Faites plus avec moins de code grâce aux #[Attributs] PHP

@GromNaN

SymfonyLive
PARIS 2024
MARCH 28-29, 2024

Jérôme Tamarelle

@GromNaN

MongoDB®

Symfony

SymfonyLive
PARIS 2024
MARCH 28-29, 2024

```
pecl install mongodb

composer require mongodb/mongodb

composer require doctrine/mongodb-odm-bundle
```

`#[Attribute]`

depuis PHP 8.0 – 26 Novembre 2020

SymfonyLive
PARIS 2024
MARCH 28-29, 2024

Syntaxe

```
<<Attribute>>

@@Attribute

#[Attribute]
```


SymfonyLive
PARIS 2024
MARCH 28-29, 2024

PHP 7.4

```
#[todo] comment
```

PHP 8.0

```
#[todo] comment
```


SymfonyLive
PARIS 2024
MARCH 28-29, 2024

Natifs

		dans Symfony
PHP 8.0	`#[Attribute]`	161
PHP 8.1	`#[ReturnTypeWillChange]`	154 en 5.4
	`#[SensitiveParameter]`	312
PHP 8.2	`#[AllowDynamicProperties]`	Zéro
PHP 8.3	`#[Override]`	
RFC	`#[NotSerializable]`	
	`#[Deprecated]`	

SymfonyLive
PARIS 2024
MARCH 28-29, 2024

```
composer require --dev jetbrains/phpstorm-attributes
```

RFC en cours

```
#[Jetbrains\PhpStorm\Deprecated(message: …)]
#[Jetbrains\PhpStorm\ArrayShape([…])]
#[Jetbrains\PhpStorm\ObjectShape([…])]
#[Jetbrains\PhpStorm\Immutable]
#[Jetbrains\PhpStorm\Pure]
#[Jetbrains\PhpStorm\ExpectedValues]
#[Jetbrains\PhpStorm\NoReturn]
#[Jetbrains\PhpStorm\Language('regex')]
```

Supplanté par :never

```php
#[ArrayShape([
    'id' => 'int',
    'name' => 'string',
    'foo' => App\PHP8\Foo::class,
])]
function user(): array{...}

$u = user();
$u;
```


SymfonyLive
PARIS 2024
MARCH 28-29, 2024

```php
namespace Symfony\Component\Scheduler\Attribute;

#[\Attribute]
class AsCronTask
{
    public function __construct(
        public readonly string $expression,
    ) {
    }
}
```

L'attribut qui définit une classe d'attribut

```php
use Symfony\Component\Scheduler\Attribute\AsCronTask;

#[AsCronTask('30 12 * * *')]
class SendDailyJokes
{
    public function __invoke()
    {
        // ...
    }
}
```


SymfonyLive
PARIS 2024
MARCH 28-29, 2024

```php
$reflection = new \ReflectionClass(SendDailyJokes::class);

foreach ($reflection->getAttributes() as $attribute) {
    $attribute->getName();      // 'AsCronTask'
    $attribute->getArguments(); // ['30 12 * * *']
    $attribute->newInstance();  // new AsCronTask('30 12 * * *')
}
```

```php
#[AsCronTask(expression: '30 12 * * *')]
class SendDailyJokes
{
    // ...
}
```

Symfony Backward Compatibility Promise

Parameter names are only covered by the compatibility promise for constructors of Attribute classes.

Using PHP named arguments for other classes might break your code when upgrading to newer Symfony versions.

```php
#[AsCronTask('30 12 * * *'), AllowDynamicProperties]


class SendDailyJokes
{
    public function __invoke()
    {
        // ...
    }
}
```


SymfonyLive
PARIS 2024
MARCH 28-29, 2024

```php
#[
  AsCronTask('30 12 * * *'),
  AllowDynamicProperties
]
class SendDailyJokes
{
    public function __invoke()
    {
        // ...
    }
}
```


SymfonyLive
PARIS 2024
MARCH 28-29, 2024

```php
#[AsCronTask('30 12 * * *')]
#[AllowDynamicProperties]

class SendDailyJokes
{
    public function __invoke()
    {
        // ...
    }
}
```


SymfonyLive
PARIS 2024
MARCH 28-29, 2024

PER Coding Style 2.0

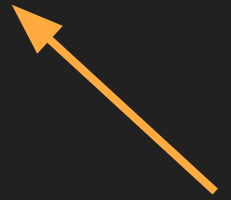
https://www.php-fig.org/per/coding-style/

