

SEPTEMBER 19 - 20, 2024 - LILLE, FRANCE & ONLINE

API PLATFORM CONFERENCE

Une documentation d'API
“aux petits oignons”

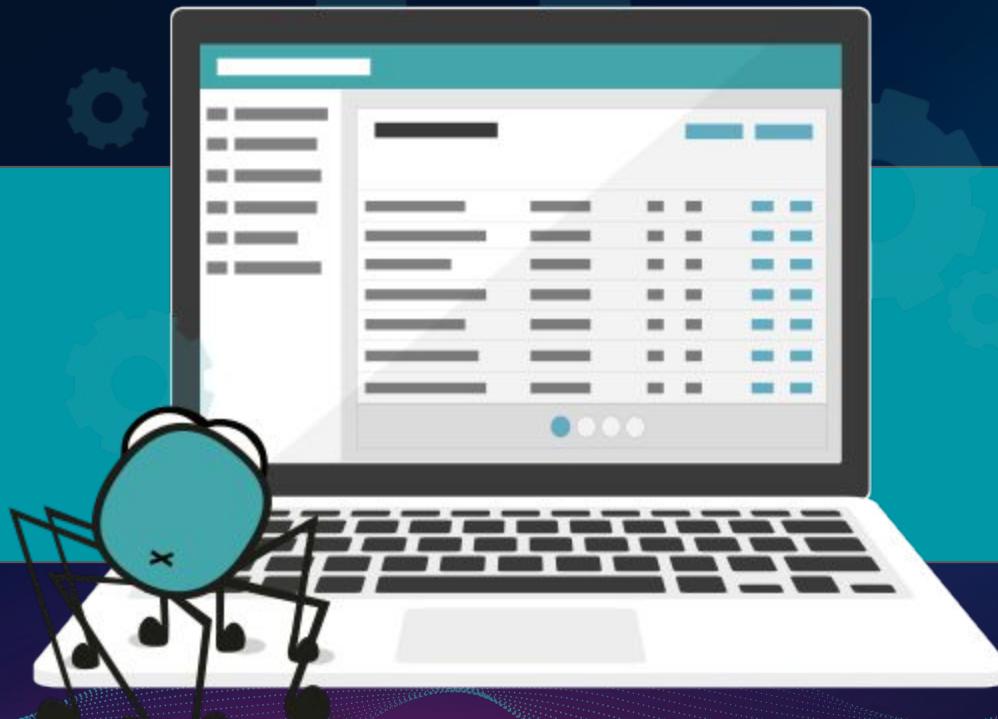
Les-Tilleuls.coop



contact@les-tilleuls.coop

- ✓ **Développement** : PHP, JavaScript, Go, Rust...
- ✓ DevOps & **SRE**
- ✓ Audit, **consulting**, et TMA
- ✓ Gestion de projet **Agile**, **UX/UI**

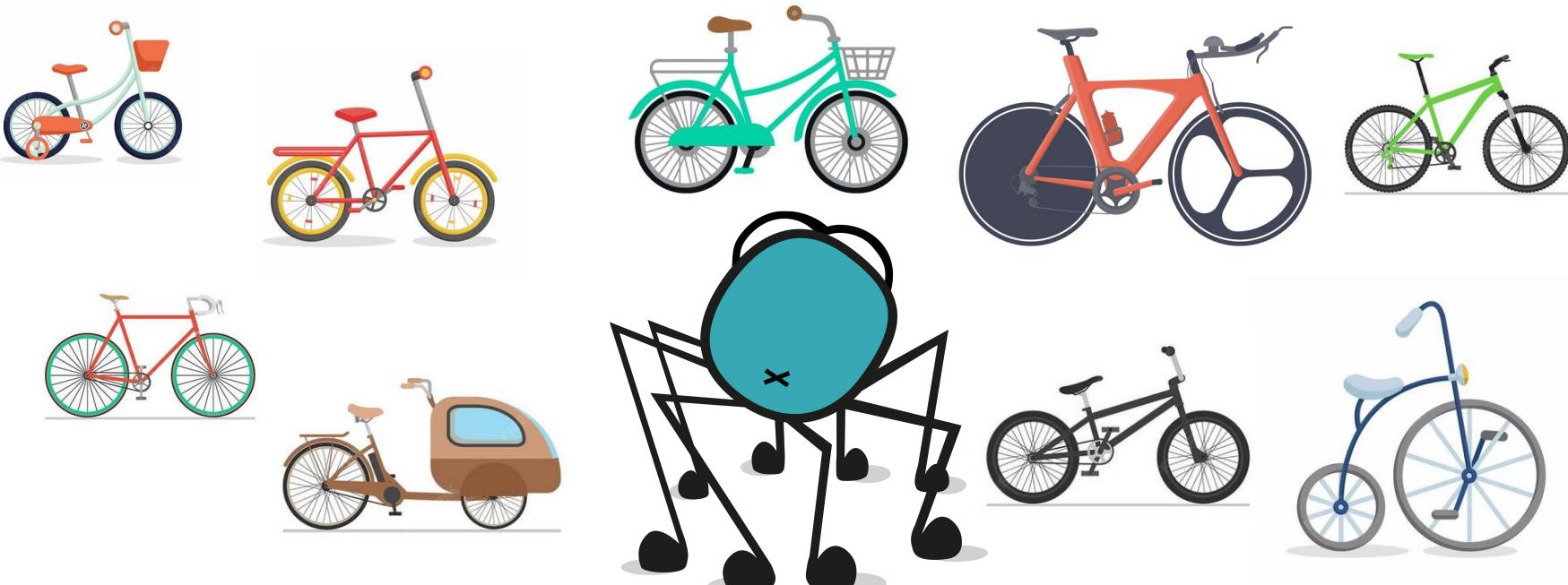
Documenter son API



API Platform c'est comme le vélo...



