolabs.com/en/courses?q=SF3C2&from=02/14/2018&to=02/15/2018>)
Berlin - February 14, 2018

[View all sessions →](#)

Symfony™ is a trademark of Fabien Potencier. All rights reserved (</trademark>).

[What is Symfony? \(/what-is-symfony\)](/what-is-symfony)

[Symfony at a Glance \(/at-a-glance\)](/at-a-glance)

[Symfony Components \(/components\)](/components)

[Projects using Symfony \(/projects\)](/projects)

[Case Studies \(/blog/category/case-studies\)](/blog/category/case-studies)

[Symfony Roadmap \(/roadmap\)](/roadmap)

[Learn Symfony \(/doc/current/index.html\)](/doc/current/index.html)

[Getting Started \(/doc/4.0/setup.html\)](/doc/4.0/setup.html)

[Components \(/doc/4.0/components/index.html\)](/doc/4.0/components/index.html)

[Best Practices \(/doc/4.0/best_practices/index.html\)](/doc/4.0/best_practices/index.html)

[Bundles \(/doc/bundles/\)](/doc/bundles/)

[Reference \(/doc/4.0/reference/index.html\)](/doc/4.0/reference/index.html)

[Security Policy \(/doc/current/contributing/code/security.html\)](#)
[Logo & Screenshots \(/logo\)](#)
[Trademark & Licenses \(/license\)](#)
[symfony1 Legacy \(/legacy\)](#)

Community (/community)

[SensioLabs Connect \(https://connect.sensiolabs.com/\)](#)
[Support \(/support\)](#)
[How to be Involved \(/doc/current/contributing/index.html\)](#)
[Code Stats \(/stats/code\)](#)
[Downloads Stats \(/stats/downloads\)](#)
[Contributors \(/contributors\)](#)

Services (/services)

[Our services \(/services\)](#)
[Train developers \(https://training.sensiolabs.com/en\)](#)
[Start a project \(/services#before\)](#)
[Manage your project quality \(https://insight.sensiolabs.com/\)](#)
[Improve your project performance \(https://blackfire.io/\)](#)
[Struggling with your project \(/services\)](#)
[Support \(https://sensiolabs.com/en/packaged-solutions/index.html\)](#)
[Contact us \(/services\)](#)

[Training \(https://training.sensiolabs.com/en/courses?q=symfony\)](#)
[Certification \(https://sensiolabs.com/certification\)](#)

Blog (/blog/)

[A week of symfony \(/blog/category/a-week-of-symfony\)](#)
[Case studies \(/blog/category/case-studies\)](#)
[Community \(/blog/category/community\)](#)
[Documentation \(/blog/category/documentation\)](#)
[Living on the edge \(/blog/category/living-on-the-edge\)](#)
[Meet the Bundle \(/blog/category/meet-the-bundle\)](#)
[Releases \(/blog/category/releases\)](#)
[Security Advisories \(/blog/category/security-advisories\)](#)
[Symfony CMF \(/blog/category/symfony-cmf\)](#)
[Community Events \(/events/\)](#)

About (/about)

[SensioLabs \(https://sensiolabs.com/en/join_us/join_us.html\)](#)
[Contributors \(/contributors\)](#)
[Careers \(https://sensiolabs.com/en/join_us/join_us.html\)](#)
[Support \(/support\)](#)

Follow Symfony

 (<https://github.com/symfony>)  (<https://stackoverflow.com/questions/tagged/symfony>)  (/slack-
 invite) (<https://twitter.com/symfony>)  (<https://www.facebook.com/SymfonyFramework>) 
(<https://www.youtube.com/user/SensioLabs>)