OUR HOTELS API Project Documentation

By

Yaseen Taha

showyaseen@hotmail.com

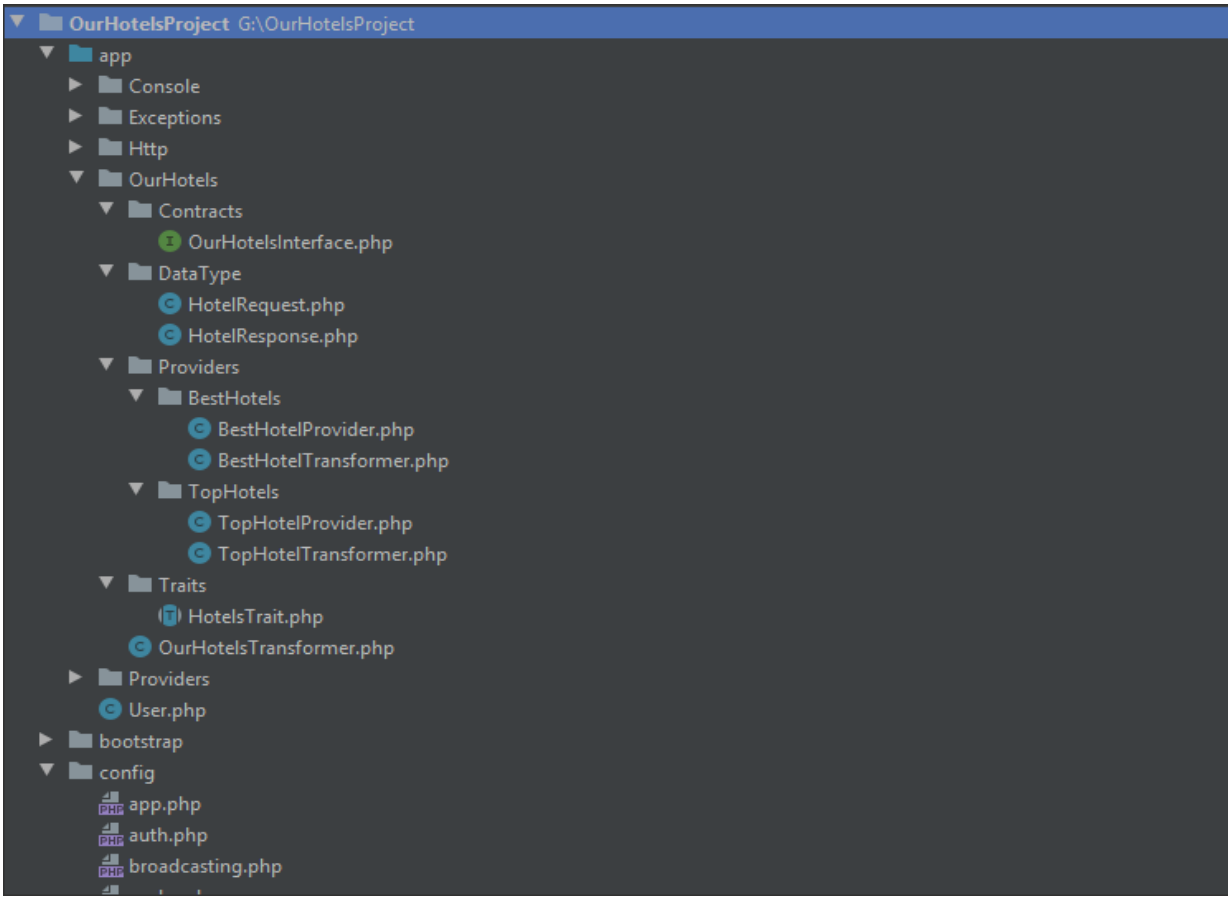
2019-11-16

1. Introduction:

In this Project we defined OurHotels interface with single method “getHotels” to:

- Unified the way we are calling the hotel providers with single method “getHotels” witch every added provider to the system should implement with its custom way.
- Support Dynamic future providers addition by maintain a dynamic list of providers with ability to add/remove providers at the runtime without need to change current implementation.