API Platform c'est comme le vélo...



Spécifier son API

“out of the box”

OpenAPI

```
{  
  "openapi": "3.1.0",  
  "info": {  
    "title": "API Title",  
    "version": "1.0.0"  
  },  
  "servers": [  
    {  
      "url": "https://api.example.com"  
    }  
  ],  
  "paths": {  
    "/example": {  
      "get": {}  
    }  
  },  
  "components": {  
    "schemas": {}  
  },  
  "security": [],  
  "tags": [],  
  "externalDocs": {}  
}
```



OpenAPI

```
{  
  "openapi": "3.1.0",  
  "info": {  
    "title": "API Title",  
    "version": "1.0.0"  
  },  
  "servers": [  
    {  
      "url": "https://api.example.com"  
    }  
  ],  
  "paths": {  
    "/example": {  
      "get": {}  
    }  
  },  
  "components": {  
    "schemas": {}  
  },  
  "security": [],  
  "tags": []  
}
```



```
$ php bin/console api:openapi:export
```



Swagger UI

Hello API Platform 1.0.0 OAS 3.1

Servers / Authorize

Bike

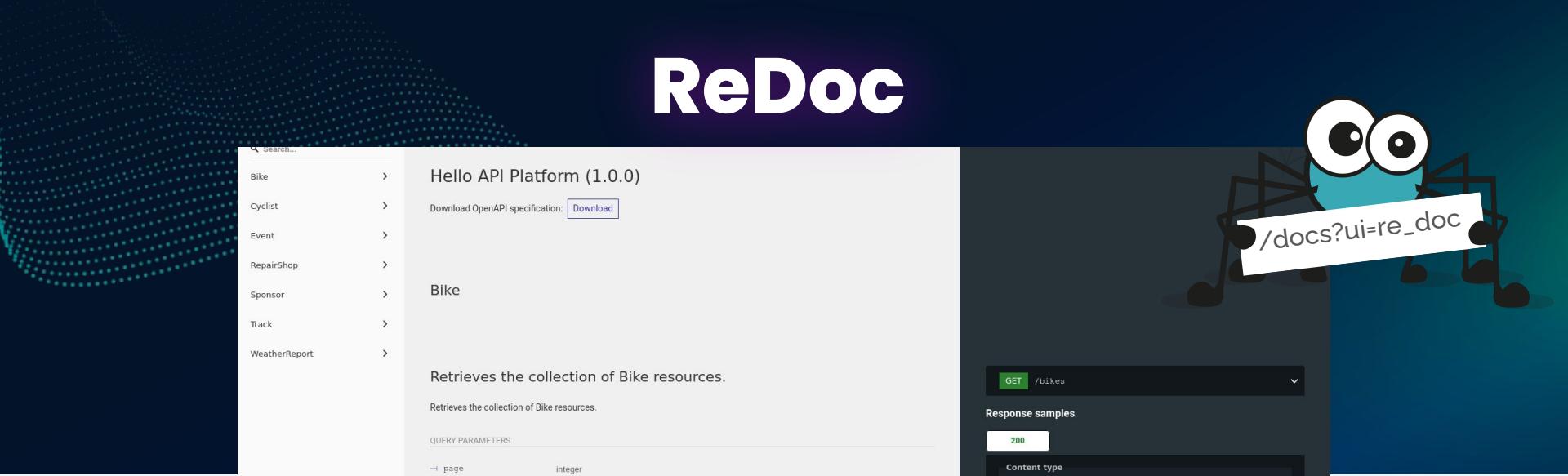
- GET /bikes Retrieves the collection of Bike resources.
- POST /bikes Creates a Bike resource.
- GET /bikes/{id} Retrieves a Bike resource.
- PUT /bikes/{id} Replaces the Bike resource.
- DELETE /bikes/{id} Removes the Bike resource.
- PATCH /bikes/{id} Updates the Bike resource.

Cyclist

- GET /cyclists Retrieves the collection of Cyclist resources.
- POST /cyclists Creates a Cyclist resource.
- GET /cyclists/{id} Retrieves a Cyclist resource.
- PUT /cyclists/{id} Replaces the Cyclist resource.
- DELETE /cyclists/{id} Removes the Cyclist resource.
- PATCH /cyclists/{id} Updates the Cyclist resource.



ReDoc



Search...

- Bike
- Cyclist
- Event
- RepairShop
- Sponsor
- Track
- WeatherReport

Hello API Platform (1.0.0)

Download OpenAPI specification: [Download](#)

Bike

Retrieves the collection of Bike resources.

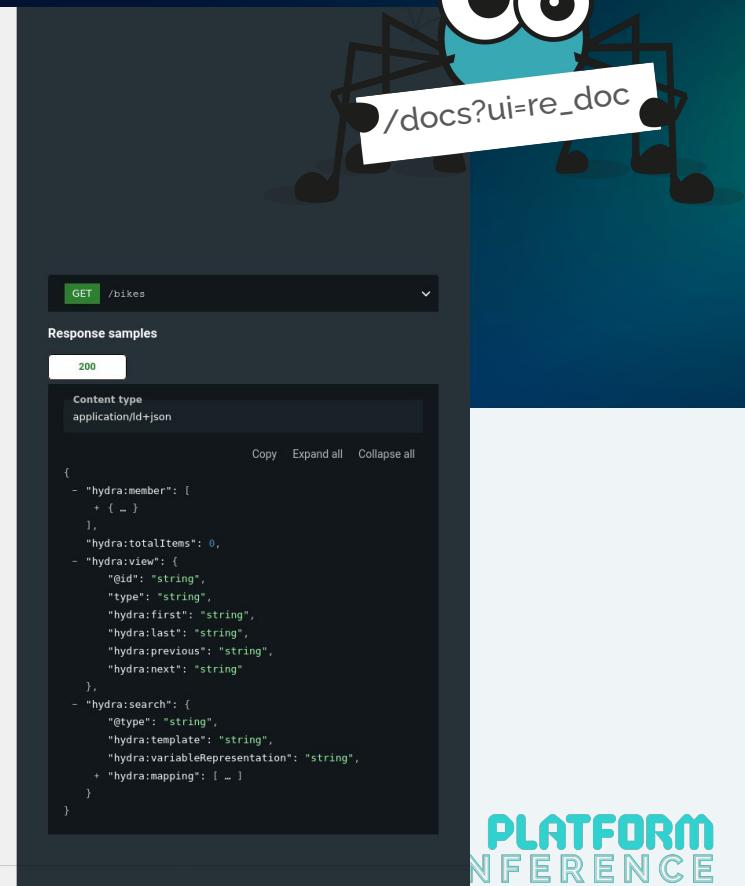
Retrieves the collection of Bike resources.