@GromNaN

SymfonyLive
PARIS 2024
MARCH 28-29, 2024

```php
$reflection = new \ReflectionClass(SendDailyJokes::class);

$reflection->getAttributes(AsCronTask::class);
```



Filtre les attributs avec ce nom de classe

```php
namespace Symfony\Component\Scheduler\Attribute;

#[Attribute]
class AsNoonTask extends AsCronTask
{

    public function __construct()
    {
        parent::__construct('30 12 * * *');
    }

}
```

Requis pour les classes enfant

Partage de configuration

```php
namespace Symfony\Component\Scheduler\Attribute;

#[Attribute(Attribute::TARGET_CLASS)]
class AsNoonTask extends AsCronTask
{
    public function __construct()
    {
        parent::__construct('30 12 * * *');
    }

}
```

Attribut utilisable sur les classes uniquement

```php
use Symfony\Component\Scheduler\Attribute\AsCronTask;

#[AsNoonTask]
class SendDailyJokes
{
    public function __invoke()
    {
        // ...
    }
}
```


SymfonyLive
PARIS 2024
MARCH 28-29, 2024

```php
$reflection = new \ReflectionClass(SendDailyDadJokes::class);

$reflection->getAttributes(AsCronTask::class,
                \ReflectionAttribute::IS_INSTANCEOF);
```

Filtre sur toutes les classes qui dérivent de `AsCronTask`

```
#[Entity]
```

```
#[Document]
```

SymfonyLive
PARIS 2024
MARCH 28-29, 2024

@GromNaN

```php
use Doctrine\ORM\Mapping as ORM;

#[ORM\Entity]
#[ORM\Table(table: 'dad_joke')]
class Joke
{
    public function __construct(
        #[ORM\Id, ORM\GeneratedValue]
        public ?int $id = null,

        #[ORM\Column]
        public string $joke = '',
    ) {}
}
```

Configuration des propriétés

Type SQL déduit du typage natif

```php
use Doctrine\ODM\MongoDB\Mapping\Annotations as ODM;
use MongoDB\BSON\ObjectId;

#[ODM\Document(collection: 'dad_jokes')]
class Joke
{
    public function __construct(
        #[ODM\Id]
        public readonly string $id = new ObjectId(),

        #[ODM\Field]
        public string $joke = '',
    ) {}
}
```

Valeur par défaut avec `new` (PHP 8.1)

Identifiant généré côté client

MongoDB.

SymfonyLive PARIS 2024
MARCH 28-29, 2024

```php
use Doctrine\ODM\MongoDB\Types\Type;


#[ODM\Field(type: Type::DATE_IMMUTABLE)]
public \DateTimeImmutable $createdAt = new DatePoint(),


#[ODM\Field(type: Type::BINDATA, name: 'body')]
public string $joke = '',


#[ODM\Field]
public int $likes = 0,
```


SymfonyLive
PARIS 2024
MARCH 28-29, 2024

```php
use Symfony\Component\Validator\Constraints as Assert;

#[Assert\DateTime]

public \DateTimeImmutable $createdAt = new DatePoint(),

#[Assert\NotBlank]

public string $joke = '',

#[Assert\PositiveOrZero]

public int $likes = 0,
```


SymfonyLive
PARIS 2024
MARCH 28-29, 2024

```php
use Symfony\Component\Validator\Constraints as Assert;

#[Assert\DateTime]
#[ODM\Field(type: Type::DATE_IMMUTABLE)]
public \DateTimeImmutable $createdAt = new DatePoint(),

#[Assert\NotBlank]
#[ODM\Field(type: Type::BINDATA, name: 'body')]
public string $joke = '',

#[Assert\PositiveOrZero]
#[ODM\Field]
public int $likes = 0,
```

```php
#[ODM\Field(type: Type::COLLECTION)]
#[Assert\Collection(
    fields: [
        'url' => new Assert\Url(),
        'caption' => new Assert\NotBlank(),
    ]
)]
#[Assert\When(
    expression: 'this.public === true',
    constraints: [
        new Assert\Count(min: 1, max: 10),
    ],
)]
public array $photos = [],
```

