

Systemutveckling PHP

Föreläsning 14 - Felsökning

Mikael Olsson
mikael.olsson@emmio.se
076-174 90 43



Dagens ämnen

- Felsökning
- Enhetstester

Felsökning

- Att hitta fel i en sida kan ofta vara komplicerat, speciellt i större projekt eller ramverk.
- Många filer inkluderas, både interna och externa (t ex tredjepartsbibliotek).

Felsökning

- Vad är en bugg?
 - Syntax-fel
 - Logiska fel
 - Exekveringsfel
 - Feature

Felsökning

- Syntax-fel
 - När man använt språket på fel sätt, när browser inte förstår våra instruktioner.
- Skrivfel
- Utelämnade tecken
- Felstavade identifierare (variabler, funktioner, konstanter klasser)
- Odefinierade variabler

Felsökning

- Logiska fel
 - Vi använder språket korrekt men får inte det resultat vi har tänkt oss, t ex svar som blir fel bara för en viss sorts indata (specialfall).

Felsökning

- Exekveringsfel
 - Dividera med noll
 - Array utanför index
 - Oändliga loopar

Felsökning

- Feature
 - Ibland är en bugg inte en bugg utan ett missförstånd i vad en funktion verkligen ska göra.

Kodstandard

- <https://blog.sideci.com/5-php-coding-standards-you-will-love-and-how-to-use-them-adf6a4855696>
- PSR-2 Coding Style Guide
- CakePHP Coding Standards
- Symfony Coding Standards
- WordPress Coding Standards
- FuelPHP Coding Standards

PSR-1

Basic Coding Standard

- Exempel:
 - Only `<?php` or `<?='` are allowed for PHP tags
 - Files must be in UTF-8 without BOM(Byte Order Mark)
 - Namespaces and class names must follow the standards in PSR-0 and PSR-4
 - Class names must be defined in `UpperCamelCase`
 - Class variables must be defined in `UPPER_SNAKE_CASE`
 - Method names must be defined in `camelCase`

PSR-2

Coding Style Guide

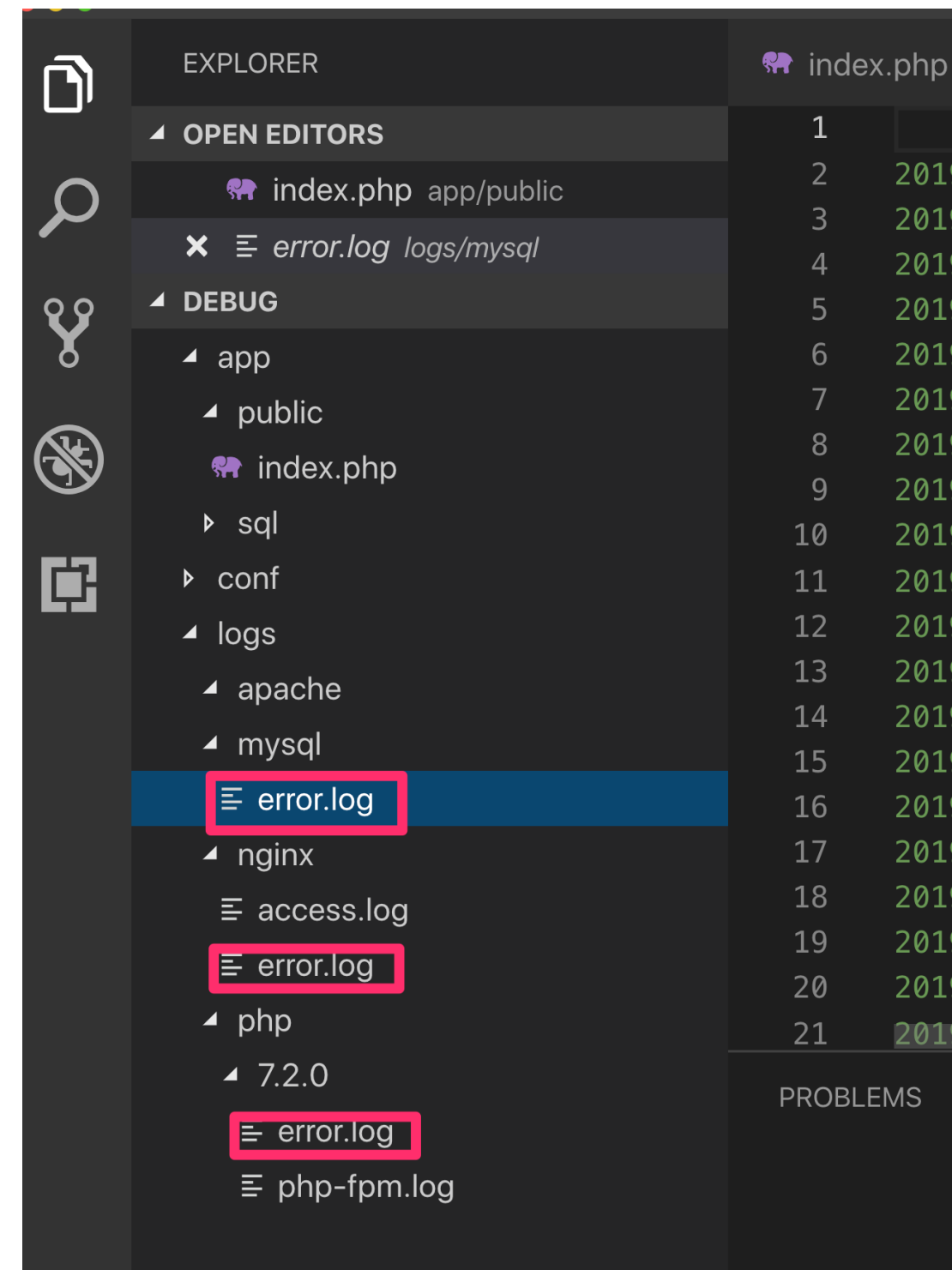
- Exempel:
 - You must follow PSR-1 coding standards
 - 4 spaces must be used for indents. Using tabs is not allowed
 - There is no limit to line length, but it should be under 120 characters, and best if under 80
 - You must put a newline before curly braces for classes and methods
 - You must not put a newline before curly braces in conditional statements
 - You must not put any spaces before (and) in conditional statements

