サーバーサイドスクリプトⅡ

１０．API

第１０回目では、APIの作成を行っていきます。（APIの説明は後述）

API呼び出し時に渡されたデータを元にDBを検索し、検索結果をJSON形式で返すAPIを作成します。下記手順の③④を実装します。（既存のモデルを使用します）

1. **マイグレーション**
2. **モデルの作成**
3. **コントローラの作成**
4. **ルーティングの設定**
5. **ビューの作成　※※注意※※　APIなので、Laravelでのビューは使用しません。**

APIの利用には、今回のように「自分で開発する」パターンの他に、「提供されているサービスを利用する」パターンもあります。

|  |  |
| --- | --- |
| 警告 単色塗りつぶし | **APIとは**  Application Programming Interfaceの略であり、アプリケーションやソフトウェアを繋ぐインターフェイス（＊）のことを指します。  （＊）インターフェイス・・・接続する際のプロトコルや接続部分の仕様を取り決めたもの。 |

１０－１．APIリソースコントローラ

第6回では、CRUDに対応するメソッドを作成するために、リソースコントローラを作成しました。ただし、リソースコントローラの中には、フォームを表示するためのメソッド（createやedit）も含まれています。今回はAPIの作成なので、createメソッドやeditメソッドは必要ありません。

＊リソースコントローラのメソッド一覧については、第6回目の資料を参照すること。

１０－１－１．　コントローラの作成

コントローラはartisan コマンドで作成しますが、APIリソースコントローラを作成する際には、オプションを付けて実行する必要があります。

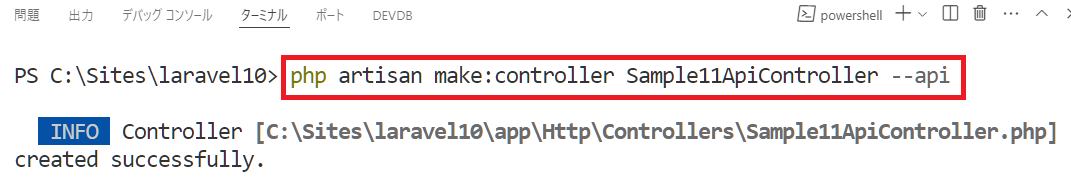
■コントローラの作成コマンド apiオプション付き



今回は「Ｓａｍｐｌｅ11ApiController」を作成します。

■サンプル実行環境

　＊「～～successfully」と表示されればOK。



作成されたSample11ApiControllerクラスを見ていきます。

Sample11ApiControllerの作成にapiオプションを付けたことにより、CRUDに対応したメソッドが予め用意されています。

また、createメソッドやeditメソッドがないことも確認してください。

（コードは次ページに記載）

■Sample11ApiController.php

　＊use文やコメントアウトは省略して記載。

class Sample11ApiController extends Controller

{

    public function index()

    {

    }

    public function store(Request $request)

    {

    }

    public function show(string $id)

    {

    }

    public function update(Request $request, string $id)

    {

    }

    public function destroy(string $id)

    {

    }

}

１０－１－２．コントローラの処理

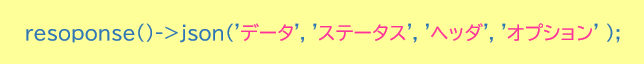
今回はDBの「samples」テーブルよりidをキーに1件のデータを取得する処理を作成していきます。

コントローラのメソッドは、指定されたリソースの詳細情報を表示するshowメソッドを使用します。

値はJSON形式で返したいので、response()->jsonメソッドを使用します。

また、ステータスには定数を利用します。

■response()->jsonメソッド



■Sample11ApiController.php

　＊showメソッドのみ記述。

DBからの抽出データがなかった場合は、固定値を返却する。

定数は

Symfony\Component\HttpFoundation\Response」クラスに定義しているものを使用する。

public function show(string $id)

