入門編

PHPとAPI Platformで作る本格的なWeb APIアプリケーション

2024/12/22 PHPカンファレンス2024 @ttskch

#phpcon #track4

このトークのゴール

「API Platform、なんかよさそう」

と思ってもらえれば十分です！

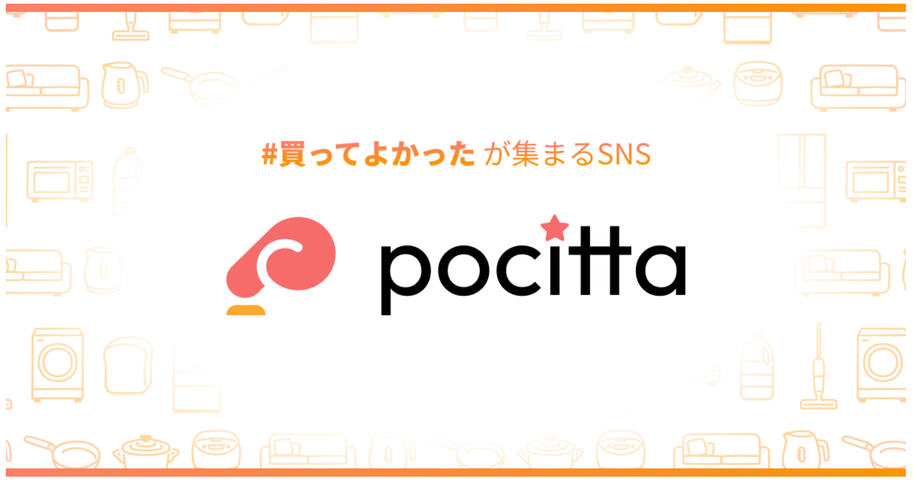
＼気楽に聞いてください／

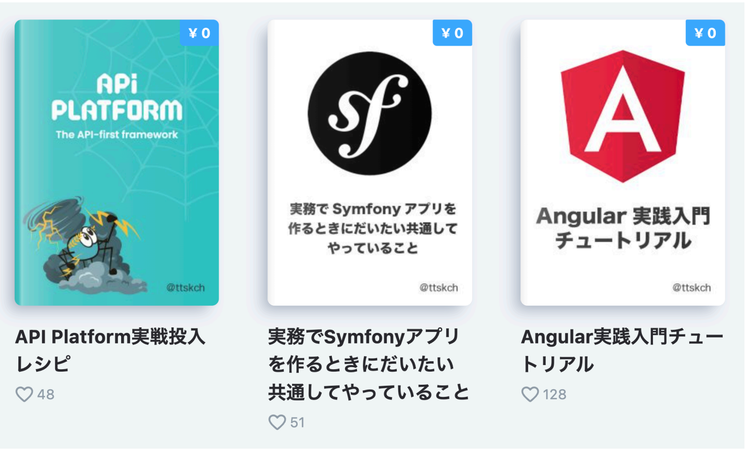
スライドは **#phpcon** で放流済みです

よかったら𝕏フォローしてください🤝✨

たつきち

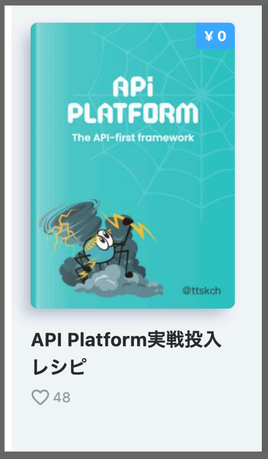
- @ttskch
- PHP歴13年、Symfony歴12年
- 株式会社Kannade代表
- 受託開発とか技術顧問とか
- 書き物も少々





たつきち

- @ttskch
- PHP歴13年、Symfony歴12年
- 株式会社Kannade代表
- 受託開発とか技術顧問とか
- 書き物も少々



API Platform実戦投入レシピ

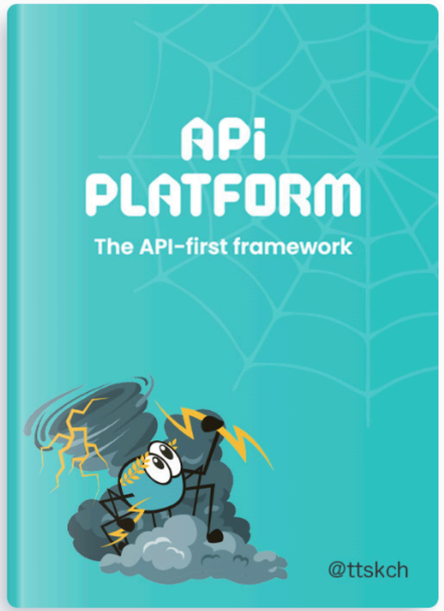
♡ 48

実務でSymfonyアプリを作るときにだいたい共通してやっていること

♡ 51

Angular実践入門チュートリアル

♡ 128



API Platform実戦投入レシピ

ttskch

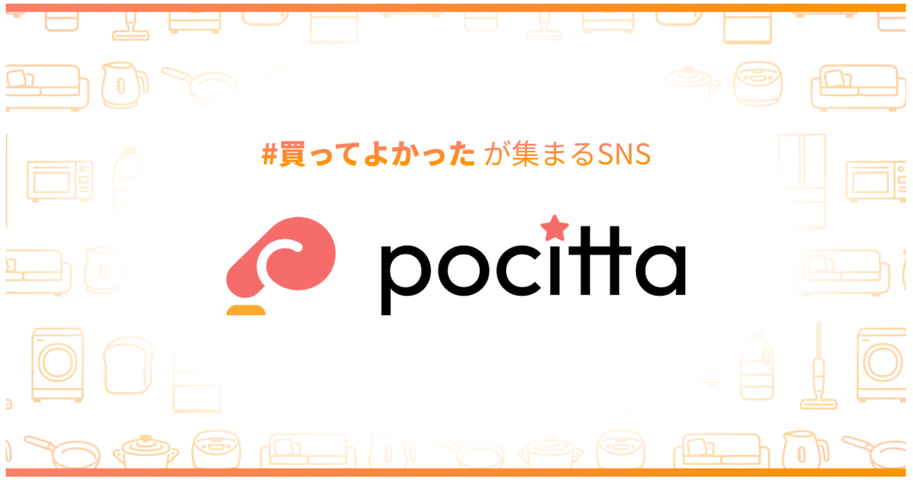
無料で読める本

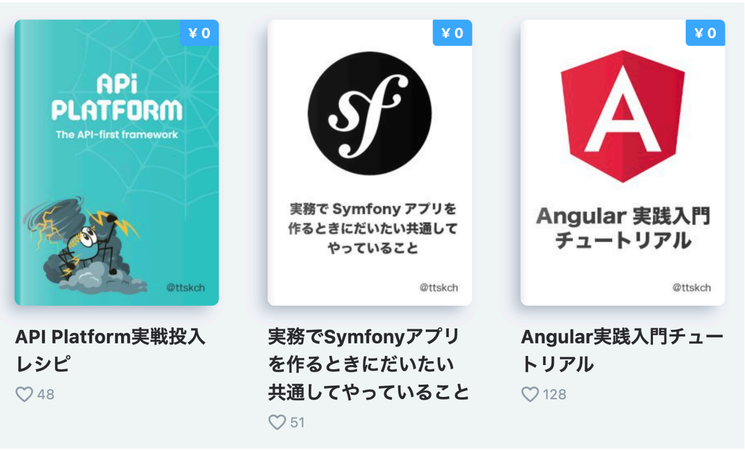
API Platform (https://api-platform.com/) は、Symfonyをベースとする PHP製のWeb APIフレームワークです。

簡単な設定を書くだけでSymfonyアプリにREST API（やGraphQL API）の機能を一瞬で追加することができて非常に便利なのですが、2023年9月現在、ある程度以上複雑なことをしようとすると途端にフレームワークについての深い理解が求められたり、痒いところに手が届かず強引なワークアラウンドが必要になったりするという面もあり、まだまだ日本国内でプロダクションに投入されている例を見聞きすることは少ない印象です。

たつきち

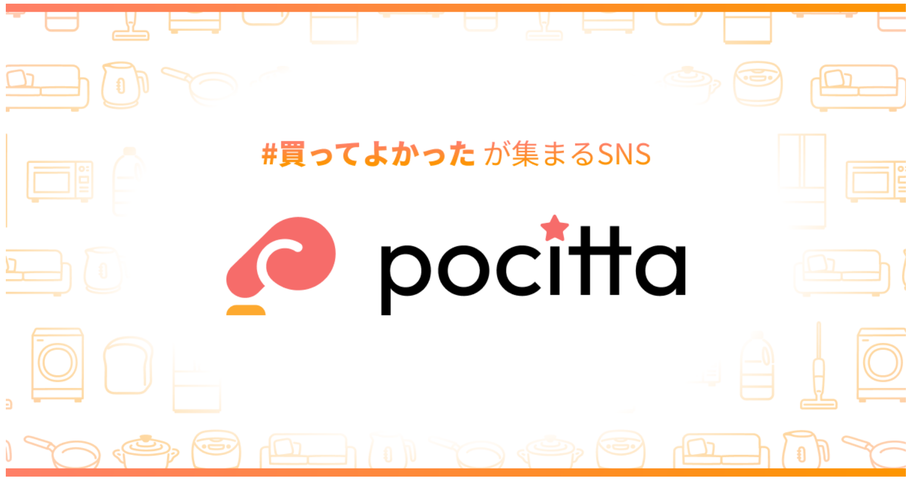
- @ttskch
- PHP歴13年、Symfony歴12年
- 株式会社Kannade代表
- 受託開発とか技術顧問とか
- 書き物も少々

