LARAVEL LIVE

LONDON 18-19 JUNE 2026 | UK

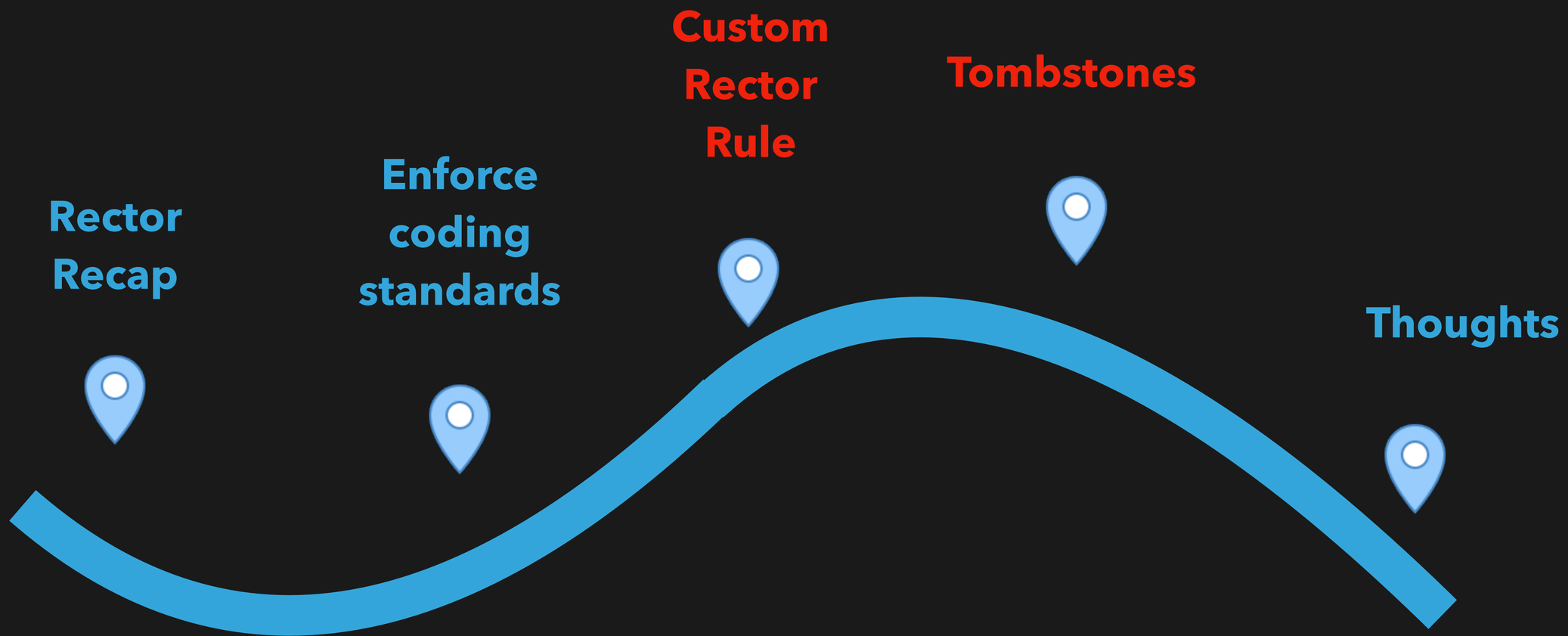
Dave Liddament

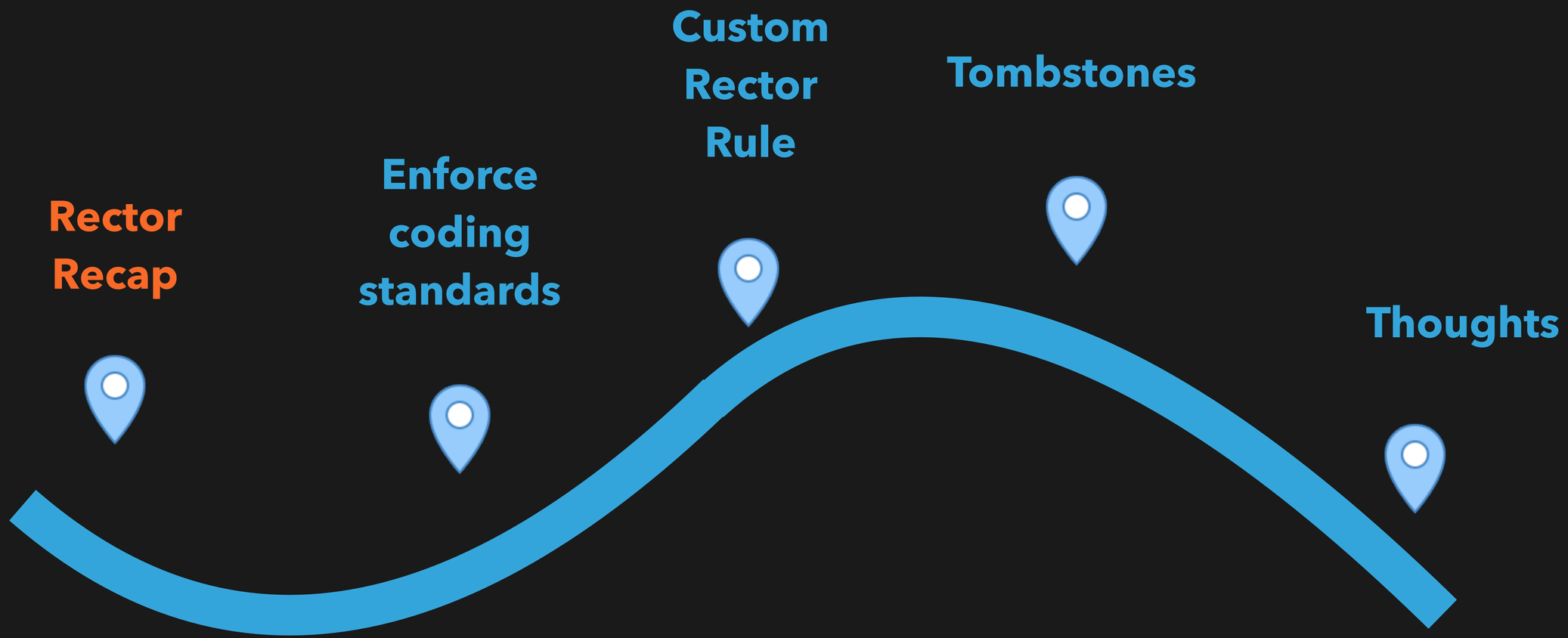
DIRECTOR, LAMP BRISTOL

https://getrector.com/

https://getrector.com/

- Upgrades
- Refactors
- Automate code review
- Investigation work


Rector Recap
Enforce coding standards
Custom Rector Rule
Tombstones
Thoughts


Rector Recap
Enforce coding standards
Custom Rector Rule
Tombstones
Thoughts

867 Rules (and counting)

RenameMethodRector

Turns method names to new ones.