Felsökning

- Hur hittar man en bugg?
 - Loggar
 - Isolering & uteslutning
 - Spårutskrifter
 - Remote debugging
- <https://divante.co/blog/find-bug-5-steps-remove-toughest-bugs-php/>

Loggar

- Apache, php och mysql har ofta logg-filer som man kan få ledtrådar ifrån.



Isolering & uteslutning

- Att isolera och utesluta buggar innebär att man testar så mycket man kan i både gränssnitt och kod för att ta bort faktorer som kan påverka problemet.
- Det kan t ex vara att testa vilka data som krävs för att återskapa buggen. Är det bara om funktionen får en viss typ av data? (Sträng, int, negativt tal osv.) Uppstår buggen bara under vissa omständigheter? Kan man kommentera bort kodrader och se om problemet fortfarande kvarstår?

Spårutskrifter

- Man kan börja med att slå på `display_errors` i `php.ini` eller i början av php-filen:
- ```
ini_set('display_errors', 'On');
error_reporting(E_ALL);
```

# Spårutskrifter

- Man kan skriva ut variabler, uppbyggda strängar osv för att undersöka vad de innehåller.
- `print_r($var);`
- `var_dump($var);`
- `get_defined_vars();`
- `debug_print_backtrace();`
- `debug_backtrace();`



# Spårutskrifter

- Man kan skriva en funktion som skriver ut `<pre>`-taggar för tydligare utskrifter.
- ```
function pr($var) {  
    echo '<pre>';  
    var_dump($var);  
    echo '</pre>';  
}
```

Spårutskrifter

- ```
function console_log($data){
 echo '<script>';
 echo 'console.log('. json_encode($data) .')';
 echo '</script>';
}
```
- Usage:  

```
$myvar = array(1,2,3);
console_log($myvar); // [1,2,3]
```

# Remote debugging

- xdebug har stöd för:
  - Set/Remove breakpoints
  - Perform an automatic stack trace
  - Set a manual variable watch
  - Enable function call logging
  - Enable profiling
  - Allow remote debugging

# Remote debugging

- Det finns inbyggt stöd för xdebug i många LAMP-stackar.
  - local by flywheel
  - XAMPP - win - mac
  - MAMP
- Ni kan behöva starta om apache.

