Paypal payment gateway integration in Laravel 8

4 months ago By Jitesh Meniya Laravel

Paypal is a international payment processing gateway working in majority of countries that support online money

websites using Laravel framework as backend technology. We will go through step by step from the beginning.

The tutorial will follows below steps:

- Step 1: Create new Laravel application
- Step 2: Install Laravel-Paypal package
- Step 3: Create Paypal account and credentials
- Step 4: Configure package
- Step 5: Create application routes
- Step 6: Create PayPalController controller
- Step 7: Create blade view for payment button
- Step 8: Create a test transaction

So, let's start from creating fresh Laravel application.

Step 1: Create new Laravel application

We are going to start tutorial from creating new Laravel 8 application. We will use Composer application to create latest Laravel application. To start with tutorial, open the Terminal or CMD and run the below Composer command to create Laravel application

```
composer create-project laravel/laravel paypal --prefer-dist
```

After the project created, change Terminal directory to project.

```
cd paypal
```

Step 2: Install Laravel-Paypal package

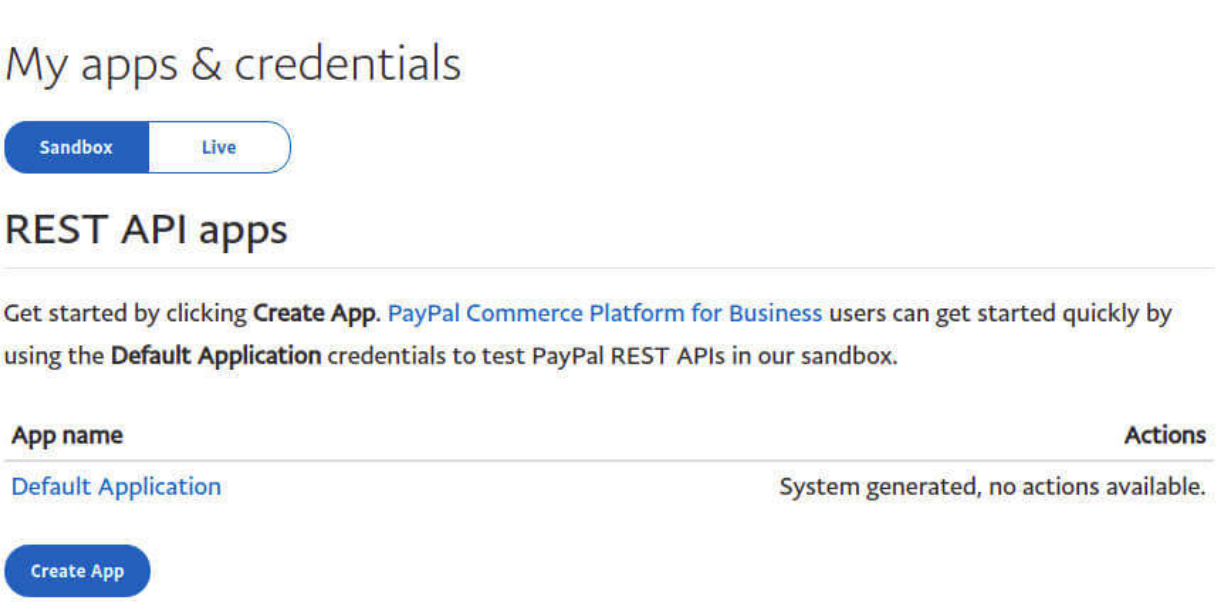
In the second step, we will install Laravel-Paypal package. Using this package, you can also handle refund, payment notification or Rest API. Run the following command to install package.

```
composer require srmklive/paypal:~3.0
```