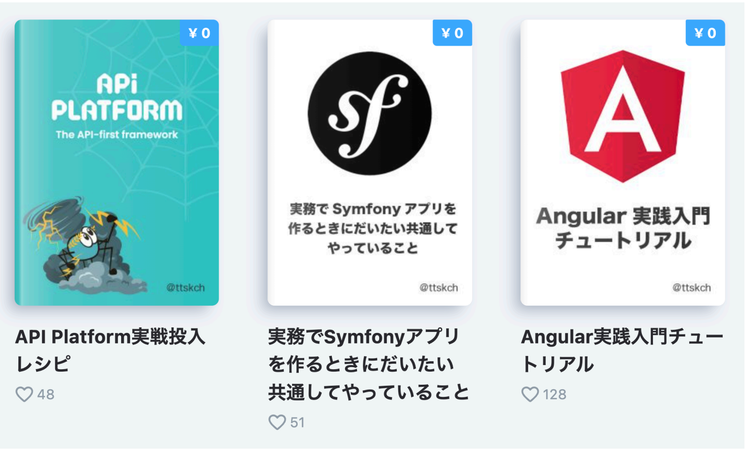




たつきち

- @ttskch
- PHP歴13年、Symfony歴12年
- 株式会社Kannade代表
- 受託開発とか技術顧問とか
- 書き物も少々
- **ペチコン名古屋2025実行委員長**





祝！初開催！

日本のど真ん中
名古屋にいこまい！

PHPCON NAGOYA 2025

2025年2月22日(土)

ウインクあいち 名古屋駅そば

参加申し込み

スポンサー応募

\ Follow us /

@phpcon_nagoya | #phpcon_nagoya

note

\ Share this /

Share this with

スポンサー様・一般参加者様

大募集中！です

入門編

PHPとAPI Platformで作る本格的なWeb APIアプリケーション

“API Platformは、あらゆるフレームワークおよび言語において最も先進的なAPIプラットフォームだ。

—Fabien Potencier (Symfonyの作者)”

https://api-platform.com/docs/symfony/

もくじ

- 自己紹介（2分）
- API Platformの全体像を把握する（8分）
- 動かしてみる（3分）
- データの読み書きをカスタマイズしてみる（3分）
- OpenAPIをカスタマイズしてみる（3分）
- まとめ（1分）
- デモ（5分）

デモ環境一式、置いときました

https://github.com/**ttskch/phpcon-2024-api-platform-demo**

たぶん今日は途中で時間切れになっちゃう予感がするので
ぜひ後ほどお手元で動かして体験してみてください🙏

📌もくじ

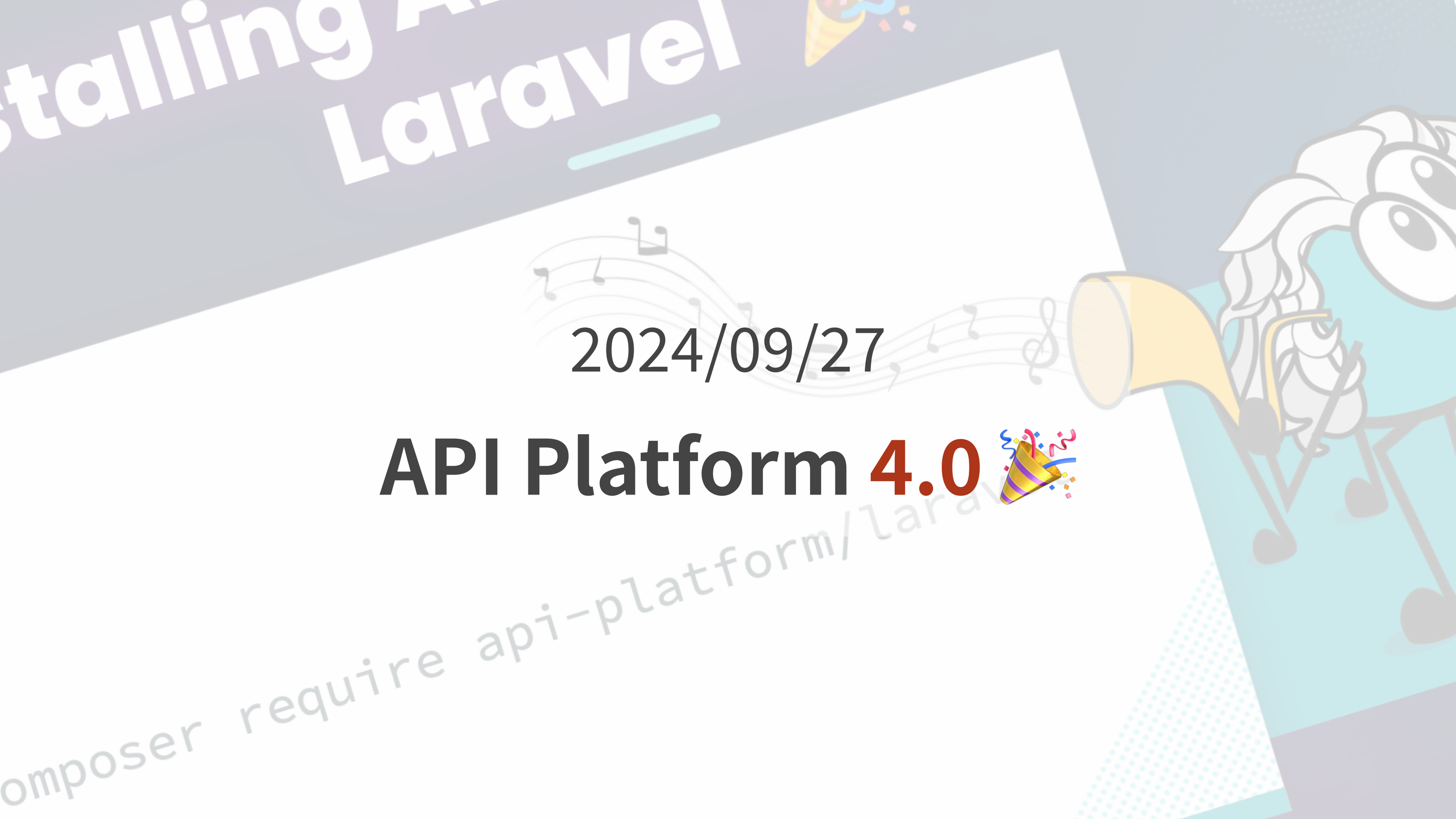
- 自己紹介（2分）
- **API Platformの全体像を把握する（8分）**
- 動かしてみる（3分）
- データの読み書きをカスタマイズしてみる（3分）
- OpenAPIをカスタマイズしてみる（3分）
- まとめ（1分）
- デモ（5分）

API Platformとは...

- **Web APIを開発**するためのPHP製の**フレームワーク**
- モデルに**アトリビュートを1行追記するだけ**でAPIとAPIドキュメントが作れる
- RESTとGraphQLに対応
- **Symfony**とシームレスに統合可能

API Platformとは...

- **Web APIを開発**するためのPHP製の**フレームワーク**
- モデルに**アトリビュートを1行追記するだけ**でAPIとAPIドキュメントが作れる
- RESTとGraphQLに対応
- ~~**Symfony**とシームレスに統合可能~~（2015/09〜2024/08）
- **SymfonyまたはLaravelとシームレスに統合可能**（2024/09〜）



2024/09/27

API Platform 4.0 🎉

API Platformとは...

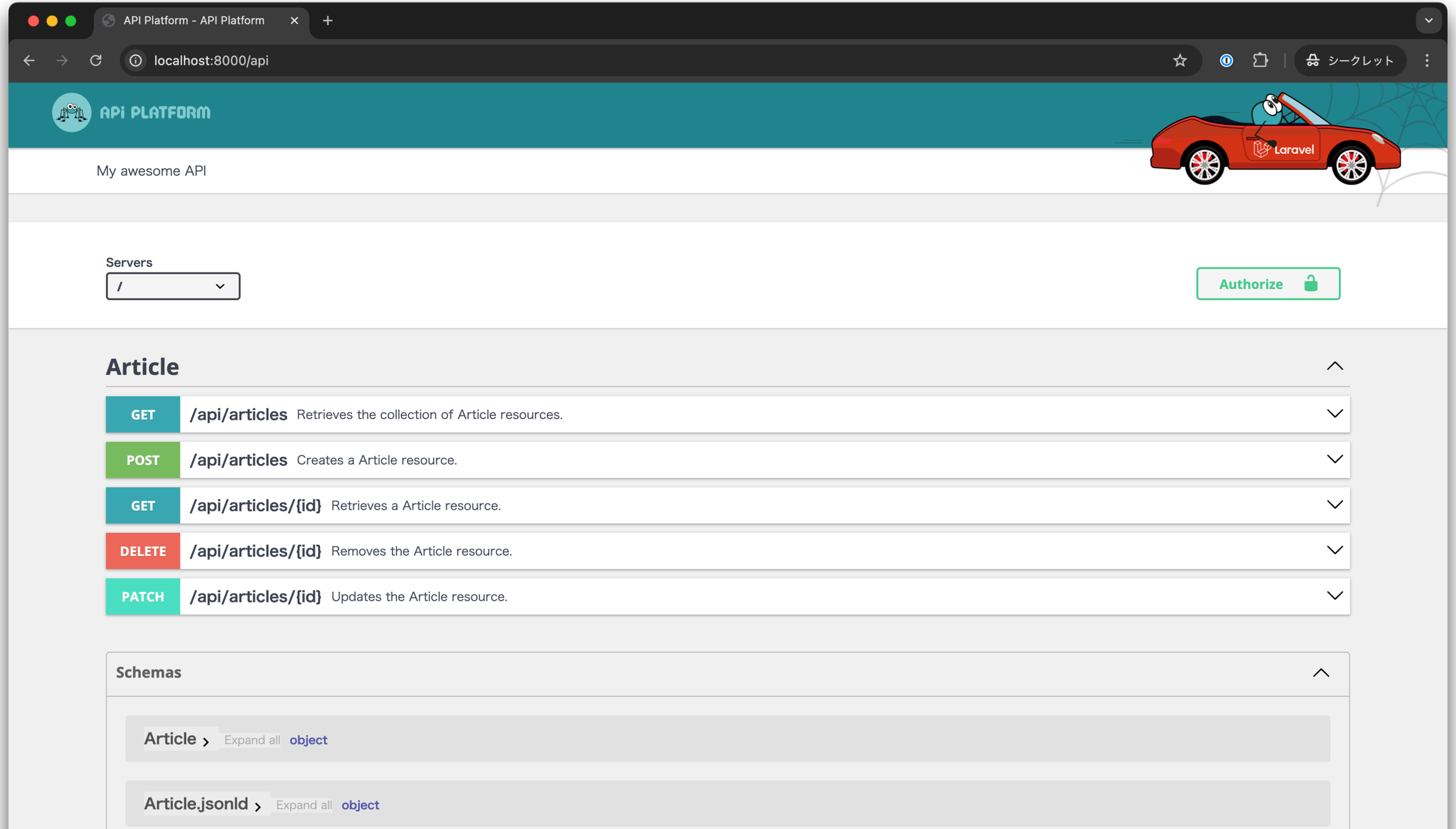
- **Web APIを開発**するためのPHP製の**フレームワーク**
- モデルに**アトリビュートを1行追記するだけ**でAPIとAPIドキュメントが作れる
- RESTとGraphQLに対応
- ~~**Symfony**とシームレスに統合可能~~（2015/09〜2024/08）
- **SymfonyまたはLaravelとシームレスに統合可能**（2024/09〜）