🔧 **configure it!**

- class: `Rector\Renaming\Rector\MethodCall\RenameMethodRector`

```
 $someObject = new SomeExampleClass;
-$someObject->oldMethod();
+$someObject->newMethod();
```

```
composer require --dev rector/rector
```

```
vendor/bin/rector
```

```
No "rector.php" config found. Should we
generate it for you? [yes]:
 > yes
```

```
 [OK] The config is added now. Re-run command
to make Rector do the work!
```

```
composer require --dev rector/rector
```

```
vendor/bin/rector
```

```
No "rector.php" config found. Should we
generate it for you? [yes]:
 > yes
```

```
 [OK] The config is added now. Re-run command
to make Rector do the work!
```

```
composer require --dev rector/rector
```

```
vendor/bin/rector


No "rector.php" config found. Should we
generate it for you? [yes]:
 > yes


 [OK] The config is added now. Re-run command
to make Rector do the work!
```

```php
<?php

use Rector\Config\RectorConfig;
use Rector\Renaming\Rector\MethodCall\RenameMethodRector;
use Rector\Renaming\ValueObject\MethodCallRename;

return RectorConfig::configure()

    ->withPaths([
        __DIR__ . '/src',
    ])

    ->withConfiguredRule(
        RenameMethodRector::class,
        [
            new MethodCallRename(
                App\Framework\Model::class,
                belongsTo',
                belongs')
        ]
    );
```

```php
<?php

use Rector\Config\RectorConfig;
use Rector\Renaming\Rector\MethodCall\RenameMethodRector;
use Rector\Renaming\ValueObject\MethodCallRename;

return RectorConfig::configure()
    ->withPaths([
        __DIR__ . '/src',
    ])

    ->withConfiguredRule(
        RenameMethodRector::class,
        [
            new MethodCallRename(
                App\Framework\Model::class,
                belongsTo',
                belongs')
        ]
    );
```

```php
<?php

use Rector\Config\RectorConfig;
use Rector\Renaming\Rector\MethodCall\RenameMethodRector;
use Rector\Renaming\ValueObject\MethodCallRename;

return RectorConfig::configure()

    ->withPaths([
        __DIR__ . '/src',
    ])

    ->withConfiguredRule(
        RenameMethodRector::class,
        [
            new MethodCallRename(
                App\Framework\Model::class,
                belongsTo',
                belongs')
        ]
    );
```

```php
<?php

use Rector\Config\RectorConfig;
use Rector\Renaming\Rector\MethodCall\RenameMethodRector;
use Rector\Renaming\ValueObject\MethodCallRename;

return RectorConfig::configure()

    ->withPaths([
        __DIR__ . '/src',
    ])

    ->withConfiguredRule(
        RenameMethodRector::class,
        [
            new MethodCallRename(
                App\Framework\Vehicle::class,
                getName',
                getMake')
        ]
    );
```

```
vendor/bin/rector [--dry-run]
```

```
vendor/bin/rector [--dry-run]


1) src/Models/Car.php:10

---------- begin diff ----------
@@ @@
 {

   public function getMake(): string {

-    return $this->vehicle->getName();
+    return $this->vehicle->getMake();
 }

---------- end diff ----------

Applied rules:

 * RenameMethodRector
```

ChangeIfElseValueAssignToEarlyReturnRector

Change if/else value to early return

- class: `Rector\EarlyReturn\Rector\If_\ChangeIfElseValueAssignToEarlyReturnRector`

```
 class SomeClass
 {
     public function run()
     {
         if ($this->hasDocBlock($tokens, $index)) {
-            $docToken = $tokens[$this->getDocBlockIndex($tokens, $index)];
-        } else {
-            $docToken = null;
+            return $tokens[$this->getDocBlockIndex($tokens, $index)];
         }
-
-        return $docToken;
+        return null;
     }
 }
```

https://getrector.com/find-rule

RECTOR Try Rector Documentation Hire Team Blog Contact

Find the best Rector rule to solve your problem. Searching through 867 rules.

Type to start searching a rule

Group: Any group

Set: Pick group to filter

Not sure how to search? Try one of these for a start:

- attributes
- add return type strict
- symfony rules

github.com/driftingly/rector-laravel

ReverseConditionableMethodCallRector

Reverse conditionable method calls

```
-$conditionable->when(!$condition, function () {});
+$conditionable->unless($condition, function () {});
```

SETS: **Code quality**

GuardedPropertyToGuardedAttributeRector

Changes model guarded property to use the guarded attribute

```
 use Illuminate\Database\Eloquent\Model;
+use Illuminate\Database\Eloquent\Attributes\Guarded;

+#[Guarded(['is_admin'])]
 class User extends Model
 {
-    protected $guarded = [
-        'is_admin',
-    ];
 }
```

SETS: laravel/framework 13.0

Rulesets: A set of rules

Laravel 13 upgrade rectors

DelayPropertyToDelayAttributeRector

HelpPropertyToHelpAttributeRector

ConnectionPropertyToConnectionAttributeRector

HiddenPropertyToHiddenAttributeRector

… And others …

Code quality rectors

RedirectRouteToToRouteHelperRector

SleepFuncToSleepStaticCallRector

AppToResolveRector

RemoveRedundantValueCallsRector

… And others …

rector.php

```php
<?php

declare(strict_types=1);

use Rector\Config\RectorConfig;
use Rector\Set\ValueObject\SetList;

return RectorConfig::configure()
    ->withPaths(…)
    ->withSets([
        SetList::PHP_85,
    ]);
```

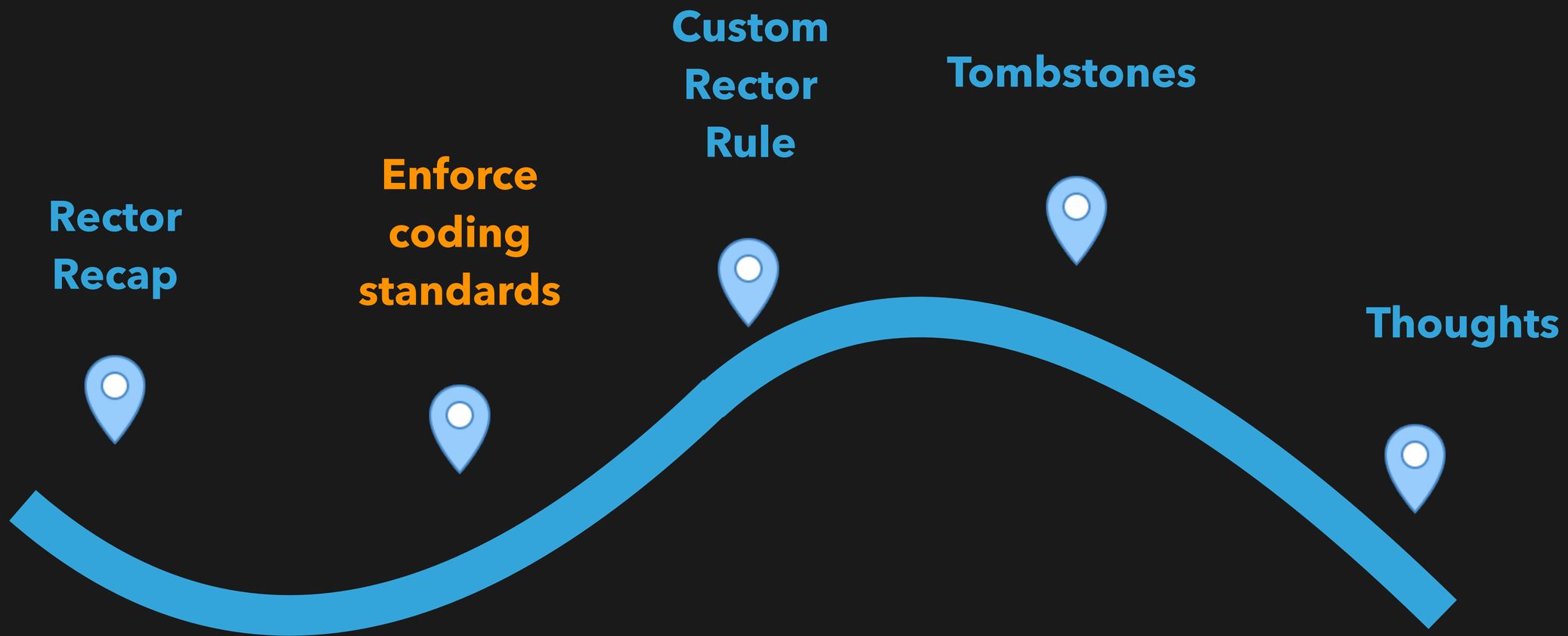
rector.php

```php
<?php

declare(strict_types=1);

use Rector\Config\RectorConfig;
use Rector\Set\ValueObject\SetList;

return RectorConfig::configure()
    ->withPaths(…)
    ->withPreparedSets(
        earlyReturn: true,
    );
```

Levels: Step by step

```
return RectorConfig::configure()
    ->withPaths(…)
    ->withTypeCoverageLevel(10)
    ->withCodeQuality(5)

    ->and others…
);
```

- Upgrades
- Refactors
- Downgrades
- Automate code review
- Investigation work


Rector
Recap
Enforce
coding
standards
Custom
Rector
Rule
Tombstones
Thoughts

Coding standards

- PSR 12
- Use Early Return pattern
- Use named arguments for #[Middleware]