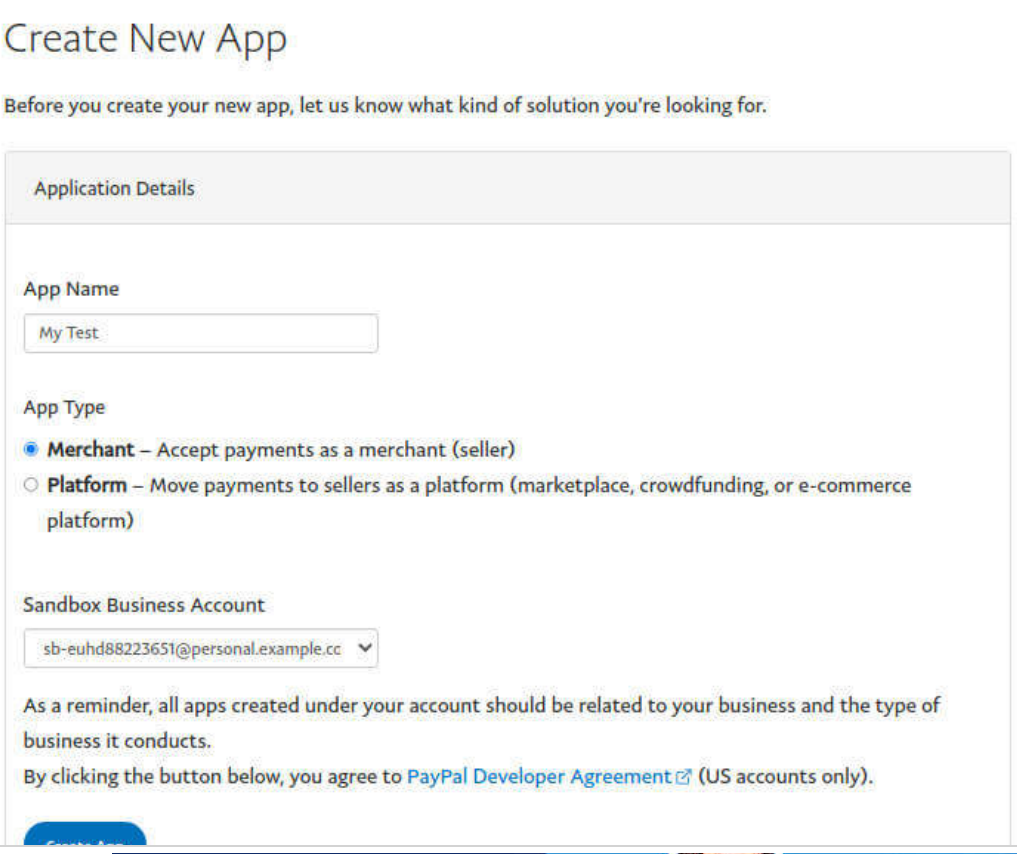
Step 3: Create Paypal account and credentials

We are going to use Sandbox credentials to create test transaction. To generate REST API credentials for the sandbox and live environments, Log in to the Developer Dashboard with your PayPal account. You need to register App and create API credentials. Under the `DASHBOARD` menu, select `My Apps & Credentials` option.

Click on Create App.



Fill the form and create App.



Step 4: Configure package

After package installation complete and getting API credentials, Open the project in your IDE. Add the Paypal API credentials details at the .env file in the root directory.

```
#sandbox or live
PAYPAL_MODE=sandbox
#Paypal sandbox credential
PAYPAL_SANDBOX_CLIENT_ID=AeExqObFcW0X........tjVEuZm
PAYPAL_SANDBOX_CLIENT_SECRET=EOvIUjNQj.......gDTnkhNd
```

```
#Paypal live credential
PAYPAL_LIVE_CLIENT_ID=Jjac8ad......f2sd2faww
PAYPAL_LIVE_CLIENT_SECRET=cak8nd0Pq3mQf2f......jawd2z2d6
```

If you want to customize package default configuration options, run the below vendor:publish command.

```
php artisan vendor:publish --provider "Srmklive\PayPal\Providers\PayPalServiceProvider"
```

This will create `config/paypal.php` configuration file with below details, that you can change it.

```
<?php

return [
    'mode'    => env('PAYPAL_MODE', 'sandbox'),

    'sandbox' => [
        'client_id'         => env('PAYPAL_SANDBOX_CLIENT_ID', ''),
        'client_secret'     => env('PAYPAL_SANDBOX_CLIENT_SECRET', ''),
        'app_id'            => 'APP-80W284485P519543T',
    ],

    'live' => [
        'client_id'         => env('PAYPAL_LIVE_CLIENT_ID', ''),
        'client_secret'     => env('PAYPAL_LIVE_CLIENT_SECRET', ''),
        'app_id'            => '',
    ],

    'payment_action' => env('PAYPAL_PAYMENT_ACTION', 'Sale'),
    'currency'       => env('PAYPAL_CURRENCY', 'USD'),
    'notify_url'     => env('PAYPAL_NOTIFY_URL', ''),
    'locale'         => env('PAYPAL_LOCALE', 'en_US'),
    'validate_ssl'   => env('PAYPAL_VALIDATE_SSL', true),
];
```