API Platformとは...

- **Web APIを開発**するためのPHP製の**フレームワーク**
- モデルに**アトリビュートを1行追記するだけ**でAPIとAPIドキュメントが作れる
- RESTとGraphQLに対応
- ~~**Symfony**とシームレスに統合可能~~（2015/09〜2024/08）
- **SymfonyまたはLaravelとシームレスに統合可能**（2024/09〜）

```php
<?php

namespace App\Models;


use Illuminate\Database\Eloquent\Model;


class Article extends Model
{
    //
}
```

```php
<?php

namespace App\Models;

use ApiPlatform\Metadata\ApiResource;
use Illuminate\Database\Eloquent\Model;

#[ApiResource]
class Article extends Model
{
    //
}
```


API Platform - API Platform
localhost:8000/api
シークレット
API PLATFORM
Laravel
My awesome API
Servers
/
Authorize
Article
GET /api/articles Retrieves the collection of Article resources.
POST /api/articles Creates a Article resource.
GET /api/articles/{id} Retrieves a Article resource.
DELETE /api/articles/{id} Removes the Article resource.
PATCH /api/articles/{id} Updates the Article resource.
Schemas
Article › Expand all object
Article.jsonld › Expand all object

使い始め方

- **Symfonyと併用で使う場合**
  - **公式テンプレート**でプロジェクトを開始
  - 既存のSymfonyアプリケーションに**API Platform Core**をインストール
- **Laravelと併用で使う場合**
  - 既存のLaravelアプリケーションに**API Platform Core**をインストール

フレームワークと併用しない場合：https://api-platform.com/docs/core/bootstrap/

公式テンプレート？API Platform Core？

- https://github.com/**api-platform/api-platform** ⭐ 8.7k
- https://github.com/**api-platform/core** ⭐ 2.5k
- https://github.com/**api-platform/admin** ⭐ 487
- https://github.com/**api-platform/create-client** ⭐ 374
- https://github.com/**api-platform/symfony** ⭐ 8
- https://github.com/**api-platform/laravel** ⭐ 49

⋮

公式テンプレート？API Platform Core？

- https://github.com/**api-platform/api-platform** ⭐ 8.7k
- https://github.com/**api-platform/core** ⭐ 2.5k
- https://github.com/**api-platform/admin** ⭐ 487
- https://github.com/**api-platform/create-client** ⭐ 374
- https://github.com/**api-platform/symfony** ⭐ 8
- https://github.com/**api-platform/laravel** ⭐ 49

⋮

API Platform Core

API Platformの本体

公式テンプレート？API Platform Core？

- https://github.com/**api-platform/api-platform** ⭐8.7k
- https://github.com/**api-platform/core** ⭐
- https://github.com/**api-platform/admin**
- https://github.com/**api-platform/create-client** ⭐374
- https://github.com/**api-platform/symfony** ⭐8
- https://github.com/**api-platform/laravel** ⭐49

⋮

API Platform Coreの**Subtree Split**

API Platform Coreの部分集合をパッケージ化したもの

公式テンプレート？API Platform Core？

- https://github.com/**api-platform/api-platform** ⭐ 8.7k
- https://github.com/**api-platform/core** ⭐ 2.5k
- https://github.com/**api-platform/admin** ⭐ 487
- https://github.com/**api-platform/create-client** ⭐ 374
- https://github.com/**api-platform/symfony** ⭐ 8
- https://github.com/**api-platform/laravel** ⭐ 49

⋮

周辺ツールなど

公式テンプレート？API Platform Core？

- https://github.com/**api-platform/api-platform** ⭐8.7k
- https://github.com/**api-platform/core** ⭐2.5k
- https://github.com/**api-platform/admin** ⭐
- https://github.com/**api-platform/create-cl**
- https://github.com/**api-platform/symfony** ⭐8
- https://github.com/**api-platform/laravel** ⭐49

⋮

公式テンプレート

Core（Symfony）、Admin、Create Clientなど全部入り

公式テンプレート？API Platform Core？

- https://github.com/**api-platform/api-platform** ⭐ 8.7k
- https://github.com/**api-platform/core** ⭐ 2.5k
- https://github.com/**api-platform/admin** ⭐ 487
- https://github.com/**api-platform/create-client** ⭐ 374
- https://github.com/**api-platform/symfony** ⭐ 8
- https://github.com/**api-platform/laravel** ⭐ 49

⋮

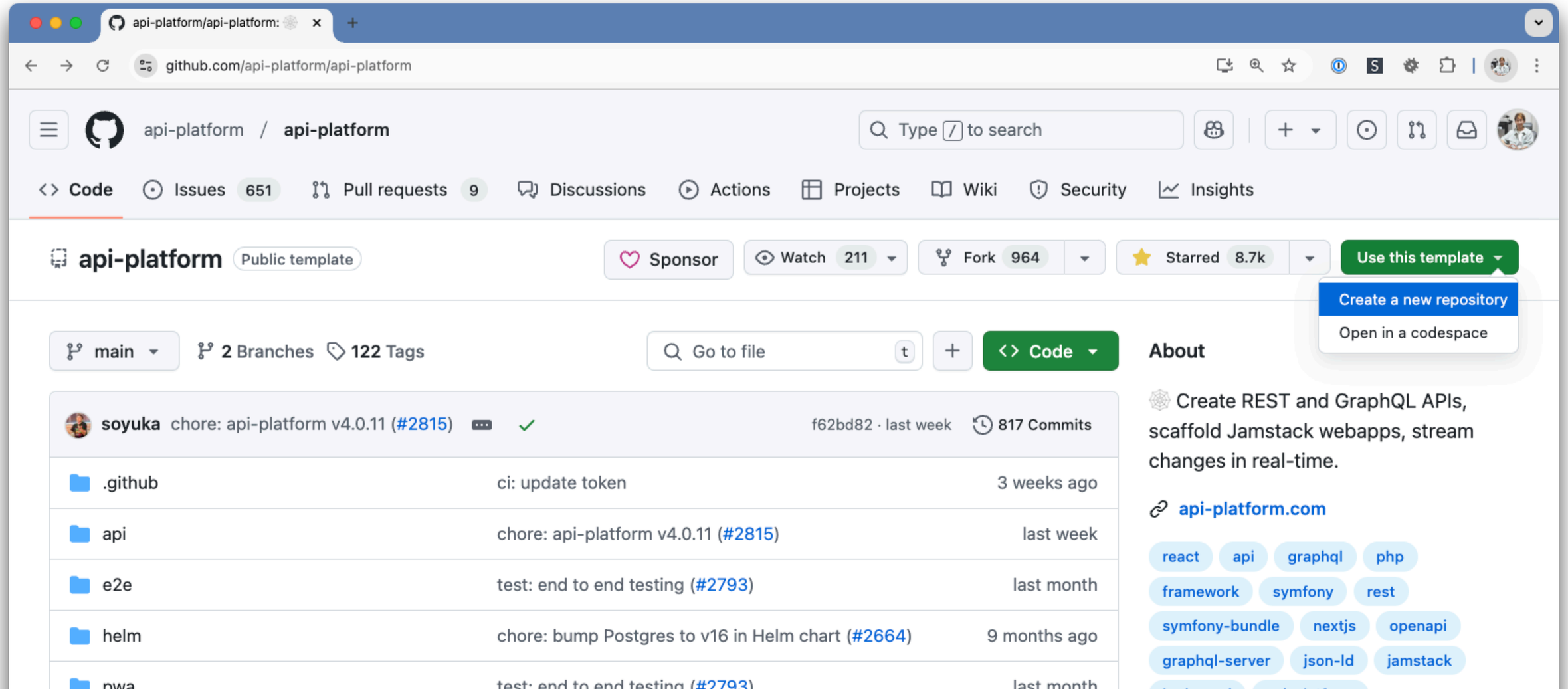
公式テンプレート？API Platform Core？

- http
- http
- http
- http
- http
- https://github.com/**api-platform/laravel** ⭐49

