

# PHP - Laravel Socialite : Authentification OAuth avec Google, Facebook et Github (Social login)

Ce TP viens suite à une demande faites par des étudiants et vous serez très important en tant que développeur. Il s'agit ici de vous apprendre à installer, configurer et utiliser le package `laravel/socialite` pour connecter et/ou inscrire les utilisateurs avec un compte Google, Facebook, Twitter, Github, Gitlab, Bitbucket ... : Social login

## #Introduction à Socialite

**Socialite** est un package officiel du framework PHP Laravel qui offre un moyen simple et pratique d'authentifier (authentification) les utilisateurs auprès des fournisseurs OAuth.

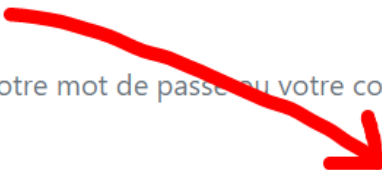
**OAuth**, abréviation de « Open Authorization », est un protocole libre qui permet à un site web ou une application tierce (dit « consommateur ») d'utiliser l'API sécurisée d'un autre site web (dit « fournisseur » ou « provider ») pour le compte d'un utilisateur. C'est l'utilisateur qui autorise ou donne, au site web « consommateur », l'accès à des informations personnelles ou services (accès à l'API) du site web « fournisseur ».

**Laravel Socialite** permet d'intégrer l'authentification avec un compte existant d'un réseau social dans une application Laravel. On appelle cela « social login ».

Ce guide explique comment installer, configurer et utiliser le package `laravel/socialite` dans un projet Laravel pour inscrire et/ou connecter un utilisateur avec son compte Google, Facebook, GitHub ou d'autres providers comme on peut le voir sur les pages de connexion des applications.

## Se connecter

Utilisez votre adresse email + votre mot de passe ou votre compte Gmail, Facebook ou Github pour accéder à votre compte.



Adresse email


Votre adresse email


Mot de passe


Votre mot de passe

☐ Se souvenir de moi


✓ Valider


 Se connecter avec Google

 Se connecter avec Facebook

 Se connecter avec Github

Autres options :

 J'ai oublié mon mot de passe

 Je n'ai pas de compte

## #Installer Socialite

Pour importer Socialite dans un projet Laravel, on exécute la commande composer suivante :

```
composer require laravel/socialite
```

## #Configurer Socialite

Une fois laravel/socialite importé dans le projet, nous avons besoin des **identifiants client OAuth** pour le configurer. En jargon OAuth, on appelle « client » l'application qui accède à des ressources protégées d'un provider OAuth pour le compte d'un utilisateur.

On renseigne les identifiants client OAuth au fichier **config/services.php** où il faut fournir pour chaque provider les informations suivantes :

- « client\_id » : L'identifiant du client
- « client\_secret » : Le mot de passe du client
- « redirect » : L'URL callback, l'adresse de retour du provider à votre application après l'authentification de l'utilisateur

Voici les liens où obtenir les identifiants et gérer les clients OAuth de quelques providers :

- Facebook : <https://developers.facebook.com/apps>
- Google : <https://console.developers.google.com>
- Github : <https://github.com/settings/apps>
- LinkedIn : <https://www.linkedin.com/developers/apps>

La vidéo suivante décrit la procédure pour créer des identifiants client OAuth pour les providers Facebook, Google, Github et LinkedIn : <https://www.youtube.com/watch?v=0TJ1lGgpU-4>

Nous devons finalement avoir le code suivant au fichier config/services.php :

```
<?php
return [
    // ... ,
    'google' => [
        'client_id' => env('GOOGLE_CLIENT_ID'),
        'client_secret' => env('GOOGLE_CLIENT_SECRET'),
        'redirect' => env('GOOGLE_CLIENT_CALLBACK')
    ],
    'facebook' => [
        'client_id' => env('FACEBOOK_CLIENT_ID'),
        'client_secret' => env('FACEBOOK_CLIENT_SECRET'),
        'redirect' => env('FACEBOOK_CLIENT_CALLBACK')
    ],
];
```

```
'github' => [  
    'client_id' => env('GITHUB_CLIENT_ID'),  
    'client_secret' => env('GITHUB_CLIENT_SECRET'),  
    'redirect' => env('GITHUB_CLIENT_CALLBACK')  
]  
];
```