Attributs imbriqués

Condition utilisant ExpressionLanguage

```
#[AsController]
```

```yaml
App\Controller\:
    resource: '../src/Controller/'
    public: true
    tags: [ 'controller.service_arguments' ]
    calls:
        - [ setContainer, [ '@abstract_controller.locator' ] ]

abstract_controller.locator:
    class: Symfony\Component\DependencyInjection\ServiceLocator
    arguments:
        - router: '@router'
          request_stack: '@request_stack'
          …
```

```yaml
App\Controller\:
    resource: '../src/Controller/'
    public: true
    tags: [ 'controller.service_arguments' ]
    autowire: true
    autoconfigure: true
```


SymfonyLive
PARIS 2024
MARCH 28-29, 2024


```php
#[AsController]
class HomeController
{
    #[Route('/', name: 'home')]
    #[Template('home/index.html.twig')]
    public function __invoke(): array
    {
        return [];
    }
}
```

```
#[Route]
```


SymfonyLive
PARIS 2024
MARCH 28-29, 2024

```
❯ bin/console debug:router
```

Name	Method	Scheme	Host	Path
_logout_main	ANY	ANY	ANY	/logout
ux_live_component	ANY	ANY	ANY	/{_locale}/_components/{_live_component}/{_live_action}
homepage	ANY	ANY	ANY	/{_locale}
admin_index	GET	ANY	ANY	/{_locale}/admin/post/
admin_post_index	GET	ANY	ANY	/{_locale}/admin/post/
admin_post_new	GET|POST	ANY	ANY	/{_locale}/admin/post/new
admin_post_show	GET	ANY	ANY	/{_locale}/admin/post/{id}
admin_post_edit	GET|POST	ANY	ANY	/{_locale}/admin/post/{id}/edit
admin_post_delete	POST	ANY	ANY	/{_locale}/admin/post/{id}/delete
blog_index	GET	ANY	ANY	/{_locale}/blog/
blog_rss	GET	ANY	ANY	/{_locale}/blog/rss.xml
blog_index_paginated	GET	ANY	ANY	/{_locale}/blog/page/{page}
blog_post	GET	ANY	ANY	/{_locale}/blog/posts/{slug}
comment_new	POST	ANY	ANY	/{_locale}/blog/comment/{postSlug}/new
blog_search	GET	ANY	ANY	/{_locale}/blog/search
security_login	ANY	ANY	ANY	/{_locale}/login
user_edit	GET|POST	ANY	ANY	/{_locale}/profile/edit
user_change_password	GET|POST	ANY	ANY	/{_locale}/profile/change-password

SensioFrameworkExtraBundle c'est fini !!!

`@Route`	`Symfony\Component\Routing\Attribute\Route`
`@Template`	`Symfony\Bridge\Twig\Attribute\Template`
`@Cache`	`Symfony\Component\HttpKernel\Attribute\Cache`
`@IsGranted`	`Symfony\Component\Security\Http\Attribute\IsGranted`
`@ParamConverter`	`Symfony\Component\HttpKernel\Attribute\ValueResolver`

```
#[ValueResolver]
```

SymfonyLive
PARIS 2024
MARCH 28-29, 2024

@GromNaN

```
#[AsDecorator]
```

SymfonyLive
PARIS 2024
MARCH 28-29, 2024

@GromNaN

```php
#[AsDecorator(decorates: 'router')]
class DecoratedRouter implements RouterInterface
{
    public function __construct(
        private RouterInterface $router,
        private LoggerInterface $logger,
    ) {}

    public function generate(string $name, array $parameters)
    {
        $this->logger->info('Generate {name}', ['name' => $name]);

        return $this->router->generate($name, $parameters);
    }
```

Service décoré

```
#[AsEventListener]
```

SymfonyLive
PARIS 2024
MARCH 28-29, 2024

@GromNaN

```php
class AppListener implements EventSubscriberInterface
{
    public static function getSubscribedEvents(): array
    {
        // ???
    }

    public function onRequest(RequestEvent $event): void
    {
        // ...
    }
}
```

```php
class AppListener implements EventSubscriberInterface
{
    public static function getSubscribedEvents(): array
    {
        return [
            KernelEvents::REQUEST => ['onRequest', 10],
        ];
    }

    public function onRequest(RequestEvent $event): void
    {
        // ...
    }
}
```

```php
class AppListener
{


    #[AsEventListener(KernelEvents::REQUEST, 10)]


    public function onRequest(RequestEvent $event): void
    {
        // ...
    }
}
```