⋮

この辺の全体像が分かっていると
ドキュメントやIssueを当たるときに
迷子にならなくて済みます🤦‍♂️

テンプレート is 何？


api-platform/api-platform:
github.com/api-platform/api-platform
api-platform / api-platform
Type / to search
Code
Issues 651
Pull requests 9
Discussions
Actions
Projects
Wiki
Security
Insights
api-platform
Public template
Sponsor
Watch 211
Fork 964
Starred 8.7k
Use this template
Create a new repository
Open in a codespace
main
2 Branches
122 Tags
Go to file
t
Code
About
soyuka chore: api-platform v4.0.11 (#2815)
f62bd82 · last week
817 Commits
.github
ci: update token
3 weeks ago
api
chore: api-platform v4.0.11 (#2815)
last week
e2e
test: end to end testing (#2793)
last month
helm
chore: bump Postgres to v16 in Helm chart (#2664)
9 months ago
Create REST and GraphQL APIs, scaffold Jamstack webapps, stream changes in real-time.
api-platform.com
react
api
graphql
php
framework
symfony
rest
symfony-bundle
nextjs
openapi
graphql-server
json-ld
jamstack

テンプレート is 何？

必要なものが色々入ってる状態で
リポジトリを作れる

「色々入ってる」の中身

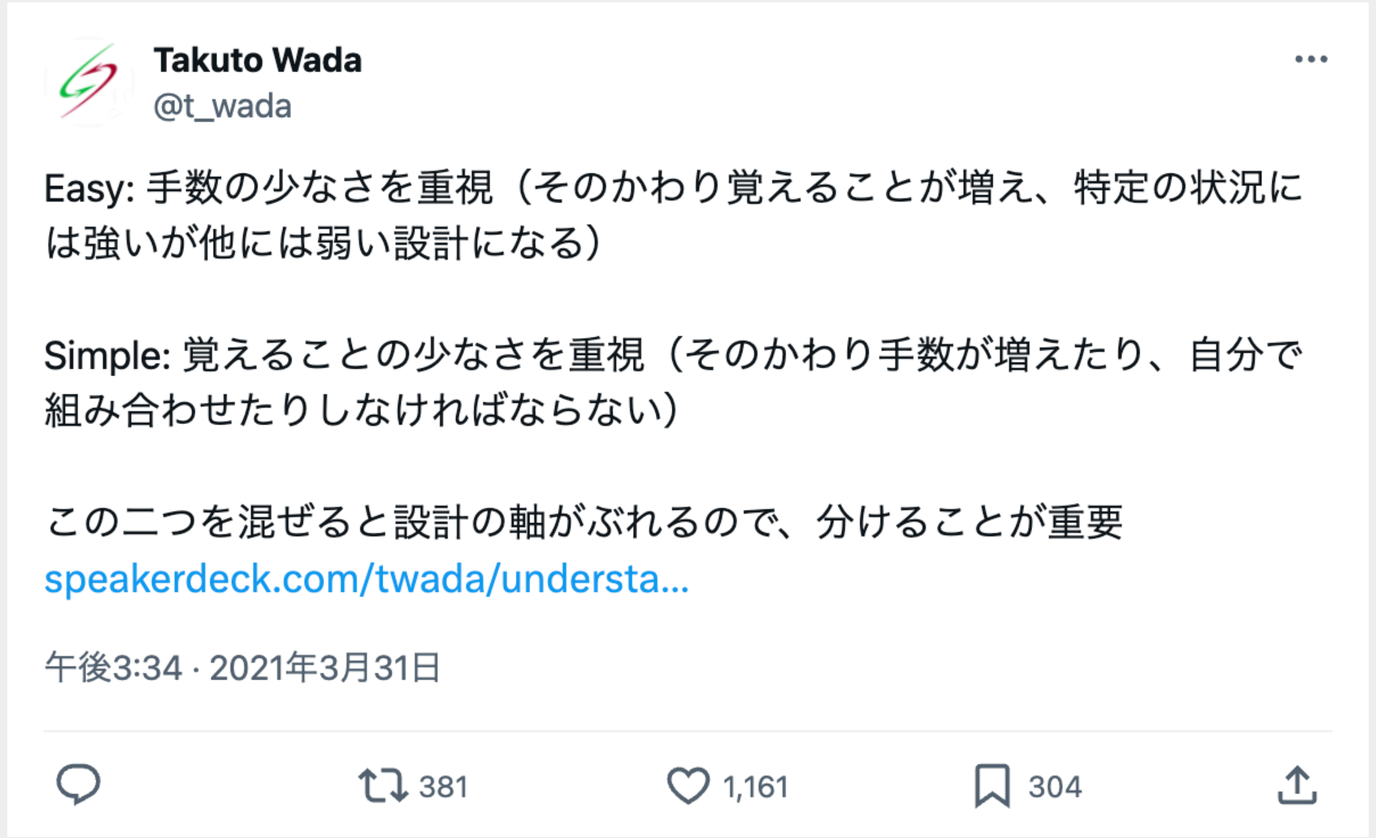
内容	ライブラリ
API Platform Coreが導入されたSymfony製のバックエンドアプリの雛形	api-platform/ **symfony**
React Admin製の管理画面の実装	api-platform/ **admin**
Next.js製のフロントエンドアプリの雛形を自動生成するためのツール	api-platform/ **create-client**
Mercureプロトコルを使用してバックエンドアプリからフロントエンドアプリへデータをプッシュできる構成	-
バックエンドアプリとフロントエンドアプリをまとめてホスティングするためのDocker定義	-
バックエンドアプリとフロントエンドアプリをKubernetesクラスターにデプロイするためのHelmチャート	-

「色々入ってる」の中身

内容	ライブラリ
Mercureプロトコルを使用してバックエンドアプリからフロントエンドアプリへデータをプッシュできる構成	-
バックエンドアプリとフロントエンドアプリをまとめてホスティングするためのDocker定義	-
バックエンドアプリとフロントエンドアプリをKubernetesクラスターにデプロイするためのHelmチャート	-

ごめん、ちょっと色々入って過ぎ🫠

SimpleとEasy



https://x.com/t_wada/status/1377147203077111814

SimpleとEasy

Easyはトレードオフ（であることが多い）

Easy / Complex

導入コスト ◎
汎用性・拡張性 △

Hard / Simple

導入コスト △
汎用性・拡張性 ◎

https://speakerdeck.com/takeru0757/simple-is-not-easy?slide=52

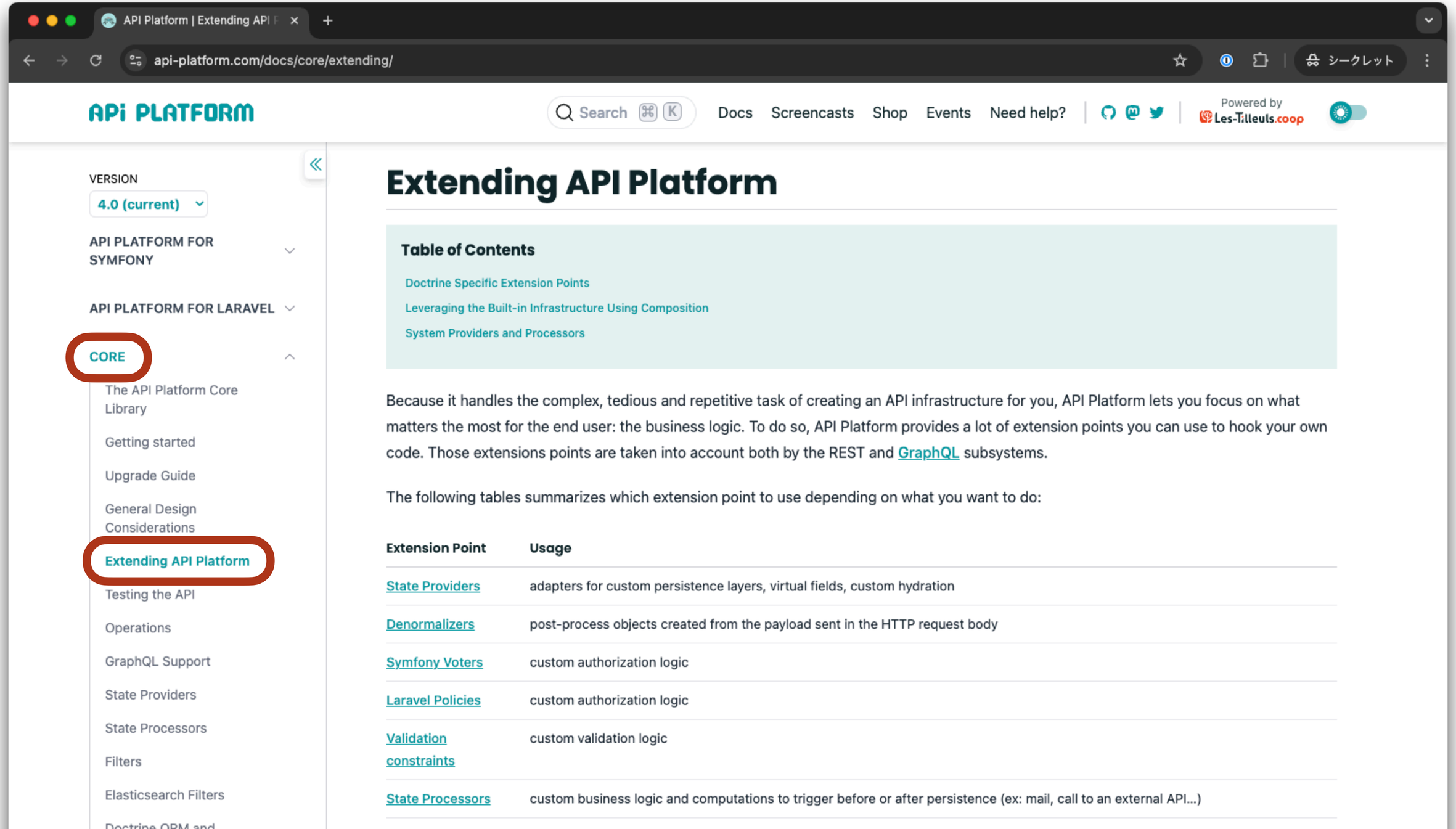
API PlatformはEasy指向？

- **公式テンプレート**
  - 「全部入り」でワンストップだけど自由度低
  - まさに創味のつゆ
- **API Platform Core**
  - アトリビュート1行でAPIが生える、これもEasy

