Stripecon
Europe 2024

(X)debug Silverstripe

recycled talk form 2016

Same place, same guys, same talk like 2016

Werner M. Krauß

wmk

Bad Ischl, Austria
netwerkstatt.at
2 kids, at least 5 guitars, pilgrim

netwerkstatt

Lukas Erni

lerni

Ruswil, Switzerland
kraftausdruck.ch
2 kids, Beekeeping

KRAFTAUSDRUCK.CH

What has changed since 2016?

Things getting slower with containered development environment but gaining speed with Xdebug 3.x, PHP 7.x & 8.x and much more predictable & streamlined development environment. With DDEV/containers, setting-up Xdebug has become a breeze.

A bit of history repeating

It’s not a bug, it’s a feature!

Most of the time… not

History of bugs

History of bugs

- the class of insects originated on Earth about 480 million years ago
- so…
- bugs have been there all the time before computers
- and annoyed engineers
- fun fact: bed bugs are the horror of all pilgrims

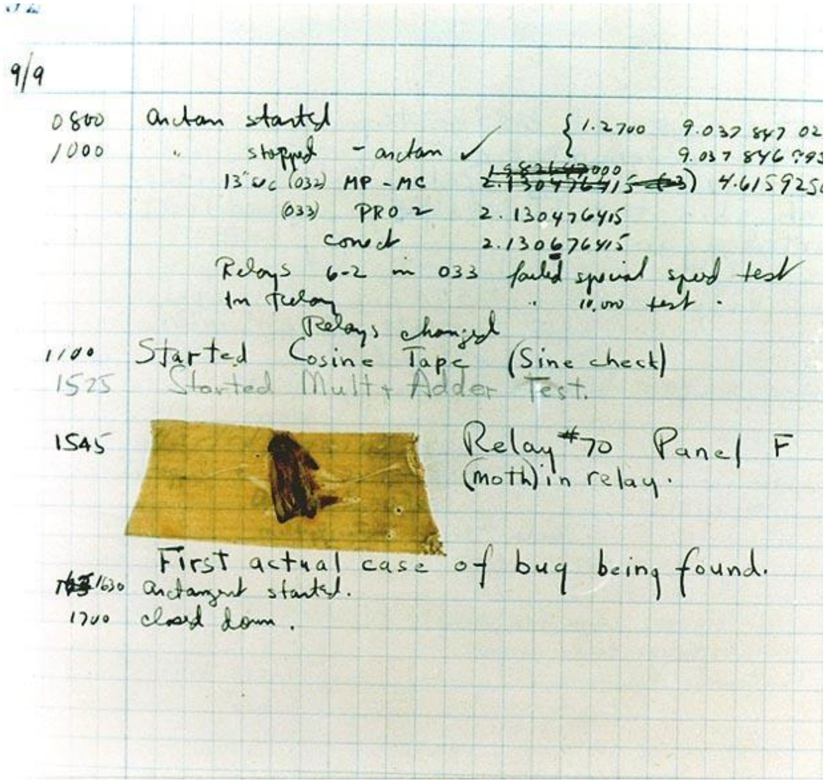
19th Century Hardware Engineering

„The first step [in all of my inventions] is an intuition, and comes with a burst, then difficulties arise – this thing gives out and [it is] then that ‘Bugs’ – as such little faults and difficulties are called – show themselves […].“

Thomas Edison, 1878

The first Bug

- 09.09.1947, Harvard Faculty at the Computation Laboratory
- Operators traced an error in the Mark II to a moth trapped in a relay, coining the term bug.
- This bug was carefully removed and taped to the log book.
- source: Wikipedia



9/9

0800 Arctan started
1000 " stopped - arctan ✓ {1.2700 9.037 847 02
9.037 846 99
13" uc (032) MP - MC ~~1.982147000~~ ~~2.130476415~~ ~~(-3)~~) 4.615925
(033) PRO 2 2.130476415
conect 2.130676415
Relays 6-2 in 033 failed special speed test
In relay " 10,000 test.
Relays changed
1100 Started Cosine Tape (Sine check)
1525 Started Mult + Adder Test.
1545 Relay #70 Panel F
(moth) in relay.
First actual case of bug being found.
~~1545~~ 1630 Arctangent started.
1700 closed down.

History Of Debugging PHP

Debug History in PHP / Silverstripe CMS

- echo($var);
- print_r($array);
- die("I'm here");

Not in Live mode

- debug::show(...);
- debug::message(...);

Also in Live mode

- debug::dump(...);

In Templates

- `$Foo.Debug()`

Debugging in PHP

Pros:

- easy to use
- immediate output

Cons:

- debugging in code tends to end up in git
- not the best tool for the job

Silverstripe Debug Parameters

Silverstripe Debug Parameters

How to get more informations from Silverstripe CMS

- **?isDev=1** Put the site into development mode, enabling debugging messages to the browser on a live server. For security, you'll be asked to log in with an administrator log-in. Will persist for the current browser session.
- **?isTest=1** See above.
- **?debug=1** Show a collection of debugging information about the director / controller operation
- **?debug_request=1** Show all steps of the request from initial HTTPRequest to Controller to Template Rendering

Silverstripe Debug Parameters #2

- **?showqueries=1** List all SQL queries executed
- **?showtemplate=1** Show the compiled version of all the templates used, including line numbers. Good when you have a syntax error in a template. Cannot be used on a Live site without **isDev** when logged in as **Admin**.

https://docs.silverstripe.org/en/5/developer_guides/debugging/

https://docs.silverstripe.org/en/5/developer_guides/debugging/url_variable_tools/