Step 5: Create application routes

Now we need to create application routes which we will test application test transaction. Open application route file `routes/web.php` and add following new routes.

```
<?php

use Illuminate\Support\Facades\Route;
```

```
Route::get('success-transaction', [PayPalController::class, 'successTransaction'])->name('successTrans
Route::get('cancel-transaction', [PayPalController::class, 'cancelTransaction'])->name('cancelTransact
```

Step 6: Create PayPalController controller

We have defined routes which redirects to `PayPalController` controller class. So, create controller class using following Artisan command.

```
php artisan make:controller PayPalController
```

This will create new controller class at `app/Http/Controllers/PayPalController.php` file. Open this file and add below code into it.

```php
<?php

namespace App\Http\Controllers;

use Illuminate\Http\Request;
use Srmklive\PayPal\Services\PayPal as PayPalClient;

class PayPalController extends Controller
{
    /**
     * create transaction.
     *
     * @return \Illuminate\Http\Response
     */
    public function createTransaction()
    {
        return view('transaction');
    }

    /**
     * process transaction.
     *
     * @return \Illuminate\Http\Response
     */
    public function processTransaction(Request $request)
    {
        $provider = new PayPalClient;
        $provider->setApiCredentials(config('paypal'));
        $paypalToken = $provider->getAccessToken();

        $response = $provider->createOrder([
            "intent" => "CAPTURE",
            "application_context" => [
                "return_url" => route('successTransaction'),
                "cancel_url" => route('cancelTransaction'),
            ],
            "purchase_units" => [
                0 => [
                    "amount" => [
                        "currency_code" => "USD",
                        "value" => "1000.00"
                    ]
                ]
            ]
        ]);

        if (isset($response['id']) && $response['id'] != null) {

            // redirect to approve href
            foreach ($response['links'] as $links) {
                if ($links['rel'] == 'approve') {
                    return redirect()->away($links['href']);
                }
            }
```

```
            return redirect()
                ->route('createTransaction')
                ->with('error', 'Something went wrong.');

        } else {
            return redirect()
                ->route('createTransaction')
                ->with('error', $response['message'] ?? 'Something went wrong.');
        }
    }

    /**
     * success transaction.
     *
     * @return \Illuminate\Http\Response
     */
    public function successTransaction(Request $request)
    {
        $provider = new PayPalClient;
        $provider->setApiCredentials(config('paypal'));
        $provider->getAccessToken();
        $response = $provider->capturePaymentOrder($request['token']);

        if (isset($response['status']) && $response['status'] == 'COMPLETED') {
            return redirect()
                ->route('createTransaction')
                ->with('success', 'Transaction complete.');
        } else {
            return redirect()
                ->route('createTransaction')
                ->with('error', $response['message'] ?? 'Something went wrong.');
        }
    }

    /**
     * cancel transaction.
     *
     * @return \Illuminate\Http\Response
     */
    public function cancelTransaction(Request $request)
    {
        return redirect()
            ->route('createTransaction')
            ->with('error', $response['message'] ?? 'You have canceled the transaction.');
```

Step 7: Create blade view for payment button

In this step, we will create a view which will redirect to process the transaction. Create a blade view file `resources/views/transaction.blade.php` file and add the below code into it.

```
<!doctype html>
<html>
<head>
    <meta charset="utf-8">
    <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.1.3/dist/css/bootstrap.min.css" rel="sty
    <title>Pay $1000</title>
```