API PlatformはEasy指向？

- **公式テンプレート**
  - 「全部入り」でワンストップだけど自由度低
  - まさに創味のつゆ
- **API Platform Core**
  - アトリビュート1行でAPIが生える、これもEasy
- **だから拡張しづらくて融通が利かない…？**


©YOSHIMOTO KOGYO CO.,LTD.


API Platform | Extending API
api-platform.com/docs/core/extending/
シークレット
API PLATFORM
Search
Docs
Screencasts
Shop
Events
Need help?
Powered by
Les-Tilleuls.coop
VERSION
4.0 (current)
API PLATFORM FOR SYMFONY
API PLATFORM FOR LARAVEL
CORE
The API Platform Core Library
Getting started
Upgrade Guide
General Design Considerations
Extending API Platform
Testing the API
Operations
GraphQL Support
State Providers
State Processors
Filters
Elasticsearch Filters
Extending API Platform
Table of Contents
Doctrine Specific Extension Points
Leveraging the Built-in Infrastructure Using Composition
System Providers and Processors
Because it handles the complex, tedious and repetitive task of creating an API infrastructure for you, API Platform lets you focus on what matters the most for the end user: the business logic. To do so, API Platform provides a lot of extension points you can use to hook your own code. Those extensions points are taken into account both by the REST and GraphQL subsystems.
The following tables summarizes which extension point to use depending on what you want to do:
Extension Point
Usage
State Providers
adapters for custom persistence layers, virtual fields, custom hydration
Denormalizers
post-process objects created from the payload sent in the HTTP request body
Symfony Voters
custom authorization logic
Laravel Policies
custom authorization logic
Validation constraints
custom validation logic
State Processors
custom business logic and computations to trigger before or after persistence (ex: mail, call to an external API...)

豊富な拡張ポイント

- **「これがやりたいんだけど、やり方が用意されてない」がほぼ無い**
- よく抽象化されており、RESTにもGraphQLにもそのまま適用される

API Platformは"Easyでラップされ たSimple指向"

- **本質的にはSimple指向**
  - 無駄を省いた1つの重要な仕事をする小さな部品の集合体
  - 拡張ポイントによって部品同士の繋ぎ目をフック可能
- **最外層でEasy指向な利用に対応**
  - 部品の組み合わせを**公式テンプレート**や**デフォルト設定**という形で提供

API Platformは"Easyでラップされ たSimple指向"

- **本質的にはSimple指向**
  - 無駄を省いた1つの重要な仕事をする小さな部品の集合体
  - 拡張ポイントによって部品同士の繋ぎ目をフック可能
- **最外層でEasy指向な利用に対応**
  - 部品の組み合わせを**公式テンプレート**や**デフォルト設定**という形で提供
- **すごくSymfonyっぽい**（作者のKévin DunglasさんはSymfonyコアチームのメンバー）

使い始め方

- **Symfonyと併用で使う場合**
  - **公式テンプレート**でプロジェクトを作成
  - 既存のSymfonyアプリケーションに**API Platform Core**をインストール
- **Laravelと併用で使う場合**
  - 既存のLaravelアプリケーションに**API Platform Core**をインストール

フレームワークと併用しない場合：https://api-platform.com/docs/core/bootstrap/

使い始め方

Easyを指向するなら

- **Symfonyと併用で使う場合**
  - **公式テンプレート**でプロジェクトを作成
  - 既存のSymfonyアプリケーションに**API Platform Core**をインストール
- **Laravelと併用で使う場合**
  - 既存のLaravelアプリケーションに**API Platform Core**をインストール

フレームワークと併用しない場合：https://api-platform.com/docs/core/bootstrap/

使い始め方

- **Symfonyと併用で使う場合**
  - **公式テンプレート**でプロジェクトを作成
  - 既存のSymfonyアプリケーションに**API Platform Core**をインストール
- **Laravelと併用で使う場合**
  - 既存のLaravelアプリケーションに**API Platform Core**をインストール

Simpleを指向するなら

フレームワークと併用しない場合：https://api-platform.com/docs/core/bootstrap/

使い始め方

- **Symfonyと併用で使う場合**
  - **公式テンプレート**でプロジェクトを作成
  - 既存のSymfonyアプリケ
- **Laravelと併用で使う場合**
  - 既存のLaravelアプリケーションに**API Platform Core**をインストール

今日はこれの実装例を紹介します🤷‍♂️

フレームワークと併用しない場合：https://api-platform.com/docs/core/bootstrap/

メモ：ここまで10分目標

もくじ

- 自己紹介（2分）
- API Platformの全体像を把握する（8分）
- **動かしてみる（3分）**
- データの読み書きをカスタマイズしてみる（3分）
- OpenAPIをカスタマイズしてみる（3分）
- まとめ（1分）
- デモ（5分）

とりあえず動くところまで

- Laravelアプリケーションを新規作成

```
$ composer create-project laravel/laravel my-app
$ cd my-app
```

とりあえず動くところまで

- API Platform Coreをインストール

```
$ composer require api-platform/laravel
```

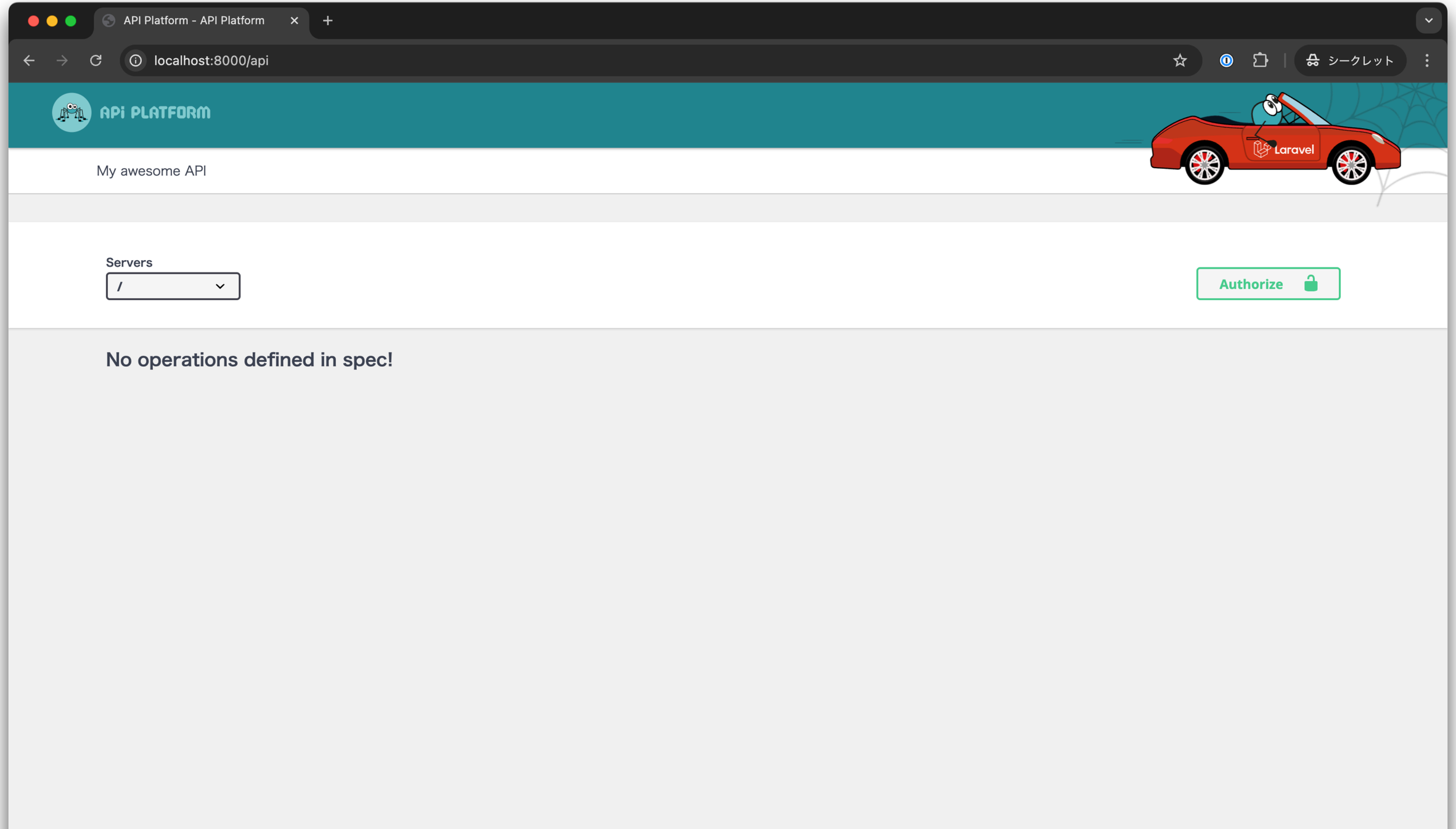
- API Platform Coreが提供するアセットと設定をインストール

```
$ php artisan api-platform:install
```

とりあえず動くところまで

- 起動

```
$ php artisan serve
```


API Platform - API Platform
localhost:8000/api
シークレット
API PLATFORM
Laravel
My awesome API
Servers
/
Authorize
No operations defined in spec!