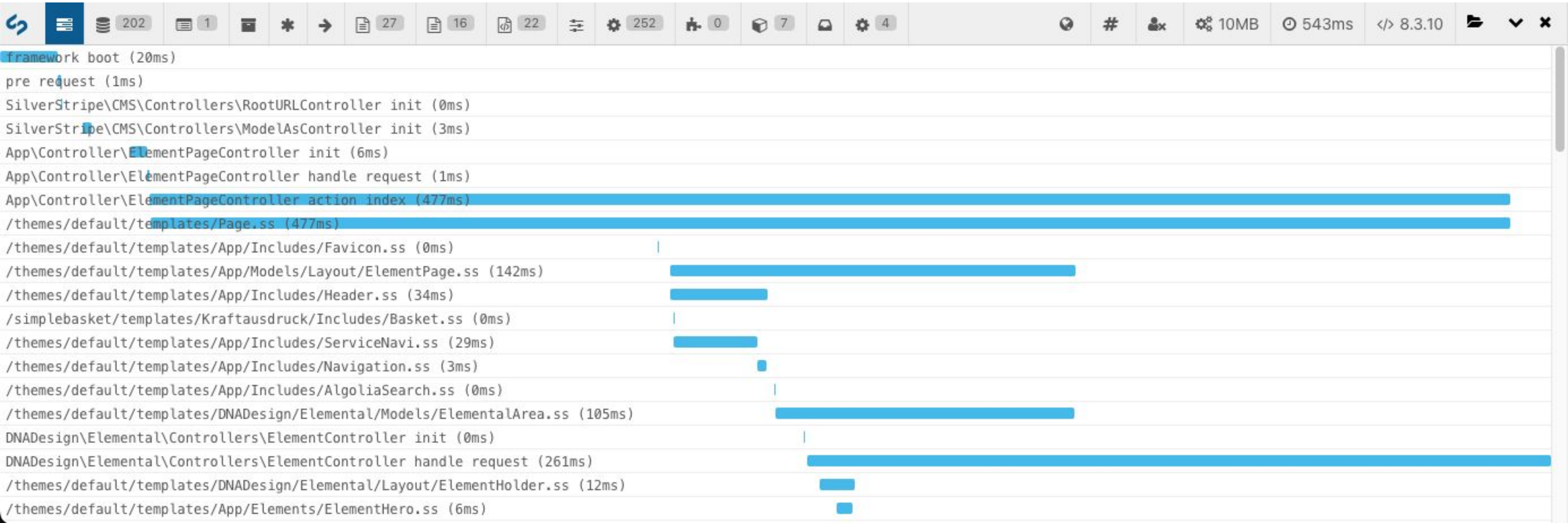
You can disable that (for security reasons)

```
---
Only:
  environment: 'live'
---
SilverStripe\Dev\DevelopmentAdmin:
  deny_non_cli: true
```

🍸 Silverstripe DebugBar 🍹

lekoala/silverstripe-debugbar

https://github.com/lekoala/silverstripe-debugbar makes many things much easier to spot. It gives you a lot of information during development.

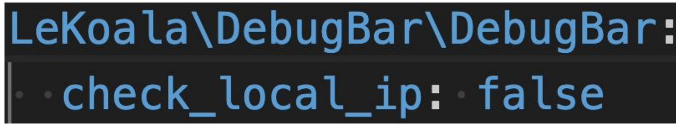


```
framework boot (20ms)
pre request (1ms)
SilverStripe\CMS\Controllers\RootURLController init (0ms)
SilverStripe\CMS\Controllers\ModelAsController init (3ms)
App\Controller\ElementPageController init (6ms)
App\Controller\ElementPageController handle request (1ms)
App\Controller\ElementPageController action index (477ms)
/themes/default/templates/Page.ss (477ms)
/themes/default/templates/App/Includes/Favicon.ss (0ms)
/themes/default/templates/App/Models/Layout/ElementPage.ss (142ms)
/themes/default/templates/App/Includes/Header.ss (34ms)
/simplebasket/templates/Kraftausdruck/Includes/Basket.ss (0ms)
/themes/default/templates/App/Includes/ServiceNavi.ss (29ms)
/themes/default/templates/App/Includes/Navigation.ss (3ms)
/themes/default/templates/App/Includes/AlgoliaSearch.ss (0ms)
/themes/default/templates/DNADesign/Elemental/Models/ElementalArea.ss (105ms)
DNADesign\Elemental\Controllers\ElementController init (0ms)
DNADesign\Elemental\Controllers\ElementController handle request (261ms)
/themes/default/templates/DNADesign/Elemental/Layout/ElementHolder.ss (12ms)
/themes/default/templates/App/Elements/ElementHero.ss (6ms)
```

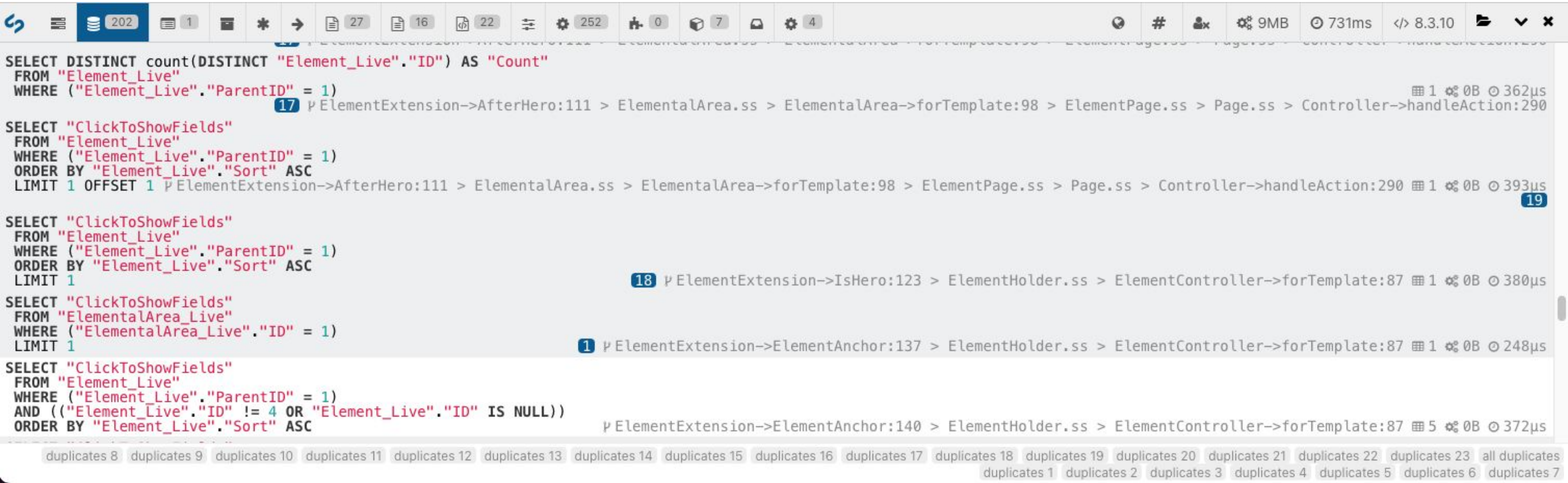
Install as dev dependency with composer:

composer require --dev lekoala/silverstripe-debugbar

Common gotcha with DDEV



```
LeKoala\DebugBar\DebugBar:
  check_local_ip: false
```



```
SELECT DISTINCT count(DISTINCT "Element_Live"."ID") AS "Count"
 FROM "Element_Live"
 WHERE ("Element_Live"."ParentID" = 1)                                                                                    1  0B  362µs
                                17 ElementExtension->AfterHero:111 > ElementalArea.ss > ElementalArea->forTemplate:98 > ElementPage.ss > Page.ss > Controller->handleAction:290

SELECT "ClickToShowFields"
 FROM "Element_Live"
 WHERE ("Element_Live"."ParentID" = 1)
 ORDER BY "Element_Live"."Sort" ASC
 LIMIT 1 OFFSET 1 ElementExtension->AfterHero:111 > ElementalArea.ss > ElementalArea->forTemplate:98 > ElementPage.ss > Page.ss > Controller->handleAction:290  1  0B  393µs
                                                                                                                                                                19

SELECT "ClickToShowFields"
 FROM "Element_Live"
 WHERE ("Element_Live"."ParentID" = 1)
 ORDER BY "Element_Live"."Sort" ASC
 LIMIT 1                                    18 ElementExtension->IsHero:123 > ElementHolder.ss > ElementController->forTemplate:87  1  0B  380µs

SELECT "ClickToShowFields"
 FROM "ElementalArea_Live"
 WHERE ("ElementalArea_Live"."ID" = 1)
 LIMIT 1                                 1 ElementExtension->ElementAnchor:137 > ElementHolder.ss > ElementController->forTemplate:87  1  0B  248µs

SELECT "ClickToShowFields"
 FROM "Element_Live"
 WHERE ("Element_Live"."ParentID" = 1)
 AND (("Element_Live"."ID" != 4 OR "Element_Live"."ID" IS NULL))
 ORDER BY "Element_Live"."Sort" ASC         ElementExtension->ElementAnchor:140 > ElementHolder.ss > ElementController->forTemplate:87  5  0B  372µs
```

Debugbar shows you:

Timeline execution time overview

Database Queries, Long running queries

System logs and messages Shows anything processed by a logger -> no need to check log

Session

Cookies

Parameter

Requirements

Middleware

Template

SiteConfig

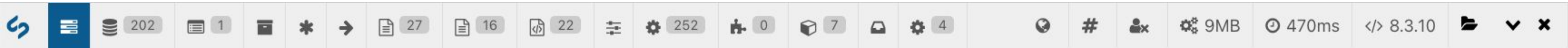
Config System

Cache

Mails

Headers

CMS & PHP Version, Time & Memory Usage



Local Development using DDEV

DDEV

- for local development
- based on docker
- has everything you need and a lot extensions for special requirements
- apache-fpm/nginx
- mariadb
- all major PHP-Versions
- project-type=silverstripe (thanks to firesphere)
- plugins for PHPStorm and VSCode
- See https://ddev.com/

XDebug
step debugging made easy

Why should I?

- find errors / bugs more easier
- know the tools for craftsmanship 🛠️
- no debugging information gets committed to git
- easy setup with ddev and PHPStorm / VScode
- actually it works out of the box 🦾
- ddev xdebug on/off
  - switch it off for a faster dev experience when not debugging
- you’ll become more sexy ❤️

What happened until now

- Xdebug is a PHP extension written by Derick Rethans - he works on it since 2002!
- It uses the DBGp debugging protocol
- It is a powerful tool for debugging and profiling PHP code.

Xdebug still worth a talk? Just click ●

DDEV makes setup much easier! Tweak a few things in your boilerplate, to make it always available. https://github.com/lerni/ootstra/tree/master/.vscode

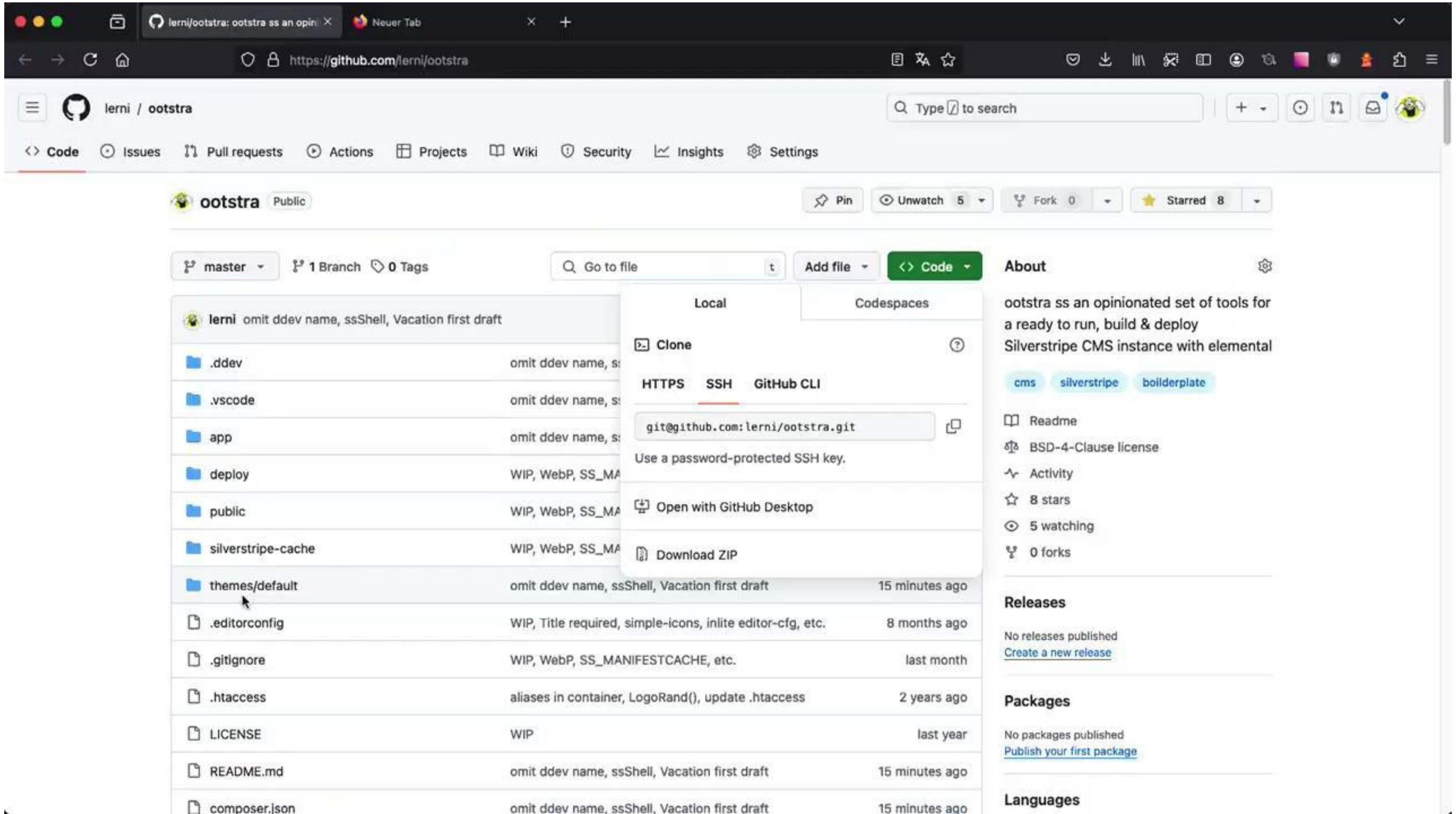
VSCode Extensions → .vscode/extensions.json