# Testa om xdebug funkar

- Skapa en sida som innehåller ett fel och gå in på den.
- Om felmeddelandet visas med en sån här orange ruta så fungerar xdebug.

```
<?php
```

```
function add($a, $b) {
 return $a + $b;
}
```

```
$c = add (5, 6);
echo $c;
echo $d;
echo 1/0;
```

Denna rad räcker.

| (!) Notice: Undefined variable: d in /app/public/index.php on line 9 |        |        |          |                 |
|----------------------------------------------------------------------|--------|--------|----------|-----------------|
| Call Stack                                                           |        |        |          |                 |
| #                                                                    | Time   | Memory | Function | Location        |
| 1                                                                    | 0.0383 | 398640 | {main}() | .../index.php:0 |

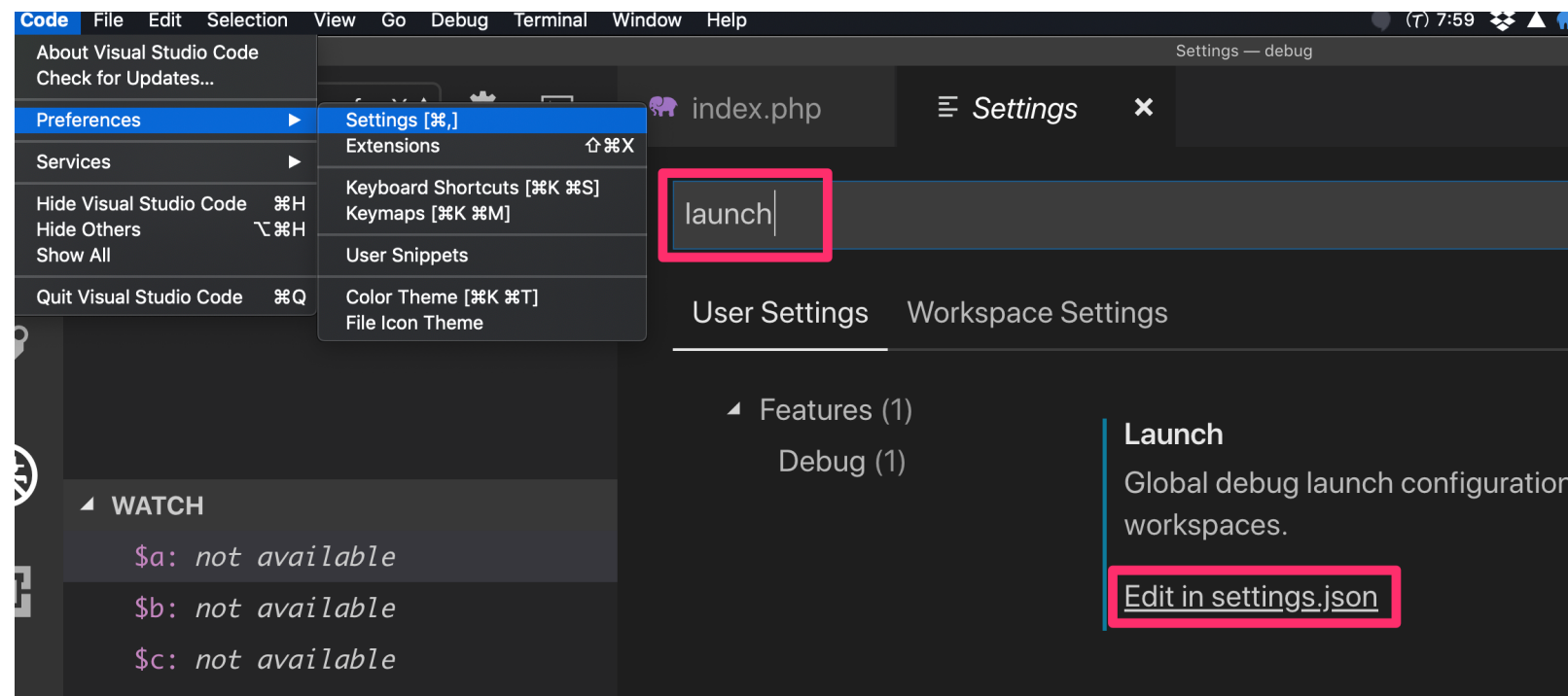
# VSC

- <https://github.com/felixfbecker/vscode-php-debug>
- `F1 + ext install php-debug`
- Redigera `php.ini`:  
`xdebug.remote_enable = 1`  
`xdebug.remote_autostart = 1`