モデルを作ってみる

- Article（記事）モデルを作成

```
$ php artisan make:model Article
```

- マイグレーションファイルを作成

```
$ php artisan make:migration create_articles_table
```

モデルを作ってみる

- マイグレーションファイルを修正

```
    public function up(): void
    {
        Schema::create('articles', function (Blueprint $table) {
            $table->id();
+           $table->string('title');
+           $table->text('content')->nullable();
+           $table->boolean('published')->default(false);
            $table->timestamps();
        });
    }
```

モデルを作ってみる

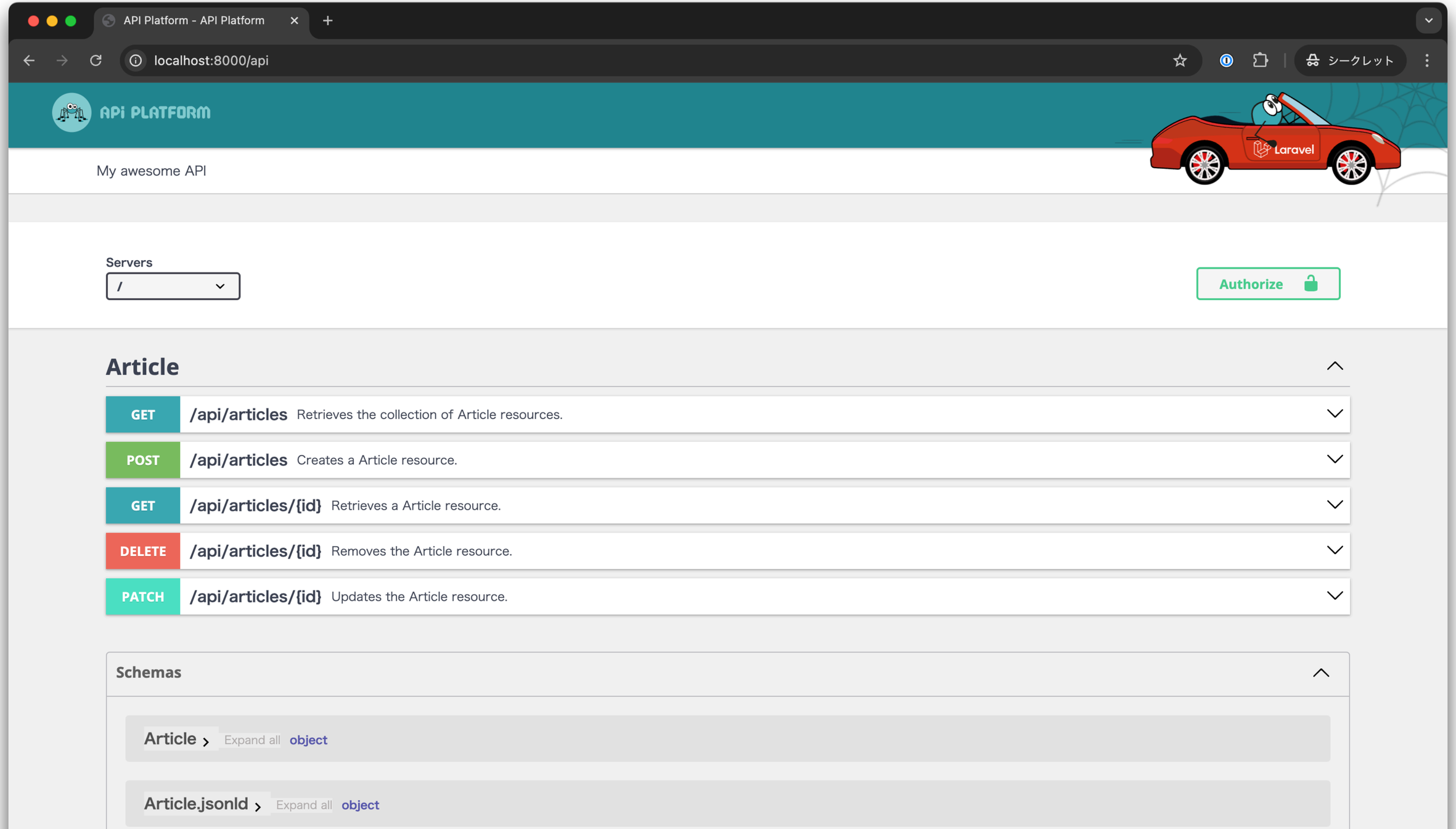
- マイグレーションを実行

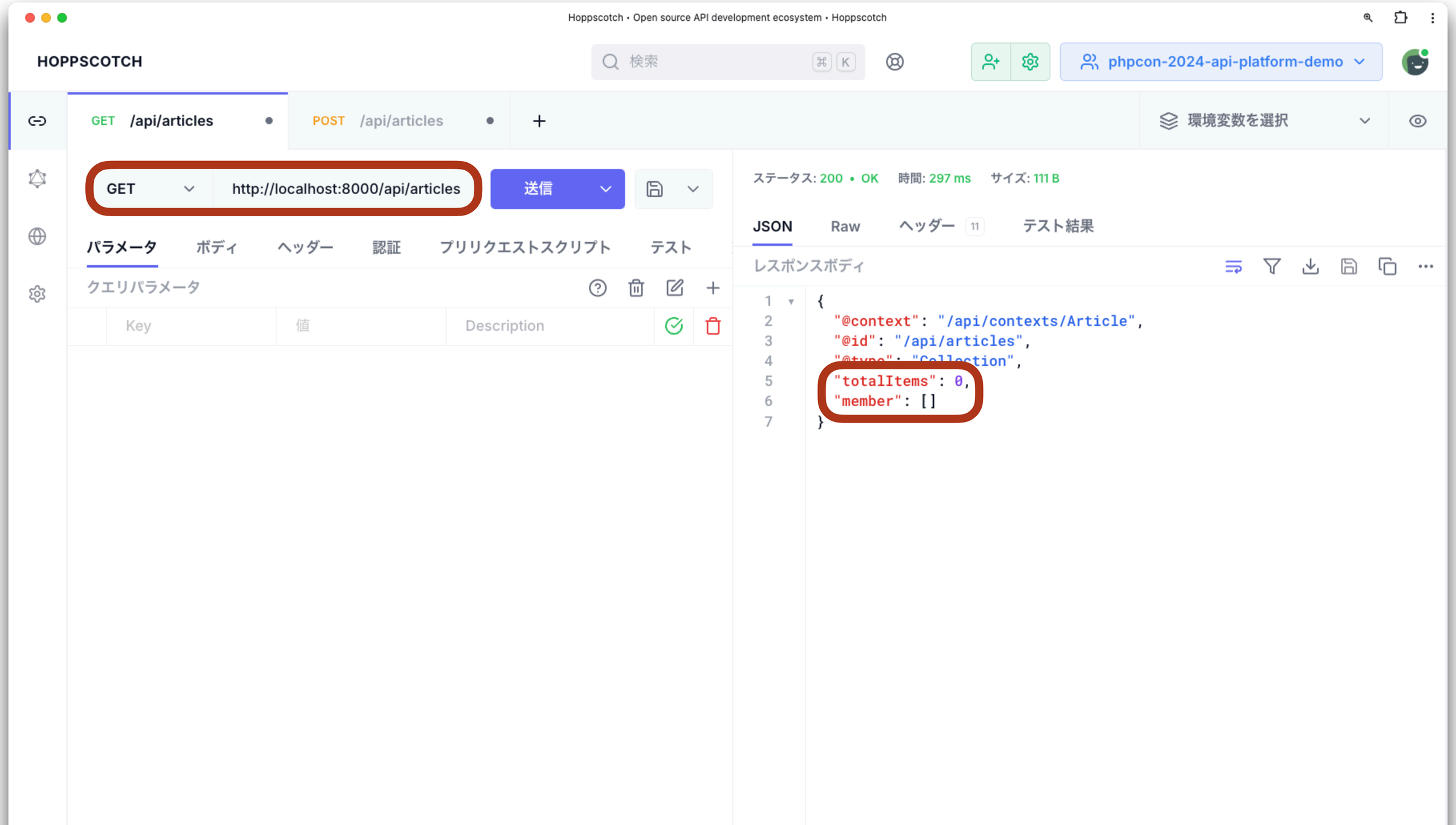
```
$ php artisan migrate
```

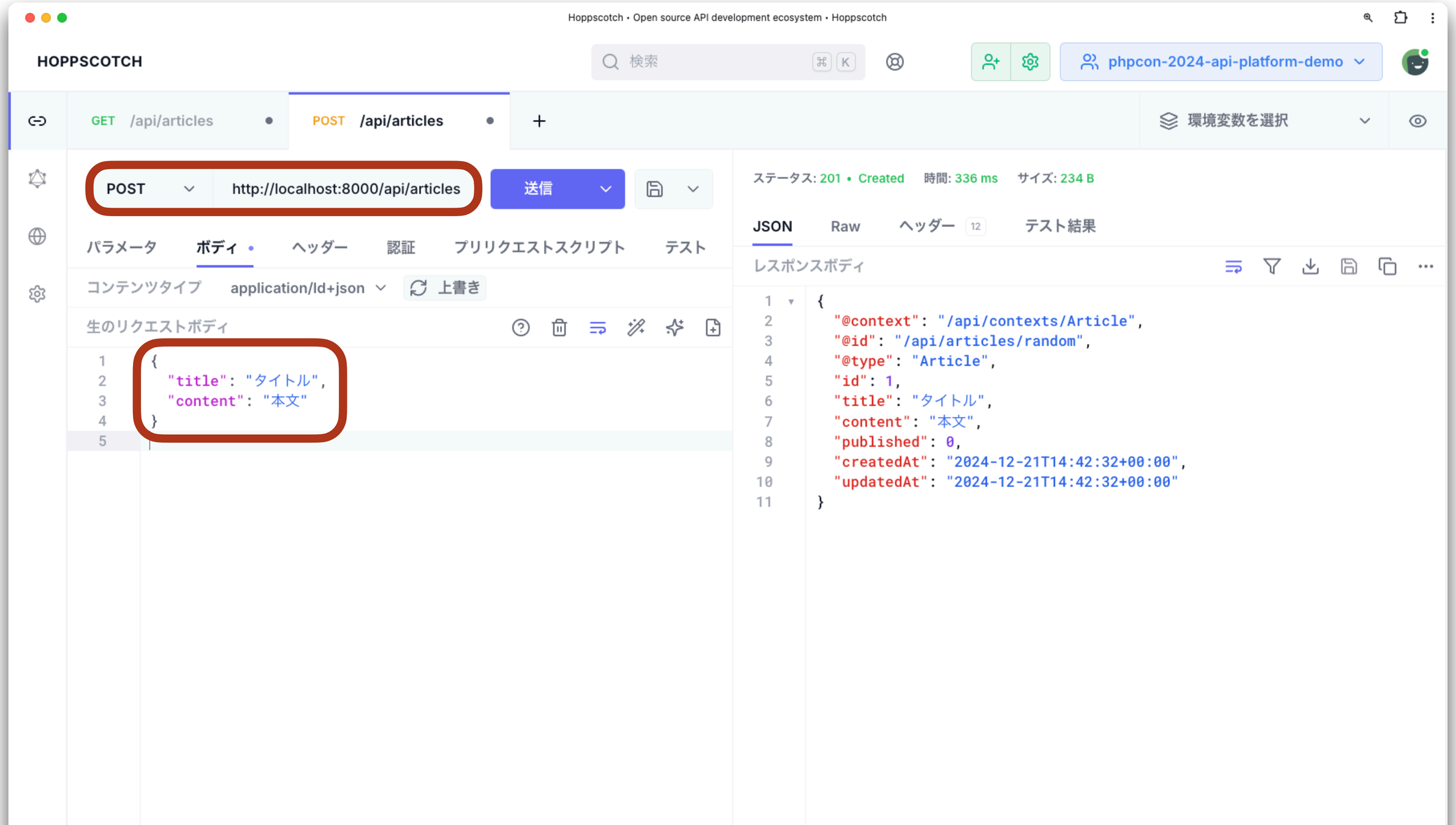
モデルを作ってみる

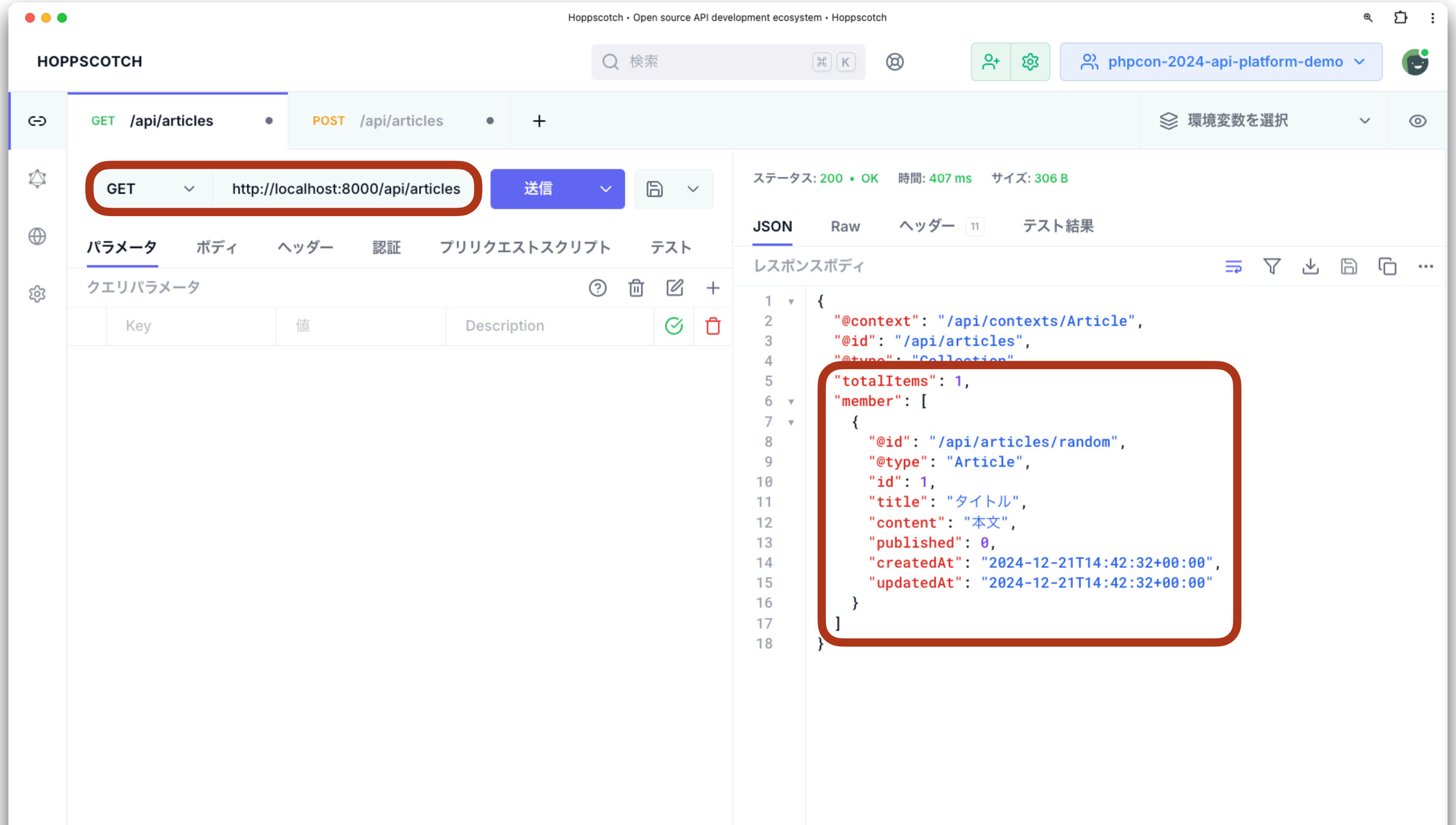
- モデルをAPIリソースとして公開

```
  <?php

  namespace App\Models;

+ use ApiPlatform\Metadata\ApiResource;
  use Illuminate\Database\Eloquent\Model;

+ #[ApiResource]
  class Article extends Model
  {
      //
  }
```


API Platform - API Platform
localhost:8000/api
シークレット
API PLATFORM
Laravel
My awesome API
Servers
/
Authorize
Article
GET /api/articles Retrieves the collection of Article resources.
POST /api/articles Creates a Article resource.
GET /api/articles/{id} Retrieves a Article resource.