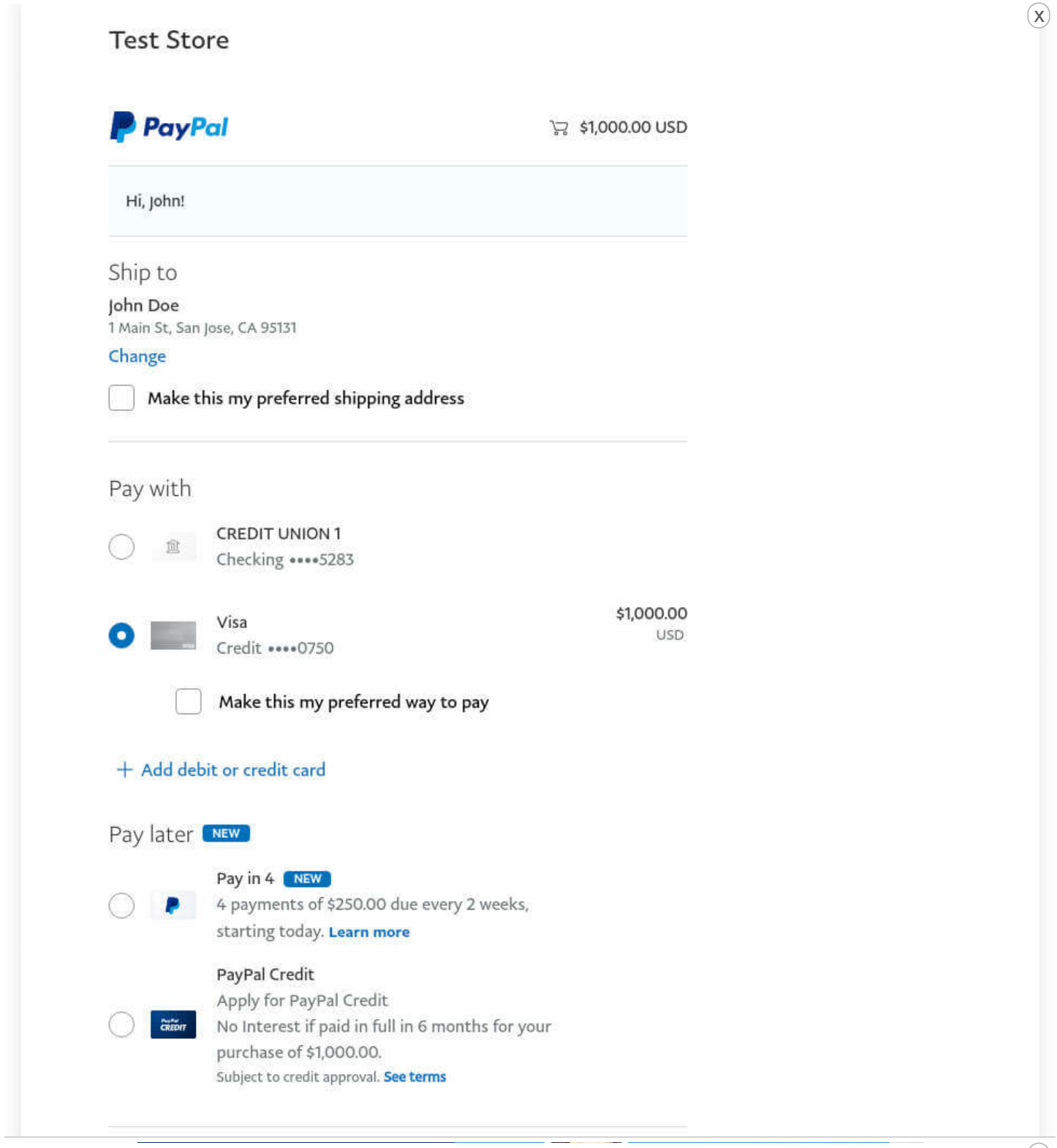
```
    <script src="https://www.paypal.com/sdk/js?client-id={{ env('PAYPAL_SANDBOX_CLIENT_ID') }}"></scri
</head>
<body>
    <a class="btn btn-primary m-3" href="{{ route('processTransaction') }}">Pay $1000</a>
    @if(\Session::has('error'))
        <div class="alert alert-danger">{{ \Session::get('error') }}</div>
        {{ \Session::forget('error') }}
    @endif
    @if(\Session::has('success'))
        <div class="alert alert-success">{{ \Session::get('success') }}</div>
        {{ \Session::forget('success') }}
    @endif
</body>
</html>
```

Step 8: Create a test transaction

Paypal integration is completed. Now we need to create the transaction. Run the Laravel server using below Artisan command.

```
php artisan serve
```

And in your browser window, run the url `http://localhost:8000/create-transaction` and process the transaction. To pay with PayPal account, you need to create sandbox account.



You can browse all API call history at PayPal dashboard.

Conclusion

Finally, our tutorial is over. We have integrated PayPal payment integration. I hope this tutorial will help you on your development. You can download the integration code of this tutorial article from GitHub.

TAGS: PAYPAL PAYMENT GATEWAY LARAVEL 8

worldstatistics.live's server IP address could not be found.

Random **Articles**

Javascript - How to preview multiple images before...

How to install Samba share and transfer files betw...

How to catch cURL errors in php

CSS Quiz

Bootstrap Quiz

Website Down Detector

Javascript Formatter

Base64 Encode

SHA256 Hash

Trending **Categories**

Laravel Articles

PHP Articles

Javascript Articles

Python Articles

Node.js Articles

Advertise

Quick Links

- About Us
- Contact Us
- Privacy Policy
- Terms & Conditions

We are providing