Les informations de mes providers OAuth proviennent du fichier `.env` à la racine du projet où j'ai :

```
# Facebook  
FACEBOOK_CLIENT_ID=523349924960945  
FACEBOOK_CLIENT_SECRET=8d7a23cd4fb58451859d84c34a05a9d2  
FACEBOOK_CLIENT_CALLBACK=http://mon-application.test/callback/facebook  
  
# Google  
GOOGLE_CLIENT_ID=792032373421-  
u6l1fanhomac0m1gfgfiuco57p1cuor1.apps.googleusercontent.com  
GOOGLE_CLIENT_SECRET=hUrXS9reFz8P_oY90BKhpjwi  
GOOGLE_CLIENT_CALLBACK=https://mon-application.test/callback/google  
  
#Github  
GITHUB_CLIENT_ID=ff392218cf3022818ef7  
GITHUB_CLIENT_SECRET=fe37883056da24542cd8296ced30cc420022acf6  
GITHUB_CLIENT_CALLBACK=http://mon-application.test/callback/github
```

Remplacez les valeurs `..._CLIENT_ID`, `..._CLIENT_SECRET` et `..._CLIENT_CALLBACK` par les vôtres.

## #Configurer Socialite

Pour travailler avec Socialite nous avons besoin des éléments suivants dans l'application :

1. **Trois routes** : L'une pour présenter la page avec les liens vers les différents providers, l'autre pour rediriger l'utilisateur vers le provider sélectionné et la dernière pour récupérer les informations provenant du provider (callback).
2. **Un contrôleur** pour gérer les actions des routes.
3. **Une vue** pour présenter les liens de connexion/inscription avec les différents providers

Nous allons donc travailler avec les trois fichiers suivants :

- `routes/web.php`
- `app/Http/Controllers/SocialiteController.php`
- `resources/views/socialite/login-register.blade.php`

## 1. Les routes

Définissons pour tous les providers les routes suivantes au fichier **routes/web.php** :

- « login-register » (GET) pour présenter la page (la vue) avec les liens qui mènent vers les différents providers
- « redirect/{provider} » (GET) pour rediriger l'utilisateur vers le provider
- « callback/{provider} » (GET) pour récupérer des informations de l'utilisateur provenant du provider

Le paramètre {provider} des routes « redirect » et « callback » représente le fournisseur OAuth. Ça sera soit « facebook », « google », « linkedin », « github », ...

**routes/web.php** :

```
# Socialite URLs

// La page où on présente les liens de redirection vers les providers
Route::get("login-register", "SocialiteController@loginRegister");

// La redirection vers le provider
Route::get("redirect/{provider}", "SocialiteController@redirect")->name('socialite.redirect');

// Le callback du provider
Route::get("callback/{provider}", "SocialiteController@callback")->name('socialite.callback');
```

## 2. Le contrôleur

Générons le contrôleur où décrire les actions de routes en exécutant la commande artisan suivante :

```
php artisan make:controller SocialiteController
```

Nous obtenons le fichier **app/Http/Controllers/SocialiteController.php**.

Editons-le en insérant les actions :

- loginRegister() pour présenter la **vue resources/views/socialite/login-register.blade.php**
- redirect() pour rediriger l'utilisateur vers le provider (Google, Github, ...) où il va se connecter (s'authentifier) et autoriser notre application
- callback() pour traiter les informations de l'utilisateur provenant du provider

Le code source du contrôleur **SocialiteController.php** :

```
<?php

namespace App\Http\Controllers;
use Illuminate\Http\Request;

use Socialite; // <-- ne pas oublier

class SocialiteController extends Controller
{
    // Les tableaux des providers autorisés
    protected $providers = [ "google", "github", "facebook" ];

    # La vue pour les liens vers les providers
    public function loginRegister () {
        return view("socialite.login-register");
    }

    # redirection vers le provider
    public function redirect (Request $request) {

        $provider = $request->provider;

        // On vérifie si le provider est autorisé
        if (in_array($provider, $this->providers)) {
            return Socialite::driver($provider)->redirect(); // On redirige vers le provider
        }
        abort(404); // Si le provider n'est pas autorisé
    }

    // Callback du provider
    public function callback (Request $request) {

        $provider = $request->provider;

        if (in_array($provider, $this->providers)) {

            // Les informations provenant du provider
            $data = Socialite::driver($request->provider)->user();

            // Les informations de l'utilisateur
            $user = $data->user;

            // voir les informations de l'utilisateur
            dd($user);
        }
        abort(404);
    }
}
```