DELETE /api/articles/{id} Removes the Article resource.
PATCH /api/articles/{id} Updates the Article resource.
Schemas
Article › Expand all object
Article.jsonld › Expand all object


Hoppscotch • Open source API development ecosystem • Hoppscotch
HOPPSCOTCH
検索
⌘ K
phpcon-2024-api-platform-demo
GET /api/articles
POST /api/articles
環境変数を選択
GET
http://localhost:8000/api/articles
送信
パラメータ
ボディ
ヘッダー
認証
プリリクエストスクリプト
テスト
クエリパラメータ
Key
値
Description
ステータス: 200 • OK
時間: 297 ms
サイズ: 111 B
JSON
Raw
ヘッダー 11
テスト結果
レスポンスボディ
{
"@context": "/api/contexts/Article",
"@id": "/api/articles",
"@type": "Collection",
"totalItems": 0,
"member": []
}


Hoppscotch • Open source API development ecosystem • Hoppscotch
HOPPSCOTCH
検索
⌘ K
phpcon-2024-api-platform-demo
GET /api/articles
POST /api/articles
環境変数を選択
POST
http://localhost:8000/api/articles
送信
パラメータ
ボディ
ヘッダー
認証
プリリクエストスクリプト
テスト
コンテンツタイプ
application/ld+json
上書き
生のリクエストボディ
{
"title": "タイトル",
"content": "本文"
}
ステータス: 201 • Created
時間: 336 ms
サイズ: 234 B
JSON
Raw
ヘッダー 12
テスト結果
レスポンスボディ
{
"@context": "/api/contexts/Article",
"@id": "/api/articles/random",
"@type": "Article",
"id": 1,
"title": "タイトル",
"content": "本文",
"published": 0,
"createdAt": "2024-12-21T14:42:32+00:00",
"updatedAt": "2024-12-21T14:42:32+00:00"
}