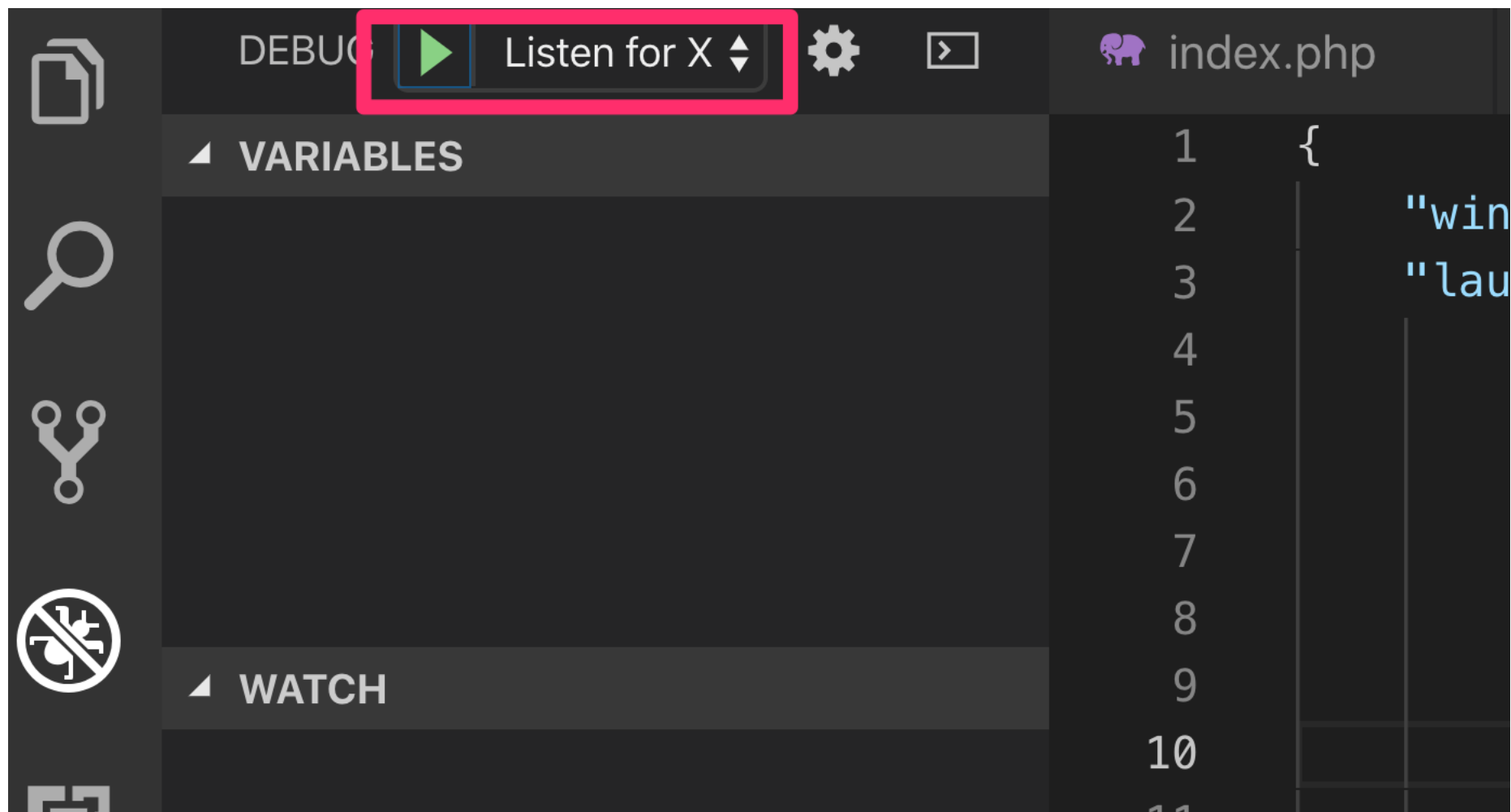
# VSC

```
"launch": {
 "version": "0.2.0",
 "configurations": [
 {
 "name": "Listen for XDebug",
 "type": "php",
 "request": "launch",
 "port": 9000,
 "pathMappings": {
 "/app": "${workspaceRoot}/app",
 },
 },
 {
 "name": "Launch currently open script",
 "type": "php",
 "request": "launch",
 "program": "${file}",
 "cwd": "${fileDirname}",
 "port": 9000
 }
]
}
```