Coding standards

- PSR 12
- Use Early Return pattern
- Use named arguments for #[Middleware]

rector-ci.php

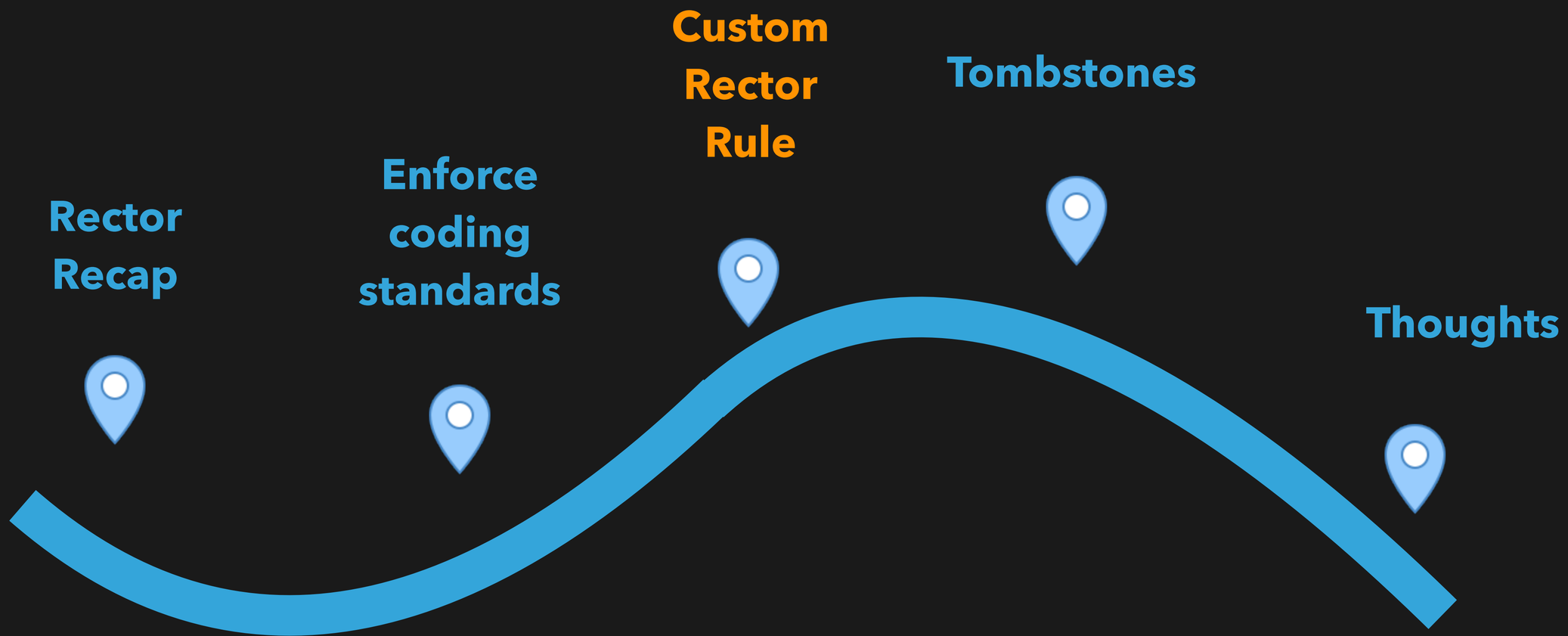
```
return RectorConfig::configure()

    ->withPaths([
        __DIR__ . '/src',
   ])

    ->withPreparedSets(earlyReturn: true)
    // Add any other rules and sets
;
```

```
vendor/bin/rector --config rector-ci.php
```

Notes:
1.Run Laravel Pint after rector
2.On CI run with --dry-run

Coding standards

- PSR 12
- Use Early Return pattern
- Use named arguments for #[Middleware]


Rector Recap
Enforce coding standards
Custom Rector Rule
Tombstones
Thoughts

```php
#[Attribute(Attribute::TARGET_CLASS |
  Attribute::TARGET_METHOD |
  Attribute::IS_REPEATABLE)]
class Middleware
{
    /**
     * @param  array<string>|null  $only
     * @param  array<string>|null  $except
     */
    public function __construct(
        public Closure|string $middleware,
        public ?array $only = null,
        public ?array $except = null,
    ) {
    }
}
```

```php
#[Attribute(Attribute::TARGET_CLASS |
  Attribute::TARGET_METHOD |
  Attribute::IS_REPEATABLE)]
class Middleware
{
    /**
     * @param  array<string>|null  $only
     * @param  array<string>|null  $except
     */
    public function __construct(
        public Closure|string $middleware,
        public ?array $only = null,
        public ?array $except = null,
    ) {
    }
}
```

Old

```
#[Middleware(RequireToken::class, ['edit'])]
class ItemController
{
    … controller methods …
}
```

New

```
#[Middleware(RequireToken::class, only: ['edit'])]
class ItemController
{
    … controller methods …
}
```

No Change: 1

```
#[Middleware(RequireToken::class, only: [‘edit’])]
class ItemController
{
    … controller methods …
}
```

No Change: 2

```
#[Middleware(RequireToken::class, except: [‘edit’])]
class ItemController
{
    … controller methods …
}
```

No Change: 2

```
#[Middleware(RequireToken::class, except: ['edit'])]
class ItemController
{
    … controller methods …
}
```

No Change: 3

```
#[MyAttribute('foo', 'bar')]
class PersonController
{
}
```

```
$ vendor/bin/rector custom-rule


 What is the name of the rule class (e.g. "LegacyCallToDbalMethodCall")?:
 > MiddlewareUseNamedArguments

Generated files
===============

 * utils/rector/src/Rector/MiddlewareUseNamedArgumentsRector.php
 * utils/rector/tests/Rector/MiddlewareUseNamedArgumentsRector/Fixture/some_class.php.inc
 * utils/rector/tests/Rector/MiddlewareUseNamedArgumentsRector/config/configured_rule.php
 * utils/rector/tests/Rector/MiddlewareUseNamedArgumentsRector/
MiddlewareUseNamedArgumentsRectorTest.php


 [OK] Base for the "MiddlewareUseNamedArguments" rule was created. Now you can fill the
missing parts


 [OK] We also update composer.json autoload-dev, to load Rector rules. Now run "composer
dump-autoload" to update paths
```

```
$ vendor/bin/rector custom-rule
```

```
 What is the name of the rule class (e.g. "LegacyCallToDbalMethodCall")?:
 > MiddlewareUseNamedArguments

Generated files
===============

 * utils/rector/src/Rector/MiddlewareUseNamedArgumentsRector.php
 * utils/rector/tests/Rector/MiddlewareUseNamedArgumentsRector/Fixture/some_class.php.inc
 * utils/rector/tests/Rector/MiddlewareUseNamedArgumentsRector/config/configured_rule.php
 * utils/rector/tests/Rector/MiddlewareUseNamedArgumentsRector/
MiddlewareUseNamedArgumentsRectorTest.php


 [OK] Base for the "MiddlewareUseNamedArguments" rule was created. Now you can fill the
missing parts


 [OK] We also update composer.json autoload-dev, to load Rector rules. Now run "composer
dump-autoload" to update paths
```

```
$ vendor/bin/rector custom-rule

 What is the name of the rule class (e.g. "LegacyCallToDbalMethodCall")?:
 > MiddlewareUseNamedArguments

Generated files
===============

 * utils/rector/src/Rector/MiddlewareUseNamedArgumentsRector.php
 * utils/rector/tests/Rector/MiddlewareUseNamedArgumentsRector/Fixture/some_class.php.inc
 * utils/rector/tests/Rector/MiddlewareUseNamedArgumentsRector/config/configured_rule.php
 * utils/rector/tests/Rector/MiddlewareUseNamedArgumentsRector/
MiddlewareUseNamedArgumentsRectorTest.php


 [OK] Base for the "MiddlewareUseNamedArguments" rule was created. Now you can fill the
missing parts


 [OK] We also update composer.json autoload-dev, to load Rector rules. Now run "composer
dump-autoload" to update paths
```

```
$ vendor/bin/rector custom-rule


 What is the name of the rule class (e.g. "LegacyCallToDbalMethodCall")?:
 > MiddlewareUseNamedArguments

Generated files
===============

 * utils/rector/src/Rector/MiddlewareUseNamedArgumentsRector.php
 * utils/rector/tests/Rector/MiddlewareUseNamedArgumentsRector/Fixture/some_class.php.inc
 * utils/rector/tests/Rector/MiddlewareUseNamedArgumentsRector/config/configured_rule.php
 * utils/rector/tests/Rector/MiddlewareUseNamedArgumentsRector/
MiddlewareUseNamedArgumentsRectorTest.php


 [OK] Base for the "MiddlewareUseNamedArguments" rule was created. Now you can fill the
missing parts


 [OK] We also update composer.json autoload-dev, to load Rector rules. Now run "composer
dump-autoload" to update paths
```