- DDEV Manager - mainly automatic ‘ddev xdebug true/false’
- PHP Debug Adapter

.vscode/tasks.json & .vscode/launch.json

- "hostname": "0.0.0.0" for CLI debugging
- "pathMappings": {“/var/www/html": "${workspaceFolder}"}

Silverstripe DDEV, Xdebug etc. setup in under 2 minutes



https://github.com/lerni/ootstra

lerni / ootstra

Code · Issues · Pull requests · Actions · Projects · Wiki · Security · Insights · Settings

ootstra Public

Pin · Unwatch 5 · Fork 0 · Starred 8

master · 1 Branch · 0 Tags · Go to file · Add file · Code

lerni omit ddev name, ssShell, Vacation first draft

.ddev	omit ddev name, s	
.vscode	omit ddev name, s	
app	omit ddev name, s	
deploy	WIP, WebP, SS_MA	
public	WIP, WebP, SS_MA	
silverstripe-cache	WIP, WebP, SS_MA	
themes/default	omit ddev name, ssShell, Vacation first draft	15 minutes ago
.editorconfig	WIP, Title required, simple-icons, inlite editor-cfg, etc.	8 months ago
.gitignore	WIP, WebP, SS_MANIFESTCACHE, etc.	last month
.htaccess	aliases in container, LogoRand(), update .htaccess	2 years ago
LICENSE	WIP	last year
README.md	omit ddev name, ssShell, Vacation first draft	15 minutes ago
composer.json	omit ddev name, ssShell, Vacation first draft	15 minutes ago

Local · Codespaces

Clone

HTTPS · SSH · GitHub CLI

```
git@github.com:lerni/ootstra.git
```

Use a password-protected SSH key.

Open with GitHub Desktop

Download ZIP

About

ootstra ss an opinionated set of tools for a ready to run, build & deploy Silverstripe CMS instance with elemental