QUERY PARAMETERS

`-i page` integer Default: 1 The collection page number

Responses

`> 200` Bike collection



GET /bikes

Response samples

200

Content type application/d+json

```
{  
  - "hydra:member": [  
    + { _ }  
  ],  
  "hydra:totalItems": 0,  
  - "hydra:view": {  
    "@id": "string",  
    "type": "string",  
    "hydra:first": "string",  
    "hydra:last": "string",  
    "hydra:previous": "string",  
    "hydra:next": "string"  
  },  
  - "hydra:search": {  
    "type": "string",  
    "hydra:template": "string",  
    "hydra:variableRepresentation": "string",  
    + "hydra:mapping": [ _ ]  
  }  
}
```

Copy Expand all Collapse all

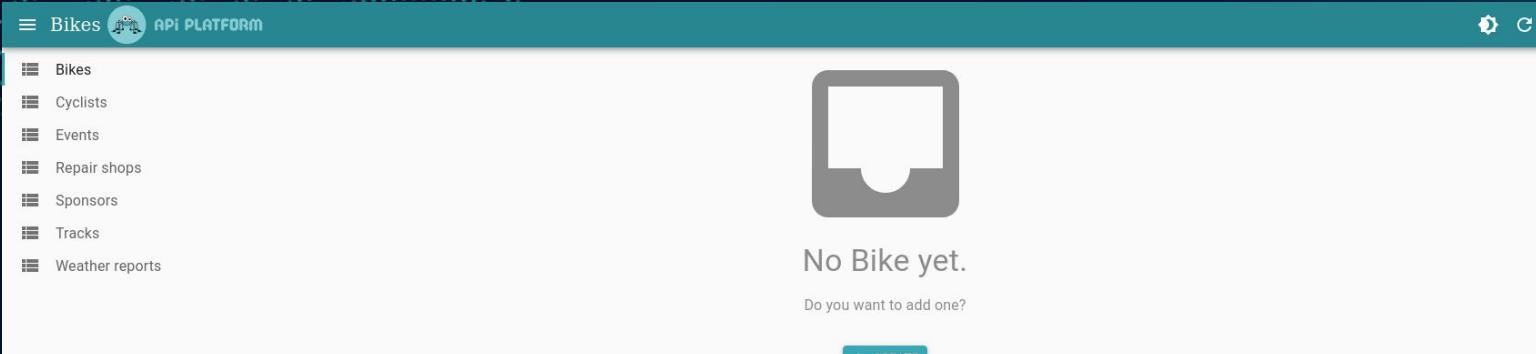
PLATFORM INFERENCE

Hydra

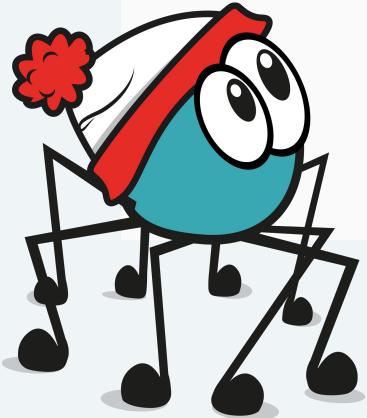
```
{  
    "@context": {  
        "@vocab": "https://localhost/docs.jsonld#",  
        "hydra": "http://www.w3.org/ns/hydra/core#",  
        "rdf": "http://www.w3.org/1999/02/22-rdf-syntax-ns#",  
        "rdfs": "http://www.w3.org/2000/01/rdf-schema#",  
        "xmls": "http://www.w3.org/2001/XMLSchema#",  
        "owl": "http://www.w3.org/2002/07/owl#",  
        "schema": "https://schema.org/",  
        ...  
    },  
    "@id": "/docs.jsonld",  
    "@type": "hydra:ApiDocumentation",  
    "hydra:title": "BikeHub API",  
    "hydra:description": "BikeHUB est une API qui regroupe une communauté de cyclistes...",  
    "hydra:entrypoint": "/",  
    "hydra:supportedClass": [  
        {  
            "@id": "#Bike",  
            "@type": "hydra:Class",  
            "hydra:title": "Bike",  
            "hydra:supportedProperty": [  
                {  
                    "@type": "hydra:SupportedProperty",  
                    "hydra:property": { "@id": "#Bike/brand" ...},  
                    "hydra:title": "brand",  
                    "hydra:required": false,  
                    "hydra:readable": true,  
                    "hydra:writeable": true  
                },  
            ],  
        }  
    ]  
}
```



API Platform Admin



The screenshot shows the API Platform Admin interface. At the top, there's a navigation bar with a sidebar icon, the text "Bikes", and a "API PLATFORM" logo. On the right of the bar are settings and refresh icons. The main content area has a sidebar on the left listing entities: Bikes, Cyclists, Events, Repair shops, Sponsors, Tracks, and Weather reports. The central area features a large square placeholder icon with a white outline and a central white shape, accompanied by the text "No Bike yet." and a smaller text "Do you want to add one?". At the bottom of this central area is a blue button labeled "+ CREATE".