Old

```
#[Middleware(RequireToken::class, ['edit'])]
class ItemController
{
    … controller methods …
}
```

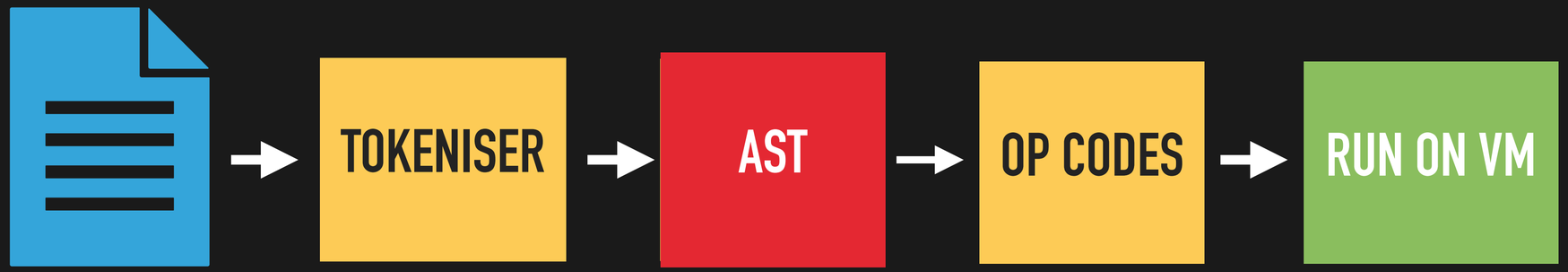
New

```
#[Middleware(RequireToken::class, only: ['edit'])]
class ItemController
{
    … controller methods …
}
```

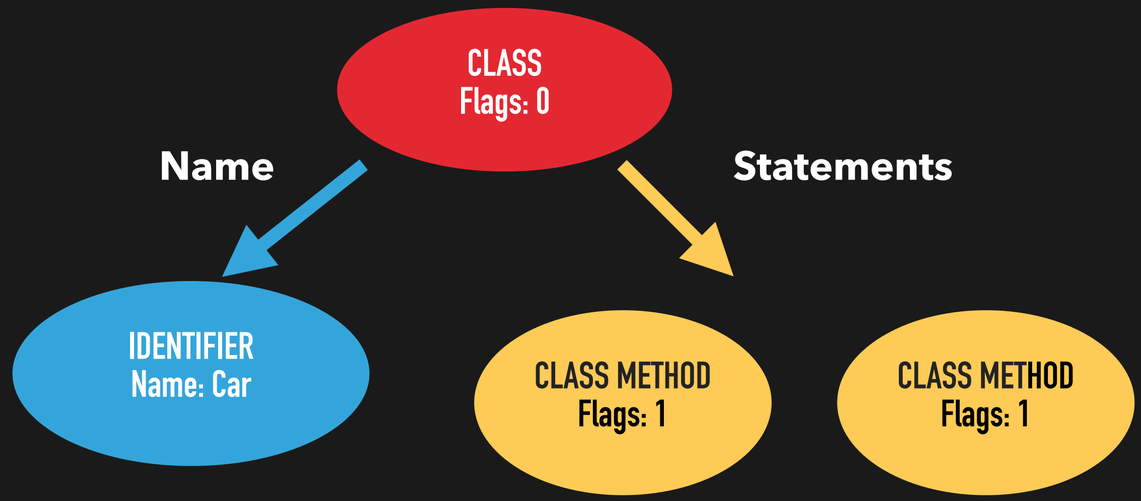
No Change

```
#[Middleware(RequireToken::class, only: [‘edit’])]
class ItemController
{
    … controller methods …
}
```

```
RouterUseNamedArguments extends AbstractRector
{

    public function getNodeTypes(): array

    public function refactor(Node $node): ?Node
}
```

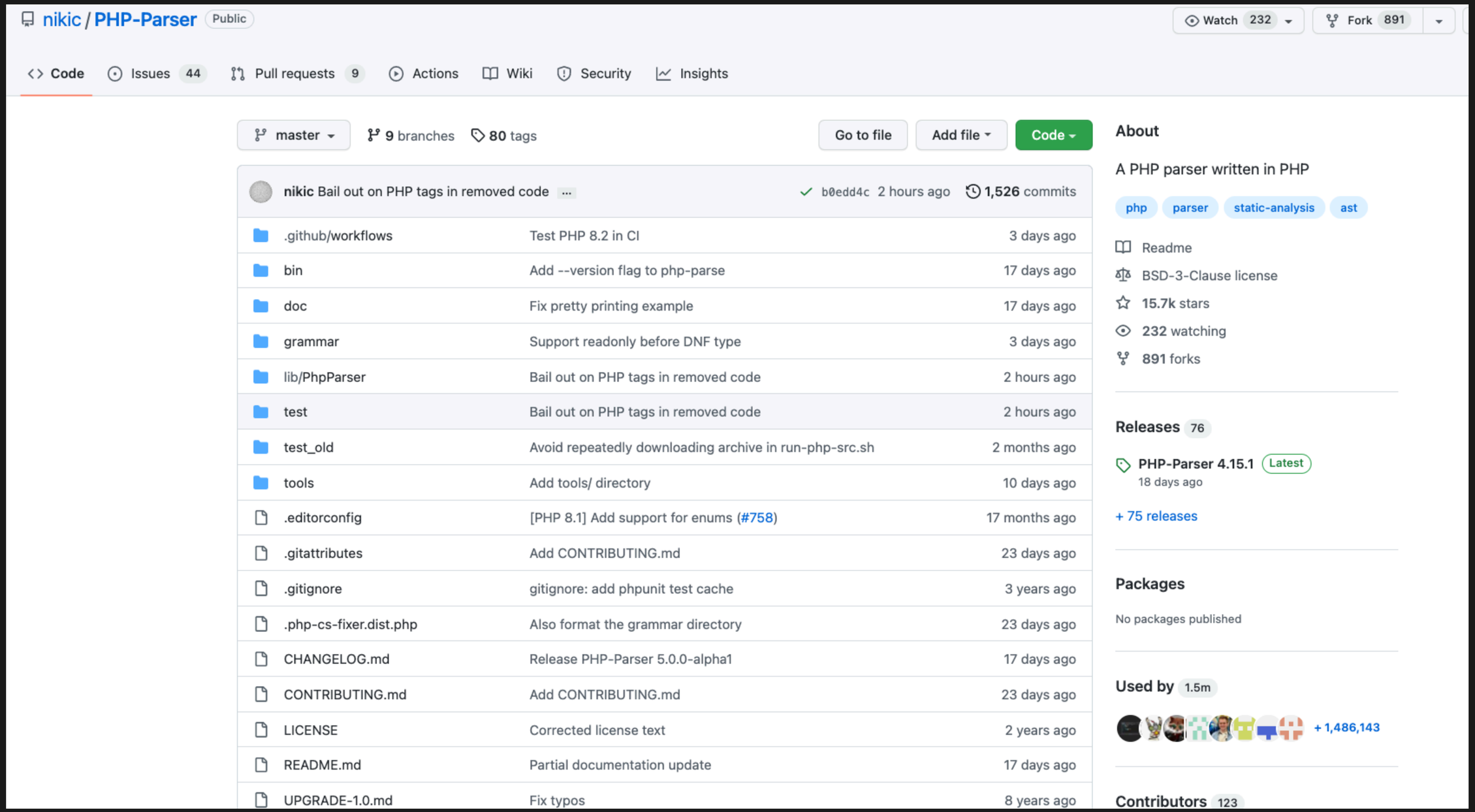

TOKENISER
AST
OP CODES
RUN ON VM

```
class Car
{
    public function users() {…}

    public function anotherMethod() {…}
}
```



```
public function user ()
{
    return $this->belongsTo('User');
}
```