SymfonyLive
PARIS 2024
MARCH 28-29, 2024

```php
class AppListener
{

    #[AsEventListener(priority: 10)]

    public function onRequest(RequestEvent $event): void
    {
        // ...
    }
}
```

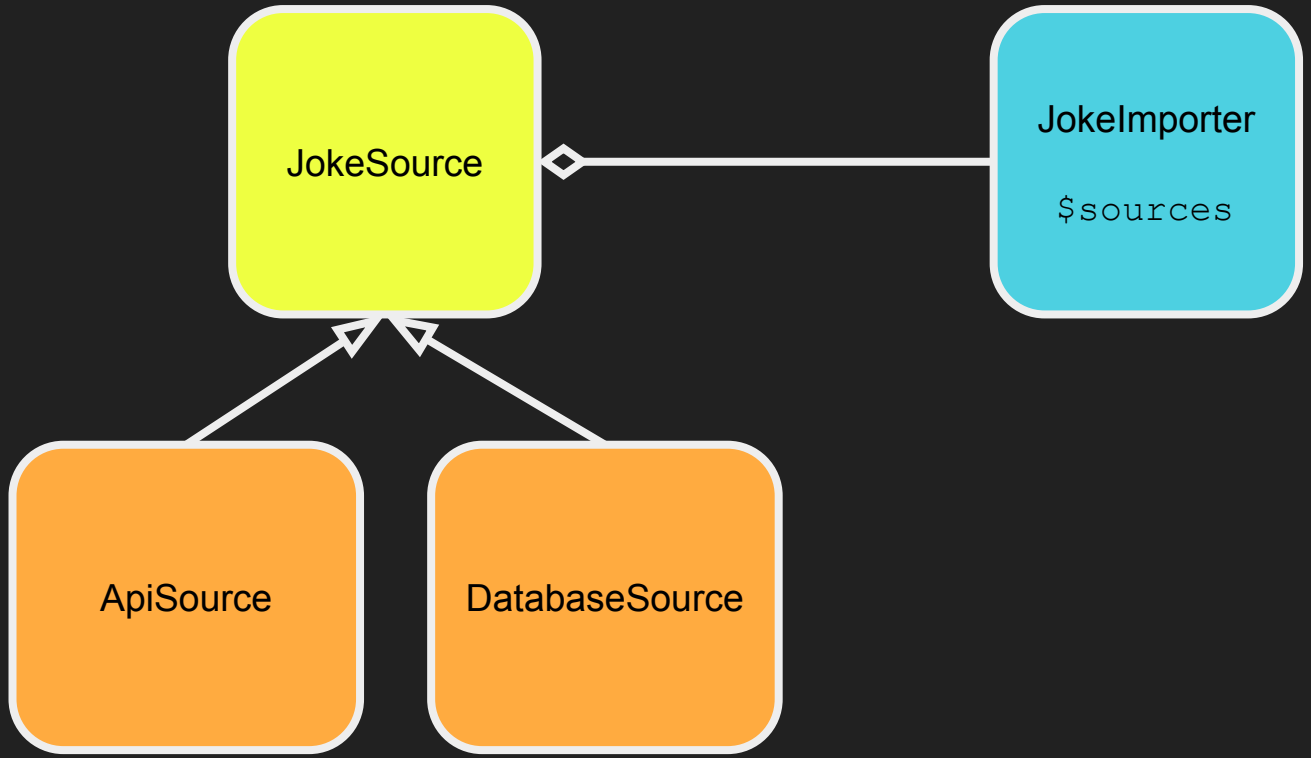
Named Argument pour omettre `$event`

Permet de déduire le nom de l'événement écouté

#[Autoconfigure]

SymfonyLive
PARIS 2024
MARCH 28-29, 2024

@GromNaN


JokeSource
JokeImporter
$sources
ApiSource
DatabaseSource

```php
interface JokeSource
{

    public function getJokes(): array;

}
```

```php
class ApiJokeSource implements JokeSource
{
    public function getJokes(): array
    {
        return [/* ... */];
    }
}
```


SymfonyLive
PARIS 2024
MARCH 28-29, 2024

```php
class JokeImporter
{
    /** @param iterable<JokeSource> $sources */
    public function __construct(
        private iterable $sources
    ) {}

    public function import()
    {
        foreach ($this->sources as $source) {
            $jokes = $source->getJokes();
        }
    }
}
```