cms · silverstripe · boilderplate

Readme

BSD-4-Clause license

Activity

8 stars

5 watching

0 forks

Releases

No releases published

Create a new release

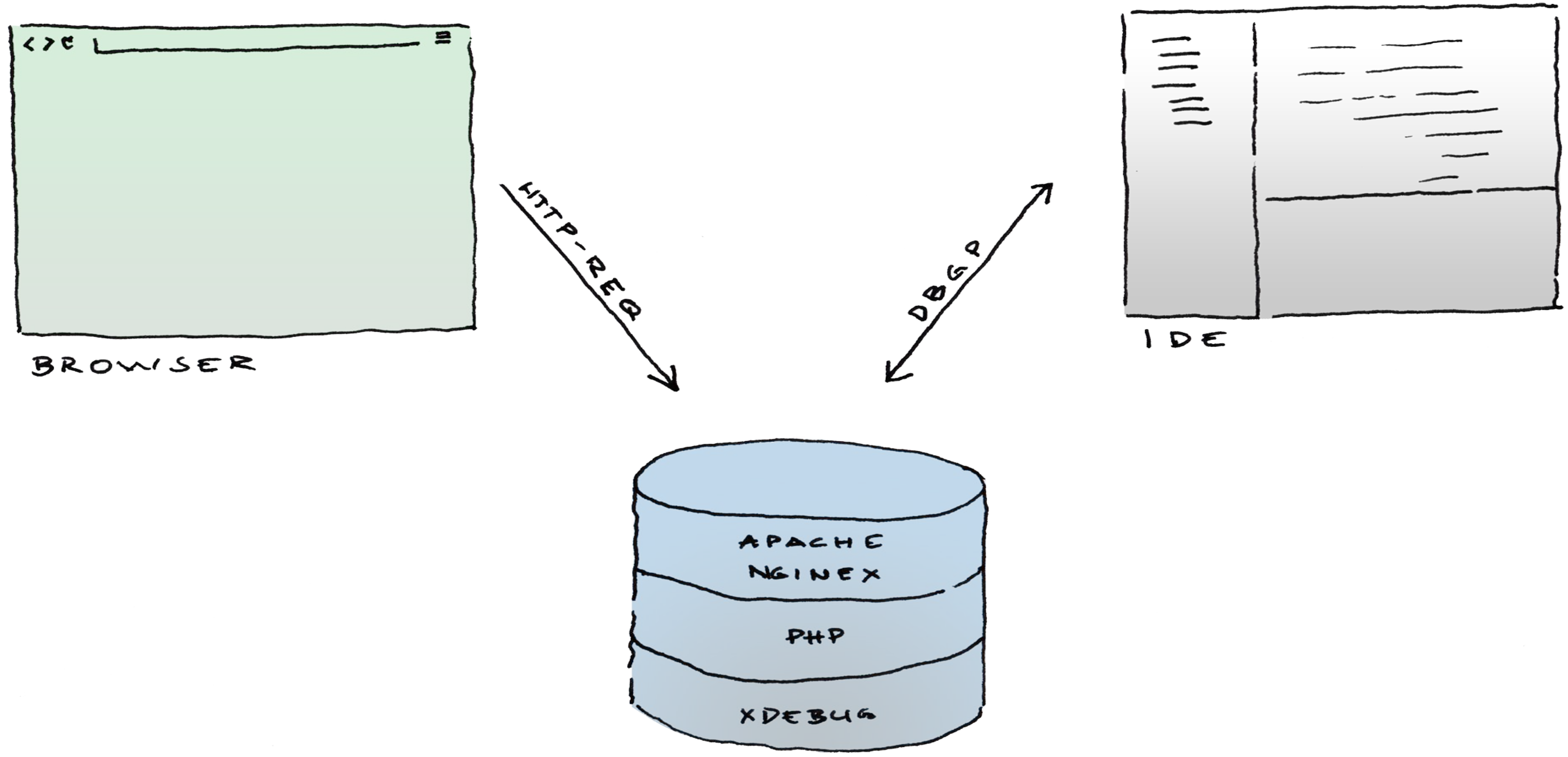
Packages

No packages published

Publish your first package

Languages

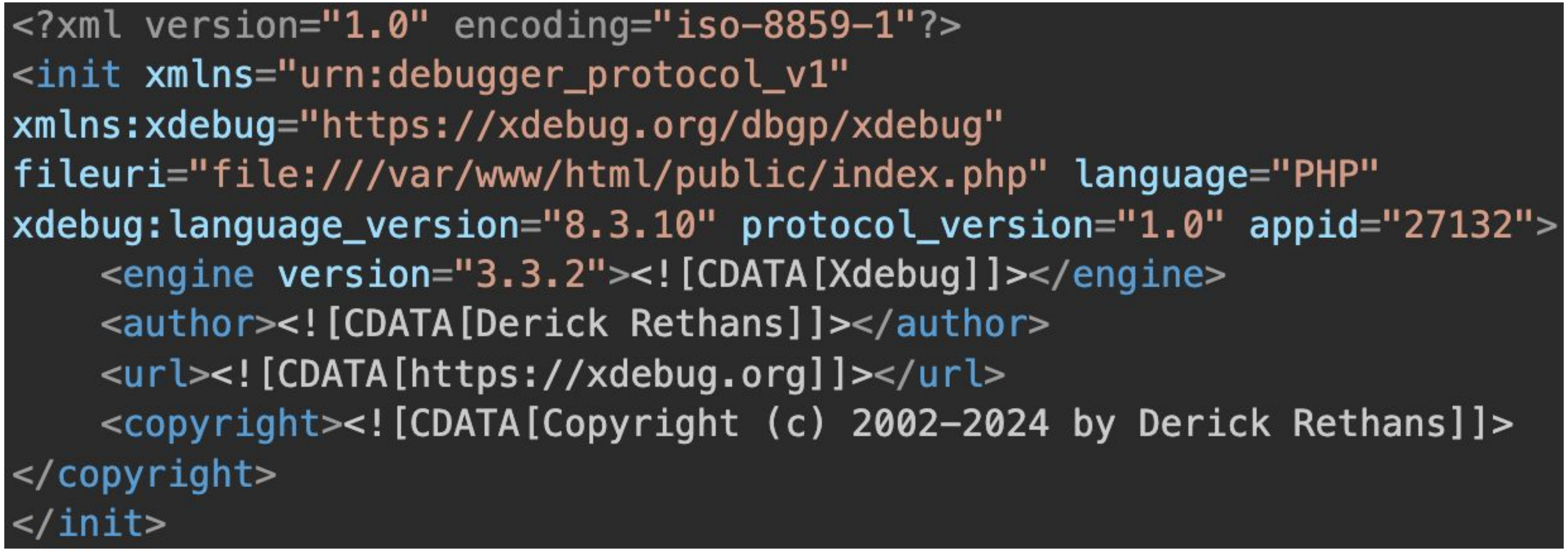
DBGp debugging protocol



Fake Client IDE/Editor and listen to Xdebug

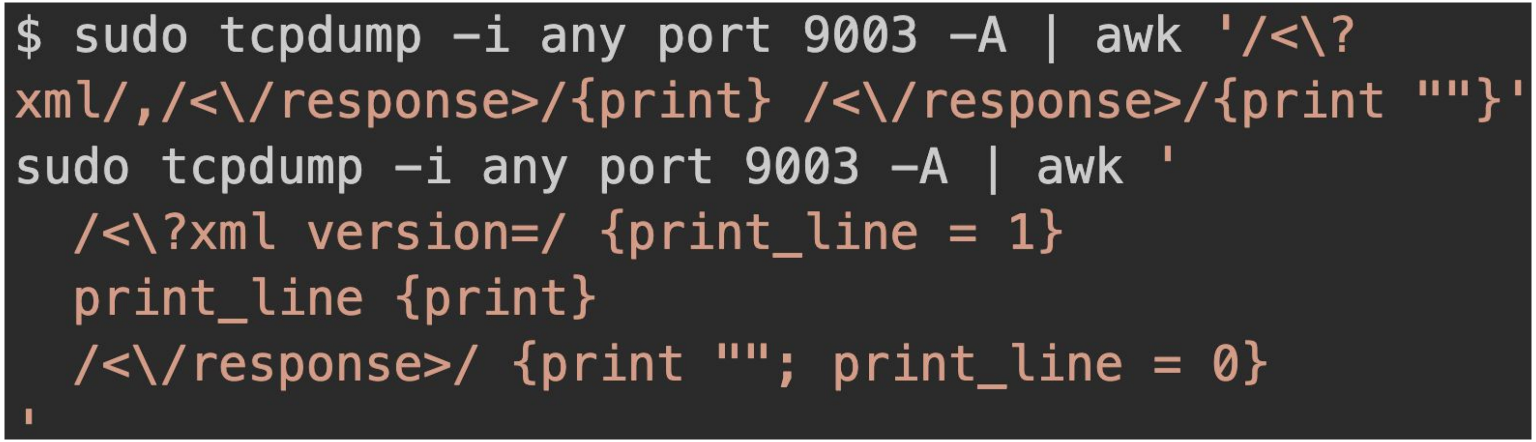
ddev xdebug on

$ nc -l 0.0.0.0 9003 - and fetch an url with the browser or curl



```
<?xml version="1.0" encoding="iso-8859-1"?>
<init xmlns="urn:debugger_protocol_v1"
xmlns:xdebug="https://xdebug.org/dbgp/xdebug"
fileuri="file:///var/www/html/public/index.php" language="PHP"
xdebug:language_version="8.3.10" protocol_version="1.0" appid="27132">
    <engine version="3.3.2"><![CDATA[Xdebug]]></engine>
    <author><![CDATA[Derick Rethans]]></author>
    <url><![CDATA[https://xdebug.org]]></url>
    <copyright><![CDATA[Copyright (c) 2002-2024 by Derick Rethans]]>
</copyright>
</init>
```

You can 👂 closer, if you really want to…