https://github.com/nikic/PHP-Parser



nikic / PHP-Parser Public

Watch 232 | Fork 891

Code | Issues 44 | Pull requests 9 | Actions | Wiki | Security | Insights

master | 9 branches | 80 tags | Go to file | Add file | Code

nikic Bail out on PHP tags in removed code ... b0edd4c 2 hours ago 1,526 commits

.github/workflows	Test PHP 8.2 in CI	3 days ago
bin	Add --version flag to php-parse	17 days ago
doc	Fix pretty printing example	17 days ago
grammar	Support readonly before DNF type	3 days ago
lib/PhpParser	Bail out on PHP tags in removed code	2 hours ago
test	Bail out on PHP tags in removed code	2 hours ago
test_old	Avoid repeatedly downloading archive in run-php-src.sh	2 months ago
tools	Add tools/ directory	10 days ago
.editorconfig	[PHP 8.1] Add support for enums (#758)	17 months ago
.gitattributes	Add CONTRIBUTING.md	23 days ago
.gitignore	gitignore: add phpunit test cache	3 years ago
.php-cs-fixer.dist.php	Also format the grammar directory	23 days ago
CHANGELOG.md	Release PHP-Parser 5.0.0-alpha1	17 days ago
CONTRIBUTING.md	Add CONTRIBUTING.md	23 days ago
LICENSE	Corrected license text	2 years ago
README.md	Partial documentation update	17 days ago
UPGRADE-1.0.md	Fix typos	8 years ago

About

A PHP parser written in PHP

php | parser | static-analysis | ast

Readme

BSD-3-Clause license

15.7k stars

232 watching

891 forks

Releases 76

PHP-Parser 4.15.1 Latest

18 days ago

+ 75 releases

Packages

No packages published

Used by 1.5m

+ 1,486,143

Contributors 123

```
class Attribute extends \PhpParser\NodeAbstract
{
    public Name $name

    /** @list<Arg> */
    public array $args;

    // Rest of class …
```

```
#[Middleware(RequireToken::class, ['edit'])]
```

```php
public function user ()
{
  return $this->belongsTo('User');
}
```


```php
class Attribute extends \PhpParser\NodeAbstract
{
  public Name $name

  /** @list<Arg> */
  public array $args;

  // Rest of class …
```

```php
#[Middleware(RequireToken::class, ['edit'])]
```

- PHP code can be represented by an AST
- Different types of Node
- Nodes contain information
- Each type of node has different information

www.getrector.com/ast

RECTOR Try Rector Docs Hire Team Find Rule Blog Contact AST

1. Write short PHP code you want to understand

```php
<?php

if ($condition === 'demo') {
    return true;
}
```

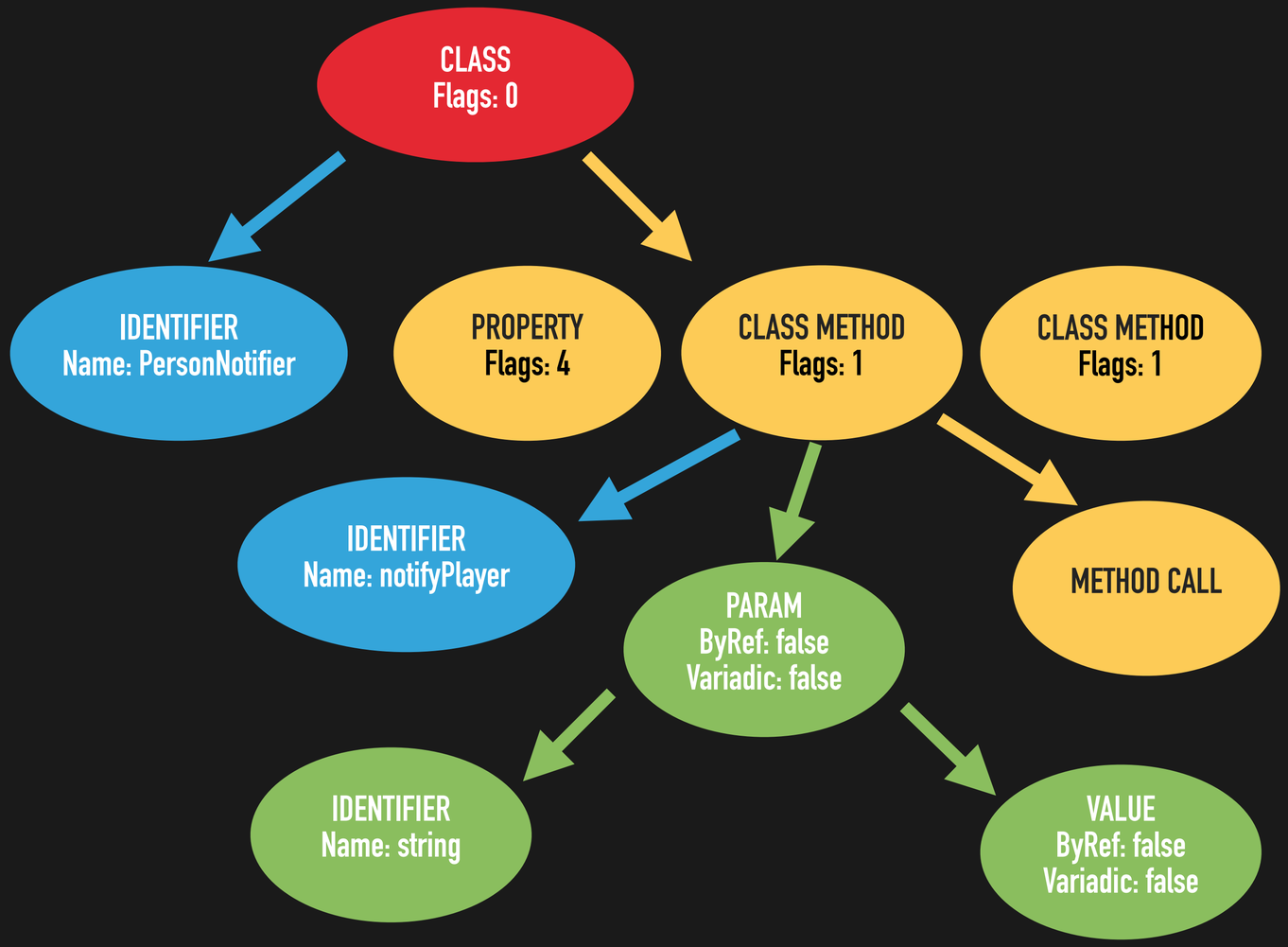
Parse

2. Click on any part of the code

```php
<?php

if ($condition === 'demo') {
    return \true;
}
```

3. See Abstract Syntax Tree created by php-parser for full file

```
PhpParser\Node\Stmt\If_(
    cond: PhpParser\Node\Expr\BinaryOp\Identical(
        left: PhpParser\Node\Expr\Variable( name: "condition" )
        right: PhpParser\Node\Scalar\String_( value: "demo" )
    )
    stmts: [
        0: PhpParser\Node\Stmt\Return_(
            expr: PhpParser\Node\Expr\ConstFetch(
                name: PhpParser\Node\Name\FullyQualified( parts: ["true"] )
            )
        )
    ]
    elseifs: []
    else: null
)
```