{

    $sample = Sample::find($id);

    if ($sample) {

        return response()->json(

            [

                "data" => $sample

            ],

            Response::HTTP\_OK,

            ['Content-Type' => 'application/json;charset=UTF-8', 'Charset' => 'utf-8'],

            JSON\_UNESCAPED\_UNICODE

        );

    } else {

        return response()->json(

            [

                "data" => [

                    "status" => "error",

                    "message" => "データが存在しません。"

                ]

            ],

            Response::HTTP\_NOT\_FOUND,

            ['Content-Type' => 'application/json;charset=UTF-8', 'Charset' => 'utf-8'],

            JSON\_UNESCAPED\_UNICODE

        );

    }

}

１０－１－３．APIステータスコード

APIの応答メッセージに含まれる数値コードであり、APIの処理結果や応答の状態を示すために使用されます。クライアントがAPIを呼び出した際に、サーバーから返されるステータスコードは、クライアントに対してリクエストの成功、失敗、またはその他の特定の状態を伝える重要な情報となります。

例題では、データが存在した時に「200」、データが存在しない時に「４０４」を、定数を用いて返却しています。

■主なステータスコード一覧

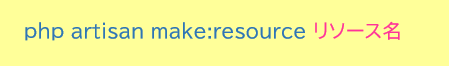
|  |  |  |  |
| --- | --- | --- | --- |
| コード | 意味 | タイプ | 説明 |
| 200 | OK | 成功 | リクエストが成功し、正常なレスポンスが返されたことを示します。 |
| 201 | Created | 成功 | リクエストが成功し、新しいリソースが作成されたことを示します。 |
| 204 | No Content | 成功 | リクエストが成功し、レスポンスにコンテンツがないことを示します。 |
| 400 | Bad Request | クライアントエラー | リクエストが不正であるため、サーバーがリクエストを解釈できないことを示します。 |
| 401 | Unauthorized | 認証とセキュリティエラー | 認証が必要なリソースに対して、クライアントが認証されていないことを示します。 |
| 403 | Forbidden | クライアントエラー | リクエストされたリソースへのアクセスが禁止されていることを示します。 |
| 404 | Not Found | クライアントエラー | リクエストされたリソースが見つからないことを示します。 |
| 409 | Conflict | クライアントエラー | リクエストが競合しているため、リソースの操作が失敗したことを示します。 |
| 422 | Unprocessable Entity | 業務エラー | リクエストデータが処理できない形式であることを示し、業務規則に違反していることを伝えます。 |
| 429 | Too  Many Requests | 業務エラー | リクエストが制限や割当量を超えているため、一時的にリクエストを制限または拒否していることを示します。 |
| 451 | Unavailable  For Legal Reasons | 業務エラー | リクエストが法的な理由で利用できないことを示します。 |
| 500 | Internal  Server Error | サーバーエラー | サーバーで予期しないエラーが発生したことを示します。 |
| 502 | Bad Gateway | サーバーエラー | ゲートウェイサーバーが無効なレスポンスを受信したことを示します。 |
| 503 | Service Unavailable | サーバーエラー | サーバーが一時的に過負荷やメンテナンスのために利用できない状態であることを示します。 |
| 504 | Gateway Timeout | サーバーエラー | ゲートウェイサーバーがタイムアウトしたことを示します。 |
| 507 | Insufficient Storage | サーバーエラー | サーバーが保存領域の容量不足で要求を処理できないことを示します。 |
| ５０９ | Bandwidth Limit Exceeded | 制限と割当量エラー | アクセス帯域幅の制限を超えたため、サーバーがアクセスを拒否していることを示します。 |
| 419 | Authentication Timeout | 認証とセキュリティエラー | セッションの認証タイムアウトが発生したことを示します。 |
| 498 | Invalid Token | 認証とセキュリティエラー | 提供されたトークンが無効であることを示します。 |