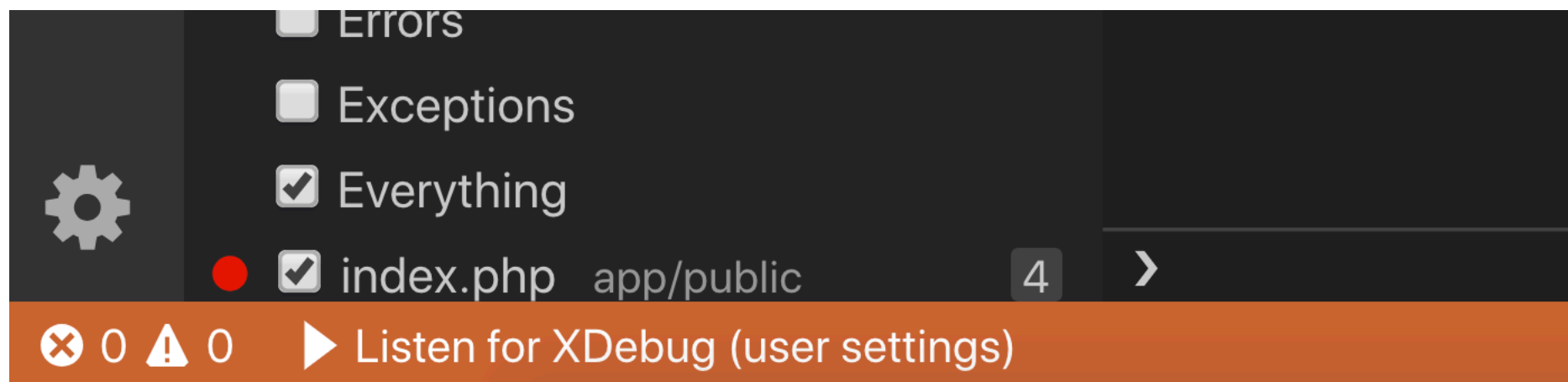
Starta om VSC



# VSC



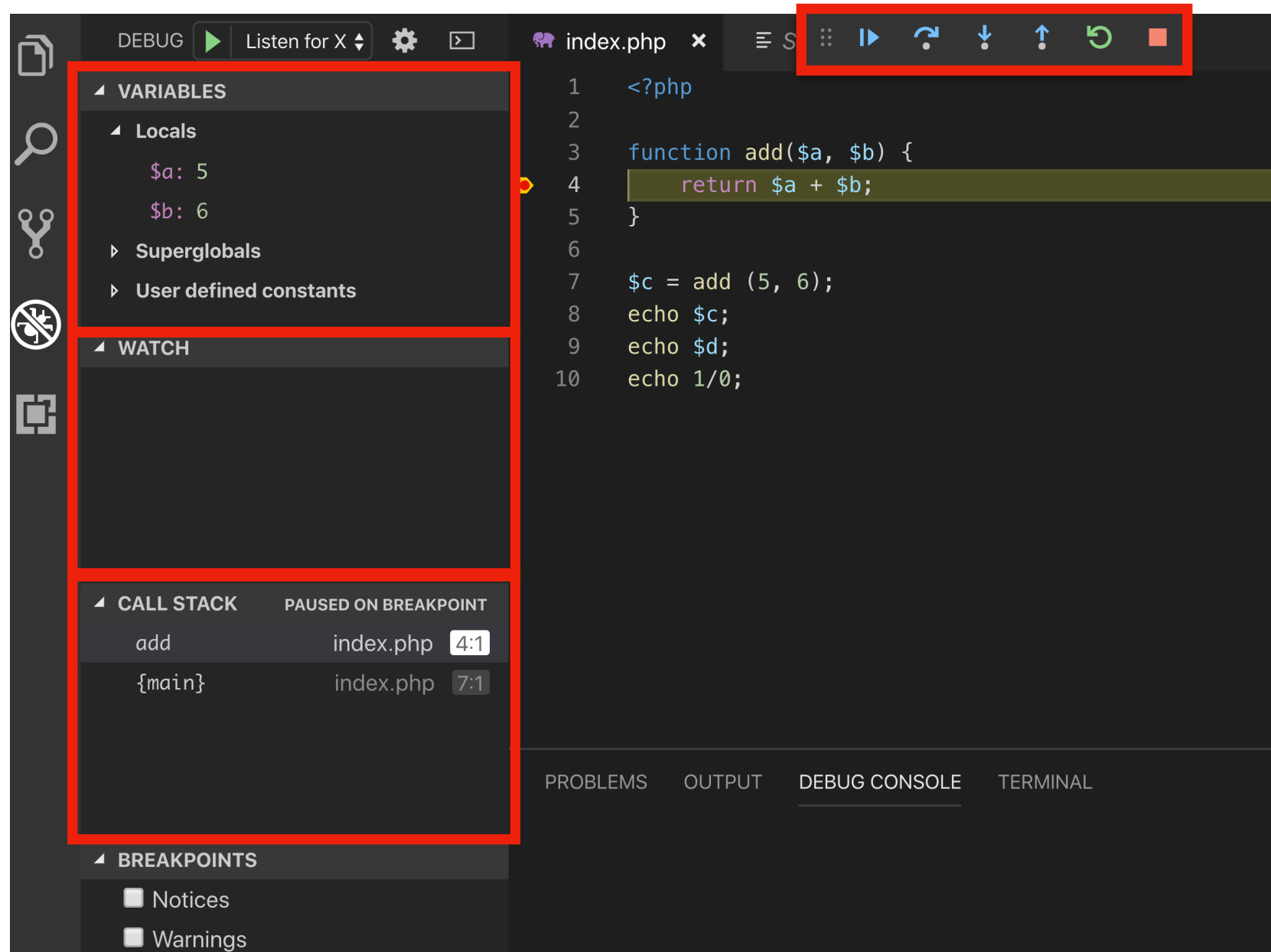
**Välj "Listen for Xdebug" och tryck på "play-knappen".**