SymfonyLive
PARIS 2024
MARCH 28-29, 2024

```php
#[Autoconfigure(tags: [['app.joke_source' => []]])]
interface JokeSource
{

    public function getJokes(): array;
}
```


SymfonyLive
PARIS 2024
MARCH 28-29, 2024

```php
#[AutoconfigureTag('app.joke_source')]
interface JokeSource
{

    public function getJokes(): array;
}
```

Etend et spécialise

```php
#[Autoconfigure]
```

```php
#[AutoconfigureTag(self::TAG)]
interface JokeSource
{
    public const TAG = 'app.joke_source';

    public function getJokes(): array;
}
```


SymfonyLive
PARIS 2024
MARCH 28-29, 2024

```php
#[AutoconfigureTag(static::TAG)]
interface JokeSource
{
    public const TAG = 'app.joke_source';

    public function getJokes(): array;
}
```

Compile Error: **static::class** cannot be used for compile-time class name resolution

```php
#[AutoconfigureTag(self::class)]
interface JokeSource
{

    public function getJokes(): array;
}
```

Le nom de l'interface comme nom de tag

```php
#[AutoconfigureTag]
interface JokeSource
{

    public function getJokes(): array;
}
```

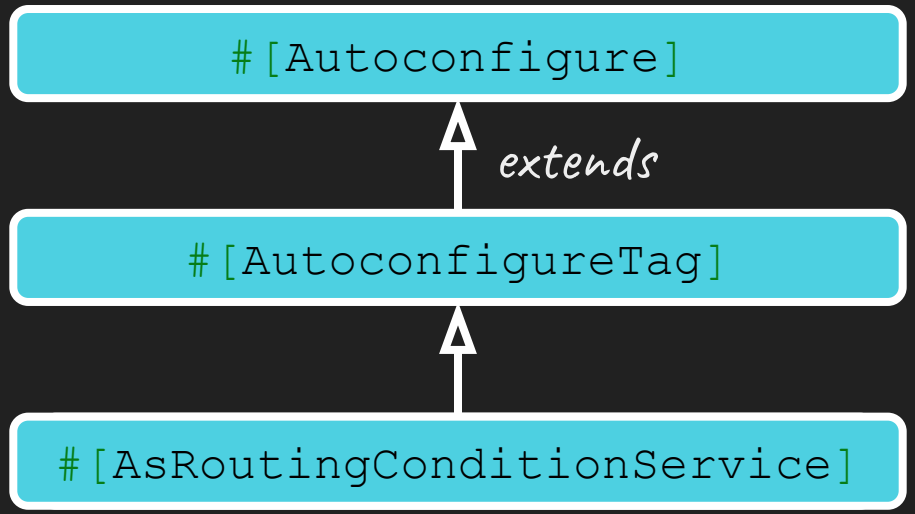

SymfonyLive
PARIS 2024
MARCH 28-29, 2024

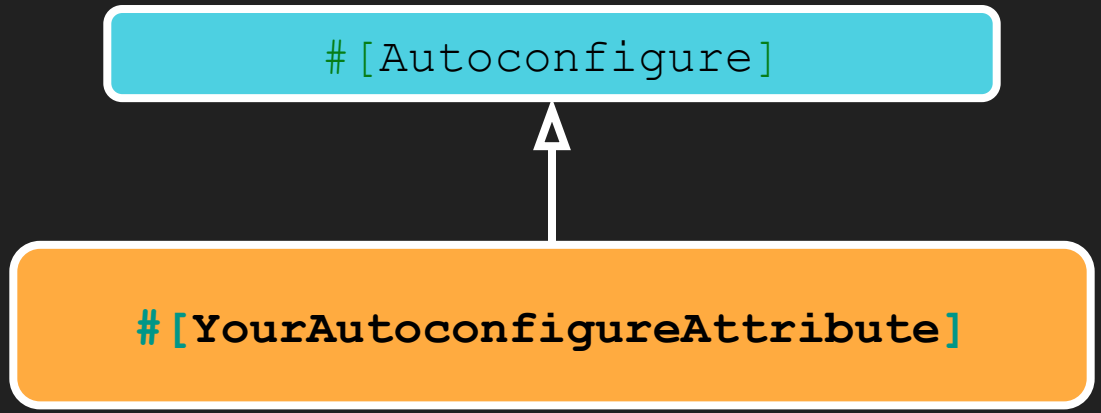
```php
class JokeImporter
{
    /** @param iterable<JokeSource> $sources */
    public function __construct(
        #[AutowireIterator(JokeSource::class)]
        private iterable $jokeSources,
    ) {}

    // ...
}
```