“ API PLATFORM ADMIN: THE ULTIMATE ADMIN GENERATOR

”



FRANCOIS
ZANINOTTO

CEO

@ MARMELAB

@francoisz



Configurer sa doc



La config globale

```
# api/config/packages/api_platform.yaml
api_platform:
    title: BikeHub API
    description: 'BikeHub est une API qui gère une communauté de cyclistes'
    version: '1.0.0'

    openapi:
        contact:
            name: 'BikeHub'
            url: 'https://xxxxxx.com'
            email: 'contact@xxxxxx.com'
        termsOfService: '/cgv'
        license:
            name: 'xxx'
            url: 'https://xxxxxx.com'

    swagger_ui_extra_configuration:
        docExpansion: 'none'
        filter: false
```

Swagger UI - docExpansion

Bike	▼
Cyclist	▼
Event	▼
RepairShop	▼
Sponsor	▼
Track	▼
WeatherReport	▼

Swagger UI - Filter

Bike

Bike

^

POST	/bikes	Buy a new bike	🔒 ↴
GET	/bikes/{id}	Retrieves a Bike resource.	🔒 ↴
DELETE	/bikes/{id}	Removes the Bike resource.	🔒 ↴
PATCH	/bikes/{id}	Updates the Bike resource.	🔒 ↴

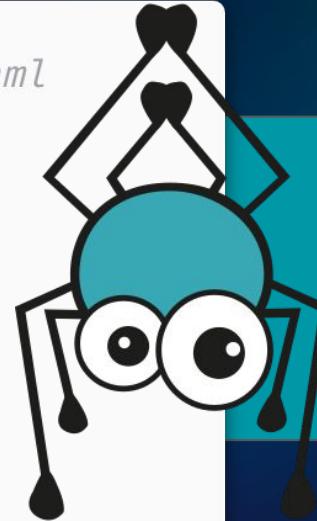
▼

Schemas

▼

Activer/désactiver les fonctionnalités

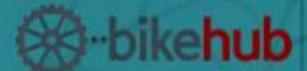
```
# api/config/packages/api_platform.yaml
api_platform:
    show_webby: true
    enable_swagger: true
    enable_swagger_ui: true
    enable_re_doc: true
    enable_entrypoint: true
    enable_docs: true
    asset_package: null
```



Surcharger le template UI

```
<!DOCTYPE html>
<html>
<head>
...
</head>

<body>
{% block header %}
<header>
  <a id="logo" href="/"></a>
</header>
{% endblock %}
...
<div id="swagger-ui" class="api-platform"></div>
...
</body>
</html>
```



Changer l'URL de l'UI



```
# app/config/routes.yaml
api_doc:
    path: /api_documentation
    controller: api_platform.swagger_ui.action
```

```
@pwa expression `(
    header({'Accept': '*text/html*'})
    && !path(
        '/docs*', '/graphql*', '/bundles*', '/contexts*', '/_profiler*', '/_wdt*',
        '/api_documentation*',
        '*.json*', '*.html', '*.csv', '*.yml', '*.yaml', '*.xml'
    )
)
|| path('/favicon.ico', '/manifest.json', '/robots.txt', '/sitemap*', '/_next*', '/__next*')
|| query({'_rsc': '*'})`
```

Configurer l'authentification

```
# api/config/packages/api_platform.yaml
api_platform:
    oauth:
        enabled: true
        clientId: 'abc'
        clientSecret: 'def'
        type: 'oauth2'
        flow: 'application'
        tokenUrl: '/oauth/v2/token'
        authorizationUrl: '/oauth/v2/auth'
        scopes:
            profile_read: "Read profile infos (firstname, lastname)"
            email: "Read email info"

    swagger:
        api_keys:
            ApiKeyAuth:
                name: 'API-KEY'
                type: 'header'
```



Configurer l'authentification

Available authorizations

oauth (OAuth2, clientCredentials)

Application: BikeHub API

OAuth 2.0 application Grant
Token URL: /oauth/v2/token
Flow: clientCredentials

client_id:

client_secret:

Scopes: [select all](#) [select none](#)

profile_read
Read profile infos (firstname, lastname)

email
Read email info

[Authorize](#) [Close](#)

ApiKeyAuth (apiKey)

Authorized

Value for the API-KEY header parameter.
Name: API-KEY
In: header
Value: *****

[Logout](#) [Close](#)

Responses

Curl

```
curl -X 'GET' \
  'https://localhost/bikes?page=1' \
  -H 'accept: application/ld+json' \
  -H 'API-KEY: helloworld'
```

Request URL

```
https://localhost/bikes?page=1
```

Configurer une ressource

```
use ApiPlatform\OpenApi\Model;  
  
#[ApiResource(  
    description: 'Magasin de vélo',  
    openapi: new Model\Operation(  
        tags: ['Services'],  
        summary: 'Magasin de vélo',  
        description: 'Ensemble de services autour du vélo : magasin, réparateur, conseils...',  
        externalDocs: new Model\ExternalDocumentation('CoopCycle', 'https://coopcycle.org/fr/'))  
)  
]  
class RepairShop
```

Services

GET	/repair_shops	Magasin de vélo
Ensemble de services autour du vélo : magasin, réparateur, conseils...		
Find more details		
CoopCycle		
https://coopcycle.org/fr/		

Configurer une opération

```
#[ApiResource]
#[Post(openapi: new Model\Operation(
    summary: 'Buy a new bike',
    description: "# New bike day!\n\n<img height='50px' src='https://api-platform.com/images/con/logo.svg' alt='New bike day' />",
    requestBody: new Model\RequestBody(
        content: new \ArrayObject([
            'application/json' => [
                'example' => [
                    'brand' => 'Origine Cycles',
                    'model' => 'Graxx III GTO',
                    'type' => 'Gravel',
                ]
            ]
        ])
    )
))]
```

Bike

POST /bikes Buy a new bike

New bike day!

