『eg-r2』の公開について

LITALICO SaaS Summit_2411　Nov 15, 2024.

v0.0.1

@katzumi(かつみ)

Press Space for next page →

自己紹介

katzumi（かつみ）と申します。

「障害のない社会をつくる」をビジョンに掲げている「LITALICO」という会社に所属しています

LITALICO りたりこ

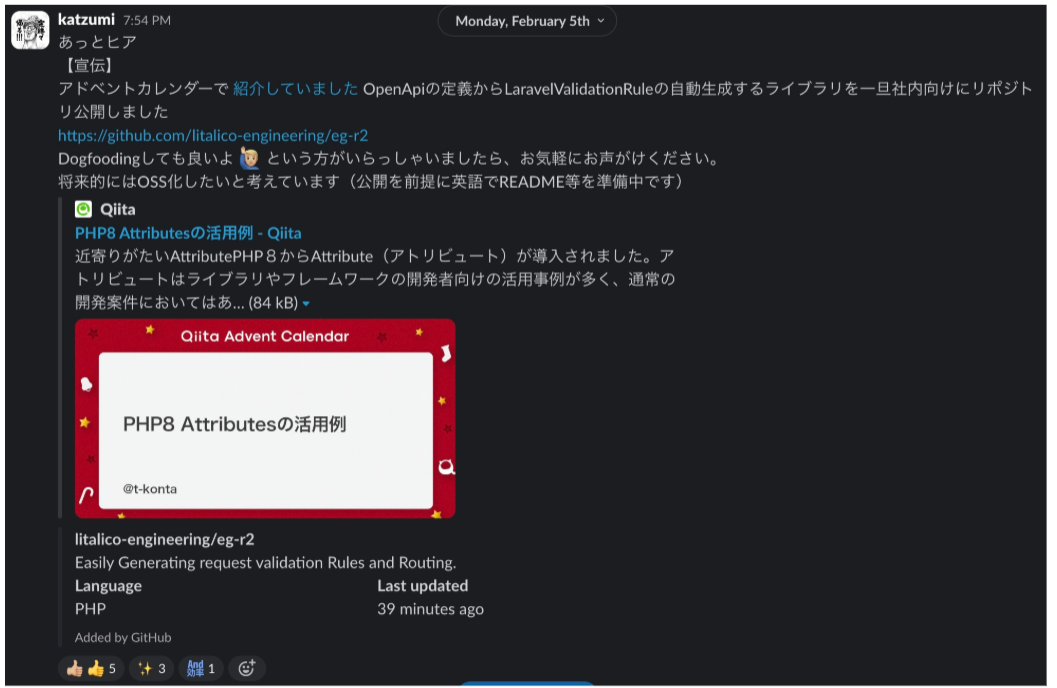
以下のアカウントで活動しています。

X katzchum

k2tzumi katzumi

eg-r2 プレリリース

法改正リリース後に社内のみ先行公開していました



祝！オープンソース化！🎉

たぶん LITALICO 初

オープンソースソフトウェアとは？

What is OSS?

> *オープンソースソフトウェア（Open Source Software、略称: OSS）とは、利用者の目的を問わずソースコードを使用、調査、再利用、修正、拡張、再配布が可能なソフトウェアの総称である。*

wikipedia より

packagistにも公開されています

https://packagist.org/packages/litalico-engineering/eg-r2

`composer require litalico-engineering/eg-r2` で直ぐに使えます。

MEMO

Packagist は Composer のメインリポジトリです。
Composer でインストール可能な公開 PHP パッケージが集約されています。
Packagistには、たくさんのパッケージが登録されています。
これらのパッケージは他の開発者が作った便利な部品で、プログラマーは自分のプロジェクトに簡単に取り込むことができます。

eg-r2 とは？

Easy request validation and route generation from open API specifications

2 つのことを簡単(Easy)にすることを目的としています

eg-r2 とは？

Easy request validation and route generation from open API specifications

2 つのことを簡単(Easy)にすることを目的としています

1. リクエストバリデーション

eg-r2 とは？

Easy request validation and route generation from open API specifications

2 つのことを簡単(Easy)にすることを目的としています

1. リクエストバリデーション
2. ルーティング設定

前提

require

- PHP8.2 以上

  PHP Attributes で API Spec を記述する為
- Laravel9.0 以上 最新バージョン(version1.0.0)以降は 11.0 以上
- swagger-phpで API 仕様書を OpenAPI V3 形式で記述

リクエストのバリデーション自動生成

Easy request validation

OpenAPI でスキーマ定義しておけば、バリデーションが自動生成されます！

MEMO

リクエストのバリデーションとは？

　例えば、オンラインショッピングでお買い物をする時に、名前や住所、クレジットカード番号等の入力内容を「リクエスト」と言い その時に、入力内容（リクエスト）が正しい形式であるか、間違いや漏れがないかを確認することを「バリデーション」といいます。

　具体的には、次のようなことをチェックします。

1. 必須項目の確認： 名前や住所など、必ず入力しなければならない情報がちゃんと入力されているかを確認します。
2. 形式の確認： メールアドレスが「@」を含んでいるか、クレジットカード番号が数字のみで構成されているかなど、特定の形式に合っているかをチェックします。
3. 範囲の確認： 年齢や金額など、入力された値が決められた範囲内にあるかを確認します。

As-is

こんな感じで FormRequest のクラス定義していますよね？

別途 API 仕様書を記述する必要があります

```
/**
 * @property int $age
 * @property string $name
 * @property bool $is_active
 */
class MyFormRequest extends FormRequest
{
  public function rules()
  {
    return [
      'age' => 'required|integer',
      'name' => 'required|string',
      'is_active' => 'required|boolean',
    ];
  }
}
```