Pour maîtriser l'ordre dans l'itérateur

```php
#[AsTaggedItem(priority: 10)]
class ApiJokeSource implements JokeSource
{
    public function getJokes(): array
    {
        return [/* ... */];
    }
}
```

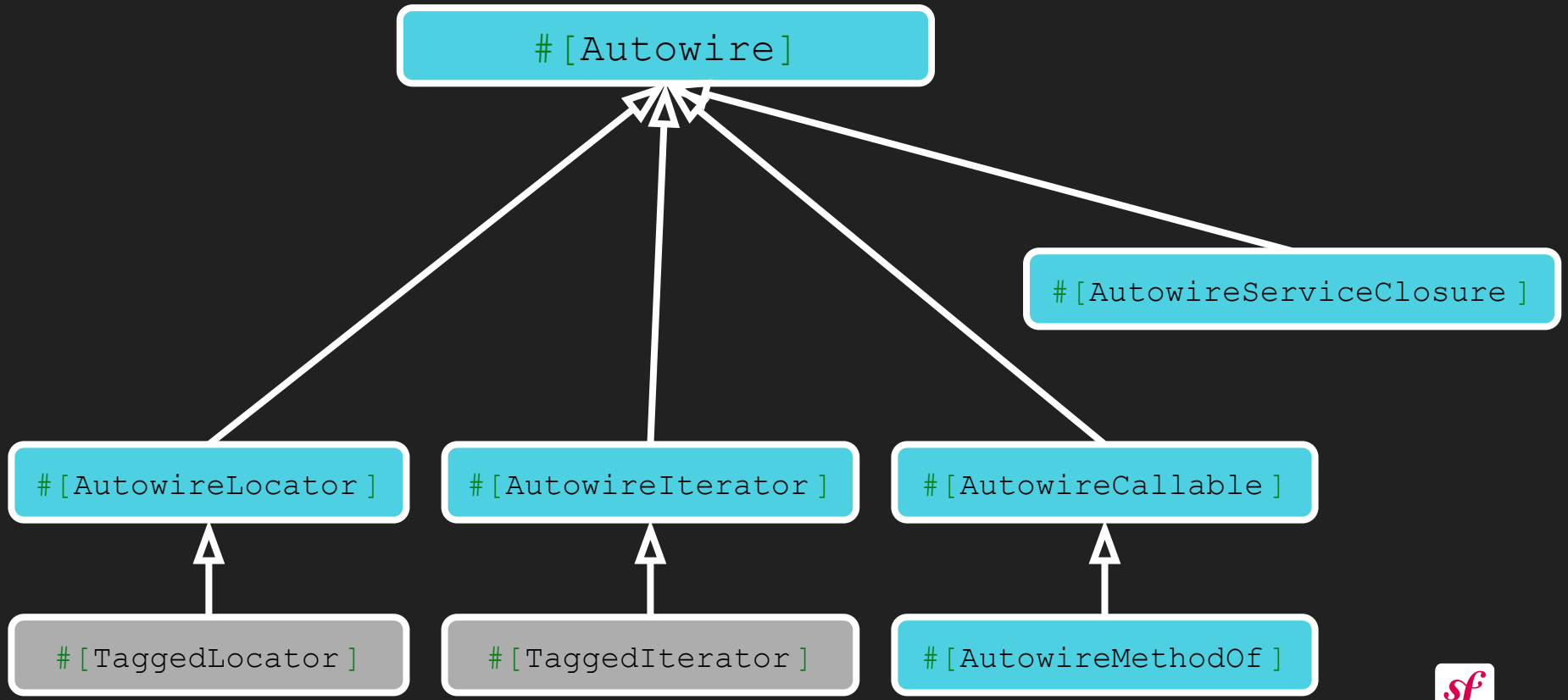

#[Autoconfigure]
extends
#[AutoconfigureTag]
#[AsRoutingConditionService]


#[Autoconfigure]
#[YourAutoconfigureAttribute]

```
#[Autowire]
```

SymfonyLive
PARIS 2024
MARCH 28-29, 2024

@GromNaN


#[Autowire]
#[AutowireServiceClosure]
#[AutowireLocator]
#[AutowireIterator]
#[AutowireCallable]
#[TaggedLocator]
#[TaggedIterator]
#[AutowireMethodOf]

```php
#[AsController]
class IndexController
{
    public function __construct(
        private JokeRepository $jokeRepository
    ) {}

    #[Route('/jokes/', name: 'joke_index')]
    public function __invoke(): array
    {
        return [
            'jokes' => $this->repository->findAll(),
        ];
    }
}
```