```
$ sudo tcpdump -i any port 9003 -A | awk '/<\?
xml/,/<\/response>/{print} /<\/response>/{print ""}'
sudo tcpdump -i any port 9003 -A | awk '
  /<\?xml version=/ {print_line = 1}
  print_line {print}
  /<\/response>/ {print ""; print_line = 0}
'
```

Debugger Functions

Why is XDebug better than var_dump() and die()?

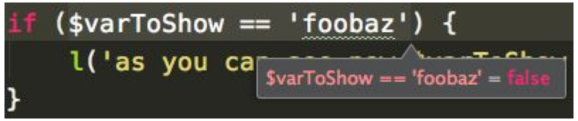
- Breakpoint
- Conditional breakpoint
- List of all available variables in current scope
- Watch
- Frames (stack of called functions)

Methods

- Step Over => goto next line
- Step Into => go inside a called function or method
- Force Step Into
- Step Out => leave the current method
- Run to Cursor

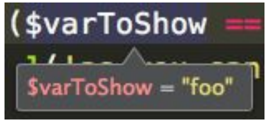
More Methods

- Resume Program => goto next breakpoint
- Evaluate Expression
- Quick Evaluate Expression => without dialog



```
if ($varToShow == 'foobaz') {
    l('as you ca
}
```

```
$varToShow == 'foobaz' = false
```

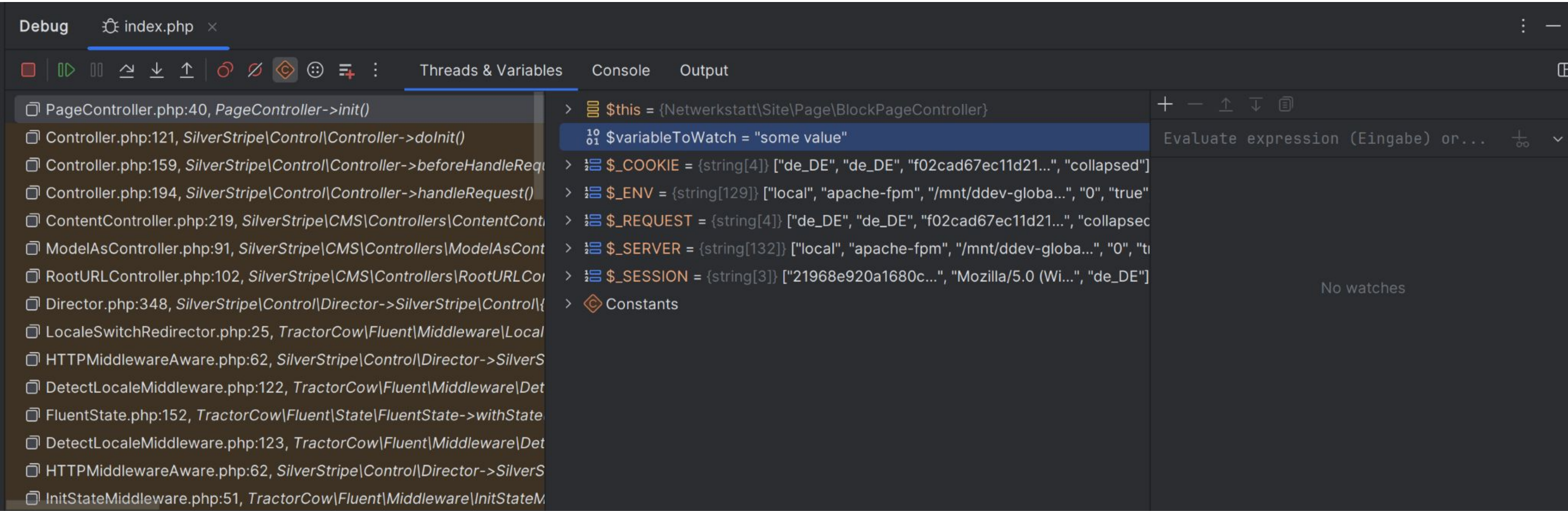


```
($varToShow ==
```

```
$varToShow = "foo"
```