As-is

こんな感じで FormRequest のクラス定義していますよね？

別途 API 仕様書を記述する必要があります

```php
/**
 * @property int $age
 * @property string $name
 * @property bool $is_active
 */
class MyFormRequest extends FormRequest
{
  public function rules()
  {
    return [
      'age' => 'required|integer',
      'name' => 'required|string',
      'is_active' => 'required|boolean',
    ];
  }
}
```

To-be

こんな感じで FormRequest に OpenAPI を Spec を Attribute を記述するだけ！

Trait を追加するだけ！その他の記述不要

```
#[Schema(title: 'My request', required:['age', 'name', 'is_active'])]
class MyFormRequest extends FormRequest
{
  use RequestRuleGeneratorTrait, FormRequestPropertyHandlerTrait;

  #[Property(property: 'age', type: 'integer', format: 'int64')]
  public int $age;

  #[Property(property: 'name', type: 'string')]
  public string $name;

  #[Property(property: 'is_active', type: 'boolean')]
  public boolean $is_active;

  // roulesメソッドはtraitで自動生成しているので不要
}
```

リクエストパラメータを型安全で扱えます！

To-be

こんな感じで FormRequest に OpenAPI を Spec を Attribute を記述するだけ！

Trait を追加するだけ！その他の記述不要

```php
#[Schema(title: 'My request', required:['age', 'name', 'is_active'])]
class MyFormRequest extends FormRequest
{
  use RequestRuleGeneratorTrait, FormRequestPropertyHandlerTrait;

  #[Property(property: 'age', type: 'integer', format: 'int64')]
  public int $age;

  #[Property(property: 'name', type: 'string')]
  public string $name;

  #[Property(property: 'is_active', type: 'boolean')]
  public boolean $is_active;

  // roulesメソッドはtraitで自動生成しているので不要
}
```

リクエストパラメータを型安全で扱えます！

何が嬉しいのか？

eg-r2 の狙い

- API 仕様書と実装の乖離を発生させない！
  そもそも同じことを 2 回書く必要がなくなる
- 開発の手数が減り、気になりポイントも減る
  全体の記述量も減るのでは？

ルーティング設定の自動化

Easy route generation

OpenAPI でスキーマ定義しておけば、ルーティング設定が半自動化されます！

MEMO

ルーティングとは？
　インターネット上で誰かが何かを求めてきた時に、そのリクエストをどこに送るかを決めることを「ルーティング」といいます。
具体的な例を挙げると

1. ホームページ： お客様が「ホームページを見たい」とリクエストした場合、そのリクエストをホームページの担当部署に送ります。
2. 商品ページ お客様が「商品ページを見たい」とリクエストした場合、そのリクエストを商品ページの担当部署に送ります。
3. 注文ページ： お客様が「注文したい」とリクエストした場合、そのリクエストを注文ページの担当部署に送ります。

ルーティング設定の自動化

Easy route generation

コマンド一発でルーティングの自動生成

`php artisan eg-r2:generate-route`

```php
/**
* This file is auto-generated.
*/

declare(strict_types=1);

Route::as('api')->group(static function (): void {
    Route::controller('App\Http\Controllers\Pet')->group(static function (): void {
        Route::post('/pet', 'addPet');
        Route::put('/pet', 'updatePet');
        Route::get('/pet/findByStatus', 'findPetsByStatus');
        Route::post('/pet/{petId}', 'updatePetWithForm');
        Route::delete('/pet/{petId}', 'deletePet');
        Route::post('/pet/{petId}/uploadImage', 'uploadFile');
    });
    Route::controller('App\Http\Controllers\Store')->group(static function (): void {
        Route::get('/store', 'getInventory');
        Route::post('/store/order', 'placeOrder');
        Route::get('/store/order/{orderId}', 'getOrderById');
        Route::delete('/store/order/{orderId}', 'deleteOrder');
    });
});
```

なぜ eg-r2 を作ったのか？

全ては法改正を爆速に対応できるシステムを構築するため

- API 仕様書の品質を高めるため！

  API 仕様書の品質が悪いと手戻りが発生してしまう。サーバー側とクライアント側共に
- API 仕様の見直しで実装との乖離を発生させない

  法解釈が違ったら直さないといけなくなる
- API 仕様書を先に公開して実装するため

  各プロダクトと並行で開発する[1]。API の完成を待っていたら開発が間に合わない

---

1. スキーマ駆動開発といいます ↩

どうなったか？

eg-r2 の効果

多数の API を高品質且つ爆速で構築

- 法改正を乗り越えることができた🎉
- 200 弱の API が eg-r2 によって作成
- 1 つの API で 100 弱のパラメータが存在
- 法改正時に 80 の API を追加
  短期間にコピペで量産することもできる

なぜ eg-r2 をオープンソース化したのか？

OSS 化の狙い

- 様々なユースケースに対応させるため

  他プロジェクトでの個別な要望にも対応できるように機能追加をしていきたい
- フィードバックサイクルを回すため

  スキーマ駆動開発の在り方をディスカッションしたい

  スキーマが先か？コードが先か？の問題を提起したい

今後について

eg-r2 の発展について

- 対応ユースケースの拡大

  ドックフーディングしてくれる方募集中です
- 社内外での発信し、認知拡大

最後に

プロジェクトに貢献してくれる方、絶賛募集中です

We're contributing.

Link

- eg-r2
  プロジェクトリポジトリ
- eg-r2-example
  サンプルリポジトリ
- 頑張らないスキーマ駆動開発を支える『eg-r2』を公開しました

ご清聴ありがとうございました