１０－２．APIリソースの作成

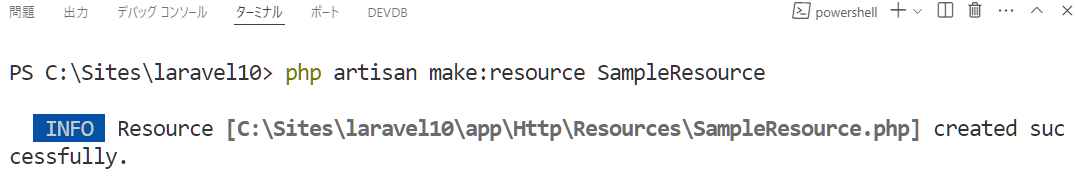
データの整形にはリソースを利用します。リソースを利用しなくても、全データを返すことは可能ですが、不要な列を除きたい場合や、テーブルのリレーションを用いる場合などに便利です。

リソースの作成には、下記のコマンドを使用します。

■APIリソースの作成



■サンプル環境



作成したリソースファイルにデータ構造を記述します。

■SampleResource.php

＊toArrayメソッドのみ記載。

　＊既存のreturn分をコメントアウトし、赤枠部分を記述。

　＊サンプルでは「id」「title」「body」「created\_at」の４列のみとしている。

　　　このリソースを使用した場合、「updated\_at」など列は省かれることに注意。

public function toArray(Request $request): array

{

    // return parent::toArray($request);

    return [

        'id' => $this->id,

        'title' => $this->title,

        'body' => $this->body,

        'created\_at' => $this->created\_at,

    ];

}

リソースファイルの記述が完了したら、コントローラからJSONデータを返却する際に、呼び出すように設定します。

■Sample11ApiController.php

　＊showメソッドのみ記述。赤枠部分を変更。

public function show(string $id)

{

　　　　～～略～～

    if ($sample) {

        return response()->json(

            [

                "data" => new SampleResource($sample)

            ],

            Response::HTTP\_OK,

　～～略～～

同様に、全件表示のindexメソッドも作成します。

■Sample11ApiController.php

　＊indexメソッドのみ記述。

　＊データが複数件の場合は、collectionメソッドを利用。

public function index()

{

    $samples = Sample::orderBy("created\_at", "desc")->get();

    if ($samples) {

        return response()->json(

            [

                "data" => SampleResource::collection($samples)

            ],

            Response::HTTP\_OK,

            ['Content-Type' => 'application/json;charset=UTF-8', 'Charset' => 'utf-8'],

            JSON\_UNESCAPED\_UNICODE

        );

    } else {

        return response()->json(

            [

                "data" => [

                    "status" => "error",

                    "message" => "データが存在しません。"

                ]

            ],

            Response::HTTP\_NOT\_FOUND,

            ['Content-Type' => 'application/json;charset=UTF-8', 'Charset' => 'utf-8'],

            JSON\_UNESCAPED\_UNICODE

        );

    }

}

１０―３．APIリソースルートの定義

APIのルーティングは、「プロジェクトフォルダ\routes」配下のapi.phpで行います。

APIに使用するリソースルートを宣言する場合、createやeditのようなHTMLテンプレートを提供するルートを除外したいことがよく起こります。

そのため、これらの ２ルートを自動的に除外する、apiResourceメソッドが使用できます。

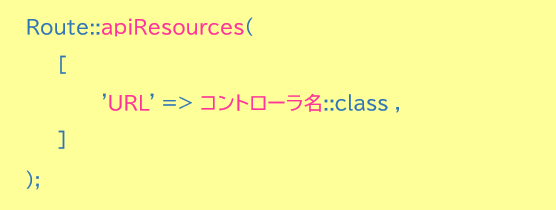
■APIリソースルート

（単一）



　または

　（複数）



URLにはバージョン番号を付けておくと管理しやすいです。

サンプルでは、「v1/samples」でルーティングします。

■api.php（単一Ver）

　＊web.php同様に使用するコントローラクラスのuse文は忘れずに。

Route::apiResource('v1/samples', Sample11ApiController::class);