API PLATFORM CONFERENCE

Parameters

No parameters

Request body

Example Value

```
{  
    "brand": "Origine Cycles",  
    "model": "Graxx III GTO",  
    "type": "Gravel"  
}
```

Configurer une opération

```
#[ApiResource(  
    operations: [  
        new Get(),  
        new GetCollection(openapi: false),  
        new Post(denormalizationContext: [  
            'groups' => 'bike:create',  
            'openapi_definition_name' => 'Buy'  
        ]),  
        // "$ref": "#/components/schemas/Bike.jsonld-Buy"  
    ]  
)]  
class Bike {}
```

Ajouter du contexte aux propriétés

```
#[ApiProperty(  
    openapiContext: [  
        'type' => 'string',  
        'example' => 'Origin Cycles',  
    ]  
)]  
public string $brand;  
  
#[ApiProperty(  
    openapiContext: [  
        'format' => 'email',  
        'example' => 'foo@example.com',  
    ]  
)]  
public string $email;
```

Décrire les paramètres

```
use ApiPlatform\Metadata\QueryParameter;
use ApiPlatform\OpenApi\Model\Parameter;

#[QueryParameter(
    key: 'q',
    schema: ['type' => 'string', 'enum' => ['gravel', 'route', 'vtt']],
    openApi: new Parameter(
        name: 'q',
        in: 'query',
        description: 'Filtrer par mot clef',
        allowEmptyValue: true,
        example: 'gravel',
    ),
    required: true
)]
class Bike
```

Décrire les paramètres

Parameters	
Name	Description
page <small>integer (query)</small>	The collection page number <i>Default value : 1</i> <input type="text" value="1"/>
q <small>string (query)</small>	Filtrer par mot clef <i>Available values : gravel, route, vtt</i> <i>Example : gravel</i> <input type="text" value="gravel"/> ▼

Aller plus loin



Les classes principales impliquées dans la génération



OpenAPI

OpenApiFactory

SchemaFactory

OpenApiNormalizer

DocumentationAction

ResourceNameCollectionFactory

ResourceClassResolver

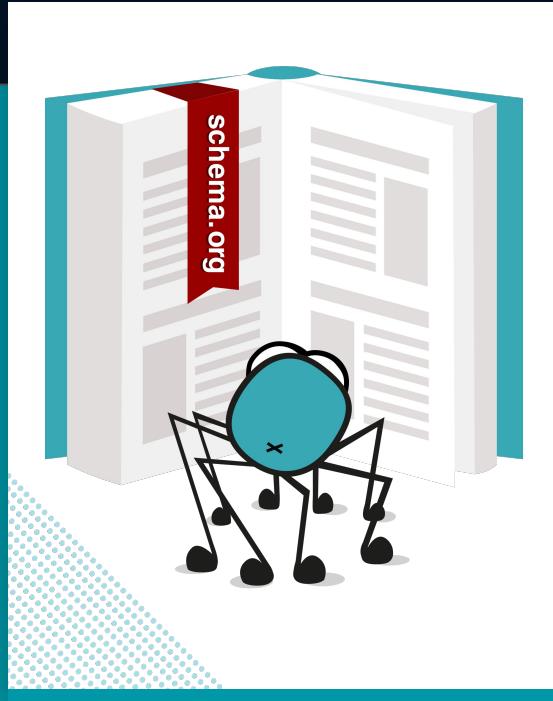
Hydra

DocumentationNormalizer

JsonSchemaFactory

EntrypointNormalizer

Comment est “devinée” la doc



- ✓ Nom de la classe
- ✓ Description des entités
- ✓ PHPDoc
- ✓ Types PHP
- ✓ Context OpenAPI
- ✓ Visibilité des propriétés
- ✓ Groupe du serializer du contexte de (dé)normalisation
- ✓ Description des filtres et paramètres
- ✓ Validator
- ✓ Doctrine ORM / MongoDB ODM

Que peut-on surcharger ?

- ✓ titre
- ✓ description
- ✓ liens
- ✓ liste des serveurs : base path URL
- ✓ filtrer
- ✓ ajout et modification de endpoints
- ✓ dépréciation de endpoints
- ✓ exemples
- ✓ schémas

Un peu **ce qu'on veut**



Décorer OpenApiFactory

```
#[AsDecorator('api_platform.openapi.factory')]
final class OpenApiFactory implements OpenApiFactoryInterface
{
    public function __construct(
        private readonly OpenApiFactoryInterface $decorated,
    ) {}

    public function __invoke(array $context = []): OpenApi
    {
        $openApi = $this->decorated->__invoke($context);

        $pathItem = $openApi->getPaths()->getPath('/bikes/{id}');
        $operation = $pathItem->getPut()->withDeprecated(true);
        $openApi
            ->getPaths()
            ->addPath('/bikes/{id}', $pathItem->withPut($operation));

        return $openApi;
    }
}
```

Déprécier une opération

```
#[AsDecorator('api_platform.openapi.factory')]
final class OpenApiFactory implements OpenApiFactoryInterface
{
    public function __construct(
        private readonly OpenApiFactoryInterface $decorated,
    ) {}

    public function __invoke(array $context = []): OpenApi
    {
        $openApi = $this->decorated->__invoke($context);

        $pathItem = $openApi->getPaths()->getPath('/bikes/{id}');
        $operation = $pathItem->getPut()->withDeprecated(true);
        $openApi
            ->getPaths()
            ->addPath('/bikes/{id}', $pathItem->withPut($operation));

        return $openApi;
    }
}
```

Déprécier une opération

```
#[AsDecorator('api_platform.openapi.factory')]
final class OpenApiFactory implements OpenApiFactoryInterface
{
    public function __construct(
        private readonly OpenApiFactoryInterface $decorated,
    ) {}

    public function __invoke(array $context = []): OpenApi
    {
        $openApi = $this->decorated->__invoke($context);

        $pathItem = $openApi->getPaths()->getPath('/bikes/{id}');
        $operation = $pathItem->getPut()->withDeprecated(true);
        $openApi
            ->getPaths()
            ->addPath('/bikes/{id}', $pathItem->withPut($operation));

        return $openApi;
    }
}
```

Déprécier une opération

Bike

GET /bikes Retrieves the collection of Bike resources.