Tous les providers OAuth fournissent les informations suivantes de l'utilisateur :

- Identifiant
- Nom
- Surnom ou pseudo
- Adresse email
- Avatar

Ces informations sont directement accessibles à travers les méthodes `getId()`, `getName()`, `getNickname()`, `getEmail()`, `getAvatar()` :

```
//...
// Les informations provenant du provider
$data = Socialite::driver($request->provider)->user();

// token
$token = $data->token;

// Les informations de l'utilisateur
$id = $data->getId();
$name = $data->getNickname();
$nickname = $data->getName();
$email = $data->getEmail();
$avatar = $data->getAvatar();
//...
```

### 3. La vue

La vue `resources/views/socialite/login-register.blade.php` présente les liens de connexion/inscription avec les différents providers :

```
@extends("layouts.app")
@section("content")
<div class="container">
    <h1>Se connecter / S'enregistrer avec un compte social</h1>
    <p>
        <!-- Lien de redirection vers Google -->
        <a href="{{ route('socialite.redirect', 'google') }}" title="Connexion/Inscription avec Google" class="btn btn-link" >Continuer avec Google</a>

        <!-- Lien de redirection vers Facebook -->
        <a href="{{ route('socialite.redirect', 'facebook') }}" title="Connexion/Inscription avec Facebook" class="btn btn-link" >Continuer avec Facebook</a>

        <!-- Lien de redirection vers Github -->
        <a href="{{ route('socialite.redirect', 'github') }}" title="Connexion/Inscription avec Github" class="btn btn-link" >Continuer avec Github</a>
    </p>
</div>
@endsection
```

## #Connecter – inscrire un utilisateur (Social login – register)

Maintenant que nous avons les informations de l'utilisateur lorsqu'il se connecte et autorise notre application auprès d'un provider, travaillons sur la logique de connexion et enregistrement de ces informations dans la table « users » de la base de données.

Commençons par migrer la table « users » dont le schéma est décrit au fichier `resources/databases/..._create_users_table.php` si ce n'est déjà fait :

```
php artisan migrate
```

La logique que nous allons implémenter pour traiter les informations provenant d'un provider OAuth est la suivante :

1. On récupère l'utilisateur de notre base de données à partir de l'adresse email
2. Si l'adresse email existe dans la base de données (utilisateur existant), on met à jour les informations de l'utilisateur
3. Si l'adresse email n'existe pas (nouvel utilisateur), on enregistre l'utilisateur
4. On connecte l'utilisateur
5. On redirige l'utilisateur vers la route « home »

Implémentons cette logique en complétant l'action **callback** au fichier `app/Http/Controllers/SocialiteController.php` :

```
<?php

namespace App\Http\Controllers;
use Illuminate\Http\Request;

use Socialite;

use App\User;

class SocialiteController extends Controller
{
    // ...

    // Callback du provider
    public function callback (Request $request) {

        $provider = $request->provider;

        if (in_array($provider, $this->providers)) {

            // Les informations provenant du provider
            $data = Socialite::driver($request->provider)->user();
```

```
# Social login - register
$email = $data->getEmail(); // L'adresse email
$name = $data->getName(); // le nom

# 1. On récupère l'utilisateur à partir de l'adresse email
$user = User::where("email", $email)->first();

# 2. Si l'utilisateur existe
if (isset($user)) {

    // Mise à jour des informations de l'utilisateur
    $user->name = $name;
    $user->save();

# 3. Si l'utilisateur n'existe pas, on l'enregistre
} else {

    // Enregistrement de l'utilisateur
    $user = User::create([
        'name' => $name,
        'email' => $email,
        'password' => bcrypt("emilie") // On attribue un mot de passe
    ]);
}

# 4. On connecte l'utilisateur
auth()->login($user);

# 5. On redirige l'utilisateur vers /home
if (auth()->check()) return redirect(route('home'));

}
abort(404);
}
}
```

A vous le « social login – register » avec Laravel Socialite !