■api.php（複数Ver）

　＊web.php同様に使用するコントローラクラスのuse文は忘れずに。

Route::apiResources(

    [

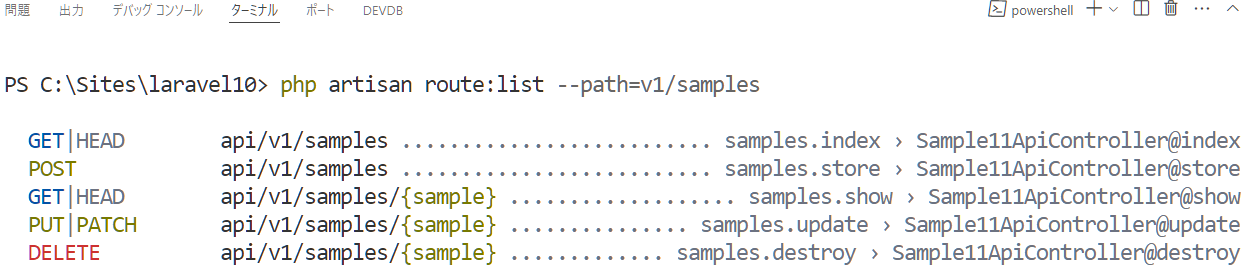
        'v1/samples' => Sample11ApiController::class,

    ]

);

ルーティングの確認を行います。

指定したURLの前に「api/」が付いていること、createやeditなどのルートがないことを確認してください。

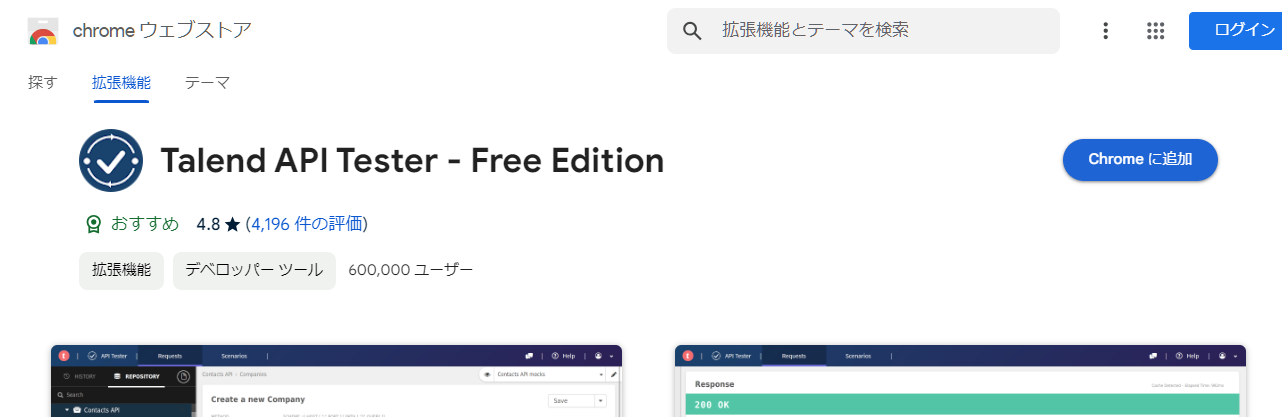


１０－４．APITesterによる確認

ブラウザから実行して、JSONデータが返ってくることを確認します。

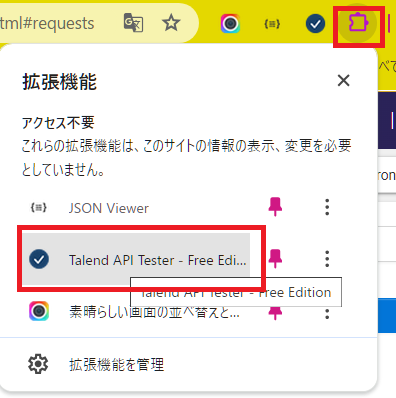
確認にはChromeの拡張機能を利用します。

Google Chromeで「<https://chromewebstore.google.com/detail/talend-api-tester-free-ed/aejoelaoggembcahagimdiliamlcdmfm?hl=ja>」にアクセスしてください。