# VSC

- Besök sidan i browsern. Du kommer att få upp VSC när PHP stöter på ett fel eller en breakpoint.



# Mer xdebug i VSC

- <https://stackify.com/php-debugging-guide/>

# CodeSniffer

- CodeSniffer används för att se om kod följer en kodstandard.
- <https://tommcfarlin.com/php-codesniffer-in-visual-studio-code/>
- <https://marketplace.visualstudio.com/items?itemName=ikappas.phpcs>
- Skulle vi kunna koppla ihop detta med Git Hooks?

# PHPUnit

- Enhetstester är kod som testar kod.
- Kan t ex testa att en metod returnerar det resultat man har tänkt sig.
- För PHP är PHPUnit absolut vanligast.
  - `composer require --dev phpunit/phpunit`
- Vi använder `--dev`-flaggan för att vi bara vill ha PHPUnit i vår utvecklingsmiljö, inte i produktionsmiljön.

# Assertions

- Assert = hävda
  - `assertEmpty($result)`
  - `assertEquals($expected, $result)`
  - `assertTrue($result)`
  - `assertFalse($result)`
  - `assertInstanceOf($expected, $result)`

# Exempel

```
use drmonkeyninja\Average;
use PHPUnit\Framework\TestCase;
```

```
class AverageTest extends TestCase
{
 public function testCalculationOfMean()
 {
 $numbers = [3, 7, 6, 1, 5];
 $Average = new \Average();
 $this->assertEquals(4.4, $Average->mean($numbers));
 }
}
```

# Exempel

```
use drmonkeyninja\Average;
use PHPUnit\Framework\TestCase;
```

```
class AverageTest extends TestCase
{
 protected $Average;
```

```
 public function setUp()
 {
 $this->Average = new Average();
 }
```

```
 public function testCalculationOfMean()
 {
 $numbers = [3, 7, 6, 1, 5];
 $this->assertEquals(4.4, $this->Average->mean($numbers));
 }
```