POST /bikes Buy a new bike

GET /bikes/{id} Retrieves a Bike resource.

PUT /bikes/{id} Replaces the Bike resource.

DELETE /bikes/{id} Removes the Bike resource.

PATCH /bikes/{id} Updates the Bike resource.

Bike resources.

POST Buy a new bike

Replaces the Bike resource. **Deprecated**

GET Retrieves a Bike resource.

Replaces the Bike resource.

PUT Replaces the Bike resource.

AUTHORIZATIONS: >

api_key

```
{  
  "openapi": "3.1.0",  
  "paths": {  
    "/bikes": {},  
    "/bikes/{id)": {  
      "put": {  
        "deprecated": true  
      },  
    },  
  },  
}
```

Ajouter un endpoint custom

```
public function __invoke(array $context = []): OpenApi
{
    $openApi = $this->decorated->__invoke($context);
    $openApi->getPaths()->addPath(
        '/user_doc',
        new Model\PathItem(
            get: new Model\Operation(
                'GET',
                tags: ['Documentation'],
                description: 'Get the user documentation',
                parameters: [
                    new Model\Parameter('version', 'query', 'Get a specific version')
                ]
            ),
            ),
    );
    return $openApi;
}
```

Documentation

GET /user_doc

Get the user documentation

Parameters

Name	Description
version (query)	Get a specific version version

Ajouter des informations

```
public function __invoke(array $context = []): OpenApi
{
    $openApi = $this->decorated->__invoke($context);
    $description = sprintf('Nous sommes le %s', now()->format('d/m/Y'));

    return $openApi
        ->withInfo((new Model\Info(
            'BikeHub API',
            '1.0.0',
            $description,
            '/cgv',
            new Model>Contact('BikeHub', 'https://bikehub.org', 'contact@bikehub.org'),
            summary: 'BikeHub est une API qui permet de gérer une communauté de cyclistes,
            de parcours, de vélos et de compétitions'
        )));
}
```

BikeHub API 1.0.0 OAS 3.1

BikeHub est une API qui permet de gérer une communauté de cyclistes, de parcours, de vélos et de compétitions

Nous sommes le 13/09/2024

[BikeHub - Website](#)
[Send email to BikeHub](#)

Configurer des serveurs (avec variables ou sans)

```
public function __invoke(array $context = []): OpenApi
{
    $variables = [
        'subdomain' => [
            'default' => 'api',
            'description' => 'Subdomain of the API',
            'enum' => ['api', 'staging', 'dev'],
        ],
        'port' => [
            'default' => '443',
            'description' => 'Port to connect to',
            'enum' => ['443', '80'],
        ],
    ];

    return $openApi
        ->withServers([
            new Model\Server('https://localhost', 'Serveur de dev'),
            new Model\Server('https://bikehub.foo', 'Serveur Foo'),
            new Model\Server('https://bikehub.bar', 'Serveur Bar'),
            new Model\Server('https://{$subdomain}.example.com:{port}', 'Server with variables', new \ArrayObject($variables)),
        ]);
}
```

The screenshot shows two separate configuration panels for servers.

Top Panel: Shows a dropdown menu labeled "Servers" containing the option "https://localhost - Serveur de dev".

Bottom Panel: Shows a dropdown menu labeled "Servers" containing the option "https://{\$subdomain}.example.com:{port} - Server with variables". Below it, a "Computed URL" field displays "https://api.example.com:443".

Server variables:

- subdomain: api
- port: 443

Ajouter les méthodes d'authentification

```
$components = $openApi->getComponents();
$schemes = $components->getSecuritySchemes() ?? new \ArrayObject();

$schemes['basicAuth'] = new \ArrayObject([
    'type' => 'http',
    'scheme' => 'basic',
]);
$schemes['cookieAuth'] = new \ArrayObject([
    'type' => 'apiKey',
    'in' => 'cookie',
    'name' => 'PHPSESSID',
]);

$security = $openApi->getSecurity() ?? [];
$security[] = ['basicAuth' => []];
$security[] = ['cookieAuth' => []];

return $openApi
    ->withSecurity($security)
    ->withComponents($components->withSecuritySchemes($schemes));
```

Ajouter les méthodes d'authentification

```
$components = $openApi->getComponents();
$schemes = $components->getSecuritySchemes() ?? new \ArrayObject();

$schemes['basicAuth'] = new \ArrayObject([
    'type' => 'http',
    'scheme' => 'basic',
]);
$schemes['cookieAuth'] = new \ArrayObject([
    'type' => 'apiKey',
    'in' => 'cookie',
    'name' => 'PHPSESSID',
]);

$security = $openApi->getSecurity() ?? [];
$security[] = ['basicAuth' => []];
$security[] = ['cookieAuth' => []];

return $openApi
    ->withSecurity($security)
    ->withComponents($components->withSecuritySchemes($schemes));
```

```
#[ApiResource()  
openapi: new Model\Operation(security: [['cookieAuth' => []]]),  
security: "is_granted('ROLE_ADMIN')"  
)]  
class Bike
```

security: méthodes d'authentification globales, peuvent être surchargées par opération