[chromeに追加]を押下。

Chromeに追加が完了したら、拡張機能を表示します。

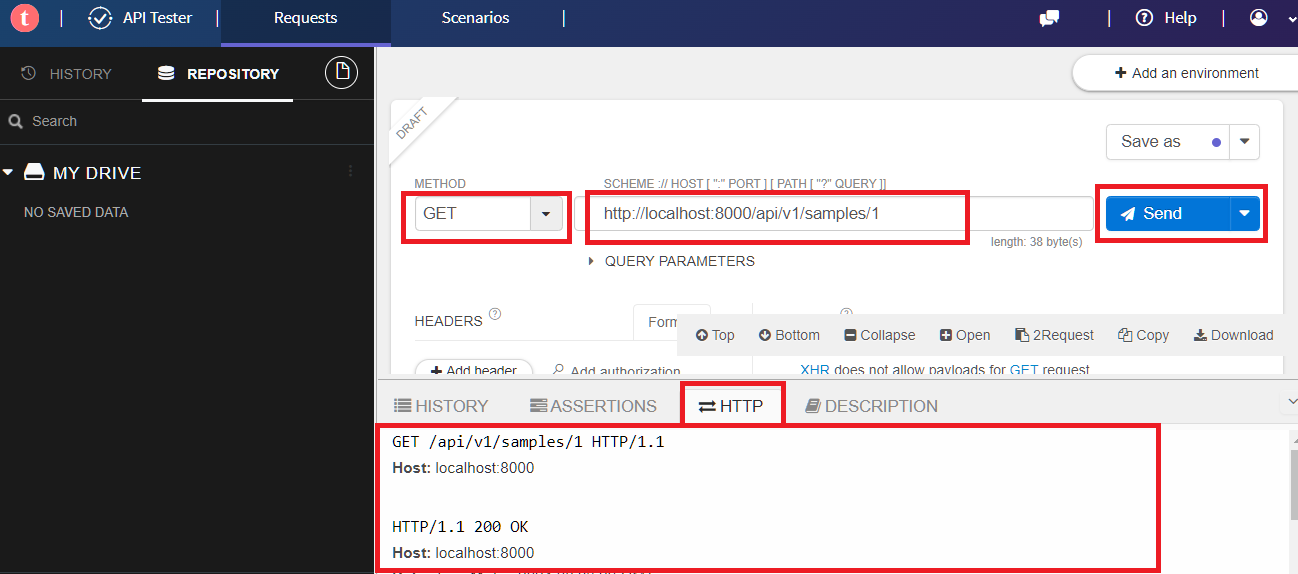


API Testerの画面で「HTTPメソッド」と「URL」を指定します。

サンプルで実装したのは、showメソッドです。HTTPメソッドはGETを使用し、下記のURLを指定します。＊{sample}にはidが入ります。

「[http://localhost:8000/api/v1/samples/{sample}](http://localhost:8000/api/v1/samples/%7bsample%7d)」