Hoppscotch • Open source API development ecosystem • Hoppscotch
HOPPSCOTCH
検索
⌘ K
phpcon-2024-api-platform-demo
GET /api/articles
POST /api/articles
環境変数を選択
GET
http://localhost:8000/api/articles
送信
パラメータ
ボディ
ヘッダー
認証
プリリクエストスクリプト
テスト
クエリパラメータ
Key
値
Description
ステータス: 200 • OK
時間: 407 ms
サイズ: 306 B
JSON
Raw
ヘッダー 11
テスト結果
レスポンスボディ
{
"@context": "/api/contexts/Article",
"@id": "/api/articles",
"totalItems": 1,
"member": [
{
"@id": "/api/articles/random",
"@type": "Article",
"id": 1,
"title": "タイトル",
"content": "本文",
"published": 0,
"createdAt": "2024-12-21T14:42:32+00:00",
"updatedAt": "2024-12-21T14:42:32+00:00"
}
]
}

メモ：ここまで13分目標

📌もくじ

- 自己紹介（2分）
- API Platformの全体像を把握する（8分）
- 動かしてみる（3分）
- **データの読み書きをカスタマイズしてみる（3分）**
- OpenAPIをカスタマイズしてみる（3分）
- まとめ（1分）
- デモ（5分）

現状で、DBに対する単純なCRUDはできる

- データの一覧を取得
- データを新規作成
- 単一のデータの詳細を取得
- 単一のデータを編集
- 単一のデータを削除

では、こういう場合は？

- DB以外のデータソース（例：外部APIなど）からデータを取得したい
- DB以外のデータソース（例：外部APIなど）にデータを保存したい
- DBにマッピングされた内部モデルをそのままAPIとして公開したくない
- 読み取りと更新の操作に別々のモデルを使いたい（CQRSパターン）
- etc...

答え

State Provider / State Processor を自作する

State Provider / State Processor

State **Provider**	リクエストに対応するデータ（状態）を**取得**するサービス
State **Processor**	リクエストに対応するデータ（状態）の**永続化**処理を行うサービス

State Provider / State Processor

State **Provider**	リクエストに対応するデータ（状態）を**取得**するサービス
State **Processor**	リクエストに対応するデータ（状態）の**永続化**処理を行うサービス

- いずれも、`api-platform/laravel` に**Eloquent向けのデフォルト実装**が組み込まれており、通常はそれが**自動で使われる**
- これを**自作**して、特定の**APIオペレーションに割り当て**ればよい

State Provider の自作

```php
namespace App\State;

use ApiPlatform\Metadata\Operation;
use ApiPlatform\State\ProviderInterface;

final class ArticleProvider implements ProviderInterface
{
    public function provide(Operation $operation,
                            array $uriVariables = [],
                            array $context = []): object|array|null
    {
        $article = /* ここに記事を取得する処理を書く */;

        return $article;
    }
}
```

State Processor の自作

```php
namespace App\State;

use ApiPlatform\Metadata\Operation;
use ApiPlatform\State\ProcessorInterface;

class ArticleProcessor implements ProcessorInterface
{
    public function process(mixed $data,
                            Operation $operation,
                            array $uriVariables = [],
                            array $context = []): mixed
    {
        /* ここに記事を永続化（$data を更新して $data->save() など）する処理を書く */

        return $data;
    }
}
```

サービスへのタグ付けを忘れずに（Laravelの場合のみ）

```php
namespace App\Providers;

use ApiPlatform\State\ProcessorInterface;
use App\State\ArticleProcessor;
use App\State\ArticleProvider;
use Illuminate\Support\ServiceProvider;

class AppServiceProvider extends ServiceProvider
{
    public function register(): void
    {
        $this->app->tag([ArticleProvider::class], ProviderInterface::class);
        $this->app->tag([ArticleProcessor::class], ProcessorInterface::class);
    }

    public function boot(): void
    {
    }
}
```

APIオペレーションへの割り当て

```
  namespace App\Models;

  use ApiPlatform\Metadata\ApiResource;
+ use App\State\ArticleProcessor;
+ use App\State\ArticleProvider;
  use Illuminate\Database\Eloquent\Model;

- #[ApiResource]
+ #[ApiResource(
+     provider: ArticleProvider::class,
+     processor: ArticleProcessor::class,
+ )]
  class Article extends Model
  {
      //
  }
```

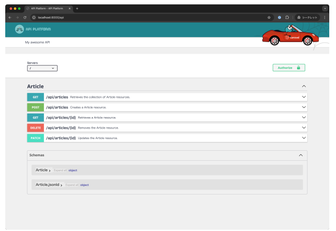
メモ：ここまで16分目標

📌もくじ

- 自己紹介（2分）
- API Platformの全体像を把握する（8分）
- 動かしてみる（3分）
- データの読み書きをカスタマイズしてみる（3分）
- **OpenAPIをカスタマイズしてみる（3分）**
- まとめ（1分）
- デモ（5分）

OpenAPIとは？

- **Web APIの仕様を記述**するための言語仕様
- OpenAPIで記述されたドキュメントを「OpenAPIドキュメント」と呼ぶ
  （JSONまたはYAML形式）
- OpenAPIドキュメントは「Swagger UI」というツールによって
  視覚的なAPIドキュメントページに変換できる（これのこと→）



- OpenAPIドキュメントから**TypeScript**（など各種言語）**のAPIクライアント実装を自動生成**することなんかもできる（スキーマ駆動開発）

API PlatformとOpenAPI

- API Platformは、OpenAPIドキュメントを**それなりの内容で**自動生成してくれる
- **必要に応じて手動で細かく制御**することもできる
- OpenAPIドキュメントの内容を**徹底的に厳密**にすれば、APIクライアントの自動生成がより強力に機能する

カスタマイズの例：プロパティの型を変更する

- 例えば、API Platformがコードから自動で推測したプロパティの型とは**異なる型情報をOpenAPIドキュメントに出力**することなどができる
- そして、実はLaravelベースで最低限のセットアップをした状態では、DBがSQLiteであることに起因して、**bool型のプロパティのOpenAPIドキュメント上での型がbooleanではなくstringになっている**🤦

Article.jsonld ^ Collapse all object

@context > Expand all read-only (string | object)

@id read-only string

@type read-only string

id read-only integer

title string

content string | null

published string

createdAt string | null date-time

updatedAt string | null date-time

カスタマイズの例：プロパティの型を変更する

- ので、これが意図どおり**boolean型として出力されるように**手動で
  カスタマイズしてみる

```
 namespace App\Models;

+ use ApiPlatform\Metadata\ApiProperty;
  use ApiPlatform\Metadata\ApiResource;
  use App\State\ArticleProcessor;
  use App\State\ArticleProvider;
  use Illuminate\Database\Eloquent\Model;

  #[ApiResource(
      provider: ArticleProvider::class,
      processor: ArticleProcessor::class,
  )]
+ #[ApiProperty(schema: ['type' => 'boolean'], property: 'published')]
  class Article extends Model
  {
      //
  }
```

Article.jsonld ^ Collapse all object

@context > Expand all read-only (string | object)

@id read-only string

@type read-only string

id read-only integer

title string

content string | null

published boolean

createdAt string | null date-time

updatedAt string | null date-time

＼かんたん！／

OpenApiFactoryクラスを拡張すれば何でもできる

- 基本的なカスタマイズはアトリビュート経由で簡単に対応できる
- もっとイレギュラーなことをやりたい場合は、OpenAPIドキュメントの生成を担っている**OpenApiFactory**というクラスを直接拡張することで無限に柔軟に拡張することもできる👌

  （API Platformを本格的に使い始めると、なんだかんだでほとんどの場合これをいじることになる）

📌もくじ

- 自己紹介（2分）
- API Platformの全体像を把握する（8分）
- 動かしてみる（3分）
- データの読み書きをカスタマイズしてみる（3分）
- OpenAPIをカスタマイズしてみる（3分）
- **まとめ（1分）**
- デモ（5分）

3行で頼む

- あのFabienも激推し！
- マジでアトリビュート1行でAPIとAPIドキュメントが作れる！
- その気になれば無限に拡張も可能！

3行で頼む

- あのFabi
- マジでア
- その気に

このトークのゴール

「API Platform、なんかよさそう」

と思ってもらえれば十分です！

＼気楽に聞いてください／

3行で頼む

- あのFabi
- マジでア
- その気に

懇親会などで
感想聞かせて
ください！😉

このトークのゴール

「API Platform、なんかよさそう」

と思ってもらえれば十分です！

＼気楽に聞いてください／

ちなみに

商業本を書いています✍️（進捗ダメです）

- まだいろいろ未確定なのであまり詳しくは書けませんが…
- Symfony/Laravelなんもﾜｶﾗﾝ状態からでもAPI Platformをマスターできるようにめっちゃ丁寧な解説を頑張っています
- OpenAPI/JSON Schema/JSON-LD/Hydraあたりの要素技術についてのめちゃくちゃ丁寧な解説は書き終わってて、ここだけでも超自信作なので絶対に世に出さなければ…と思っています
- PhpStormで書いています（？）

＼Thanks!／

@ttskch

質問・相談などいつでもどうぞ🤞

📌もくじ

- 自己紹介（2分）
- API Platformの全体像を把握する（8分）
- 動かしてみる（3分）
- データの読み書きをカスタマイズしてみる（3分）
- OpenAPIをカスタマイズしてみる（3分）
- まとめ（1分）
- **デモ（5分）**

お見せする（したい）もの

- とりあえず動くところまでセットアップしたAPIを実際に叩く様子
- OpenAPIドキュメントをJSON形式で出力して、それをもとにTypeScriptの型定義を自動生成し、フロントエンドから利用する様子