SymfonyLive
PARIS 2024
MARCH 28-29, 2024

```php
#[AsController]
class IndexController
{
    public function __construct(
        /** @see JokeRepository::findAll() */
        #[AutowireCallable(
            service: JokeRepository::class,
            method: 'findAll'
        )]
        private \Closure $findAllJokes,
    ) {}

    // ...
```


SymfonyLive
PARIS 2024
MARCH 28-29, 2024

```php
#[AsController]
class IndexController
{
    // ...

    #[Route('/jokes/', name: 'joke_index')]
    public function __invoke(): array
    {
        return [
            'jokes' => ($this->findAllJokes)(),
        ];
    }
}
```

Appel de la Closure

```php
#[AsController]
class IndexController
{
    public function __construct(
        /** @see JokeRepository::findAll() */
        #[AutowireMethodOf(
            service: JokeRepository::class
        )]
        private \Closure $findAll,
    ) {}

    // ...
```

Nom de la méthode à injecter

```php
#[AsController]
class IndexController
{
    #[Route('/jokes/')]
    #[Template(...)]
    public function __invoke(
        #[AutowireMethodOf(service: JokeRepository::class)]
        \Closure $findAll,
    ): array
    {
        return [
            'jokes' => ($findAll)(),
        ];
    }
}
```

Autowire en paramètre de contrôleur

```php
class IndexControllerTest extends TestCase
{
    public function testInvoke()
    {
        $findAll = fn () => ['joke1', 'joke2'];

        $controller = new IndexController();

        $this->assertSame(
            ['jokes' => ['joke1', 'joke2']],
            $controller($findAll)
        );
    }
}
```

Simple Closure de test

Invocation directe du contrôleur

```
#[Target]
```

SymfonyLive
PARIS 2024
MARCH 28-29, 2024

@GromNaN

```yaml
# config/packages/framework.yaml
framework:
    http_client:
        scoped_clients:
            githubApi:
                scope: 'https://api\.github\.com'
                headers:
                    Accept: 'application/vnd.github.v3+json'
                    # ...
```

```php
use Symfony\Component\DependencyInjection\Attribute\Target;
use Symfony\Contracts\HttpClient\HttpClientInterface;

class GitHubDownloader
{
    public function __construct(
        #[Target('githubApi')]
        private HttpClientInterface $httpClient
    ) {}
}
```

`#[Test]`

à partir de PHPUnit 10

@GromNaN

SymfonyLive
PARIS 2024
MARCH 28-29, 2024

```php
/**
 * @test
 * @group slow
 * @dataProvider method
 * @testWith [0, 1]
 *           [2, 3]
 * @testdox
 */
```

```php
#[Test]
#[Group('slow')]
#[DataProvider('method')]
#[TestWith([0, 1])]
#[TestWith([2, 3])]
#[Testdox]
```


SymfonyLive
PARIS 2024
MARCH 28-29, 2024

```php
/**
 * @testWith [0, 0, 0]
 *           [0, 1, 1]
 *           [-1, 0, 1]
 * @testdox Substracting $b from $a results is $expected
 * @test
 */
public function substract(int $expected, int $a, int $b)
{
    $this->assertSame($expected, $a - $b);
}
```

```php
#[
    TestWith([0, 0, 0]),
    TestWith([0, 1, 1]),
    TestWith([-1, 0, 1]),
    TestDox('Substracting $b from $a results is $expected'),
    Test
]
public function substract(int $expected, int $a, int $b)
{
    $this->assertSame($expected, $a - $b);
}
```

```php
#[TestWith([0, 0, 0])]
#[TestWith([0, 1, 1])]
#[TestWith([-1, 0, 1])]
#[TestDox('Substracting $b from $a results is $expected')]
#[Test]

public function substract(int $expected, int $a, int $b)
{
    $this->assertSame($expected, $a - $b);
}
```

```php
namespace PHPUnit\Framework\Attributes;

/**
 * @no-named-arguments Parameter names are not covered
 * by the backward compatibility promise for PHPUnit
 */
#[Attribute(Attribute::TARGET_CLASS|Attribute::TARGET_METHOD)]
final readonly class TestDox
{
    public function __construct(private string $text) {}
}
```