1. 左記、入力後「Send」を押下。

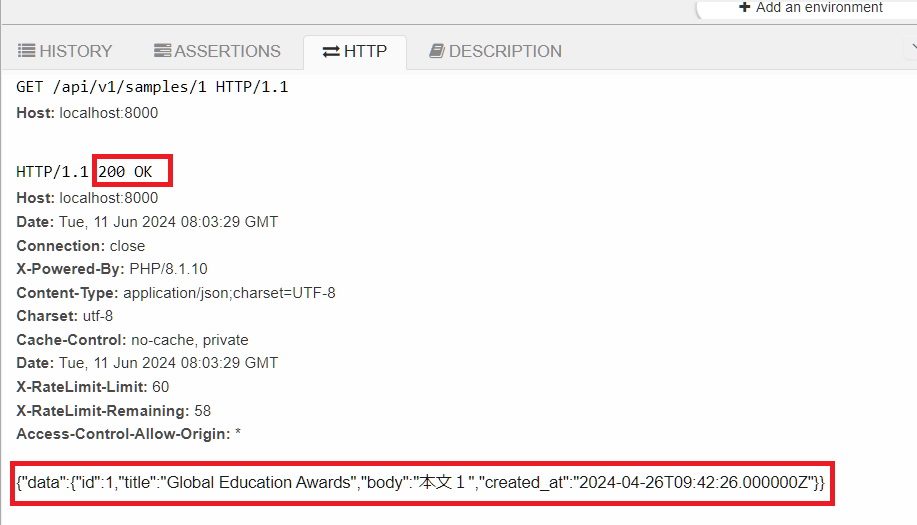


②結果が表示される。

結果は下部の「HTTP」タグで確認可能です。

データが存在した場合、ステータス「200 OK」と末尾にJSONデータが表示されています。

■API Tester（ステータス200）



データがない（存在しないidを指定した）場合、ステータス「404 Not Found」とともに、データがない場合に指定したJSONデータが表示されています。

■API Tester（ステータス404）



１０―４．JSON ViewerによるJSONデータの確認

JSONデータのみを読みやすい形式で確認します。

Chrome拡張機能でJSON Viewerを追加します。

「<https://chromewebstore.google.com/detail/gbmdgpbipfallnflgajpaliibnhdgobh?hl=ja>」にアクセスしてください。



[chromeに追加]を押下。

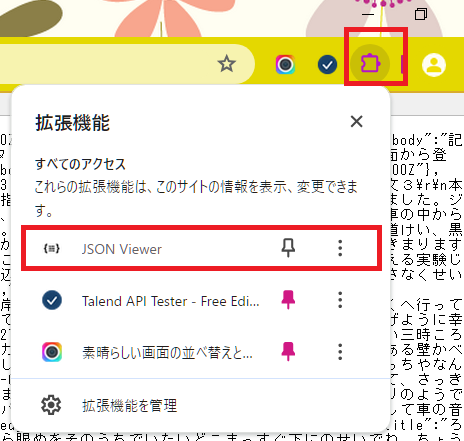
indexメソッドでの全件表示処理を呼び出します。ブラウザで下記を入力してください。