components.securitySchemes : détails de chaque méthode

```
{  
  "paths": {  
    "/bikes": {  
      "get": {  
        "operationId": "api_bikes_get_collection",  
        "security": [  
          {  
            "cookieAuth": []  
          }  
        ]  
      },  
      "post": {  
        "operationId": "api_bikes_post_collection",  
        "parameters": [  
          {  
            "name": "bikes",  
            "in": "body",  
            "schema": {  
              "type": "array",  
              "items": {  
                "type": "object",  
                "properties": {  
                  "name": {  
                    "type": "string"  
                  },  
                  "color": {  
                    "type": "string"  
                  },  
                  "year": {  
                    "type": "integer",  
                    "format": "int32"  
                  }  
                }  
              }  
            }  
          }  
        ]  
      }  
    }  
  },  
  "components": {  
    "securitySchemes": {  
      "oauth": {},  
      "ApiKeyAuth": {  
        "type": "apiKey",  
        "description": "Value for the API-KEY header parameter.",  
        "name": "API-KEY",  
        "in": "header"  
      },  
      "basicAuth": {  
        "type": "http",  
        "scheme": "basic"  
      },  
      "cookieAuth": {  
        "type": "apikey",  
        "in": "cookie",  
        "name": "PHPSESSID"  
      }  
    },  
    "security": [  
      {  
        "oauth": []  
      },  
      {  
        "ApiKeyAuth": []  
      },  
      {  
        "basicAuth": []  
      },  
      {  
        "cookieAuth": []  
      }  
    ],  
  }  
}
```

Ajouter un modèle (JSON schema)

```
$schemas = $openApi->getComponents()->getSchemas() ?? new \ArrayObject();
$schemas['CyclistCustom'] = new \ArrayObject([
    'type' => 'object',
    'properties' => [
        'id' => ['type' => 'integer', 'example' => 1],
        'firstName' => ['type' => 'string', 'example' => 'John'],
        'lastName' => ['type' => 'string', 'example' => 'Doe'],
        'username' => ['type' => 'string', 'example' => 'Toto'],
    ],
]);
$components = $openApi->getComponents()->withSchemas($schemas);
$openApi = $openApi->withComponents($components);
```

```
$pathItem = $openApi->getPaths()->getPath('/cyclists/{id}');
$operation = $pathItem->getGet();
$responses = $operation->getResponses();
$responses['200'] = new Model\Response(
    description: 'Réponse custom',
    content: new \ArrayObject([
        'application/json' => [
            'schema' => [
                'oneOf' => [
                    ['name' => 'Ref1', '$ref' => '#/components/schemas/Cyclist.jsonld'],
                    ['$ref' => '#/components/schemas/CyclistCustom'],
                    ['type' => 'object', 'properties' => [...]],
                ]
            ],
            'examples' => []
        ]
    ])
);
$pathItem = $pathItem->withGet($operation->withResponses($responses));
$openApi->getPaths()->addPath('/cyclists/{id}', $pathItem);
```

```
$responses['200'] = new Model\Response(
    description: 'Réponse custom',
    content: new \ArrayObject([
        'application/json' => [
            'schema' => [],
            'examples' => [
                'user' => [
                    'summary' => 'Exemple pour un utilisateur régulier',
                    'value' => [
                        ['id' => 1, 'model' => 'Gravel'], ['id' => 2, 'model' => 'Road']
                    ]
                ],
                'admin' => [
                    'summary' => 'Exemple pour un admin',
                    'value' => [
                        ['id' => 1, 'model' => 'Gravel', 'availability' => 'available'],
                        ['id' => 2, 'model' => 'Road', 'availability' => 'in maintenance']
                    ]
                ],
            ],
        ]
    ]
);
$pathItem = $pathItem->withGet($operation->withResponses($responses));
$openApi->getPaths()->addPath('/cyclists/{id}', $pathItem);
```

Ajouter des exemples différents

Code	Description
200	Réponse custom Media type application/json Examples Exemple pour un utilisateur régulier Controls Accept header. Example Value Schema <pre>[{ "id": 1, "model": "Gravel" }, { "id": 2, "model": "Road" }]</pre> <p>Resource not found</p>

Code	Description
200	Réponse custom Media type application/json Examples Exemple pour un admin Controls Accept header. Example Value Schema <pre>[{ "id": 1, "model": "Gravel", "availability": "available" }, { "id": 2, "model": "Road", "availability": "in maintenance" }]</pre>

```
^ Collapse all (object | object | object)
One of ^ Collapse all (object | object | object)
#0 ^ Collapse all object
@context > Expand all read-only (string | object)
@id read-only string
@type read-only string
id read-only integer
firstName string
lastName string
email string
age integer | null
bikes > Expand all array<string>
tracks > Expand all array<string>
events > Expand all array<string>
#1 ^ Collapse all object
id > Expand all integer
firstName > Expand all string
lastName > Expand all string
username > Expand all string
#2 ^ Collapse all object
id integer
status string
```

Filtrer les ressources

```
final class FilteredResourceNameCollectionFactory implements ResourceNameCollectionFactoryInterface
{
    public function __construct(private readonly ResourceNameCollectionFactoryInterface $decorated, private readonly RequestStack $requestStack) {}

    private function getFilter(): ?string
    {
        $currentRequest = $this->requestStack->getCurrentRequest();
        if (!$currentRequest || 'api_doc' !== $currentRequest->attributes->get('_route')) {
            return null;
        }
        return $currentRequest->query->get('type');
    }

    public function create(): ResourceNameCollection
    {
        return match ($this->getFilter()) {
            'organizer' => new ResourceNameCollection([Event::class, Sponsor::class, WeatherReport::class]),
            'cyclist' => new ResourceNameCollection([Event::class, Cyclist::class, Bike::class, Track::class]),
            'provider' => new ResourceNameCollection([RepairShop::class, Bike::class]),
            default => $this->decorated->create(),
        };
    }
}
```