SymfonyLive
PARIS 2024
MARCH 28-29, 2024

```
/**
 * @before                    #[Before]
 * @after                     #[After]
 * @beforeClass               #[BeforeClass]
 * @afterClass                #[AfterClass]
 */
```


SymfonyLive
PARIS 2024
MARCH 28-29, 2024

```
#[AsTwigFunction]
```

SymfonyLive
PARIS 2024
MARCH 28-29, 2024

@GromNaN

```php
class AppExtension extends \Twig\Extension\AbstractExtension
{

    public function reverse(string $value): string
    {
        return strrev($value);
    }

    public function getFunctions(): array
    {
        return [
            new TwigFunction('reverse', $this->reverse(...)),
        ];
    }
}
```


SymfonyLive
PARIS 2024
MARCH 28-29, 2024

```php
class AppExtension
{
    #[AsTwigFunction('reverse')]
    public function reverse(string $value): string
    {
        return strrev($value);
    }
}
```


SymfonyLive
PARIS 2024
MARCH 28-29, 2024

```
composer require zenstruck/twig-service-bundle
```

Ce bundle vous est offert
par Kevin Bond (@kbond)

SymfonyLive
PARIS 2024
MARCH 28-29, 2024

Create attributes `AsTwigFilter`, `AsTwigFunction` and `AsTwigTest` to ease extension development #3916

Edit | Code

Open **GromNaN** wants to merge 16 commits into `twigphp:3.x` from `GromNaN:attribute`

Conversation 40 | Commits 16 | Checks 66 | Files changed 10 | +769 −1

GromNaN commented on Nov 26, 2023 • edited — Contributor

One drawback to writing extensions at present is that the declaration of functions/filters/tests is not directly adjacent to the methods. It's worse for runtime extensions because they need to be in 2 different classes. See `SerializerExtension` and `SerializerRuntime` as an example.

By using attributes for filters, functions and tests definition, we can make writing extensions more expressive, and use reflection to detect particular options (`needs_environment` , `needs_context` , `is_variadic`).

Example if we implemented the `serialize` filter:

Twig/extra/intl-extra/IntlExtension.php
Lines 392 to 395 in aeeec9a

```
392     public function formatDate(Environment $env, $date, ?string $dateFormat = 'medium', string
        $pattern = '', $timezone = null, string $calendar = 'gregorian', string $locale = null): string
393     {
394         return $this->formatDateTime($env, $date, $dateFormat, 'none', $pattern, $timezone, $calendar,
        $locale);
395     }
```

By using the `AsTwigFilter` attribute, it is not necessary to create the `getFilters()` method. The `needs_environment` option is detected from method signature. The name is still required as the method naming convention (camelCase) doesn't match with Twig naming convention (snake_case).

```
use Twig\Extension\Attribute\AsTwigFilter;
use Twig\Extension\Attribute\AsTwigExtension;

#[AsTwigExtension]
class IntlExtension
{
    #[AsTwigFilter(name: 'format_date')]
    public function formatDate(Environment $env, $date, ?string $dateFormat = 'medium', string $pattern = '',
    {
        return $this->formatDateTime($env, $date, $dateFormat, 'none', $pattern, $timezone, $calendar, $locale
    }
}
```

This approach does not totally replace the current definition of extensions, which is still necessary for advanced needs. It does, however, make for more pleasant reading and writing.

This makes writing lazy-loaded runtime extension the easiest way to create Twig extension in Symfony:
symfony/symfony#52748

Reviewers
- nicolas-grekas
- smnandre
- stof
- derrabus

Still in progress? Convert to draft

Assignees
No one assigned

Labels
None yet

Milestone
No milestone

Development
Successfully merging this pull request may close these issues.
None yet

Notifications — Customize
Unsubscribe
You're receiving notifications because you're watching this repository.

6 participants

Allow edits and access to secrets by maintainers

@GromNaN

SymfonyLive PARIS 2024
MARCH 28-29, 2024

Rénover

les `/** @annotation */`

en `#[Attribute]`

avec **rector**

SymfonyLive
PARIS 2024
MARCH 28-29, 2024

@GromNaN

Où trouver cette présentation

Jérôme Tamarelle

@GromNaN

SymfonyLive
PARIS 2024
MARCH 28-29, 2024