「<http://localhost:8000/api/v1/samples/>」

下記のように全件表示されます。このままだと見づらいので、JSON Viewerで表示形式を整えます。



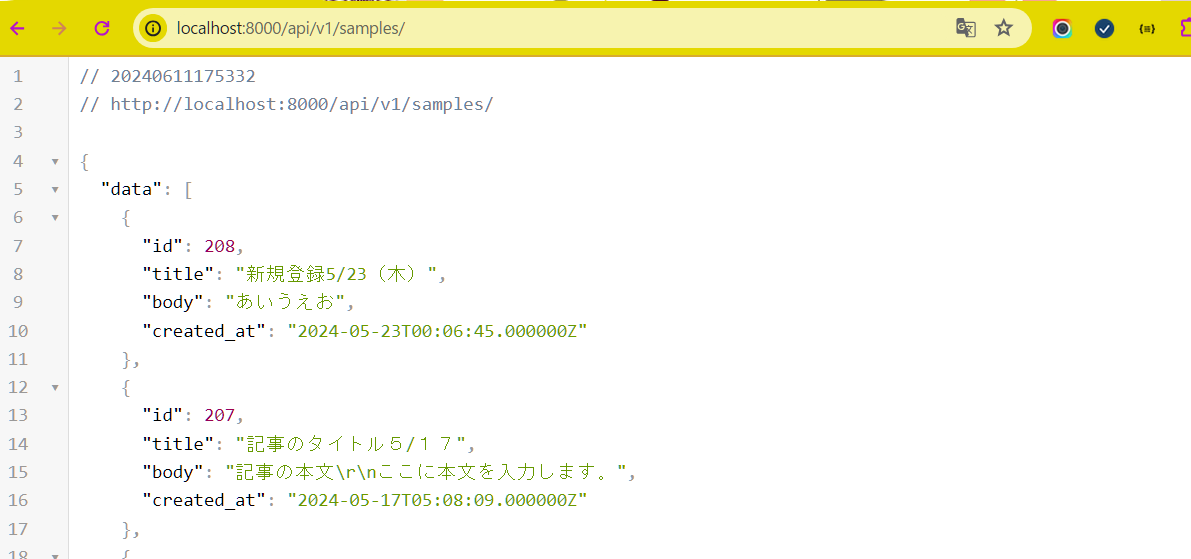
拡張機能から「JSON Viewer」を押下します。



1. ブラウザリロード。
2. 「JSON Viewer」押下。
3. 拡張機能押下。

ブラウザを再読み込みすると、表示が切り替わったことを確認します。

データ内容を確認する場合は、JSON Viewerを活用しましょう。



１０―５．共通処理のメソッド化

返却データにより部分的に異なりますが、JSON形式でのデータ指定を複数回使用しました。

また、response()->jsonメソッドでの第３引数「ヘッダ」と第４引数「オプション」は、今回共通としています。このままだとプログラムが長くなるので、response()->jsonメソッド部分は共通部分としてクラスを分け、メソッド化します。

サンプルは一例なので、実装したい機能などにより設計は異なります。

■共通化したい部分

response()->json(

    [

        "data" => /\* 返却データにより異なる。 \*/

    ],

    /\* レスポンス（OKやNOTFOUND）により異なる。 \*/,

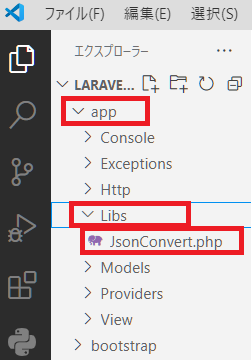
    ['Content-Type' => 'application/json;charset=UTF-8', 'Charset' => 'utf-8'],

    JSON\_UNESCAPED\_UNICODE

);

自作クラスは「プロジェクトフォルダ\app」配下にフォルダを作成します。

サンプルでは「Libs」フォルダを作成し、「JsonConvert.php」ファイルを作成します。



Libsフォルダと

JsonConvert.phpファイルを作成。

■JsonConvert.php

　＊toJsonメソッドを新規作成。

　＊引数はデータ（ApiResource適用後）、およびステータスとする。

<?php

namespace App\Libs;

class JsonConvert

{

    /\*

     \* @param  mixed  $data

     \* @param  int  $status

     \* @return \Illuminate\Http\JsonResponse

     \* \*/

    public function toJson($data = [], $status)

    {

        if (!$data) {

            $data = [

                "status" => "error",

                "message" => "該当する投稿はありません"

            ];

        }

        return response()->json(

            [

                "data" => $data

            ],

            $status,

            ['Content-Type' => 'application/json;charset=UTF-8', 'Charset' => 'utf-8'],

            JSON\_UNESCAPED\_UNICODE

        );

    }

}

Sample11ApiControllerのshowメソッドでこのtoJsonメソッドを利用するように修正します。

■Sample11ApiController.php

　＊showメソッドのみ記述。

　＊「use App\Libs\JsonConvert;」文を忘れずに！！

自作メソッドtoJsonを使用。

データがあるときとない時で、ステータスを管理するようの変数$statusを追加。

public function show(string $id)

{

    //

    $sample = Sample::find($id);

    $convert = new JsonConvert();

    if ($sample) {

        $sample = new SampleResource($sample);

        $status = Response::HTTP\_OK;

    } else {

        $status = Response::HTTP\_NOT\_FOUND;

    }

    return $convert->toJson($sample, $status);

}

同様に、indexメソッドも自作のtoJsonメソッドを使用するように修正します。

■Sample11ApiController.php

　＊indexメソッドのみ記述。

public function index()

{

    $samples = Sample::orderBy("created\_at", "desc")->get();

    $convert = new JsonConvert();

    if ($samples) {

        $samples = SampleResource::collection($samples);

        $status = Response::HTTP\_OK;

    } else {

        $status = Response::HTTP\_NOT\_FOUND;

    }

    return $convert->toJson($samples, $status);

}

１０回目は以上です。