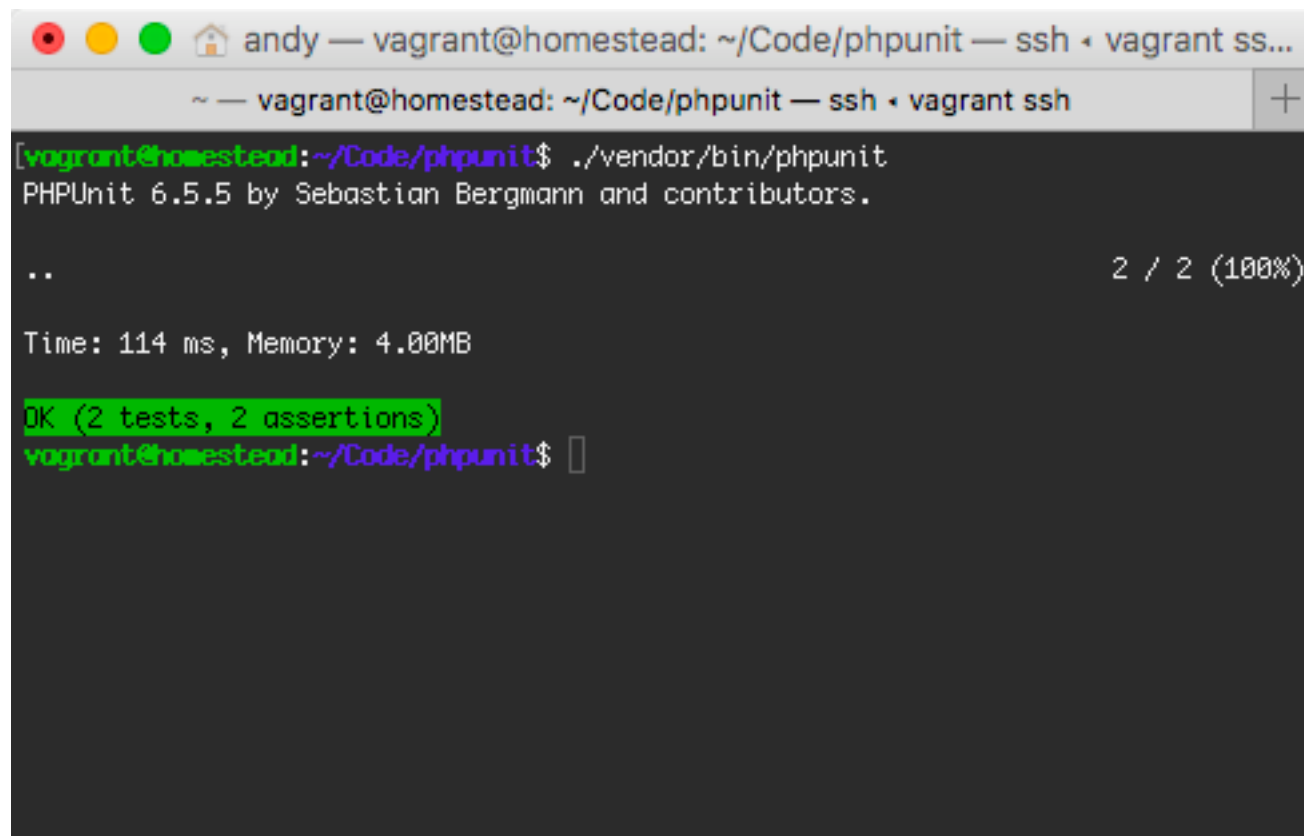
```
 public function testCalculationOfMedian()
 {
 $numbers = [3, 7, 6, 1, 5];
 $this->assertEquals(5, $this->Average->median($numbers));
 }
}
```

# Test

```
./vendor/bin/phpunit
```

```
// Testa en specifik fil:
./vendor/bin/phpunit tests/AverageTest.php
```

```
// Köra ett specifikt test:
./vendor/bin/phpunit tests/ --filter testCalculationOfMean
```