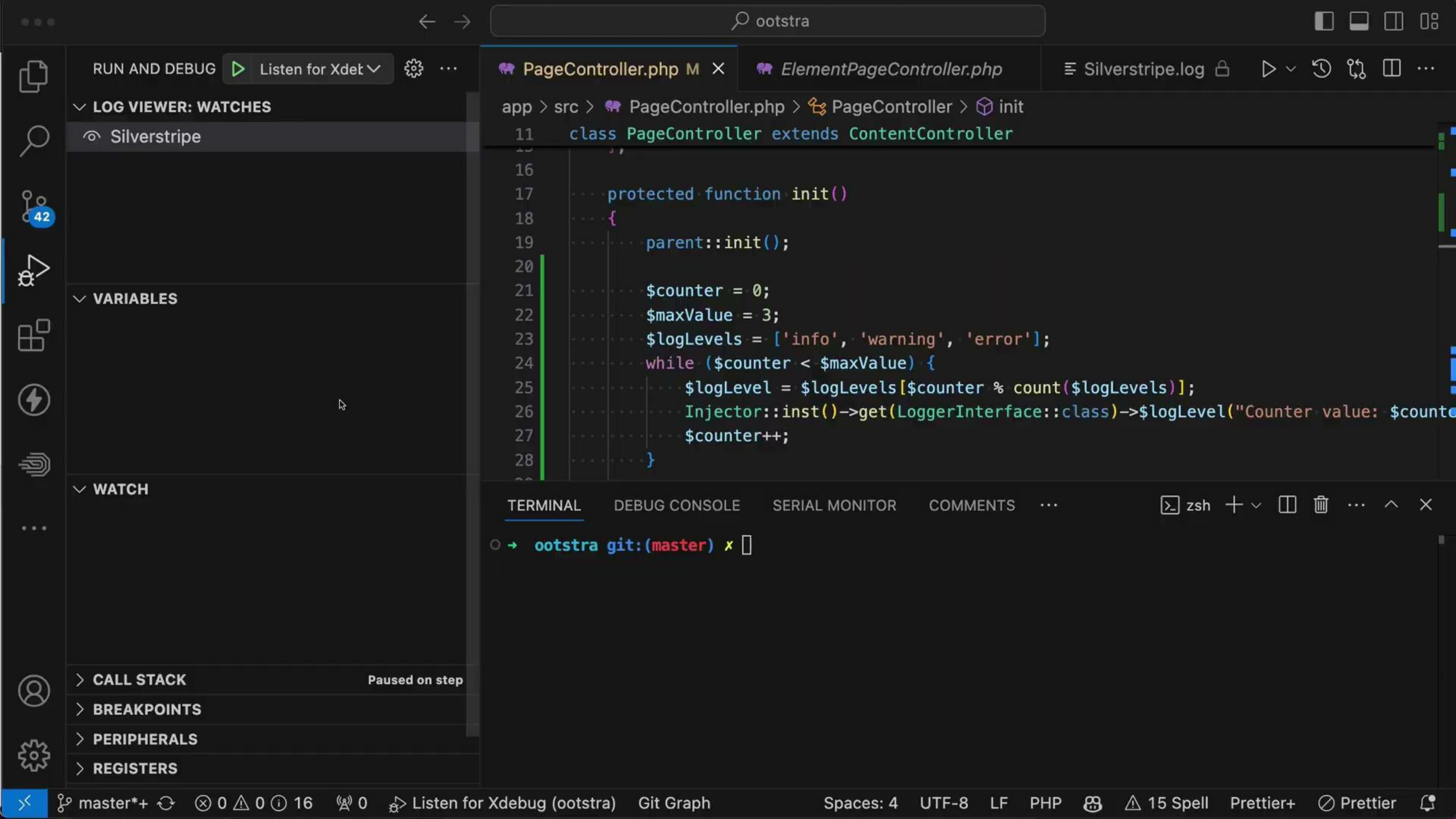
- Toggle Breakpoint
- View Breakpoints

GUI Overview PHPStorm



```
PageController.php:40, PageController->init()
Controller.php:121, SilverStripe\Control\Controller->doInit()
Controller.php:159, SilverStripe\Control\Controller->beforeHandleRequ
Controller.php:194, SilverStripe\Control\Controller->handleRequest()
ContentController.php:219, SilverStripe\CMS\Controllers\ContentCont
ModelAsController.php:91, SilverStripe\CMS\Controllers\ModelAsCont
RootURLController.php:102, SilverStripe\CMS\Controllers\RootURLCo
Director.php:348, SilverStripe\Control\Director->SilverStripe\Control\{
LocaleSwitchRedirector.php:25, TractorCow\Fluent\Middleware\Local
HTTPMiddlewareAware.php:62, SilverStripe\Control\Director->SilverS
DetectLocaleMiddleware.php:122, TractorCow\Fluent\Middleware\Det
FluentState.php:152, TractorCow\Fluent\State\FluentState->withState
DetectLocaleMiddleware.php:123, TractorCow\Fluent\Middleware\Det
HTTPMiddlewareAware.php:62, SilverStripe\Control\Director->SilverS
InitStateMiddleware.php:51, TractorCow\Fluent\Middleware\InitStateM
```

```
> $this = {Netwerkstatt\Site\Page\BlockPageController}
  $variableToWatch = "some value"
> $_COOKIE = {string[4]} ["de_DE", "de_DE", "f02cad67ec11d21...", "collapsed"]
> $_ENV = {string[129]} ["local", "apache-fpm", "/mnt/ddev-globa...", "0", "true"
> $_REQUEST = {string[4]} ["de_DE", "de_DE", "f02cad67ec11d21...", "collapsed
> $_SERVER = {string[132]} ["local", "apache-fpm", "/mnt/ddev-globa...", "0", "t
> $_SESSION = {string[3]} ["21968e920a1680c...", "Mozilla/5.0 (Wi...", "de_DE"]
> Constants
```



```php
class PageController extends ContentController

    protected function init()
    {
        parent::init();

        $counter = 0;
        $maxValue = 3;
        $logLevels = ['info', 'warning', 'error'];
        while ($counter < $maxValue) {
            $logLevel = $logLevels[$counter % count($logLevels)];
            Injector::inst()->get(LoggerInterface::class)->$logLevel("Counter value: $counte
            $counter++;
        }
```

```
○ → ootstra git:(master) ✗ 
```

More debugging knowledge

When you can call it, you can debug it

Example: Debugging unit tests

- good for more complicated tasks that are not easy to reach on the site
  - e.g. shop checkout functionality
- fixtures maybe a bit hard to setup
- when a test works you’re done
- best done via CLI

PHPStan - Static Analyzer

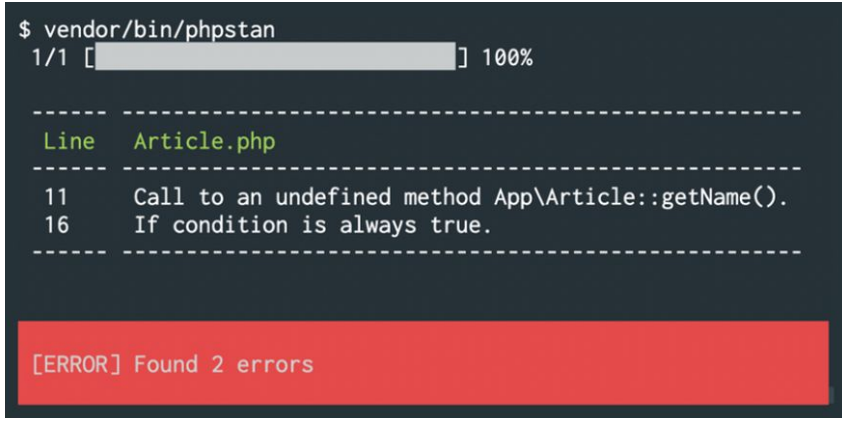
PHPStan

Pros:

- Totally annoying
- can check your code for bugs before they reach production
- works better with well typed classes
- forces you to think about types

Cons:

- Did I say it’s totally annoying… at least in the beginning



```
$ vendor/bin/phpstan
 1/1 [████████████████████████████] 100%

 ------ --------------------------------------------------
  Line   Article.php
 ------ --------------------------------------------------
  11     Call to an undefined method App\Article::getName().
  16     If condition is always true.
 ------ --------------------------------------------------


 [ERROR] Found 2 errors
```

PHPStan: installation

- Of course using composer as a dev requirement
- There’s a package to make PHPStan understand Silverstripe, e.g. DataObject::get() and its magic properties.

```
composer require --dev syntro/silverstripe-phpstan ^5
```

SS Shell

SSShell (Silverstripe’s shell, not SSS hell!)

- SSShell is a REPL for SilverStripe running on Psy Shell 🚀
- PsySH is a runtime developer console, interactive debugger and REPL for PHP.
- REPL = Read-eval-print loop

You can

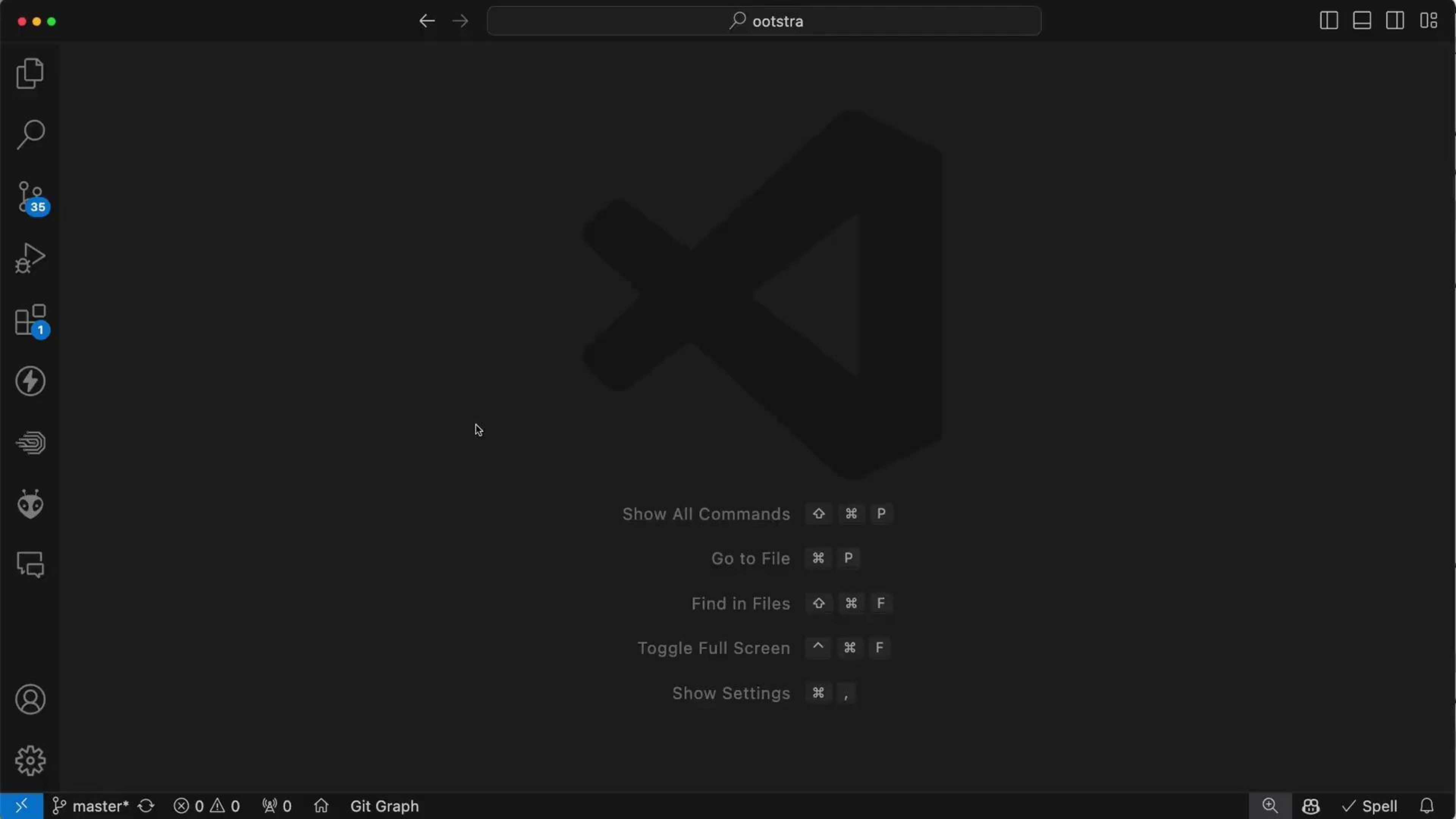
- view classes/objects and static properties
- run methods on objects
- run sake commands and flush

Why should I use SSShell

- good for tinkering around in Silverstripe
- The interactive debugger saves lives! Stop *die()*ing all the time.
- an alternative for executing simple one time tasks

More informations:

- https://github.com/pstaender/ssshell
- https://psysh.org/


ootstra
35
1
Show All Commands ⇧ ⌘ P
Go to File ⌘ P
Find in Files ⇧ ⌘ F
Toggle Full Screen ^ ⌘ F
Show Settings ⌘ ,
master*
0 0
0
Git Graph
Spell

Conclusion

Bugs’ natural habitat is code

Your code

There are tools to get rid of bugs

Use them

See you again in 2032!

👋