CLASS
Flags: 0
IDENTIFIER
Name: PersonNotifier
PROPERTY
Flags: 4
CLASS METHOD
Flags: 1
CLASS METHOD
Flags: 1
IDENTIFIER
Name: notifyPlayer
PARAM
ByRef: false
Variadic: false
METHOD CALL
IDENTIFIER
Name: string
VALUE
ByRef: false
Variadic: false

```
#[Middleware(RequireToken::class, ['edit')]]
#[Middleware(RequireToken::class, only: ['edit')]]
```

```
class Attribute extends \PhpParser\NodeAbstract
{
    public Name $name

    /** @list<Arg> */
    public array $args;

    // Rest of class …
```

```
#[Middleware(RequireToken::class, ['edit'])]
```

```
public function getNodes(): array
{
  return [Attribute::class];
}
```

```
public function refactor(Node $node): ?Node
{

}
```

```
#[Middleware(RequireToken::class, ['edit'])]
```

name

args

- **name is Middleware**
- **2nd or 3rd arg does not use a name**

```
public function refactor(Node $node): ?Node
{
    // Is name Middleware?

    if (!$this->isName($node->name, Middleware::class)) {
        return null;
    }
```

```
$updated = false;

 // Does the second argument use a name?

 $arg2 = $node->args[1] ?? null;

if (($arg2 !== null) && ($arg2->name === null)) {
   $arg2->name = new Identifier('only');
   $updated = true;
}
```

```
$updated = false;


// Does the second argument use a name?

$arg2 = $node->args[1] ?? null;

if (($arg2 !== null) && ($arg2->name === null)) {
  $arg1->name = new Identifier('only');
  $updated = true;
}
```

```
$updated = false;


// Does the second argument use a name?

$arg2 = $node->args[1] ?? null;

if (($arg2 !== null) && ($arg2->name === null)) {

  $arg2->name = new Identifier('only');
  $updated = true;
}
```

```
class Arg extends NodeAbstract
{
    /**
     * @var Identifier|null Parameter name
     * (for named parameters)
     */
    public ?\PhpParser\Node\Identifier $name;

    /** @var Expr Value to pass */
    public \PhpParser\Node\Expr $value;

    /** @var bool Whether to pass by ref */
    public bool $byRef;

    /** @var bool Whether to unpack the argument */
    public bool $unpack;
```

```
$updated = false;


// Does the second argument use a name?

$arg2 = $node->args[1] ?? null;

if (($arg2 !== null) && ($arg2->name === null)) {

  $arg2->name = new Identifier('only');
  $updated = true;
}
```

```
$updated = false;


// Does the second argument use a name?

$arg2 = $node->args[1] ?? null;

if (($arg2 !== null) && ($arg2->name === null)) {
    $arg2->name = new Identifier('only');
    $updated = true;
}
```

```
    // Does the third argument use a name?

    $arg3 = $node->args[2] ?? null;

    if (($arg3 !== null) && ($arg3->name === null)) {
      $arg3->name = new Identifier('except');
      $updated = true;
    }

    // Return

    return $updated ? $node : null;

}
```

```
    // Does the third argument use a name?

    $arg3 = $node->args[2] ?? null;

    if (($arg3 !== null) && ($arg3->name === null)) {
      $arg3->name = new Identifier('except');
      $updated = true;
    }

    // Return

    return $updated ? $node : null;
}
```

```
class MiddlewareUseNamedArguments extends AbstractRector {

  public function getNodes(): array
  {
     return [Attribute::class];
  }

  public function refactor(Node $node): ?Node
  {
     // Is name Middleware?
     if (!$this->isName($node->name, Middleware::class)) {
        return null;
     }

    $updated = false;

    // Does the 2nd argument use a name?
    $arg2 = $node->args[1] ?? null;
    if (($arg2 !== null) && ($arg2->name === null)) {
      $arg1->name = new Identifier('only');
      $updated = true;
    }

    // Does the third argument use a name?
    $arg3 = $node->args[2] ?? null;
    if (($arg3 !== null) && ($arg3->name === null)) {
      $arg3->name = new Identifier('except');
      $updated = true;
    }

    return $updated ? $node : null;
  }
}
```

Improvement 1

```
public function refactor(Node $node): ?Node
{
 // Is name Route?
 if (!$this->isName($node->name, Middleware::class)) {
    return null;
 }

 // Get information on parameters

 $parameters = new ReflectionClass(Middleware::class)
                          ->getConstructor()
                          ->getParameters();
```

```php
$updated = false;

// Add name to each argument if needed
foreach($node->args as $index => $arg) {

  if ($index < 1) {
    continue;
  }

  $parameterName = $parameters[$index]?->name;
  if ($parameterName === null) {
    continue;
  }

  if ($arg->name === null) {
    $arg->name = new Node\Identifier($parameterName);
    $updated = true;
  }
}

return $updated ? $node : null;
```

```
$updated = false;

// Add name to each argument if needed
foreach($node->args as $index => $arg) {

  if ($index < 1) {
    continue;
  }

  $parameterName = $parameters[$index]?->name;
  if ($parameterName === null) {
    continue;
  }

  if ($arg->name === null) {
    $arg->name = new Node\Identifier($parameterName);
    $updated = true;
  }
}

return $updated ? $node : null;
```

```
$updated = false;

// Add name to each argument if needed
foreach($node->args as $index => $arg) {

    if ($index < 1) {
        continue;
    }

    $parameterName = $parameters[$index]?->name;
    if ($parameterName === null) {
        continue;
    }

    if ($arg->name === null) {
        $arg->name = new Node\Identifier($parameterName);
        $updated = true;
    }
}

return $updated ? $node : null;
```

```
$updated = false;

// Add name to each argument if needed
foreach($node->args as $index => $arg) {

  if ($index < 1) {
    continue;
  }

  $parameterName = $parameters[$index]?->name;
  if ($parameterName === null) {
    continue;
  }

  if ($arg->name === null) {
    $arg->name = new Node\Identifier($parameterName);
    $updated = true;
  }
}

return $updated ? $node : null;
```

```
$updated = false;

// Add name to each argument if needed
foreach($node->args as $index => $arg) {

  if ($index < 1) {
    continue;
  }

  $parameterName = $parameters[$index]?->name;
  if ($parameterName === null) {
    continue;
  }

  if ($arg->name === null) {
    $arg->name = new Node\Identifier($parameterName);
    $updated = true;
  }
}

return $updated ? $node : null;
```

```php
$updated = false;

// Add name to each argument if needed
foreach($node->args as $index => $arg) {

  if ($index < 1) {
    continue;
  }

  $parameterName = $parameters[$index]?->name;
  if ($parameterName === null) {
    continue;
  }

  if ($arg->name === null) {
    $arg->name = new Node\Identifier($parameterName);
    $updated = true;
  }
}

return $updated ? $node : null;
```

Improvement 2

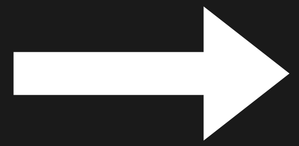
```
public function refactor(Node $node): ?Node
{
  if (!$this->isName($node->name, $this->className)) {
      return null;
   }

   // Get information on parameters
   $parameters = new ReflectionClass($this->className)
     ->getConstructor()->getParameters();

   $updated = false;

   foreach($node->args as $index => $arg) {

   if ($index < $this->firstArgumentIndexToUpdate) {
     continue;
   }

   … code as before …
```

```
return RectorConfig::configure()

    ->withPaths(…)

    ->withConfiguredRule(
        AttributeNameArgsRector::class,
        [
            Middleware::class, 1
        ]
    );
```

```php
return RectorConfig::configure()

    ->withPaths(…)

    ->withConfiguredRule(
        AttributeNameArgsRector::class,
        [
            new AttributeNamedArgs(Middleware::class, 1),
            new AttributeNamedArgs(AnotherAttribute::class, 0),
            … and more …
        ]
    )
;
```

Custom rector rule flow

Solve a specific problem



Generalise

RECTOR

The Power of Automated Refactoring

2024 Edition
Rector 1.0

Matthias Noback
Tomas Votruba

Or ask AI to create a rule

https://github.com/peterfox/agent-skills

---

rector.php ⟵ Established rules

rector-beta.php ⟵ Check output carefully

```
vendor/bin/rector --config <config file>
```