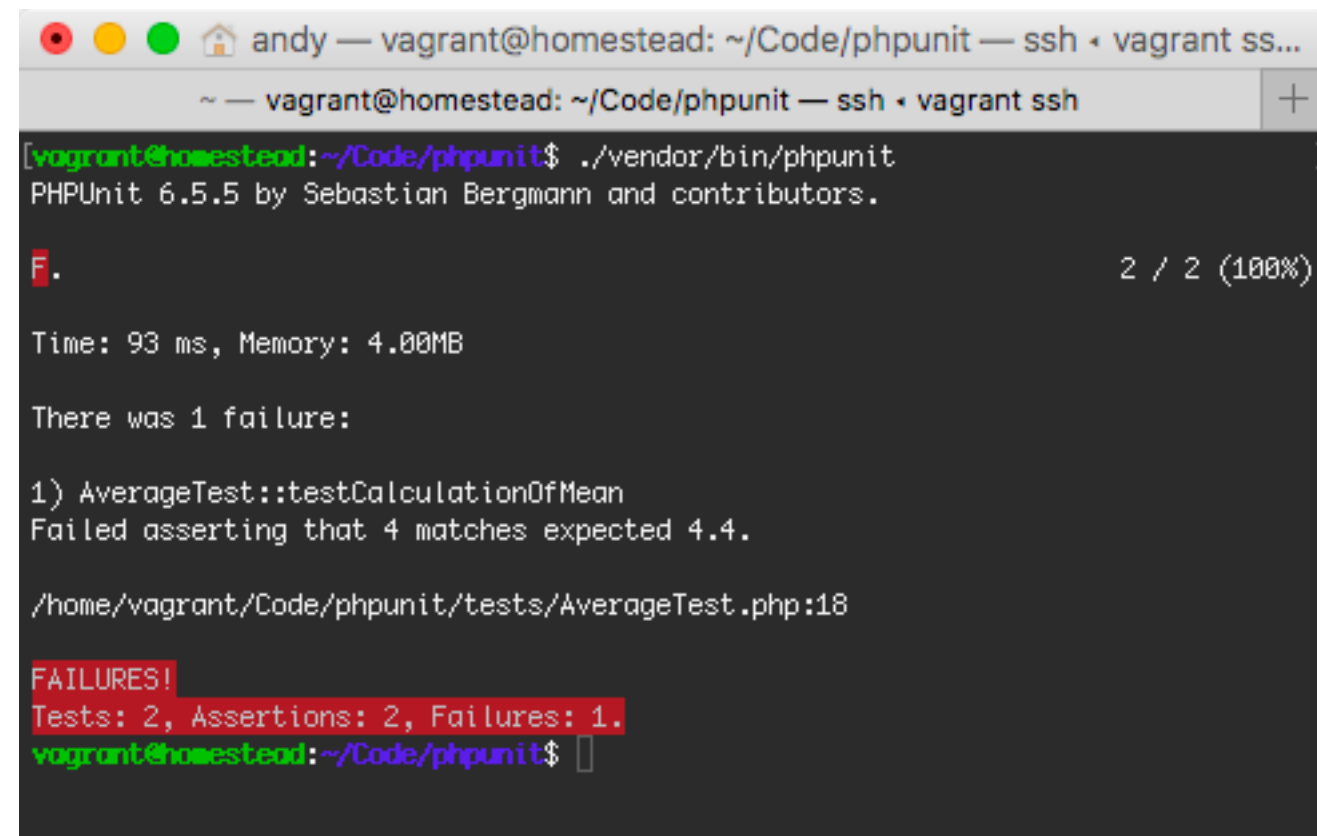


```
andy — vagrant@homestead: ~/Code/phpunit — ssh • vagrant ss...
~ — vagrant@homestead: ~/Code/phpunit — ssh • vagrant ssh
[vagrant@homestead:~/Code/phpunit$./vendor/bin/phpunit
PHPUnit 6.5.5 by Sebastian Bergmann and contributors.

..
2 / 2 (100%)

Time: 114 ms, Memory: 4.00MB

OK (2 tests, 2 assertions)
vagrant@homestead:~/Code/phpunit$
```



```
andy — vagrant@homestead: ~/Code/phpunit — ssh • vagrant ss...
~ — vagrant@homestead: ~/Code/phpunit — ssh • vagrant ssh
[vagrant@homestead:~/Code/phpunit$./vendor/bin/phpunit
PHPUnit 6.5.5 by Sebastian Bergmann and contributors.

F.
2 / 2 (100%)

Time: 93 ms, Memory: 4.00MB

There was 1 failure:

1) AverageTest::testCalculationOfMean
Failed asserting that 4 matches expected 4.4.

/home/vagrant/Code/phpunit/tests/AverageTest.php:18

FAILURES!
Tests: 2, Assertions: 2, Failures: 1.
vagrant@homestead:~/Code/phpunit$
```



# Test

- <https://andy-carter.com/blog/phpunit-what-why-how>
- Skulle vi kunna koppla ihop detta med Git Hooks?

# Livekodning

- Hur lägger man till rader i en integritetsskyddad tabell?
- Hur för man över produkter från en varukorg till en order?
- Helper (select)

# Sammanfattning

- Felsökning
- Enhetstester

# Utvärdering

- Prata i grupper om 2-3 personer i två minuter.
- Vad har varit bra idag?
- Vad skulle kunna förbättras?

# Tack för idag!

**Mikael Olsson**  
**[mikael.olsson@emmio.se](mailto:mikael.olsson@emmio.se)**  
**076-174 90 43**