Filtrer les ressources

```
final class FilteredResourceNameCollectionFactory implements ResourceNameCollectionFactoryInterface
{
    public function __construct(private readonly ResourceNameCollectionInterface $decorated, private readonly RequestStack $requestStack) {}

    private function getFilter(): ?string
    {
        $currentRequest = $this->requestStack->getCurrentRequest();
        if (!$currentRequest || 'api_doc' !== $currentRequest->attributes->get('_route')) {
            return null;
        }
        return $currentRequest->query->get('type');
    }

    public function create(): ResourceNameCollection
    {
        return match ($this->getFilter()) {
            'organizer' => new ResourceNameCollection([Event::class, Sponsor::class, WeatherReport::class]),
            'cyclist' => new ResourceNameCollection([Event::class, Cyclist::class, Bike::class, Track::class]),
            'provider' => new ResourceNameCollection([RepairShop::class, Bike::class]),
            default => $this->decorated->create(),
        };
    }
}
```

Filtrer les ressources



Filter by tag

Bike

Cyclist

Event

Track

Filtrer les ressources par permission

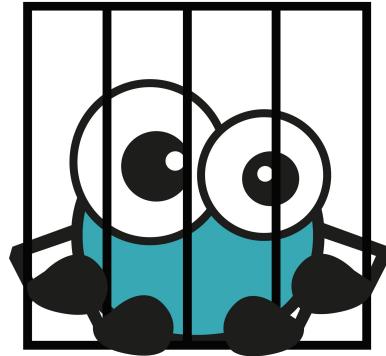
```
public function create(): ResourceNameCollection
{
    $filteredResources = [];
    $collection = $this->decorated->create();

    foreach ($collection as $resourceClass) {
        $reflection = new ReflectionClass($resourceClass);
        $apiResource = $reflection->getAttributes(ApiResource::class)[0] ?? null;

        if (
            null !== $this->resourceAccessChecker &&
            (null !== $security = $apiResource?->getArguments()['security'] ?? null) &&
            !$this->resourceAccessChecker->isGranted($resourceClass, (string) $security)
        ) {
            continue;
        }
        $filteredResources[] = $resourceClass;
    }

    return new ResourceNameCollection($filteredResources);
}
```

```
#[ApiResource(
    security: "is_granted('ROLE_ADMIN')",
)]
class Cyclist
```



Filtrer les ressources par permissions

```
public function normalize(mixed $object, /* .. */)
{
    $classes = [];
    $entrypointProperties = [];

    foreach ($object->getResourceNameCollection() as $resourceClass) {
        $resourceMetadataCollection = $this->resourceMetadataFactory->create($resourceClass);
        $resourceMetadata = $resourceMetadataCollection[0];
        /* .. */

        $security = $resourceMetadata->getSecurity();
        if ($security && !$this->resourceAccessChecker->isGranted($resourceClass, (string) $security)) {
            continue;
        }

        $this->populateEntrypointProperties($resourceMetadata, /* .. */);
        $classes[] = $this->getClass($resourceClass, $resourceMetadata, /* .. */);
    }

    return $this->computeDoc($object, $this->getClasses($entrypointProperties, $classes));
}
```

OpenAPI overlays

```
overlay: 1.0.0
info:
  title: Mise à jour de la spec OpenApi avec overlay
  version: 1.0.0
actions:
  - target: $.info
    update:
      title: Mise à jour du titre par overlay
      description: Nouvelle description
  - target: $.paths['/bikes'].get.responses['200'].content['application/json'].schema
    update:
      items:
        $ref: "#/components/schemas/BikeList"
  - target: $.paths['/bikes/{id}']
    description: Suppression de la route
    remove: true
  - target: $.paths[*][*].parameters[?(@.in=='query')]
    update:
      description: Mise à jour des paramètres de type query
      example: exemple_de_valeur
      required: true
```



Appliquer l'overlay



- ✓ Bump CLI [github.com/bump-sh/cli]

```
$ bump overlay openapi.yaml overlays.yaml > openapi.override.yaml
```

- ✓ Speakeasy CLI [[github.com/speakeasy-api/speakeeasy](https://github.com/speakeasy-api/speakeasy)]

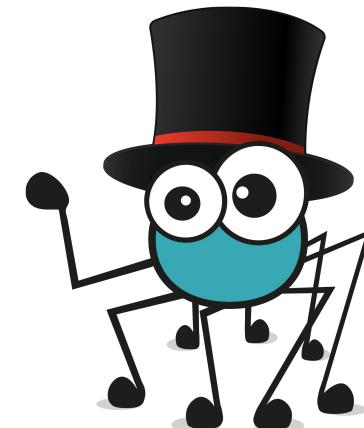
```
$ speakeasy overlay apply -s openapi.yaml -o overlays.yaml > openapi.override.yaml
```

- ✓ openapi-overlays-js [github.com/lornajane/openapi-overlays-js]

```
$ overlayjs --openapi openapi.yaml --overlay overlays.yaml
```

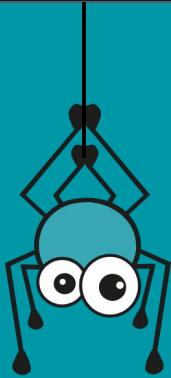
OpenAPI overlays

```
openapi: 3.1.0
info:
  title: 'Mise à jour du titre par overlay'
  description: "Nouvelle description"
  version: 1.0.0
paths:
  /bikes:
    get:
      operationId: api_bikes_get_collection
      responses:
        '200':
          content:
            application/json:
              schema:
                type: array
                items:
                  $ref: '#/components/schemas/BikeList'
      parameters:
        - name: page
          in: query
          description: 'Mise à jour des paramètres de type query'
          required: true
          example: exemple_de_valeur
```



Merci!

Des questions ?



RETROUVEZ LES-TILLEULS.COOP
SUR LES RÉSEAUX ET AU STAND

- @coopTilleuls
- les-tilleuls.coop