Coding standards

Problem

Improve tooling every review

phpday

MAY, 15-16th 2025
VERONA
+ ONLINE

Grusp

Custom PHPStan Rules: Automate Standards and Save Time

DAVE LIDDAMENT

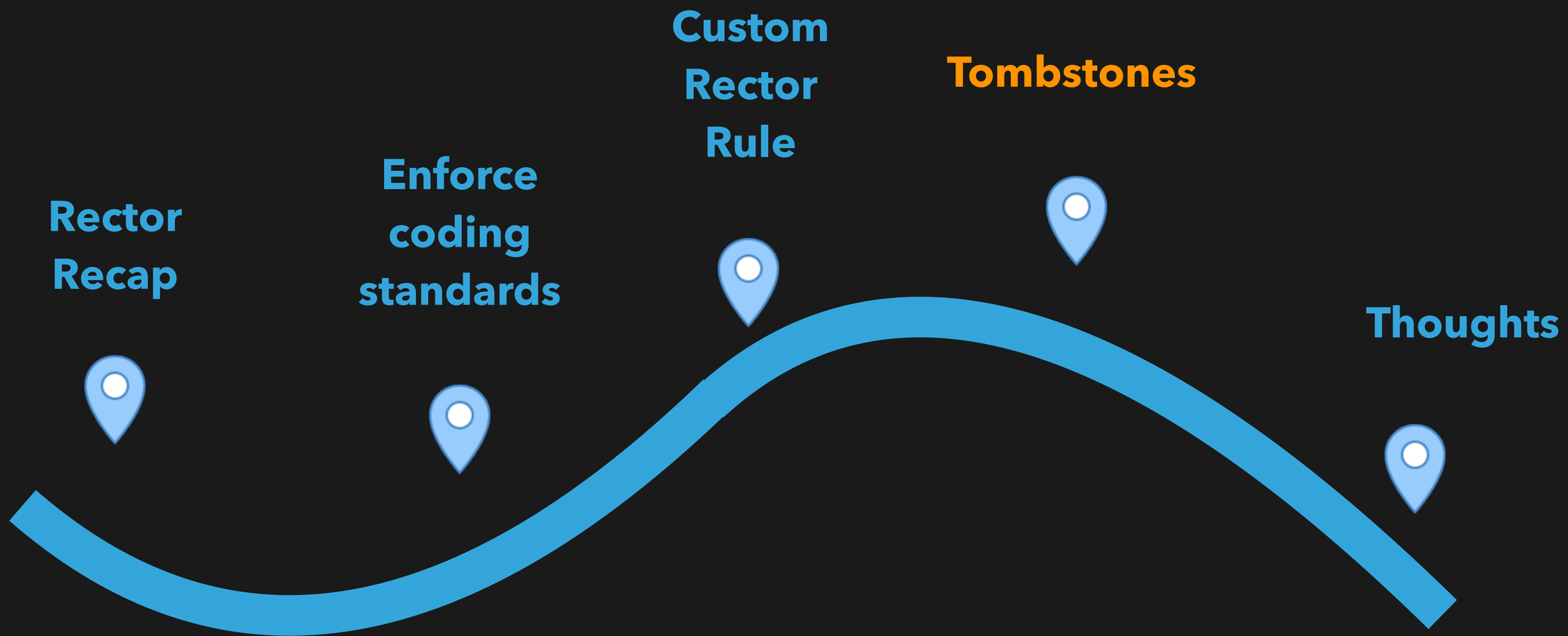

Rector Recap
Enforce coding standards
Custom Rector Rule
Tombstones
Thoughts

Photo by Antoine GIRET on Unsplash

Add tombstones

```
final class PriceCalculator
{
 public function calculate(): void
 {

   TombstoneReporter::trigger(
     'PriceCalculator', 'calculate');

  // Rest of method's code
 }
}
```

TombstoneReporter::trigger

- Gets list of all triggered tombstones from database (once per request)
- If triggered tombstone not on list, create DB entry

TombstoneReporter::dump

triggered_tombstones.json

```
[
  'PriceCalculator::calculate',
  'PriceController::index',
  'PriceRepository::getItems,
  ...
];
```

Remove triggered tombstones

```
final class PriceCalculator
{
 public function calculate(): void
 {

   TombstoneReporter::trigger(
     'PriceCalculator', 'calculate');

  // Rest of method's code
 }
}
```

Delete code

```
“REMOVE: Product listing page”

“REMOVE: 2 for 1 discount code”
```

Photo by Claudio Schwarz on Unsplash

How do we add tombstones?

How do we add tombstones?

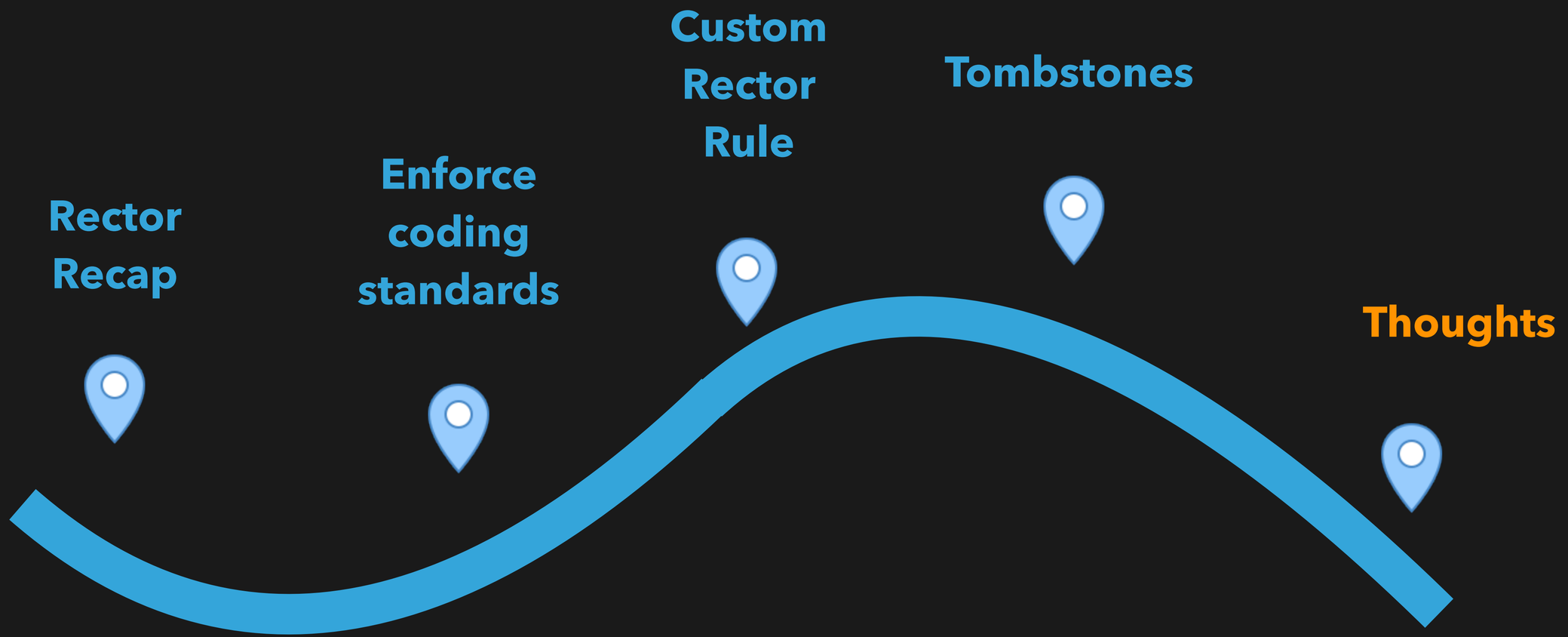
```
final class AddTombstoneRector extends AbstractRector
{


    public function getNodeTypes(): array
    {
        return [ClassMethod::class];
    }
```

```php
public function refactor(Node $node): ?Node
{
    $scope = ScopeFetcher::fetch($node);
    $className = $scope->getClassReflection()?->getName() ?? null;
    if ($className === null) {
        return null;
    }

    $staticCall = $this->nodeFactory->createStaticCall(
        Tombstone::class,
        'trigger',
        [
            $this->nodeFactory->createArg($className),
            $this->nodeFactory->createArg($node->name->name),
        ]
    );
    $staticCallStatement = new Node\Stmt\Expression($staticCall);

    $statements = $node->stmts ?? [];
    array_unshift($statements, $staticCallStatement);
    $node->stmts = $statements;
    return $node;
}
```

```php
public function refactor(Node $node): ?Node
{

  $scope = ScopeFetcher::fetch($node);
  $className = $scope->getClassReflection()?->getName() ?? null;
  if ($className === null) {
    return null;
  }

  $staticCall = $this->nodeFactory->createStaticCall(
    Tombstone::class,
    'trigger',
    [
      $this->nodeFactory->createArg($className),
      $this->nodeFactory->createArg($node->name->name),
    ]
  );
  $staticCallStatement = new Node\Stmt\Expression($staticCall);

  $statements = $node->stmts ?? [];
  array_unshift($statements, $staticCallStatement);
  $node->stmts = $statements;
  return $node;
}
```

```
public function refactor(Node $node): ?Node
{

  $scope = ScopeFetcher::fetch($node);
  $className = $scope->getClassReflection()?->getName() ?? null;
  if ($className === null) {
    return null;
  }

  $staticCall = $this->nodeFactory->createStaticCall(
    Tombstone::class,
    'trigger',
    [
      $this->nodeFactory->createArg($className),
      $this->nodeFactory->createArg($node->name->name),
    ]
  );
  $staticCallStatement = new Node\Stmt\Expression($staticCall);

  $statements = $node->stmts ?? [];
  array_unshift($statements, $staticCallStatement);
  $node->stmts = $statements;
  return $node;
}
```

