

Debugging & Profiling

Debug with Print

PHP

```
die(print_r($variable));
```

```
var_dump($variable);
```

```
dump($variable);
```

```
\Symfony\Component\VarDumper\VarDumper::dump($variable);
```

Twig

```
{{ dump(variable) }}
```

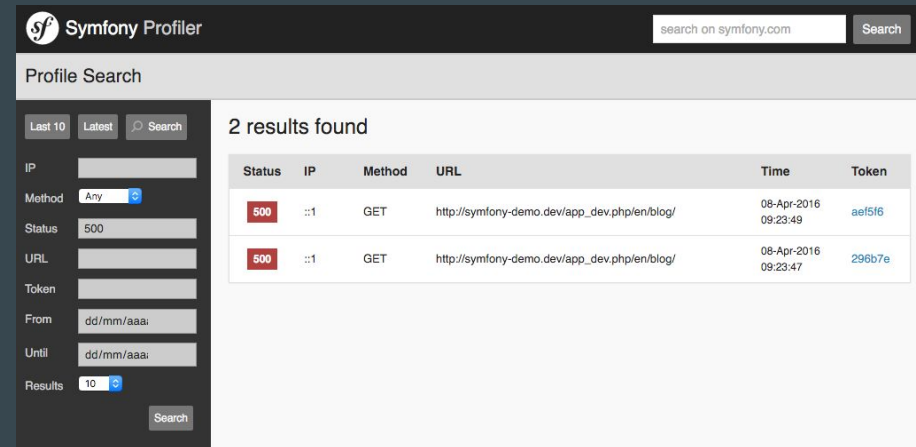
```
{% dump(variable) %}
```

Symfony Debug Component

- `Symfony\Component\Debug\Debug`
- `Symfony\Component\Debug\ErrorHandler`
- `Symfony\Component\Debug\ExceptionHandler`
- `Symfony\Component\Debug\DebugClassLoader`

Debug Toolbar

- Profile request
- Profile ajax requests
- Profile curl call
- Extend
- Overview Oro
- Toolbar tabs



The screenshot shows the Symfony Profiler interface. At the top, there's a search bar with the text "search on symfony.com" and a "Search" button. Below this is the "Profile Search" section. On the left, there are filters: "Last 10", "Latest", and "Search". Below these are input fields for "IP", "Method" (set to "Any"), "Status" (set to "500"), "URL", "Token", "From" (set to "dd/mm/aaa:"), "Until" (set to "dd/mm/aaa:"), and "Results" (set to "10"). A "Search" button is at the bottom of the filters. On the right, it says "2 results found". Below this is a table with the following data:

Status	IP	Method	URL	Time	Token
500	::1	GET	http://symfony-demo.dev/app_dev.php/en/blog/	08-Apr-2016 09:23:49	ae1516
500	::1	GET	http://symfony-demo.dev/app_dev.php/en/blog/	08-Apr-2016 09:23:47	296b7e

Symfony Debug Class Wrappers

Traceable*

- EventDispatcher/Debug
- TraceableUrlMatcher
- TraceableValidator
- etc.

Helps to Debug Toolbar, but complicates debug with xDebug

Xdebug

Helps to understand what code actually doing

- Doesn't require code changes
- Breakpoints
- Full stack trace
- Step into the code
- Live values

Xdebug

- Debugger
- Profiler (not recommended)
- Configurable
- Supports remote debugging
 - <https://xdebug.org/docs/remote>
- Quick way to enable/disable xdebug
 - `phpenmod / phpdismod xdebug + restart`
 - bash script

Xdebug in PhpStorm

- Debug from web browser
 - Chrome extensions, eg. Xdebug helper
- Debug from CLI
 - `export XDEBUG_CONFIG="remote_enable=on idekey=PHPSTORM remote_host=127.0.0.1 remote_port=9000 remote_handler=dbgp" &&`
`export PHP_IDE_CONFIG="serverName=YOUR_SERVER_NAME"`

Setup Xdebug

- Configuring with PhpStorm
- xDebug on Demand
- Evaluate expressions
- Watches
- Conditional breakpoints
- Max simultaneous connections

Blackfire.io

- High overhead, could be used in production
- A lot of ways to use, from CURL to Player
- Profile comparing
- Sharing
- Cross platform support
- Testing by all the metrics
- Periodic builds on production
- Profile every N request on overloaded application