2. Project Structure:



The project has standard Laravel project structure, along with additional folder called OurHotels contain OurHotels implementation, which consist of the following:

2.1 Contract Folder:

Contain “OurHotelsInterface” which is an interface that should every providers implement it method “getHotels”.

```php
<?php
namespace App\OurHotels\Contracts;

use App\OurHotels\DataType\HotelRequest;

Interface OurHotelsInterface
{
    public function getHotels(HotelRequest $request);
}
```

2.2 DataType folder:

Contain the data type of request/response of OurHotels API.

```php
<?php
namespace App\OurHotels\DataType;

class HotelRequest
{
    public $from_date;
    public $to_date;
    public $city;
    public $adults_number;

    public function __construct(Array $data)
    {
        $this->from_date = $data['from_date'];
        $this->to_date = $data['to_date'];
        $this->city = $data['city'];
        $this->adults_number = $data['adults_number'];
    }
}
```

2.3 Providers Folder:

This folder contain the implementation of any hotel providers added to the system, each provider should have this three components in order to work:

- **First component** the standard Laravel ServiceProvider class that inject the hotel provider implementation as dependency, for example BestHotels ServiceProvider:

```php
<?php

namespace App\Providers\HotelsServiceProvider;

use Illuminate\Support\ServiceProvider;

use App\OurHotels\Providers\BestHotels\BestHotelProvider;

class BestHotelsServiceProvider extends ServiceProvider
{
    /**
     * Register services.
     *
     * @return void
     */
    public function register()
    {
        $this->app->bind('App\OurHotels\Contracts\OurHotelsInterface', function
($app) {
            return new BestHotelProvider();
        });
    }
```

- **Second component**, hotel provider implementation for OurHoteIntrface method “getHotels”, which contain custom way that the provider use to retrieve the hotels data whatever was it is for example database, third party Api ..etc., below BestHotelsProvider class that retrieve dummy data from static array:

```php
class BestHotelProvider implements OurHotelsInterface
{

    public function __construct()
    {
    }

    public function getHotels (HotelRequest $request)
    {
        $hotelCollection  = $this->CallBestHotelAPI(["fromDate"=>$request->from_date,
"toDate"=>$request->to_date,
            "city"=>$request->city, "numberOfAdults"=>$request->adults_number]);

        $transformer = new BestHotelTransformer();
        return $transformer->transformCollection($hotelCollection)['data'];
    }

    private function CallBestHotelAPI($data) {
```

```
        // dummy data return to simulate calling external best hotel api
        return [
            ["hotel"=>"ABC Hotel", "hotelRate"=>2,
"hotelFare"=>"120.01","roomAmenities"=>"aa amenity,bb amenity,vv amenity"],
            ["hotel"=>"XYZ Hotel", "hotelRate"=>3,
"hotelFare"=>"200.00","roomAmenities"=>"df amenity,dsd amenity,sdf amenity"],
            ["hotel"=>"HFG Hotel", "hotelRate"=>1,
"hotelFare"=>"201.09","roomAmenities"=>"df2 amenity,d1 amenity,s2f amenity"],
            ["hotel"=>"SSR Hotel", "hotelRate"=>2,
"hotelFare"=>"900.00","roomAmenities"=>"df2 amenity,dsd2 amenity,sdf2 amenity"],
            ["hotel"=>"NHD Hotel", "hotelRate"=>5,
"hotelFare"=>"1000.00","roomAmenities"=>"df1 amenity,ds2d amenity,sdf2 amenity"],
            ["hotel"=>"ADX Hotel", "hotelRate"=>4,
"hotelFare"=>"100.00","roomAmenities"=>"df1 amenity,dsd2 amenity,sdf3 amenity"],
            ["hotel"=>"AAA Hotel", "hotelRate"=>4,
"hotelFare"=>"1000.00","roomAmenities"=>"df12 amenity,ds2d amenity,sd3f amenity"],
            ["hotel"=>"CCC Hotel", "hotelRate"=>3,
"hotelFare"=>"200.00","roomAmenities"=>"df23 amenity,dsd2 amenity,sdf3 amenity"],
            ["hotel"=>"AQQ Hotel", "hotelRate"=>4,
"hotelFare"=>"900.00","roomAmenities"=>"df23 amenity,dsd2 amenity,sdf3 amenity"],
        ];
    }

}
```

- **Third component**, in any hotel provider implementation is transformer which is a Laravel package that convert custom hotel provider response to unified response of OurHotels , below BestHotelsTransformer as you can see it map BestHotels response to OurHotels response:

```
class BestHotelTransformer extends Fractal\TransformerAbstract
{
    public function transformCollection($hotelCollection)
    {
        $fractal = new Manager;
        return $fractal->createData(new
Fractal\Resource\Collection($hotelCollection, new BestHotelTransformer))->toArray();
    }

    public function transform($hotel)
    {
        return [
            'provider' => 'Best Hotels',
            'hotelName' => $hotel['hotel'],
            'rate' => $hotel['hotelRate'],
            'fare' =>  $hotel['hotelFare'],
            'amenities' =>  explode(',',$hotel['roomAmenities']),
        ];
    }
}
```