https://github.com/DaveLiddament/tombstone-demo


Rector Recap
Enforce coding standards
Custom Rector Rule
Tombstones
Thoughts

Type coverage is good

0% type coverage

```
function getNames($users)
{

  $names = [];
  foreach($users as $user) {
    $names[] = $user->getUser();
  }

  return join(‘,’, $names);
}
```

100% type coverage

```
/** @param array<int,User> $users */
function getNames(array $users): string
{

  $names = [];
  foreach($users as $user) {
    $names[] = $user->getUser();
  }
  return join(‘,’, $names);

}
```

Legacy applications: Add to docblock first

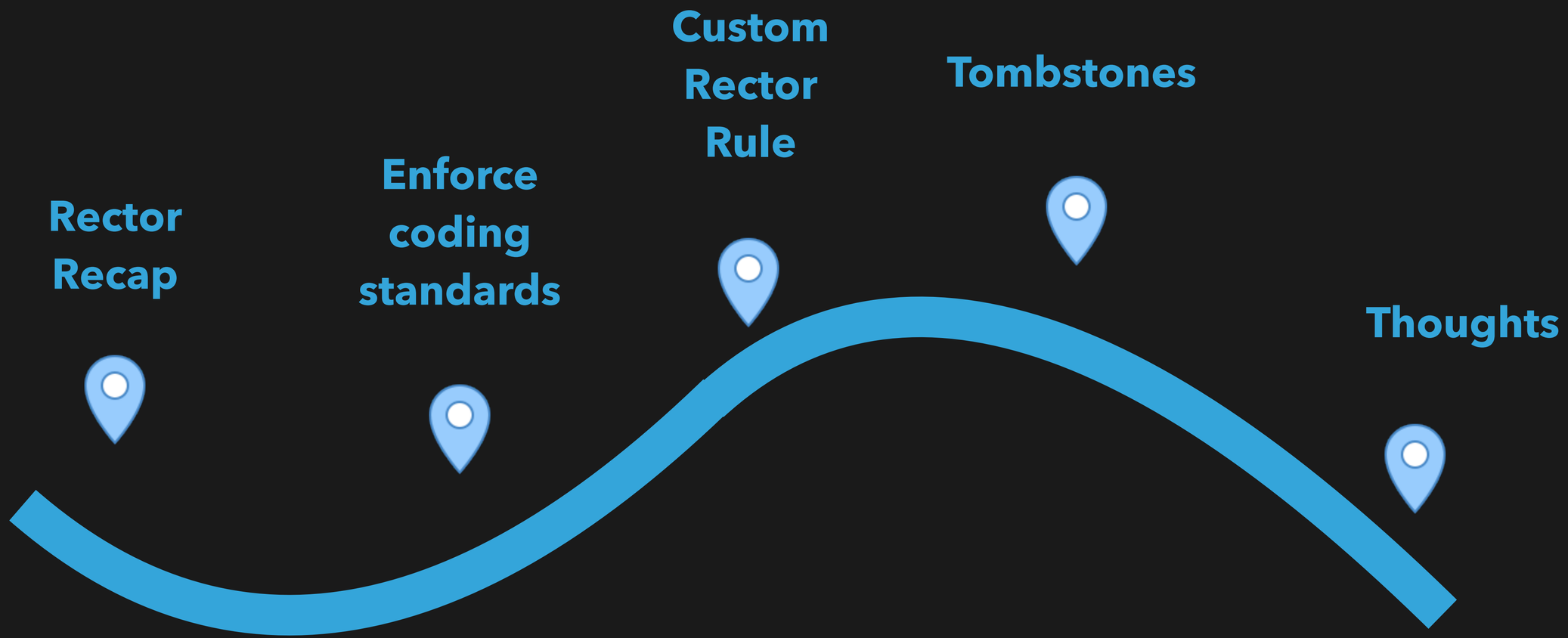
```
/**
 * @param array<int,User> $users
 * @return string
 */
function getNames($users)
{

  $names = [];
  foreach($users as $user) {
    $names[] = $user->getUser();
  }

  return join(‘,’, $names);
}
```

Could we track type information?

```
class Person
{
  public function process($name, $age)
  {
    TypeRecorder::record(
        method: “Person::process”
        argument: 1,
        value: $name);

    TypeRecorder::record(
        method: “Person::process”
        argument: 2,
        value: $age);

    … rest of method …
  }
```

Is this still relevant?

Deterministic


Rector Recap
Enforce coding standards
Custom Rector Rule
Tombstones
Thoughts

- Upgrades
- One off Refactors
- Downgrades
- Automate code review
- Investigation work

Dave Liddament

github.com/DaveLiddament/php-language-extensions
github.com/DaveLiddament/phpstan-rule-test-helper
github.com/DaveLiddament/test-splitter
github.com/DaveLiddament/sarb

www.linkedin.com/in/daveliddament/
@daveliddament@phpc.social
@daveliddament.bsky.social
@daveliddament

Me when I’m not coding!

All Value Objects must be final and read only

Old

```
#[ValueObject]
class Person
{
}
```

New

```
#[ValueObject]
final readonly class Person
{
}
```

```
final class MakeValueObjectsFinalRector
     extends AbstractRector
{

public function getNodeTypes(): array
 {
   return [Class_::class];
 }
```

```
public function __construct(
  private PhpAttributeAnalyzer $attributeAnalyzer,
 ) {}
```

```php
public function refactor(Node $node): ?Node
{
    if (!$this->attributeAnalyze->hasPhpAttribute(
        $node, ValueObject::class)) {
        return null;
    }

    if ($node->isFinal() && ($node->isReadonly())) {
        return null;
    }

    $node->flags |= Modifiers::FINAL | Modifiers::READONLY;

    return $node;
}
```

```
public function refactor(Node $node): ?Node
{

if (!$this->attributeAnalyze->hasPhpAttribute(
     $node, ValueObject::class)) {
    return null;
 }

 if ($node->isFinal() && ($node->isReadonly())) {
   return null;
 }

 $node->flags |= Modifiers::FINAL | Modifiers::READONLY;

 return $node;
}
```

```
public function refactor(Node $node): ?Node
{

if (!$this->attributeAnalyze->hasPhpAttribute(
     $node, ValueObject::class)) {
    return null;
 }

 if ($node->isFinal() && ($node->isReadonly())) {
   return null;
 }

 $node->flags |= Modifiers::FINAL | Modifiers::READONLY;

 return $node;
}
```

Previously working code breaks

Before

```
#[ValueObject]
class Person {…}

class SomeClass extends Person
{…}
```

After

```
#[ValueObject]
final readonly class Person {…}

class SomeClass extends Person
{…}
```