2.4 Hotels List Configuration file:

In order to add hotel providers dynamically to the system, without altering existing main code we maintain the implemented hotel provider classes path into list, which will be initiated and invoked during the runtime, this list stored on **hotels.php** configuration file which stored on the following path: Config -> hotels.php, the content of the file is below :

```
return [
    'providers' => [
        'BestHotel' => 'App\OurHotels\Providers\BestHotels\BestHotelProvider',
        'TopHotels' => 'App\OurHotels\Providers\TopHotels\TopHotelProvider',
    ],

];
```

3. Main Controller:

In the project there is main controller that use generic implementation to provide the response of OurHotels Api,

a. First this main class is simply load the list of registered providers from **hotels.php** configuration file, initiate them, and then call all the providers **"getHotels"** method to get hotels data from each provider.
b. Final step do necessary operation on data like sorting and the pass the result to transformer that return user response.

Below code show these operations:

```
class OurHotelsController {
    use HotelsTrait;
    private $providers = [];
    private $hotels = [];

    public function __construct()
    {
        $hotels_providers = config('hotels.providers');
        // add providers from configuration file
        foreach($hotels_providers as $key=>$provider) {
            $this->providers[$key] = new $provider;
        }
    }

    public function index (Request $request)
    {

        $validator = Validator::make($request->all(), [
            'from_date' => 'required|date_format:Y-m-d',
            'to_date' => 'required|date_format:Y-m-d',
            'city' => 'required|string|min:3|max:3',
            'adults_number' => 'required|integer|digits_between:1,11',
        ],[
            'from_date.*' => __('message.from_date'),
```

```
            'to_date.*' => __('message.to_date'),
            'city.*' => __('message.city'),
            'adults_number.*' => __('message.adults_number'),
        ]);

        if ($validator->fails()) {
            return response()->json(['status' => '400', 'errors'=>$validator->errors()],
400);
        }

        $hotel_request = new HotelRequest($request->all());

        foreach($this->providers as $provider) {
            $this->hotels = Arr::collapse([$this->hotels, $provider-
>getHotels($hotel_request)]);
        }

        $final_response_transformer = new OurHotelsTransformer();
        $sorted_by_rate_hotels = $this->sortByRate($this->hotels);
        return $final_response_transformer->transformCollection($sorted_by_rate_hotels);
    }


}
```

4. **How to add new provider?**

   To add new provider to our system through three steps:

   a. Step one, Create the three components of the provider which is: ServiceProivder class, interface implementation class and provider transformer class which map custom provider response to OurHotel standard response.
   b. Step two, Add our ServiceProvider to laravel providers list in app.php.
   c. Step three, add the provider implementation of ourHotels interface to hotels list in hotels.php file.

After performing these three steps the result of the new provider will append to ourHotels Api response dynamically.

5. **Unit Test:**

In this implementation we have developed three unit tests:

1. Calling the Api without authentication which should return unauthorized error with code 400.
2. Calling the Api without request parameters which should return validation errors with code 400.
3. Calling Api with correct parameters and authentication which should return code 200.

Below the implemented test case class:

```
class OurHotelsAPITest extends TestCase
{
    /**
     * A basic feature test example.
     *
     * @return void
     */

    public function testCallAPIWithoutAuthentication()
    {
        $data = [
            'from_date' => "2019-11-03",
            'to_date' => "2019-12-03",
            'city' => 'AUH',
            'adults_ number' => 10,
        ];

        $response = $this->json('GET', '/api/OurHotels',$data);
        $response->assertStatus(200);
    }


    public function testCallAPIWithoutAPIParameters()
    {
        $data = [
            'api_username' => "our_hotels",
            'api_password' => "123456",];
        $response = $this->json('GET', '/api/OurHotels',$data);
        $response->assertStatus(400);
    }


    public function testGetHotelsFromTwoAPIs()
    {
        $data = [
            'api_username' => "our_hotels",
            'api_password' => "123456",];
        $response = $this->json('GET', '/api/OurHotels',$data);
        $response->assertStatus(400);
    }
}
```